Un nouvel algoritme pour la simulation DNS et LES des ecoulements cavitants

Anton Znidarcic

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

ARTS ET MÉTIERS ParisTech

2016-ENAM-0056

École doctorale n° 432 : Science des Métiers de l'ingénieur

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École Nationale Supérieure d'Arts et Métiers

Spécialité " Génie énergétique (AM) "

présentée et soutenue publiquement par

Anton ŽNIDARČIČ

le 16 Décembre 2016

Un nouvel algorithme pour la simulation DNS et LES des écoulements cavitants

Directeur de thèse : **Olivier COUTIER-DELGOSHA**
Co-encadrement de la thèse : **Matevž DULAR**

Jury

M. Krishnan MAHESH, Professeur, AEM, University of Minnesota	Président
M. Uwe EHRENSTEIN, Professeur, IRPHE, Université Aix-Marseille	Rapporteur
M. Enrico NOBILE, Professeur, Department of Engineering and Architecture, University of Trieste	Rapporteur
M. Matevž DULAR, Professeur, Laboratory for Water and Turbine Machines, University of Ljubljana, Faculty of Mechanical Engineering	Examinateur
M. Olivier COUTIER-DELGOSHA, Professeur, Arts et Métiers ParisTech (Lille)	Examinateur
M. Stéphane ABIDE, Maître de conférences, LAMPS, Université Perpignan	Examinateur
M. Jean-Philippe LAVAL, Chargé de recherche, Laboratoire de Mécanique de Lille	Examinateur
M. Benoit DE LAAGE DE MEUX, Ingénieur de recherche, EDF Recherche et Développement, EDF	Examinateur
M. Yann DOUTRELEAU, DGA-Direction générale de l'armement	Invité

T H È S E

Arts et Métiers ParisTech - Campus de Lille
Laboratoire de Mécanique de Lille UMR 8107

École Nationale Supérieure d'Arts et Métiers

THÈSE

présentée en vue
d'obtenir le grade de

Docteur

en

Mécanique

par

Anton Žnidarčič

DOCTORAT DELIVRÉ PAR L'ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS

Titre de la Thèse :

Un nouvel algorithme pour la simulation DNS et LES des écoulements cavitants

A novel algorithm for DNS and LES simulations of cavitating flows

Soutenue le 16 Décembre 2016 devant le jury d'examen :

Président du jury,	M. Krishnan MAHESH, Professeur, AEM, University of Minnesota.
Rapporteur,	M. Uwe EHRENSTEIN, Professeur, IRPHE, Université Aix-Marseille.
Rapporteur,	M. Enrico NOBILE, Professeur, Department of Engineering and Architecture, University of Trieste.
Directeur de thèse,	M. Olivier COUTIER-DELGOSHA, Professeur, Arts et Métiers ParisTech, Lille.
Examinateur,	M. Matevž DULAR, Professeur, Laboratory for Water and Turbine Machines, Faculty of Mechanical Engineering, University of Ljubljana.
Examinateur,	M. Stéphane ABIDE, Maître de conférences, LAMPS, Université Perpignan.
Examinateur,	M. Jean-Philippe LAVAL, Chargé de recherche, Laboratoire de Mécanique de Lille.
Examinateur,	M. Benoit DE LAAGE DE MEUX, Ingénieur de recherche, EDF Recherche et Développement, EDF.
Invité,	M. Yann DOUTRELEAU, DGA-Direction générale de l'armement.,

Thèse préparée au Laboratoire de Mécanique de Lille

École Doctorale n° 432 : Sciences des Métiers de l'ingénieur

Acknowledgements

This thesis is the result of the work done during my PhD study under the supervision of prof. Olivier Coutier-Delgosha at ENSAM (École Nationale Supérieure d'Arts et Métiers) Centre Lille. The four years of study on the subjects the reader is about to examine and which in the end led me to successful obtaining of a PhD, were an important time in my life with a lot of enriching experience. Looking narrowly on the discussed topic, the study led me to significantly broaden my knowledge about computational fluid dynamics and numerical methods used in them. Speaking more widely, the study helped me to shape my approach towards conducting not just numerical simulations but also research work in general. It gave me chances to meet and work with pleasant and knowledgeable people, thus making for a professionally very meaningful time. However, the study did not affect only my professional life and experience. It was also a time of very rich and thus important personal experience. Because of this all, I would hereby like to express my gratitude towards many people and organisations who made this work and experience possible and supported me during the study.

First, I would like to mention and thank the people and organisations who made the PhD study and conducting the work presented in this thesis possible. The work was financed and thus enabled by DGA-Direction générale de l'armement (3 years) and EDF-Électricité de France (final year). For this I want to express my deep gratitude to both organisations and a wish that the presented work will be of great benefit for them. My gratitude also goes to my supervisor, prof. Olivier Coutier-Delgosha, and two co-supervisors, prof. Matevž Dular and dr. Matthieu Marquillie. Prof. Coutier-Delgosha enabled me to pursue my PhD by accepting to be my supervisor and providing me with all the support needed in order to perform the work in this thesis. Prof. Dular on the other hand was already my master thesis supervisor, where he encouraged me to pursue the PhD and also helped me to find a position of a PhD student. Both prof. Coutier-Delgosha's and Dular's patient, helpful and open personality meant a lot of support and encouragement during my study, for which I am very thankful. The third mentioned co-supervisor, dr. Marquillie, was my co-supervisor sadly only in the first year of my PhD study. Nevertheless, his extensive help and advice on the used code and numerical methods did not only help me to better advance in the first year but proved very helpful also in the following study. Therefore I would like to thank dr. Marquillie for his crucial contribution to my study. Regarding the used code I am also very thankful to prof. Jean-Philippe Laval and his PhD student, Ilkay Solak. With them, I had many fruitful discussions on the subject of the used code and numerical methods, which were for me very helpful. Considering performing numerical simulations, I would like to express my gratitude to three institutions for granting me the chance to conduct needed calculations on their computers. These are IDRIS (Institut du développement et des ressources en informatique scientifique), CINES (Centre Informatique National de l'Enseignement Supérieur) and Faculty of Mechanical Engineering, University of Ljubljana, Slovenija. ADA computer in IDRIS and OCCIGEN in CINES were used to perform computationally most expensive tasks, while the HPC Prelog computer from Faculty of Mechanical Engineering in Ljubljana was used mainly for

code development and testing. Concerning HPC Prelog I would also like to acknowledge the help of doc.dr. Leon Kos, HPC Prelog system administrator, who spent many hours helping me to trace and resolve the issues I faced on this machine.

On the other side, I want to express my gratitude to people who helped and supported me during this part of my life. At first, I would like to thank all the colleagues, technicians and professors from the laboratory of mechanics in ENSAM Centre Lille. There are many and I cannot name them all here. But I am thankful to each and everyone who I spent time with, as the received help, discussions and exchanges we had brought much of new information and also views into my life. I would nevertheless like to specifically mention and thank following colleagues and friends: Vlasios Leontidis, Monica Veglio, Ilyass Khlifa, Rezki Chebli, Patrick Cherdieu, Ilkay Solak, Merouane Hamdi, Christophe Cuvier and Egoi Ortego. There are also other friends who had an important part in the four years I spent in Lille. Again, I cannot mention them all. But amongst them I want to express my sincere gratitude to Patricija Cimprič, without whom the whole, also personally so enriching experience of these four years, would just not be the same.

Finally, I want to thank all of my relatives, who always provided me with help whenever I needed it. But especially I want to express my deep gratefulness to the people who I hold dearest and actually laid the foundation for me to attend my studies, which in the end also led me to achieve the PhD. These are my parents and my brother. It is difficult to describe the vast amount of support and motivation I received from them and also the sacrifices that especially both my mother and father made in order for me to pursue my studies and thrive. I feel happy and grateful to have such a family and I shall remain always thankful for all the support and help!

Contents

Acknowledgements **3**

Contents **5**

List of Figures **9**

List of Tables **15**

Nomenclature **17**

Introduction **19**

1 Theoretical background **23**
1 Phenomena of cavitation . . . 23
1.1 Cavitation definition . . . 23
1.2 Hydrodynamic cavitation . . . 25
1.2.1 Cavitation number . . . 27
1.2.2 Cavitation nuclei . . . 28
1.3 Effects of cavitation . . . 29
2 Numerical simulations of cavitating flows . . . 30
2.1 Interface tracking methods . . . 30
2.2 Homogeneous mixture methods . . . 31
2.2.1 Two phase models group . . . 32
2.2.2 Single phase models group . . . 32
2.2.2.1 Barotropic models . . . 33
2.2.2.2 Empirical models . . . 34
2.2.2.3 Models based on bubble dynamics . . . 35
3 The problem of cavitation turbulence interactions . . . 38
3.1 Experimental studies . . . 38
3.2 Numerical resolution of considered interactions . . . 44
3.3 On present Direct Numerical Simulations of cavitation . . . 50

2 Goals of this work **53**

3 Used numerical methods, theoretical background and MFLOPS-3D description **57**
1 Incompressible flow governing equations . . . 57
2 Fractional step approach and projection methods . . . 58
2.1 Projection methods definition . . . 58

2.2 Some versions of projection methods . . . 60
2.2.1 The non-incremental pressure-correction scheme . . . 60
2.2.2 The standard incremental pressure-correction scheme . . . 60
2.2.3 The rotational incremental pressure-correction schemes . . . 61
2.2.4 The Kim and Moin scheme . . . 61
2.2.5 Projection method as it is used in MFLOPS-3D code . . . 62
3 Discretization methods . . . 63
3.1 Temporal discretization . . . 63
3.2 Spatial discretization . . . 64
3.2.1 Compact finite differences, general description . . . 65
3.2.2 Compact finite differences near the boundaries . . . 67
3.2.3 Compact finite difference schemes on non uniform grids . . . 68
3.2.4 Compact finite difference schemes as used in MFLOPS-3D code . . . 70
3.3 Non-linear term discretization . . . 71
4 Solving techniques . . . 74
4.1 Mono domain solver . . . 74
4.2 Multi domain solver . . . 77
4.3 Boundary conditions application . . . 82
5 Mapping . . . 84
6 Conclusions . . . 88

4 Development of the new algorithm for cavitating flow simulations 91
1 The governing equations . . . 91
1.1 The reasons for such governing equations . . . 92
1.1.1 Reasons for single phase modelling . . . 92
1.1.2 Reasons for inclusion of α transport equation . . . 92
1.1.3 Reasons for incompressible treatment of phases . . . 93
1.2 Solving cavitating flow governing equations with projection methods . . . 96
2 The new algorithm . . . 101
2.1 Predictor velocity $\vec{v}^*$ solution . . . 101
2.1.1 Skew symmetric form of non linear term . . . 103
2.2 Solution of intermediate variable Φ, pressure and real velocity . . . 105
2.2.1 Possible forms of projection equation and $\Phi - p$ connections . . . 105
2.2.2 Poisson equation for Φ and issues encountered when solving it . . . 107
2.2.2.1 Compatibility condition constraint . . . 108
2.2.3 Solution of the Poisson equation issues . . . 111
2.2.3.1 Relaxation of compatibility constraint . . . 111
2.2.3.2 Source term and velocity update . . . 113
2.3 Solution of vapour volume fraction α . . . 118
2.4 Developed versions of the new algorithm . . . 121
2.4.1 Equalities in all versions of the algorithm . . . 122
2.4.2 Pressure incremental version of the algorithm . . . 124
2.4.2.1 Constant source term basis . . . 124
2.4.2.2 Updated source term basis . . . 127
2.4.3 Pressure non incremental version of the algorithm . . . 130
2.4.3.1 Constant source term basis . . . 130
2.4.3.2 Updated source term basis . . . 133

3 Conclusions . . . 135

5 Used code and algorithm verification approach **139**
1 Method of Manufactured Solutions theory . . . 139
2 Implementation of MMS . . . 142
3 Devised solutions for MMS . . . 143
3.1 Incompressible flow analytical solutions . . . 144
3.2 Cavitating flow analytical solutions . . . 149
4 Conclusions . . . 155

6 Algorithm and code verification and performance tests **157**
1 Test case choice . . . 158
1.1 Chosen domain dimension and factors . . . 158
1.2 Chosen meshes and time steps . . . 161
1.3 Chosen simulation time and VOA criteria . . . 164
2 Verification of the order of accuracy for incompressible flow case . . . 166
2.1 VOA of the original incompressible flow algorithm and code . . . 166
2.1.1 Uniform grid results . . . 167
2.1.2 Non uniform grid results . . . 168
2.2 VOA of the new algorithm and code in case of incompressible flow . . . 174
2.3 Conclusions from VOA with incompressible flow . . . 178
3 Verification of the order of accuracy in cavitating flow . . . 179
3.1 Results and discussion . . . 179
3.2 Conclusions from cavitating flow VOA . . . 192
4 Tests concerning increased factors in forcing terms and stability of the new algorithm . . . 194
4.1 Increased C factor . . . 195
4.2 Increased g factor . . . 200
4.3 Increased a factor . . . 200
4.4 Increase in density and viscosity ratios . . . 203
4.5 Conclusions . . . 208
5 Performance analysis of the code and algorithms . . . 208
5.1 Strong and weak scaling tests . . . 208
5.1.1 Strong scaling . . . 209
5.1.2 Weak scaling . . . 212
5.1.3 Conclusions from scaling results . . . 216
5.2 Old and new algorithm performance comparison through incompressible flow case . . . 216
5.2.1 Conclusions from direct comparison of old and new algorithm computational performance . . . 221
5.3 Performance of new algorithm in MMS cavitating flow case . . . 221
5.3.1 Conclusions . . . 227

7 Towards DNS simulations **229**
1 Instabilities caused by mapping and their removal . . . 229
2 Issues with channel flow simulations . . . 234
2.1 Turbulent periodic flow simulations . . . 234
2.2 Laminar channel flow simulations . . . 237
2.3 Conclusions from turbulent and laminar channel flow simulations . . . 246

3 Attempt to address discrete compatibility problem . . . 247
3.1 Conservative formulation of first derivative . . . 247
3.2 Laplace operator definition . . . 249
3.3 Conclusions about attempts to improve discrete compatibility . . . 256

Conclusions and future work **259**

8 Appendix **265**
1 Compact finite difference schemes, convergence tests . . . 265
1.1 First derivative orders of accuracy . . . 266
1.2 Second derivative orders of accuracy . . . 267
1.3 First derivative order of accuracy with new schemes for discrete compatibility . . . 268
1.4 Second derivative order of accuracy with new schemes for discrete compatibility . . . 269
2 3D eigendecomposition in MFLOPS-3D . . . 270
3 Proof for used connections between p and Φ . . . 272
3.1 Φ - p connection proof . . . 272
3.2 Φ_2 - p connection proof . . . 274
4 Used forcing terms from analytical solutions for incompressible flow . . . 275

Bibliography **277**

List of Figures

1.1 $p-T$ and $p-v$ diagrams . . . 24
1.2 Travelling cavitation . . . 26
1.3 Attached cavitation . . . 26
1.4 Vortex cavitation . . . 26
1.5 Cavitation nuclei density . . . 28
1.6 Barotropic cavitation model . . . 33
1.7 $4,3° - 4°$ converging-diverging part venturi geometry . . . 39
1.8 $18° - 8°$ converging-diverging part venturi geometry . . . 39
1.9 Re-entrant jet mechanism illustration . . . 40
1.10 Experimentally observed re-entrant jet cloud shedding . . . 41
1.11 Cavitating flow results in $18° - 8°$ converging-diverging part venturi with standard $k-\varepsilon$ RNG and $k-\omega$ model . . . 45
1.12 Reboud's limiter function . . . 46
1.13 Cavitating flow results in $18° - 8°$ converging-diverging part venturi with modified $k-\varepsilon$ RNG and $k-\omega$ model . . . 46
1.14 μ_t prediction on the basis of experimentally measured velocity and density fields . . . 49

2.1 Test section from experiments with X-rays . . . 53
2.2 Geometry used in DNS simulations of adverse pressure gradient with MFLOPS-3d . . . 55

3.1 An example of mapping of a geometry into rectangular domain . . . 85

4.1 Graphical illustration of Rayleigh-Plesset equation . . . 95
4.2 Example of discontinuous ρ gradient caused by the use of 8^{th} order compact finite differences . . . 120
4.3 Example of smaller discontinuities in ρ gradient caused by the use of second order compact scheme . . . 120
4.4 Graphical presentation of PICS algorithm version . . . 126
4.5 Graphical presentation of PIUS algorithm version . . . 129
4.6 Graphical presentation of PNCS algorithm version . . . 132
4.7 Graphical presentation of PNUS algorithm version . . . 134

5.1 Time development of terms in x direction NS momentum equation when incompressible flow MMS forcing terms are used . . . 146
5.2 Time development of terms in y direction NS momentum equation when incompressible flow MMS forcing terms are used . . . 146
5.3 Instantaneous values of terms in x and y direction NS momentum equations at $t = 0,25\ s$ when incompressible flow MMS forcing terms are used . . . 147

5.4 Instantaneous values of terms in x and y direction NS momentum equations at $t = 1,75\ s$ when incompressible flow MMS forcing terms are used 148
5.5 Time development of terms in x direction NS momentum equation when cavitating flow MMS forcing terms are used 152
5.6 Time development of terms in y direction NS momentum equation when cavitating flow MMS forcing terms are used 152
5.7 Instantaneous values of terms in x and y direction NS momentum equations at $t = 0.25\ s$ when cavitating flow MMS forcing terms are used 153
5.8 Instantaneous values of terms in x and y direction NS momentum equations at $t = 1.75\ s$ when cavitating flow MMS forcing terms are used 154

6.1 Highest pressure and source term changes during two time steps 160
6.2 Pressure change in limiting positions during one cycle 160
6.3 u and v velocity convergence slopes for original algorithm in mono and multi domain calculations on uniform grids 167
6.4 Pressure convergence slopes for original algorithm in mono and multi domain calculations on uniform grids 167
6.5 u and v velocity convergence slopes for original algorithm in mono and multi domain calculations on non uniform grids 169
6.6 Pressure convergence slope for original algorithm in mono and multi domain calculations on non uniform grids 169
6.7 u velocity results and errors at $t = 25,2\ s$ in incompressible flow calculations with original algorithm on starting grid 171
6.8 Pressure results and errors at $t = 25,2\ s$ in incompressible flow calculations with original algorithm on starting grid 171
6.9 u velocity results and errors at $t = 25,2\ s$ in incompressible flow calculations with original algorithm on finest grid 172
6.10 Pressure results and errors at $t = 25,2\ s$ in incompressible flow calculations with original algorithm on finest grid 172
6.11 u and v velocity convergence slopes with PIUS and PNCS versions of the new algorithm for the incompressible flow case 174
6.12 Pressure convergence slopes with PIUS and PNCS versions of the new algorithm for the incompressible flow case 175
6.13 Pressure solution and its relative errors for PIUS and PNCS algorithm in incompressible flow case at $t = 25,2\ s$ 177
6.14 u velocity convergence slopes for all four versions of the new algorithm in case of cavitating flow 180
6.15 v velocity convergence slopes for all four versions of the new algorithm in case of cavitating flow 180
6.16 Pressure convergence slopes for all four versions of the new algorithm in case of cavitating flow 181
6.17 α convergence slopes for all four versions of the new algorithm in case of cavitating flow 181
6.18 Cavitating flow case v velocity results for all four new algorithm versions and multi domain computations 182
6.19 α results for all four new algorithm versions and multi domain computations . . . 183

6.20 u velocity relative errors on a medium grid at $t = 25,2\ s$ for all four new algorithm versions in mono and multi domain calculations . . . 185
6.21 u velocity relative errors on finest grid at $t = 12,6\ s$ for all four new algorithm versions in mono and multi domain calculations . . . 186
6.22 Pressure relative errors on a medium grid at $t = 25,2\ s$ for all four new algorithm versions in mono and multi domain calculations . . . 187
6.23 Pressure relative errors on finest grid at $t = 12,6\ s$ for all four new algorithm versions in mono and multi domain calculations . . . 188
6.24 α relative errors on a medium grid at $t = 25,2\ s$ for all four new algorithm versions in mono and multi domain calculations . . . 189
6.25 α relative errors on finest grid at $t = 12,6\ s$ for all four new algorithm versions in mono and multi domain calculations . . . 190
6.26 Pressure results at $t = 12,6\ s$ on finest grid for PICS and PNUS version of the new algorithm . . . 191
6.27 Pressure results at $t = 12,6\ s$ on finest grid for PICS and PIUS version of the new algorithm in mono and multi domain calculations . . . 191
6.28 Results with PICS algorithm version in case of increased C constant . . . 196
6.29 Results with PIUS algorithm version in case of increased C constant . . . 197
6.30 u velocity relative errors from all four algorithm versions for increased C and Δt . . . 198
6.31 v velocity relative errors from all four algorithm versions for increased C and Δt . . . 198
6.32 Pressure relative errors from all four algorithm versions for increased C and Δt . . . 199
6.33 α relative errors from all four algorithm versions for increased C and Δt . . . 199
6.34 Results with PICS algorithm version in case of increased g constant . . . 201
6.35 Results with PICS algorithm version in case of increased a constant . . . 202
6.36 Results with PICS algorithm version in case of increased density and viscosity ratios 204
6.37 Results with PNUS algorithm version in case of increased density and viscosity ratios 205
6.38 u velocity relative errors from all four algorithm versions for increased density and viscosity ratios . . . 206
6.39 v velocity relative errors from all four algorithm versions for increased density and viscosity ratios . . . 206
6.40 Pressure relative errors from all four algorithm versions for increased density and viscosity ratios . . . 207
6.41 α relative errors from all four algorithm versions for increased density and viscosity ratios . . . 207
6.42 Strong scaling results for average time needed to obtain a complete time step solution 209
6.43 Strong scaling results for average amount of iterations performed to obtain an influence matrix solution . . . 210
6.44 Strong scaling results for average time to obtain an influence matrix solution . . . 210
6.45 Strong scaling results for average time of an iteration in an influence matrix solution 210
6.46 Strong scaling results for average time needed per grid point to perform an influence matrix solution iteration . . . 211
6.47 Weak scaling results for average time needed to obtain a complete time step solution 213
6.48 Weak scaling results for average amount of iterations performed per time step to obtain an influence matrix solution . . . 213
6.49 Weak scaling results for average time to obtain an influence matrix solution . . . 214
6.50 Weak scaling results for average time of an iteration in an influence matrix solution 214

6.51 Weak scaling results for average time needed per grid point to perform an influence matrix solution iteration . . . 214
6.52 Average computational times to obtain a time step solution in incompressible flow case using mono or multi domain calculations . . . 217
6.53 Average computational times to obtain a time step solution in incompressible flow case using mono or multi domain calculations on Ada cluster . . . 218
6.54 Average influence matrix iterations of all four observed variables per time step in incompressible flow case . . . 218
6.55 Average complete time step computational times with PIUS and PNCS algorithm versions in incompressible flow case on HPC Prelog . . . 219
6.56 Average complete time step computational times with PIUS and PNCS algorithm versions in incompressible flow case on Ada cluster . . . 220
6.57 Average amount of CG iterations with PIUS and PNCS algorithm versions in incompressible flow case . . . 220
6.58 Average amount of influence matrix solution iterations per one CG iteration with PIUS and PNCS algorithm versions in incompressible flow case . . . 220
6.59 Average computational times for a time step solution for all four versions of the new algorithm . . . 222
6.60 Ratios between computational times in mono and multi domain computations or cavitating and incompressible flow cases with all four new algorithm versions . . . 222
6.61 Average amounts of CG iterations for u velocity in multi and mono domain cavitating case calculations with all four new algorithm versions . . . 224
6.62 Average amounts of CG iterations for v velocity in multi and mono domain cavitating case calculations with all four new algorithm versions . . . 224
6.63 Average amounts of CG iterations for w velocity in multi and mono domain cavitating case calculations with all four new algorithm versions . . . 224
6.64 Average amounts of CG iterations for Φ in multi and mono domain cavitating case calculations with all four new algorithm versions . . . 225
6.65 Average amounts of influence matrix iterations for all considered variables in multi domain cavitating case calculations with the four new algorithm versions . . . 225

7.1 Example of studied venturi test section and mesh applied to it . . . 230
7.2 Example of problematic results obtained for the venturi test section . . . 230
7.3 Amplitude of y direction non Cartesian part of Laplacian operator in a simple venturi geometry . . . 231
7.4 Changed amplitude of y direction non Cartesian part of Laplacian operator in a simple venturi geometry . . . 232
7.5 Stable flow simulations in venturi geometry after applying double solution for Φ . 233
7.6 Mean velocity profile results obtained in LES simulations of periodic turbulent channel flows . . . 236
7.7 Examples of instantaneous u velocity for $Re_\tau = 400$ flow . . . 236
7.8 Example of u velocity oscillations appearing in dependence of number of sub domains 238
7.9 Example of pressure oscillations appearing in dependence of number of sub domains 238
7.10 u velocity fields on middle xy and xz planes in cases of different simulation settings 240
7.11 Pressure fields on middle xy and xz planes in cases of different simulation settings 241
7.12 $\nabla \cdot \vec{v}$ fields on middle xy planes in cases of different simulation settings . . . 242
7.13 u velocity and Φ gradient discontinuities on interfaces . . . 244

7.14 u velocity on middle xz plane in cases of using exact outlet velocity 245
7.15 Φ gradient in x direction at middle y position for exact outlet velocity 245
7.16 Relative difference between inflow and outflow 246
7.17 Relative velocity divergence plots 246
7.18 Results with redefined and original Laplace operator in mono domain calculations 253
7.19 Results with redefined and original Laplace operator in multi domain calculations 254
7.20 First derivative results and absolute error for both usual and new schemes 255
7.21 Second derivative results and absolute error for both usual and new schemes . . . 255

8.1 First derivative convergence slopes for 4^{th} order schemes 266
8.2 First derivative convergence slopes for 8^{th} order schemes 266
8.3 Second derivative convergence slopes for 4^{th} order schemes 267
8.4 Second derivative convergence slopes for 8^{th} order schemes 267
8.5 First derivative convergence slopes of new schemes for uniform grid 268
8.6 First derivative convergence slopes of new schemes for non uniform grid 268
8.7 Second derivative convergence slopes of new schemes for uniform grid 269
8.8 Second derivative convergence slopes of new schemes for non uniform grid 269

List of Tables

1.1 Empirical cavitation models 35
1.2 Bubble dynamics based cavitation models 37

3.1 Implicit and explicit stencils of used compact schemes 70
3.2 Implicit and explicit stencils of used compact schemes for second derivatives in Laplace operator 84

4.1 Differences between original and new MFLOPS-3D code 137

6.1 Meshes used for mono domain VOA computations 162
6.2 Meshes used for multi domain VOA computations 162
6.3 Used time steps for the chosen meshes and the corresponding CFL_{max} 163
6.4 Orders of accuracy for original algorithm results on uniform grids 168
6.5 Orders of accuracy for original algorithm results on non uniform grids 170
6.6 Orders of accuracy for PIUS and PNCS algorithm versions in incompressible flow case 176
6.7 Orders of accuracy for all four versions of the new algorithm in multi domain cavitating flow calculations 179
6.8 Orders of accuracy for all four versions of the new algorithm in mono domain cavitating flow calculations 179
6.9 Cases used to perform strong scaling tests 209
6.10 Cases used to perform weak scaling tests 212

7.1 LES periodic turbulent channel flow simulation settings at $Re_\tau = 400$ and $Re_\tau = 550$ 235
7.2 Performed laminar flow cases in order to study appearance of oscillations 239

Nomenclature

Geometry and computational domain reference frame

x	Streamwise coordinate in Cartesian or mapped coordinate system;
y	Wall normal coordinate in Cartesian or mapped coordinate system;
z	Spanwise coordinate in Cartesian or mapped coordinate system;
$\bar{x}$	Streamwise coordinate in physical domain;
$\bar{y}$	Wall normal coordinate in physical domain;
$\bar{z}$	Spanwise coordinate in physical domain;
η_1, η_2	Lower and upper wall height in a channel or physical domain;
y_a, y_b	Lower and upper wall height in a mapped channel;
t	Time;
Δt	Computational time step;
n	Computational time step level;
i, j, k	Point enumeration in x, y and z direction;
N_z	Number of Fourier modes;
R_{cont}	Continuity equation volume integral;
cr	Continuity convergence criteria;

Abbreviations

NS	Navier-Stokes;
RMS	Root Mean Square;
LDA	Laser Doppler Anemometry;
PIV	Particle Image Velocimetry;
LIF	Laser Induced Fluorescence;
LHS	Left-hand side of equations;
RHS	Right-hand side of equations;
MMS	Method of Manufactured Solutions;

Flow variables and material properties

U, $\bar{u}$	Mean flow velocity;
$\vec{v}$	Velocity vector;
u	Velocity in streamwise (x) direction;
v	Velocity in wall normal (y) direction;
w	Velocity in spanwise (z) direction;
$\vec{v}^*$	Predictor velocity vector;
u^*	Predictor velocity in streamwise (x) direction;
v^*	Predictor velocity in wall normal (y) direction;
w^*	Predictor velocity in spanwise (z) direction;
p	Pressure;
Φ	Intermediate variable, including pressure in itself;
Φ_2	Intermediate variable Φ divided by density;
Φ_D	Dual value of intermediate variable used in mapping;
T	Temperature;
σ	Cavitation number;
∞	Subscript for variable value defined from surrounding liquid;
γ	Phase mass fraction (of vapour, if not specified otherwise);
α	Phase volume fraction (of vapour, if not specified otherwise);
α_{min}	Lowest, always present phase volume fraction (of vapour, if not specified otherwise);
R	Bubble radius;
k	Turbulent kinetic energy;
K	Kinetic energy;
μ_t	Turbulent or eddy viscosity;
c	LES model constant;
Re	Reynolds number;
Re_τ	Friction Reynolds number;
CFL	Courant Friedrichs Lewy number;
CFL_{max}	Highest instantaneous CFL number;
ρ	Density;
μ	Dynamic viscosity;
l, v	Subscripts denoting liquid and vaporous phase respectively;
ψ	Surface tension;

Introduction

This PhD work is devoted to the development of an algorithm and code, suitable to perform efficient or fast DNS simulations of cavitating flows using homogeneous mixture approach and the most practically used cavitation models. The need to create a code like this comes from a reason that nowadays, although making large progress over last decades, the explanation and modelisation of cavitating flows is still incomplete. Cavitating flows feature large density and compressibility variations, turbulence effects and instabilities at various scales. Moreover, we are faced with strong interactions between the biphasic and turbulence effects, which pose a lot of difficulty to develop cavitation and turbulence models separately. This is also one of the main issues with the currently most practically used cavitating flows simulations, which feature RANS turbulence models. Added to that, we also have problems when it comes to validation of numerical results, since it is difficult to have detailed experimental data. Therefore, although making a lot of progress in the last 20 years, the existing approaches still do not provide the detailed information regarding the physical mechanisms that control the flow properties.

With having a tool which can enable fast DNS simulations of cavitation, we intend to obtain an approach to tackle one part of the difficulties posed in simulating cavitating flows. These are the issues we have when cavitation and turbulence are being simulated, as it happens so often in development of turbomachinery, propulsion systems or even on smaller study systems, such as simulations of basic cavitation development and dynamics in a venturi test section or simulations of material erosion caused by cavitation. As it was mentioned, the usual approach today is to use RANS models to simulate cavitation, while LES turbulence models are also making their way into more practical use. These models still do only model, therefore simplify turbulence, which on the other hand affects cavitation appearance. And the effect also goes in the reverse way. We tend to use cavitation models, based on many simplifications, which therefore do not describe cavitation completely, and in the end also affect the turbulence description. It is at least very hard to have a clear development of both models without separating them. But if we separate them, we have only one option in how to make this separation. And this is the use of DNS simulation strategy, where turbulence models are not used. This leads to development of a code like proposed here, which can then be used as a tool to separate cavitation and turbulence models, thus making good conditions for their future improvements.

However, it has to be said that a code for DNS simulations of cavitating flows as a tool should already be available. If we allow ourselves to speak generally, what one would need to do, is just to use an existing code, which already has cavitation models included, and make it run without turbulence models. This, with some simplifications means, refine the mesh enough and use enough computing power to run a DNS simulation. However, this is not practical. A code not created specifically for DNS simulations can show a lot of limitations in such a case, most notably, limitations in the numerical precision, amount of processors which one can use or scalability and time to obtain a solution. This made another motivation in our work, which is to use a code created specifically for DNS simulations. Because a special code for DNS simulations

of incompressible flow, MFLOPS-3D, was already developped in our laboratory, the decision where to start was straightforward. Especially since MFLOPS-3D code was created for fast DNS simulations. This ability is also very important when one considers performing DNS simulations of cavitation, where different cavitation models can be used and compared. Having a code, which enables us to perform simulations faster, opens possibilities to test and compare different models more easily. This also made the decision for the MFLOPS-3D code easier. On the other hand, the choice of MFLOPS-3D code, which was originally specifically created for fast DNS simulations of incompressible flow, meant we also had to chose which cavitation models we will use in our simulations.There exist different cavitation models, which differ in the amount of effects they try to capture and approaches on which they are based. We decided to use models based on homogeneous mixture approach which utilise additional governing equation to describe transport of gaseous phase. Chosen models also feature source terms to govern creation and destruction of cavitation on the basis of applying bubble dynamics or empirical equations. This kind of cavitation modelling is widely used in practical and academical simulations nowadays and therefore makes for a good foundation to introduce improvements.

In the scope of this thesis we came to the point where the code is developed but additional work needs to be done to perform actual DNS simulations of cavitating flows. At the beginning of the work, we expected to utilise the mentioned abilities of the MFLOPS-3D code to perform fast DNS simulations in order to make simulations with different cavitation models and propose improvements in regards to cavitation and turbulence models. This was also our main motivation, but the end results lay on a step lower than this, that is, a code which features a new and verified algorithm suitable for fast DNS simulations. The fact is that the specific numerical methods used in MFLOPS-3D to enable fast simulations demanded at first development of a new algorithm. Important discoveries and development were achieved during this, which results in an important step towards reaching final goal of obtaining better simulations, models and therefore knowledge about cavitation. However, issues imposed by specific methods in MFLOPS-3D also prevented performing wanted real flow simulations. These latter difficulties were actually found to not be exclusive to cavitating flow simulations, but come from the used original version of the code and therefore affect also incompressible flow simulations. We consider these issues and proposed improvements for them very important for future use, not just in the area of cavitating flow simulations, but generally for fast DNS simulations. Therefore the thesis will be in large part dedicated to describing the methods used in the code, encountered problems and our applied solutions or proposed improvements for them.

The thesis includes first theory and literature overview, which is focused on approaches to cavitation modelisation and cavitation-turbulence interactions experimental description and numerical capturing, since these are also the main motivation of our work. This is then followed by an overview of methods in MFLOPS-3D, from which the new code and algorithm were developed. These are the methods, from which the incompressible flow algorithm in MFLOPS-3D code is built and are also widely used in incompressible flow simulations. Moreover, many of today widely performed most practical cavitating flow simulations utilise them as well. Following this, a description of the MFLOPS-3D code in its original form is given, with the explanation on the features enabling fast DNS simulations. After the explanation of the basis code, the new algorithm, with which the code capable of cavitating flow simulations is obtained, is presented. A lot of focus is given here to the development of the approaches that make this new algorithm suitable for the solving techniques as the ones in MFLOPS-3D code. Verification of the new algorithm and the code is then presented. At first, used verification method is explained. This is the Method of Manufactured Solutions and development done to obtain a case for cavitating flow verification

with it enables a novel and thorough verification procedure suitable for algorithms and codes used in cavitating flows. A case for incompressible flow was also defined and comparison between the old and new algorithm and the code is therefore also carried out. Performance analysis of the solvers in MFLOPS-3D code and of both algorithms was also done using the proposed cases and is included in same chapter as verification results. The part after this presents preparation of the code for real flow simulations and issues which affected and finally prevented them. Improvements to resolve them are also given, one was particularly successful, others either demand further development or are more a proposal at the moment. Finally, conclusions are given, with the discussion on the topic of possible future work.

This work was in large part funded by DGA (Direction générale de l'armement) and towards the end also by EDF (Électricité de France). It was also presented in four conferences (ICNAAM 2014, Iahrwg2015, CAV2015 and ISROMAC2016). A paper on the topic of developed new algorithm and verification procedure is to be submitted.

Chapter 1

Theoretical background

It is usual that some theoretical overview of the considered problem and existing approaches or solutions for it are given before the main description of a certain work. This is, as the title reveals, also the goal of this chapter. Following to this, cavitation as a phenomena is at first described. Numerical modelling of it is then described with focus on the cavitation modelling type considered in this thesis. Issues of cavitation-turbulence interactions are explained afterwards, as the fundamental interest for the present work comes from them. These issues are presented with an overview of experimental and numerical studies, concerning the type of cavitation which is the focus of this work. Finally, an overview of DNS simulations on the subject of cavitating flows and turbulence-cavitation interactions captured in them is given.

1 Phenomena of cavitation

1.1 Cavitation definition

Following [1], cavitation describes presence and activity of bubbles or cavities in a liquid. As a phenomena it also marks phase change from liquid to gaseous phase because of pressure drop at a practically unchanged temperature. It is not often precised, but cavitation involves also return from gaseous to liquid state [2], which is the most violent part of this phenomena and as such responsible for the majority of negative cavitation effects observed. Cavitation can be well described with the use of $p-T$ and $p-v$ diagrams, where it is compared also to its related phenomena of boiling. The two graphs are shown on figure 1.1. In both, phase change from liquid to gaseous state happens. But as it can be seen on $p-T$ and $p-v$ diagrams, the change is in the case of boiling conditioned by temperature increase at constant pressure (line A-C), while cavitation involves the mentioned opposite way (line A-B). As cavitation involves also return into liquid state, after which pressure recovers almost to the value as at the start, the process undergoes nearly the same way in the opposite direction (line B-A).

Similarity between cavitation and boiling is also in presence of gaseous phase. Although $p-T$ and $p-v$ diagrams include only vapour as gaseous phase as they are given for a pure substance (water), it should be realised that various gases can also be present in a liquid. They can be present either as dissolved gases, entrapped bubbles or different pockets of gas [1, 2]. When cavitation or boiling takes place, these gases start to separate from the liquid together with vapour. The basic shape which is then formed are bubbles, which can join and grow together or separate, forming different structures. In cavitation, pressure drop causes explosive bubble growth and cavitation structure appearance. Then, return into liquid form follows, after bubbles or structures stop to

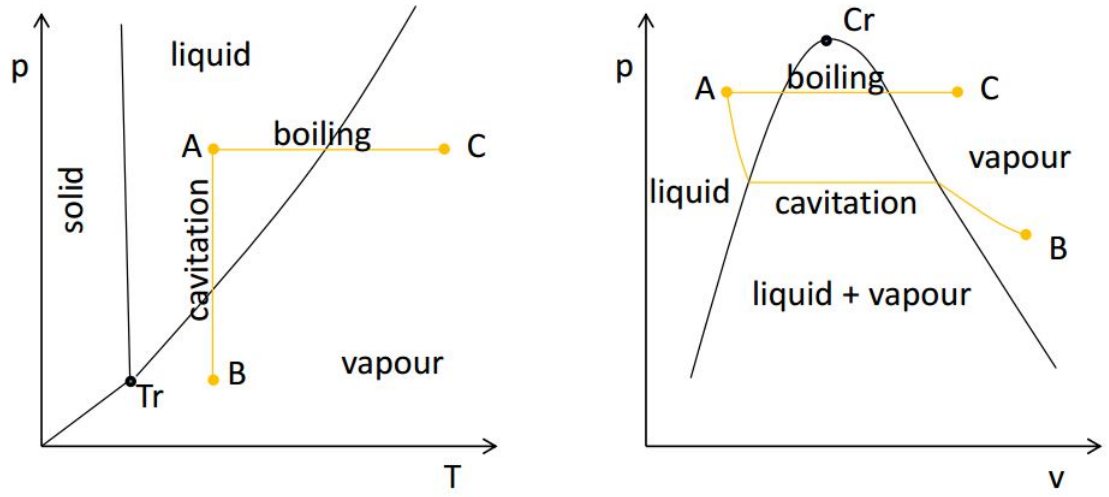


Figure 1.1: $p-T$ and $p-v$ diagrams showing phase changes induced by cavitation and boiling.

grow, because the surrounding pressure starts to increase. This return is called cavitation collapse and is the most violent part of cavitation. Bubbles and other cavitation structures collapse while their space is filled with liquid. High local velocities and pressures are caused during this, which can result even in damage being done to the surrounding solid surfaces [2].

Because of gases being present in a liquid and affecting cavitation appearance, two types of cavitation can be distinguished theoretically. If bubbles, which start to grow at pressure drop, are filled by these gasses, cavitation is referred to as gaseous cavitation. On the contrary, if the bubbles and structures are filled by vapour, cavitation is called vaporous cavitation. Distinction is purely theoretical since bubbles are normally composed by a mix of gases and vapour [1]. It is much more practical to distinguish between different cavitation types according to the main mechanism of production. Four different types of cavitation follow from this [1] :

- Hydrodynamic cavitation, which is caused by pressure changes in a system due to the flow of a liquid and geometry of the system.
- Acoustic cavitation, which occurs in a stationary liquid by sound waves.
- Optical cavitation, which is caused by high intensity photons, hitting liquid molecules and rupturing them, thus causing cavities.
- Particle cavitation, which is relative to the optical cavitation as it involves any other elementary particles causing rupture of a liquid.

The most often encountered is hydrodynamic cavitation. The second most often encountered type is acoustic cavitation. Hydrodynamic and acoustic cavitation are caused by tensions in the liquid, which are otherwise result of a pressure drop. On the other hand, optical and particle cavitation are caused by locally deposited energy in form of particles. Since hydrodynamic cavitation is the one considered by the simulations in the scope of this thesis, it deserves a bit better explanation.

1.2 Hydrodynamic cavitation

The main feature of hydrodynamic cavitation is that it occurs in a moving liquid. Hence it is conditioned by local velocities which determine static pressure. If local velocities are high enough that static pressure drops below a certain critical value, cavitation occurs. Two distinguishing principles exist for this type of cavitation. One is based on the form of cavitation while the other depends on the state of cavitation development [1]. Regarding the form, cavitation can be divided into three main groups. Each of them can be then further divided into more specific cases, which in major part follow from [3] with some distinctions added from [1] and [2]. The three groups and their specific cases are:

- Travelling cavitation, which appears when bubbles or cavities form in the liquid and then travel with it as they grow and collapse. Such cavitation appears in shapes of travelling bubble partial or supercavitation. These two shapes are given on figure 1.2.
- Attached cavitation, which appears when a cavity forms on an edge or a rigid boundary of a body and remains there, in steady or unsteady state. Regarding the state, the steady or quasi-steady state is known as the sheet type of cavitation. Cavitation of this form exhibits smaller and thin structures closely present to the body surface. Unsteady type of attached cavitation would be the cloud cavitation, where a main attached cavity exists with a strong transient regime in which one can observe cavitation structures of different sizes being detached from it (phenomena known as cloud shedding). The final type of attached cavitation is supercavitation, which appears when the cavity covers or envelopes whole body on which it appears (same definition applies to the case of travelling bubble supercavitation). These types are shown on figure 1.3.
- Vortex cavitation, which occurs in the cores of vortices present in high shear regions. Vortex cavitation can express attached or travelling cavitation form, but it is opted here to list it separately from previous two types, as in [1]. The most well known type of vortex cavitation is observed on the tip of axial turbo machines, such as Kaplan turbines or ship propellers. Therefore it is often referred to also as tip cavitation. Three other types of vortex cavitation exist. For two of them it is common that they express Von Karman street and appear either after a bluff body or are convected from the leading edge of a body downstream. Third one appears in the draft tube of Francis turbine and is called cavitating vortex rope. Vortex cavitation types are shown on figure 1.4.

The distinction for hydrodynamic cavitation based on the state of cavitation development makes again for three different groups. Following [1] and [2] these are:

- Incipient cavitation, which marks the first detectable cavitation.
- Developed cavitation, which is the state of cavitation being present with all features of its form.
- Desinent cavitation, which is the last phase in cavitation presence, right before the flow become monophasic again.

Figure 1.2: Examples of partial travelling bubble cavitation (left) and travelling bubble supercavitation (right). Both images are taken from [3].

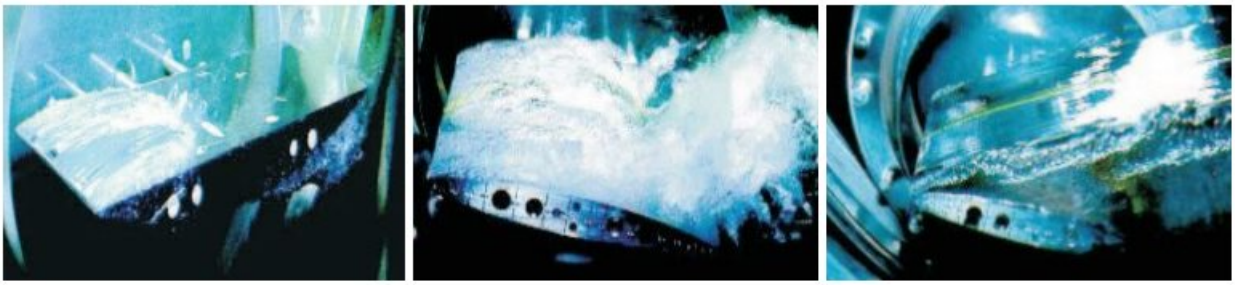


Figure 1.3: Examples of attached cavitation types. Left image shows sheet cavitation, middle one cloud cavitation and the last one supercavitation. Images are taken from [4, 5].

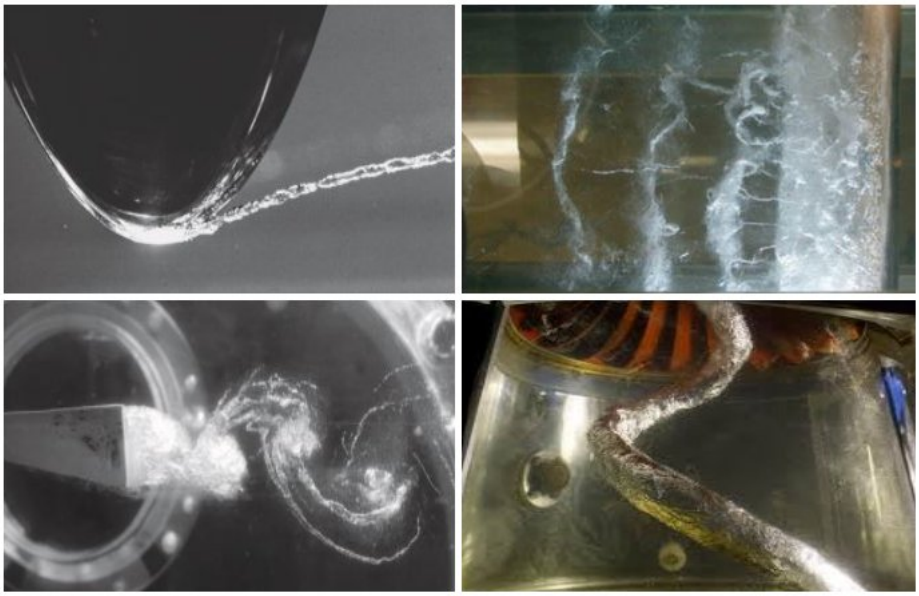


Figure 1.4: Examples of vortex cavitation types. Top left image shows tip vortex cavitation, top right the convected vortex cavitation, bottom left the bluff body vortex cavitation and the bottom right the vortex cavitation rope. Images are taken from [3, 6]

Regarding the given distinctions it is also important to know what affects appearance of cavitation. The conditions which influence hydrodynamic cavitation appearance and disappearance are not always the same, but they can still be put into the following groups:

- Hydraulic parameters, which include flow characteristics such as velocity, pressure, geometry and surface condition of systems or immersed bodies.
- Material properties, which include properties of a certain liquid and its vapour (most often this is water). Cavitation is mostly affected by viscosity, vaporisation pressure, surface tension and dissolution properties of a liquid at certain pressure and temperature.
- Quality of a liquid, which is described with amount of dissolved gases, amount and size of present bubbles of undissolved gasses and presence of solid particles. Existence of all these increases susceptibility of a liquid towards occurrence of cavitation.

Two elements following from the mentioned groups of conditions deserve a bit more explanation since they are often used to describe how prone a certain liquid is to cavitation in a specific system. Furthermore, they are also used in cavitating flow simulations. These two elements are cavitation number and cavitation nuclei, which are presented in the following two paragraphs.

1.2.1 Cavitation number

Cavitation number is a non dimensional number following from hydraulic parameters and gives an estimation for possibility of cavitation appearance in a system [7, 2]. Before introducing this number it should be stated that cavitation is usually roughly considered to occur when static pressure in a certain point reaches vaporisation pressure p_v (for a certain temperature of the liquid) [2]. Because of this it is only logical that a parameter which describes susceptibility of a system to cavitation will include vaporisation pressure in it. The other important parameter is flow velocity U, as velocity increase causes static pressure drop. Cavitation number is then given with equation (1.1).

$$\sigma = \frac{p_\infty - p_v(T_\infty)}{\frac{1}{2}\rho_l U_\infty^2} \tag{1.1}$$

Vaporisation pressure is taken for the temperature of the liquid in a system T_∞. This is considered to be constant in whole system. On the other hand, one cannot know in advance where minimum static pressure and therefore cavitation will occur in a certain system, therefore other variables, pressure p_{infty} and velocity U_∞, are taken from a certain reference point in a system. This is usually at the inlet or outlet.

Every flow has a certain σ. When value of σ is high, the value of p_∞ is much larger than p_v or value of U_∞ is low. Consequently there are smaller chances for cavitation to appear in a system. The opposite applies to the cases of σ being lower. If σ is reduced gradually it follows that cavitation will be observed at a certain value. This is otherwise known as incipient cavitation number σ_i. If σ is further decreased cavitation starts to become more and more pronounced [7].

Cavitation number is important for simulations of cavitating flows since it gives a reference for conditions at which certain hydraulic cavitation is observed. Cavitation number is specifically important in order to set appropriate pressure boundary conditions. Usually the pressure p_∞ is known for the reference point, but cannot be directly applied in cavitating flow simulations, where certain pressure value is applied on the outlet. Therefore cavitation number is used in order to assign appropriate pressure value on the outlet.

1.2.2 Cavitation nuclei

The reason why before mentioned estimation, that cavitation appears when static pressure in a system drops to p_v, is only a rough estimation, is in existence and effect of cavitation nuclei. This topic is indeed a very important one and affects all types of cavitation, not only hydrodynamic one. Here, only a short explanation is given, in order to illustrate this importance and build a basis for later consideration of single cavitation bubble behaviour, on which a large group of cavitation models is based.

When observing occurrence of cavitation, it can be noted that cavitaiton in a certain system does not occur always at the same pressure or same incipient cavitation number [7]. On the other hand, water is known to be capable to withstand tensional loads. Theory states that pure water would rupture at approx. tension of 1000 *bar* [1]. This follows from the molecular forces acting between water molecules. Experiments show that such numbers are impossible to be achieved. Moreover, the load at which the water ruptures changes between different experiments and depends heavily on purity of water and quality of the surface surrounding it [7]. These last two factors are also commonly referred to as cavitation nuclei. These are the reason why cavitation can occur at various σ_i and also explain why it occurs at relatively high pressures close to p_v (compared to theoretical tensile strength). Cavitation nuclei are therefore weak spots in a liquid [7]. Their presence leads to liquid rupture and consequently cavitation at pressures higher than 0 Pa absolute pressure. Two different types of cavitation nuclei exist, homogeneous and heterogeneous [7]. Homogeneous are a consequence of thermal activity or movement of molecules, which cause minute microscopic voids in the liquid. Heterogeneous nuclei are on the other hand a consequence of entrapped gasses in container surface cracks, solid particles and bubbles immersed in the liquid. Since many nuclei of such kind are present in a liquid (especially in water there are many bubbles of contaminant gas) they are more interesting in engineering applications.

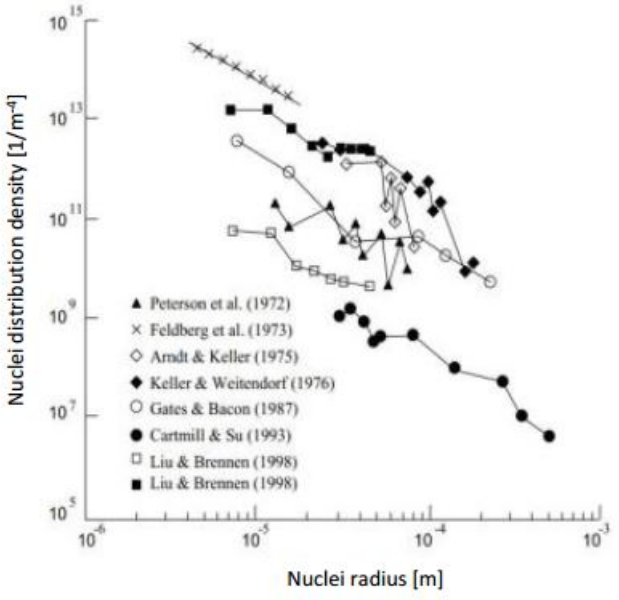


Figure 1.5: Example of cavitation nuclei amount measurements by different authors. Amount of nuclei is given according to the size of nuclei. Graph is taken from [8].

Cavitation nuclei are in simulations usually represented by some amount of gas bubbles in the liquid. This is the simplest way in which nuclei can be included in simulations and is also

reasonably close to reality. For instance, nuclei in form of gas bubbles and solid particles, which can also be modelled as gas bubbles, usually represent highest amount of nuclei present in a liquid. Furthermore, amount of cavitation nuclei in form of entrapped gas bubbles can be also reasonably well defined or predicted since experimental data about it exists, as shown on figure 1.5. Assumption of gas bubbles from which cavitation bubbles and structures grow is also used with great benefit for cavitation modelling, which will be seen later.

1.3 Effects of cavitation

Cavitation as a phenomena is usually considered to be negative and a lot of work is being done in order to prevent it. The negative effects of cavitation can be divided into four groups [2]:

- Mechanical effects, which include changes in flow kinematics, leading to serious consequences as lower flow rate, decreased efficiency, vibration of machine's parts and following damage to them.
- Erosive effecs, which refer to material loss on the parts exposed to cavitation. Erosive effects can be so severe that critical damage to a machine can be done.
- Acoustic effects, which include noise generation because of violent collapses of cavitation structures.
- Other effects, amongst which the most interesting are locally increased temperature and following light emission (sonoluminiscence).

It is clear cavitation effects can be very severe and affect different machines. However, positive or useful effects also exist and are applied to following areas and use [1, 2]:

- Medicine, where cavitation is most often used for lithotripsy, liposuction and lymph drainage,
- Cleaning, where cavitation is used with great benefit to clean hardly reachable spaces. At the same time, cavitation can be used also for most demanding cleaning.
- Material processing, where cavitation for example improves cutting with water jet with forming of pulsating jet.
- Diesel fuel injection, where cavitation enables better homogenisation of droplets, thus increasing the efficiency of a diesel engine.
- Drag reduction, where cavitation can create a sheet of gas covering the surface of a body, thus lowering its skin friction.

These are only a few areas where cavitation is used to improve existing possibilities. Common to all of them and also to prevention of negative effects is that there is a great need to control this otherwise very chaotic phenomena. Numerical simulations are one of the areas which can be used to achieve this.

2 Numerical simulations of cavitating flows

This section gives an overview of existing cavitating flow simulation approaches. Since different types of cavitation are known, different approaches towards modelling and simulating cavitation exist. As hydrodynamic cavitation is being considered in the scope of this work, section deals only with models and simulations which are used for such cavitation. The cavitation modelling chosen in this thesis from available approaches is also revealed while reasons for it will be given later in the thesis.

Cavitation is a very difficult phenomenon to model and simulate. Reasons for this are in the large differences in time, size and material scales present in such flows. As mentioned in [4], time scales in a simulation of pump flow can vary between 10^{-5} s for a cavitation bubble collapse and 10^{0} s for a flow through time for whole pump. Same applies to the size scales, where bubbles or nuclei can be of order 10^{-5} m while complete system has size in 10^{0} m order. As a consequence, computational grid would have to be very refined to capture all important effects, leading to very short time steps. Furthermore, high density and viscosity differences between the present phases (ratio of cca 70000 exists for densities and of cca 100 for viscosities) also presents a challenge to be overcome. Finally, correct phase change description should be added to this, with presence of locally high velocities and pressures at cavitation collapse. It is obvious that cavitating flow simulations therefore demand a lot of simplifications and use of specific numerical procedures. It is no surprise that the first attempts at predicting cavitating flows were very simple. Pressure isolines obtained from a monophasic flow simulation were observed and used for prediction of cavitation occurence. Similarily, potential flow theory was used. With the development of faster computers and also numerical methods, Navier-Stokes equations (also referred to as NS from here on) started to be solved for multiphase flows as well and different simulation approaches for cavitating flow and cavitation models were developed in the last three decades. The simulation approaches or methods can be divided into two main groups, based on the treatment of cavitation structures and the surrounding liquid flow [8, 4, 3, 5]. These are the front or interface tracking and capturing methods. The front capturing methods are also referred to as homogeneous mixture of continuum methods. The names already reveal that the difference between the two is in the sharp interface between the phases being kept in interface capturing methods and no interface presence in the homogeneous mixture methods. The following sections give a bit more detailed description of the two groups.

2.1 Interface tracking methods

As stated, the main feature of this methods is that they assume sharp interface between the two phases. Consequently, they present higher computational cost as deforming interface has to be followed. There are not many examples of such simulations in the case of hydrodynamic cavitation, especially since many cavitation structures, from bubbles to whole clouds, are present and render such simulations very difficult to be done.

Because of this, a typical hydrodynamic cavitation simulation with such approach usually calculates only flow in the liquid phase and assumes constant pressure in the cavitation structure. A wake model is also utilised in order to define the shape of cavity and ensure pressure increase after cavity end [5, 4]. As a result, such approach is limited to the steady cavitating flow cases, where sheet cavitation is the most appropriate example to be simulated. Examples can be seen in [9], where steady cavitation is simulated on different axisymmetric bodies, meaning the simulations are 2D. A mix between the use of potential flow theory and NS equations with such approach is

seen in [10], where cavity shape is at first estimated with potential flow theory and better results are then obtained with solving the NS equations. An example of how interface tracking can be used for a pump geometry is shown in [11]. Nevertheless, simulation is still 2D, which seems to be the limitation, as 3D interface tracking present considerable increase in computational demands.

Contrary to the examples given above, which consider constant pressure in a cavity and solve for the flow only in the liquid phase, the work published in [12, 13] involves application of NS (Euler) equations to both phases. The studied problem is bubbly flow in a jet, where pressure and shock waves with corresponding bubble behaviour (growth, collapse) are observed and following deformation of the jet is predicted. The problem is therefore also unsteady but it has to be noted that it is hard to compare it with other examples mentioned before. The reason is in the fact that behaviour of separate bubbles under the influence of pressure waves is observed, while the movement of the jet or velocity of the liquid plays no role. The case is therefore more resemblant to acoustic cavitation. Nevertheless, it gives an interesting counterpart to the usually encountered interface tracking simulations, especially as it seems to be easily applicable to more hydrodynamic cavitation configuration. It uses the axisymmetric approach, meaning it is still only a 2D simulation.

2.2 Homogeneous mixture methods

Homogeneous mixture methods are presently the most often used methods in cavitating flow simulations. They are based on treating the flow composed from multiple phases as essentially one, continuous phase flow, whose properties change in time and space depending on the local presence of phases. This continuous phase is called the mixture phase. Presence of phases is described by either mass (γ) or volume (α) fractions, where it applies that the sum of all fractions has to equal one, as given in equation (1.2). In it, i stands for the $i-th$ phase from n present phases. Following this, the local properties of mixture phase are defined as an average depending on the fractions, as given in example for mixture density ρ in equation (1.3).

$$\sum_{i=1}^{n} \alpha_i = \sum_{i=1}^{n} \gamma_i = 1 \tag{1.2}$$

$$\rho = \sum_{i=1}^{n} \rho_i \tag{1.3}$$

Such an approach was suggested back in 1960 [2] and it gained a lot of popularity since. The reasons why it is so widely used is in the lower computational costs it offers. As the mixture phase is considered, there is no need for computationally expensive interface tracking methods which could also demand use of shock capturing methods in discretization of equations because of high density and pressure gradients over interfaces. Furthermore, the homogeneous mixture methods still retain physical correctness of results. Various effects can be included in the flow with reasonable simplicity, most importantly the phase change from liquid to gaseous state and back. Depending on the observed phenomena, slip velocity, momentum and energy exchange with various impacts on them can be included as well. However, it should be considered that such an approach has best physical correctness when the phases are well intermingled or mixed, such as in cloud and vortex cavitation. It can also be used for cavitating flows where the structures are clearly separated, such as supercavitation, but with some considerations.

Different approaches based on homogeneous mixture methods exist for cavitation modelling. They can be separated in two main groups, each of them using different assumptions for flow

modellisation. The two groups are referred to as the groups of two phase models and single phase models. They are presented in the following sections.

2.2.1 Two phase models group

This group or models from it can be called also two fluid models [3, 14]. Although homogeneous mixture method is utilised, governing equations are still solved for different phases separately, which resembles the interface tracking methods. However, the phases are intermixed and a governing equation for the mixture phase can also be solved. In fact, solutions for the flow can be obtained in two manners. Either governing equations for all phases are solved or one set of governing equations for a certain phase is replaced with mixture phase governing equations. An example of first approach can be seen in [14], which considers cavitating flow in a fuel injector. The second approach is used in [3], which deals with simulations of cavitating flows in turbo pumps. In both cases, relations for momentum, energy and mass exchange between the phases are needed and modelled.

Regarding the mass transfer or cavitation model, the group of two phase models can be further divided. Different cavitation models can be used and their distinction is the same as in the case of single phase models group. Since models from this are more often used, this distinction will be considered in more detail in the following section. Here, only a division of models into those concerning the present phases as incompressible or compressible is mentioned. In [14], the present phases are assumed to be incompressible, while [3] treats the phases as compressible. Both works use similar cavitation models however, on account that they are based on bubble dynamics.

2.2.2 Single phase models group

In previous paragraph it is mentioned that models from group of single phase models are more often used than those from two phase one. They are in fact most often used models for simulations of hydrodynamic cavitation, from academic to industrial sphere. The reasons are in the assumptions which make the use of this models even easier than the use of two phase models. The no slip assumption between the phases is the most important one. This assumption states that there is no difference in velocity of present phases. Therefore there is no need to resolve two or more groups of governing equations, neither to model the transfer of momentum or energy. Transfer of mass is governed with cavitation models. Same models as in this group can be used also in two phase model group, as mentioned before. Their use originates in works such as [15, 16, 17] from two to almost three decades ago. From then, a whole group of different models was developed.

The developed cavitation models, and with them also single (or two) phase models, can be divided into various groups, which is the distinction mentioned before. Generally, the models can be divided on a basis if they use additional transport equation for vaporous phase or not [4]. More precise distinction between the models would then depend on the assumptions used to model the mass transfer or phase change. Three different groups of cavitation models exist. First group includes models which use equation of state to define the phase change. Such models are here all referred to as barotropic models. Second group consists of models based on bubble dynamics and the final group involves the so called empirical models. Finally, the models can also be divided on a basis if the present phases are considered compressible or not. For the following text, the distinction based on the phase change assumptions are used to present different models.

Because cavitation models from single phase models group present the most practical way to perform simulations of hydrodynamic cavitation and are nowadays used in many applications, they are also chosen as the models to be used for the work in this thesis.

2.2.2.1 Barotropic models

Barotropic models are one of the first models proposed for cavitation simulations with homogeneous mixture methods. The first example of their use is in [17], where a simple barotropic model was used for simulation of cavitation in convergent-divergent channel with inviscid flow assumption. Barotropic models define local vapour volume fraction α through the connection they propose between density and pressure, that is, a barotropic state law $\rho = \rho(p)$. The transition from liquid to vapour occurs in a pressure interval Δp_v around the vaporisation pressure p_v. If the local pressure is higher than $p_v + \Delta p_v/2$, local fluid density equals liquid density ρ_l, while the opposite applies if the local pressure is lower than $p_v - \Delta p_v/2$. In between the two limiting pressures, barotropic models defines the mixture density, and through it also vapour volume fraction, with a chosen transition function or state law. In the case of [17], this is proposed as a sinusoidal transition, which is also shown on figure 1.6. The transition is characterised with its maximum slope $\frac{\partial p}{\partial \rho} = C^2_{min}$, where C^2_{min} is the minimum speed of sound in the mixture. For original barotropic model in [17], this is equal to 2 m/s.

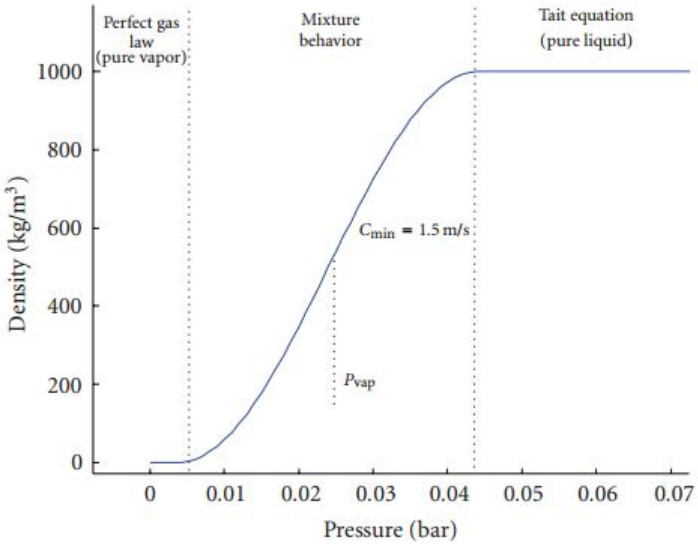


Figure 1.6: Barotropic model as proposed by [17] (taken from [18]).

The mentioned barotropic model was widely used. Its other features are also that it assumes both phases as incompressible and it uses no transport equation for vapour volume fraction α. Hence the model assumes instantaneous change of density with pressure, resulting in demanding use of it. Examples or model use are various. In [19], the model was used to simulate cavitating flow in two venturi test sections, showing quasi-steady behaviour of cavitation in one and unsteady self-oscillating behaviour in the other. The use in 2D or 3D simulations of cavitation in an inducer is shown in [20]. In [18] the model is used in comparison with other models for 2D simulations of cavitating flow on a hydrofoil.

The barotropic model was since much improved and many different forms exist. There was a lot of work done on inclusion of compressible treatment of phases with various equations of state to close the system of governing equations and also on replacement of originally proposed barotropic model with a transition, modelled through a mixture of equations of state and volume fractions of phases. Such approach is not always referred to as barotropic model, but for the sake of simplicity and because it uses same assumptions of density being a direct function of pressure (and also other

variables) it is here referred to as barotropic model. Furthermore, there were also developments to implement combination of barotropic model with transport equation for vapour volume fraction. Example of compressible treatment of phases can be found in [21] where barotropic model is used to simulate cavitation in diesel injector nozzles. In [22] the barotropic model is used in simulations where compressible pressure projection method is presented. Although the phases seem to be effectively incompressible, the simulation presents a method which is appropriate for use in case of compressible phases. The approach where barotropic model is set through properties of phases is presented in [23], where the properties are defined with the help of equations used in steam tables and cavitation on a hydrofoil is simulated. In [24] the barotropic model with included thermodynamic effects is developed from modified bubbly isenthalpic flow model, following development in [7], and results on a hydrofoil are given. Barotropic model, where the transition is modelled through the mix of equations of state for pure phases, is given in [25]. Stiffened gas equations of state are used. The work gives a comparison of the model with a barotropic model based on the originally proposed one in [17], but adapted to include compressibility of phases (in order to avoid infinite sound speeds). Simulations in a venturi geometry were performed and the simple model gave better results. In [26], both models were applied to cavitation simulations, where vapour volume fraction transport equation was also used. Simulations in a venturi geometry were performed and it was shown that the use of the model with stiffened gas equations of state can produce same or better results than use of a simple barotropic model.

2.2.2.2 Empirical models

Empirical models are in a way comparable to the originally proposed barotropic model on the basis that they propose some function which does not respect physical laws strictly to describe transition from liquid to gaseous phase and back. The function, as the name says, follows from empirical results or is shaped in a way to ensure their recreation in simulations. Its main difference from barotropic models is that the phase change according to it happens because of the pressure difference between the local and vaporisation pressure $p - p_v$ and not only in a pre-set interval Δp_v. It also introduces time dependency into it. Empirical models originate from the work in [16], where the phase change was set with equation (1.4). The time dependency in it is given through the use of material derivative. Constant C governs the amplitude of phase change and is chosen to have large value. The reason is the wish to have large phase change with small pressure differences in order to keep pressure in cavitation regions equal or close to p_v.

$$\frac{D\rho}{Dt} = C(p - p_v) \tag{1.4}$$

Empirical models then moved from the above stated to the models which utilise transport equation for vapour volume fraction α as given in (1.5), where source terms govern mass transfer between the used phases. In this equation, m^+ stands for evaporation term and m^- for condensation. First such model, which is nowadays widely used in various simulations, was proposed in 1997. It is called the Merkle model, after [27], although it was before this mentioned in work of other researchers [28]. Interestingly, the empirical models which followed it can be found to be very similar. In order to show this, some models are listed in table 1.1 where they are named by their authors. In the following explanation, the models are shortly presented, with some other models also mentioned.

$$\frac{\partial \alpha_v}{\partial t} + \nabla \cdot (\alpha_v \vec{v}) = m^+ + m^- \tag{1.5}$$

Table 1.1: Empirical cavitation models.

model	vaporisation terms (m^+)	condensation terms (m^-)
Merkle et al	$\frac{C_{prod}\rho_l\alpha_l min(0,p-p_v)}{0.5\rho_l U_\infty^2 t_\infty \rho_v}$	$\frac{C_{dest}(1-\alpha_l)max(0,p-p_v)}{0.5\rho_l U_\infty^2 t_\infty}$
Kunz et al	$\frac{C_{prod}\rho_v\alpha_l min(0,p-p_v)}{0.5\rho_l U_\infty^2 t_\infty}$	$\frac{C_{dest}\rho_v\alpha_l^2(1-\alpha_l)}{t_\infty}$
Senocak&Shyy	$\frac{max(0,p-p_v)(1-\alpha_l)}{(U_{v,n}-U_{l,n})^2(\rho_l-\rho_v)t_\infty}$	$\frac{\rho_l min(0,p-p_v)\alpha_l}{\rho_v(U_{v,n}-U_{l,n})^2(\rho_l-\rho_v)t_\infty}$
Utturkar et al	$\frac{\rho_l max(0,p-p_v)(1-\alpha_l)}{\rho(U_{m,n}-U_{l,n})^2(\rho_l-\rho_v)t_\infty}$	$\frac{\rho_l min(0,p-p_v)\alpha_l}{\rho(U_{m,n}-U_{l,n})^2(\rho_l-\rho_v)t_\infty}$

Following the mass transfer terms listed in table 1.1, it can be seen that the evaporation terms are active only when $p < p_v$ and opposite applies for condensation ones. It can be also seen that α_l, liquid volume fraction, is preferably used than vapour volume fraction. Time t_∞ and velocity U_∞ represent characteristic time and velocity. The models of Merkle [27] and Kunz [29] utilise coefficients C_{prod} and C_{dest}, which are constants used to govern the amplitudes of evaporation or condensation of models. The two constants are therefore used to adjust the models in order to obtain better results in a certain simulation. The Kunz model actually follows from Merkle, but with an important difference. Condensation in it is not assumed to be pressure dependent, it is only triggered when $p > p_v$. Furthermore, Kunz model in the original work [29] also features ability to account for presence of non condensed gasses. Both of them suppose existence of incompressible pure phases. Other two models, the Senocak&Shyy model [5] and Utturkar model [30], are also related. The latter is an evolution of the first. However, it should be first mentioned that both models do not feature adjustable parameters such as C_{prod}, C_{dest}. This follows from their development, where the mass transfer between the phases is developed from modelling vapor-liquid interface behaviour with corresponding mass transfer. Senocak& Shyy model is based on assuming sharp interfaces while Utturkar model assumes non sharp or so called mushy interfaces. The difference leads to different velocities of the interfaces to be included in the model equations, where $U_{m,n}$ is the velocity of a mixture in normal direction and $U_{l,n}, U_{v,n}$ are liquid or vapour velocities in same direction on an interface between the phases. Another difference between the two models is that Senocak model treats present phases as incompressible, while Utturkar model includes compressible phases. Many other empirical models also exist and share similarities to the here presented ones. For instance, in [31] a model following from Merkle model is used in simulations of criogenic inducers, where the phases are contrary to originally proposed model in [27] treated as compressible. A much simpler empirical model in which the source term resembles the one in equation (1.4) is shown in [32].

2.2.2.3 Models based on bubble dynamics

An immense amount of work has been done in cavitation research on the basis of observing and describing single cavitation bubble behaviour, starting from the nuclei form. A must read for anyone who deals with cavitation is the theory concerning Rayleigh-Plesset equation. This equation describes the behaviour of a single bubble in a pressure field. It exists in many forms and it was originally proposed by lord Rayleigh in 1917 without inclusion of all effects on bubble behaviour. After, many researchers have improved it and proposed various versions, a good review

can be found in [1]. The version which is usually referred to as Rayleigh-Plesset equation will be presented later. For the sake of consideration of some models mentioned here, the form which considers only inertial effects on bubble growth is given with equation (1.6). This equation states that velocity of a bubble surface $\frac{dR}{dt}$ depends on the pressure difference between inside of the bubble, where vapour pressure is assumed, and outside or bubble exterior.

$$\frac{3}{2}\left(\frac{dR}{dt}\right)^2 = \pm\frac{\mid p_v - p \mid}{\rho_l} \tag{1.6}$$

Since a single bubble and before it a nuclei, which grows into a bubble, is the basic cavitation structure, it seems only reasonable to model phase transfer with the use of some form of Rayleigh-Plesset equation. Such an approach means that whole cavitation simulated structures in the flow follow bubble behaviour in their origin. Since Rayleigh-Plesset equation considers also non linear effects in bubble behaviour and thus cavitation, the models using it include therefore also these effects. This is an important improvement. However, some assumptions need to be taken, since homogeneous mixture method approach is used. Great majority of the models in this group are based on assuming all bubbles or nuclei, which are present in the liquid, are spherical and have same initial radius. Their distribution is homogeneous with some initial amount in a unit of liquid n_0. Furthermore, bubble coalescence or splitting is not modelled. In all these models, vapour volume fraction is connected with the number and radius of these bubbles. Although the connection would be expected to have equal form in all models, this in not the case. In [15, 33, 34, 35] the connection differs from the one used in [8, 4]. The latter is also given here in equation (1.7) (it is considered as clearer definition) in order to better introduce the source terms proposed by some models. In it, V, V_v and V_l stand for the observed volume and in it included volumes of vapour and liquid. R is the bubble radius, while N gives amount of bubbles in volume V. This is defined through setting an initial amount of nuclei N_0 $[1/m^3]$ in pure liquid. The connection between N_0 and N is then simply $N = N_0 V_l$.

$$\alpha = \frac{V_v}{V} = \frac{N\frac{4}{3}\pi R^3}{N\frac{4}{3}\pi R^3 + V_l} \tag{1.7}$$

Another issue can raise a question regarding the connection between α and bubbles. This is that since the models do not suppose bubble coalescence, their growth could lead to certain imaginary bubble becoming bigger than the distance between two points on the grid or a finite volume. This is possible even though $N = N_0 V_l$ implies decrease in bubble amount when α increases, with the limiting case of one or zero bubbles present when $\alpha = 1$. Nevertheless, it should be realised that the bubbles are only imaginary and the models can and do produce valuable results even when α grows towards 1.

Over the last thirty years, there has been a lot of bubble dynamics based models proposed and they have gained a lot of popularity. One of the first models using Rayleigh-Plesset equation to govern phase transfer was proposed in 1990 [15]. It directly connects phase transition with the governing equations, like the barotropic models. The basis which it uses is a formulation of sub grid scale bubble interactions in order to resemble bubble cloud or group behaviour. This is then introduced into Rayleigh-Plesset equation and a connection between p and bubble radius R (and with it α) is gained and forms a quasi Poisson equation for pressue. Model assumes existence of incompressible pure phases and is on the basis of its formulation restricted to simulation of unsteady cloud cavitation. It is also prone to instabilities. Nevertheless, the model was used in various studies and also further developed. Examples are found in [36], where the model has damping mechanism introduced (through viscous dissipation) and quasi 1D nozzle flow is

Table 1.2: Bubble dynamics based cavitation models.

model	vaporisation terms (m^+)	condensation terms (m^-)
Schnerr-Sauer	$\frac{\rho_l\rho_v}{\rho}\frac{3\alpha(1-\alpha)}{R}\sqrt{\frac{2(p_v-p)}{3\rho_l}}$	$\frac{\rho_l\rho_v}{\rho}\frac{3\alpha(1-\alpha)}{R}\sqrt{\frac{2(p-p_v)}{3\rho_l}}$
Singhal	$F_{vap}\frac{max(1;\sqrt{k})(1-f_g-f_v)}{\psi}\rho_l\rho_v\sqrt{\frac{2(p_{v,s}-p)}{3\rho_l}}$	$F_{cond}\frac{max(1;\sqrt{k})(f_p)}{\psi}\rho_l^2\sqrt{\frac{2(p-p_{v,s})}{3\rho_l}}$
Zwart-Gerber-Belamri	$F_{vap}\frac{3\alpha_i(1-\alpha)\rho_v}{R}\sqrt{\frac{2(p_v-p)}{3\rho_l}}$	$F_{cond}\frac{3\alpha\rho_v}{R}\sqrt{\frac{2(p-p_v)}{3\rho_l}}$

simulated. In [37], a study which will be concerned in bit more detail later, the model was used to perform vortex cavitation studies of submerged jets.

However, more models of this kind exist in configuration where α transport equation including mass transfer term is used. Common to them or their development is that α transport equation is combined with $\alpha - R$ connection, like the one in (1.7). This then leads to a source term, which demands a formulation for a bubble radius change velocity. Here, equation (1.6) comes handy and is applied usually. Three most often encountered models of such kind are shown in table 1.2 where they are named by their authors. One of the first models is the so called Schnerr-Sauer model and its development can be seen in [8]. Same model with nearly equal development is also restated in [4]. Interesting about this model is that its development follows directly from α transport equation and $\alpha - R$ connection. The only empirical constant used in it is the initial nuclei density N_0. And even this follows from physical observations or averaging the amount of nuclei in a certain liquid (using data as the one on figure 1.5). In the model itself, this value is used to define bubble radius R, through the use of equation (1.7). Schnerr-Sauer model is today extensively used and also present in a commercial CFD code Ansys Fluent as the defualt cavitation model [38]. The model is developed on the assumption of incompressible phases but it was applied in cases where phases are treated as compressible too, such as in [39]. Moreover, the model was successfully used also to form a basis of a model accounting for bubble acceleration effects during bubble growth, which proved suitable for simulation of acoustic cavitation on ultrasonic horn where hydrodynamic properties were also exhibited [40]. Opposite to this model, the Singhal model [33] utilises a lot of empirical assumptions once a first form of the source term is obtained from α transport equation and $\alpha - R$ connection. Furthermore, the model does not operate with vapour volume fraction but rather with mass fractions f, although α could be used as well. This model is often referred to also as full cavitation model. The reason for this is in the effects the model tries to account for. At first, the model assumes the presence of non condesable gases, which are included with mass fraction of these gases f_g. Secondly, it accounts for the effects turbulence has on the occurrence of cavitation through increase of vaporisation pressure p_v with the estimation of local turbulent pressure fluctuations P'_{turb}. This results in equation (1.8), which defines new vaporisation pressure $p_{v,s}$. Variable k in it stands for turbulent kinetic energy, which is used also in another empirical estimation. The model namely replaces inclusion of bubble radius R with accounting for maximum possible bubble radius R_b. This is estimated from the balance between aerodynamic drag and surface tension forces, which is finally included in the source term through evaporation and condenstation empirical coefficients F_{vap}, F_{cond}, liquid surface tension ψ and turbulent kinetic energy. This is used as an estimation for the relative velocity between liquid and vapour. The model is, like Schnerr-Sauer model, developed on the assumption of incompressible phases. It is also used in commercial CFD code Ansys Fluent [38].

$$p_{v,s} = p_v + P'_{turb} = p_v + \frac{0,39\rho k}{2} \tag{1.8}$$

Final model in table 1.2 is the Zwart-Gerber-Belamri model [35]. This, like Singhal model, uses empirical constants to form the final source term after an expression for it is obtained from α transport equation and $\alpha - R$ connection. Interestingly, the model proposes an initial vapour volume fraction α_i and bubble radius R, which are then kept constant during simulations. Like the two models mentioned before, this model is originally built on assumption of incompressible phases and was used in various cases. Additionally to Ansys Fluent code it is present also in Ansys CFX [41].

Additionally to here presented models based on bubble dynamics consideration, there exist many other similar models. They can be even more complex and try to account also for bubble splitting, various nuclei amount distribution etc. As mentioned for the case of Schnerr-Sauer model and work with it published in [39], these models can be combined with the assumption of compressible phases as well.

3 The problem of cavitation turbulence interactions

Cavitating flows are difficult to describe since they do not only involve two phases but are also subjected to strong interactions between cavitation and turbulence. There have been many experimental studies performed in order to better describe the mechanisms of turbulent effects on appearance of cavitation and on the interactions between the phases. Numerical studies follow the experimentally obtained results and try to recreate them. Since cavitating flows make experiments or measurements to obtain complete set of informations about velocity, pressure and density fields very difficult, matching of numerical results and modelling with experiments gives a lot of additional insight into the physics of the flow. In this part, the problem of cavitation turbulence interactions and their capturing in numerics will be presented. Since there are many studies on the matter, the presentation gives more attention to the cases which concern or are related to the type of cavitation and simulations considered by the work group in our laboratory, and are also most often used as a benchmark test cases generally. These are experiments and simulations concerning flow in a venturi test section. To be more specific, the test sections considered by our group are the often used venturi geometries with 18° or 4, 3° converging and 8° or 4° diverging part, used in many often referred to studies, especially those performed by LEGI group in Grenoble [42, 43, 44]. Some other experiments and simulations are also mentioned, to make the presentation more complete. At first, an overview of experimental work and findings is given, with an overview of numerical simulations and there encountered issues following afterwards.

3.1 Experimental studies

The broadness of cavitation turbulence interactions can be illustrated first with mentioning some general experimental work done on the subject. The basic structure in cavitation, a cavitation bubble, can already play important effect on these interactions and therefore flow characteristics. For instance, in [45], a travelling bubble cavitation was studied and it was found that micro fluid mechanics is important. Micro fluid mechanics in the case involves separation of bubbles from solid surfaces by thin films, formation of streaks in the bubble wakes and bubble fission and roll-up during the collapse. Because of this, the so often used analysis by Rayleigh-Plesset equation, which assumes spherical and individual bubbles, was said to possibly be insufficient to describe the observed process. In a similar study [46], travelling bubble cavitation was observed to estimate interactions with boundary layer. It was found that bubbles close to the surface can squeeze boundary layer and create streamwise vorticity, thus causing local turbulent regions.

These can then result in sweeping away a portion of the attached cavity. The effect of bubbles was observed also in [47], where vapour bubbles and corresponding vapour volume fraction effect on turbulence was estimated. When vapour appeared in the flow, the turbulent intensity increased. However, this effect was overcome by increase in α because of higher turbulent dissipation which then led to decreased turbulent intensity close to the walls. Other authors also observed increase in turbulent intensity when α was close to zero or when only weak cavitation was present. For instance, work in [48] argues this is a result of bubble interactions.

Where the previous works considered the effect of bubbles on the flow, there were also works which considered the reverse influence, that is, how does the increase of turbulence in one phase flow affect appearance of cavitation. Indeed a strong link between cavitation inception and turbulent flow structures exists [37]. This can be easily shown with the use of vortex or turbulence generators put upstream of the cavitating region. In [49], such a generator was put in front of a venturi and a cavitation inception delay was observed. Turbulence generation was in [50] provoked by rectangular sill on the bottom wall in a channel. 3D features of cavitation structures in turbulent shear layers were then observed and horseshoe shaped cavitation tubes were noted in them. For a case of submerged water jet, experiment in [51] showed location and degree of cavitation structures depends on the nature of vortices. If the jet had turbulence tripped, cavitation appeared in the core of vortex rings, while it otherwise occured in the cores of comparatively strong streamwise vortex tubes downstream of the nozzle. In [52], secondary vortex cavitation was suppressed if the jet boundary layer was tripped. These are only a few works which consider effect of turbulence increase on the development of cavitation, to give an illustration of the importance of interactions in question. As it can be noted, interactions can be expressed as cavitation affecting turbulence, turbulence affecting cavitation, or most generally, as both phenomena affecting each other.

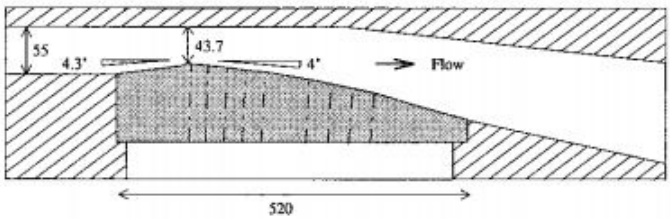


Figure 1.7: The venturi geometry with $4,3° - 4°$ converging-diverging part. (taken from [42]).

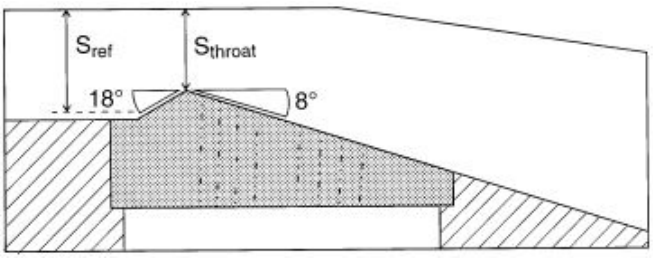


Figure 1.8: The venturi geometry with $18° - 8°$ converging-diverging part (taken from [43]).

For before mentioned single phase models, the most interesting cavitating flows, where important turbulence cavitation interactions can also be observed, are flows which involve sheet or cloud cavitation. Geometries considered in such flows are various hydrofoils or venturi test

sections. Amongst these, the most often used are venturi geometries, where before mentioned two geometries used extensively by LEGI laboratory team stand out. These two are venturi geometries with $4,3° - 4°$ converging-diverging part and $18° - 8°$ converging-diverging part. The two geometries are for illustration shown on figures 1.7 and 1.8. They were used as they offer good representation of a flow in an inducer of a rocket engine. More precisely, venturi with $4,3° - 4°$ converging-diverging part can on the lower wall resemble pressure distribution on a suction side of a blade in an inducer at nominal flow rate, while the sharper venturi geometry can resemble same pressure distribution for partial flow rate. Quasi stable, quasi periodic and unstable cavitation was studied in these two geometries [42, 43, 44, 53]. A very important mechanism for cavitation occurence and interaction with the liquid flow was observed in these experiments. This is the re-entrant jet phenomena, which was shown to be the mechanism causing the shedding of cavitation clouds from attached cavitation on a certain body. Re-entrant jet and its mechanism of cavitation cloud shedding is shown on figure 1.9, in order to give an illustrated view, and on figure 1.10, which shows a cloud shedding captured in an experiment. The illustration actually follows from [42], where it is used to give an explanation of internal flows in cavitation pocket or inside what is observed as sheet cavitation. The numbers shown in the figure describe different phenomena which takes place. Zone 1 includes intensive vaporisation or cavitation appearance. In zone 2, this is followed by cavitation structure growth, which then reaches region of condensation in 3. The collapsing cavity in zone 4 reaches the region of reversed flow caused by adverse pressure gradient. This reverse flow travels upstream and splits the cavity. At the same time, it also contributes to the cavity, as liquid in it can vaporise. Both procedures are shown under zone 5. This might be in equilibrium, meaning that the size of vapour structure can remain constant and sheet cavitation can be observed. But if the re-entrant jet continues to move upstream, it can split the whole cavity and cause cloud shedding. The shed cloud is then convected downstream, shown with zone 6.

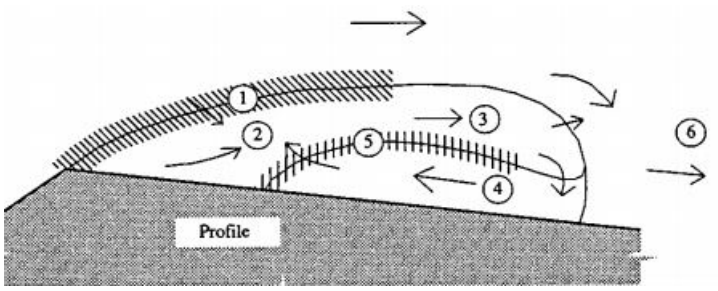


Figure 1.9: Illustration of the re-entrant jet mechanism (taken from [42]).

The reason, why re-entrant jet mechanism was described here and venturi flow experiments were pointed out, is that there have been many experiments done with similar flow patterns observed. Moreover, the experiments with such flows, especially those conducted in the mentioned venturi geometries, showed to be of great use for further development and validation of various cavitation models based on single phase models [53]. Some examples of such validations will be considered later. The mentioned experimental flow configurations continue to be essential in such activities even nowadays as they are considered a benchmark for different proposed cavitation models. As such, they seem only correct to be used also in LES or DNS approaches. Finally, important turbulence cavitation interactions were observed in them, therefore some of the experimental studies are mentioned in the following text.

Figure 1.10: Experimentally observed cloud shedding caused by re-entrant jet (taken from [53]).

The mechanism of cavitation cloud shedding was observed already in 1955 using high speed cine camera [54]. The re-entrant jet forming at the trailing edge of cavity and moving quickly upstream, leading to cloud shedding, was noted. The mechanism existence was then confirmed by various other experiments. For instance, very clearly with injection of dye at the cavity trailing edge, which was then convected upstream and finally remained in the clouds [55, 56]. In [57], velocity measurements showed that the reverse flow velocity of re-entrant jet is of the same order as that of free stream. Interestingly, the experiment also showed that it is possible to control the cloud shedding by placing small obstacles downstream, close to the end point of sheet cavity. This was confirmed also in [58]. This however, hugely affects also the turbulence characteristics of the flow. In the clouds which are shed from the cavity, turbulence was found to be higher. One of older works stating so was performed with cloud cavitation observations on hydrofoil [59]. Laser Doppler Anemometry (LDA) was used to observe velocities of the flow. It was found that velocities concerning cavitation are divided into large and small scale components. Large scale components are connected with unsteady cloud cavitation motion, which is also a large scale structure. The small scale velocities are on the other hand found on the boundaries of cloud cavitation. Although the clouds with large scale velocity components move slower than mean flow velocity, they have higher, concentrated vorticity in their centre. The clouds are also composed of bubbles, showing importance of interactions between large scale vortices and bubbles. An example is a statement there is therefore lower pressure in the cores of large vortices and also cavitation clouds. Otherwise the work deals with a slightly different cloud shedding mechanism, as re-entrant jet was observed but did not play an important role. Instead a wall jet caused by shear layer instabilities upstream was found to be causing cloud shedding.

Findings from this earlier work were much upgraded with the before mentioned work of LEGI laboratory team, which can be found in [42, 43, 44]. In their first work, [42], they studied a flow within and outside of a sheet cavity in the venturi with $4,3° - 4°$ converging-diverging part, while in the other two works, cavitating flow in $18° - 8°$ converging-diverging part venturi was observed.

In all these works, double optical probes were used to measure velocity, α and also chord length of vapour structures (or simply bubble sizes) inside cavities. In [42], LDA was used to observe the flow around cavities, while a video set in combination with a stroboscopic light sheet was used in [43]. In all these measurements, pressure was measured on the wall with pressure transducers. It was found that the pressure inside cavities is close to vaporisation pressure. This well supports the choice to use vaporisation pressure as a threshold at which cavitation starts to appear or disappear, which is regularly used in cavitation models. After cavity closure pressure was found to be higher than vaporisation pressure. Analysis of streamlines obtained with LDA showed that some of the flow passes through the cavity interface upstream. Reverse flow and re-entrant jet was found to be present in all observed types of cavitation, also quasi steady sheet one. Importantly, no reverse flow was found in the case of non-cavitating flows. Globally highest α was found to be at the front of the cavity, while locally, α is highest in the shear layer. This importantly shows the effect of higher velocity fluctuations on lowering the pressure and consequently affecting cavitation. In the vapour clouds, velocities were found to be highly unsteady and bubbly nature of the flow in them was confirmed with presence of raw waveform of the signal from the optical probes. Importantly, the attached cavity on venturi was also found to be composed of liquid and vapour, and not just filled by pure vapour as it was assumed before. As it will be seen in following chapter, the results obtained by these measurements were often used in validation of cavitation models.

Other studies which add to the results obtained by LEGI team were also performed. For instance, in [60], cavitating flow in a different venturi geometry was observed. PIV method was used for measurements in closure region of cavities and downstream. Therefore the data about the flow was obtained only for the liquid phase. Nevertheless, measurements showed that collapse of vapour structures is the primary source of vorticity production. Thin sheet cavities were observed, where no re-entrant jet was noted. The cavities collapsed as vapour started to condense and created hairpin like vortices which had microscopic bubbles inside and dominated downstream flow. It was observed that increase in turbulence, momentum and displacement thickness followed this process. The large eddies created after the collapse of cavities were found not to have a significant additional level of vorticity production. This work contradicts the before mentioned earlier work in [59]. There, it is mentioned that large vortex structures or cloud cavities and therefore increase in vorticity were formed because of the instabilities in the shear layer upstream. But these measurements cleary show that the source of increased vorticity is downstream, at the cavity closure or collapse. In connection with turbulence cavitation interactions it was also found that small change in cavitation number caused considerable changes in turbulence characteristics of the flow, although cavities remain almost the same. Regarding the lack of re-entrant jet, the work refers to another experimental work which can be found in [61]. It this, it is stated that re-entrant jet formation demands strong adverse pressure gradient in the closure region. Moreover, the cavity has to be thick enough that adverse pressure is strong enough and re-entrant jet forms in a way to shed it. These conclusions are confirmed also by [60].

Lately, another and more detailed study using the venturi geometry shown in figure 1.8 was carried and more results than in before mentioned experiments were obtained. This work can be seen in [53], and it was an extension of before performed experiments with two optical probes. The intention was to enable evaluation of instantaneous velocities as well as time average and RMS α values. The local α value measurements show that fluctuations in the lower part of cavity at the wall, where the re-entrant jet is present, are in phase with the upper part of the cavity. The thickness of the cavity on the other hand remains the same during cloud shedding or break off cycle. Meaning there is always some vapour present in the cavitating region, even between

two clouds of vapour. Experiment also provides mean wall pressure data which shows the mean pressure on the wall never drops completely to vaporisation pressure. Despite this, cavitation still appears and grows. It is then argued that vapour is created because of turbulence of the flow, where turbulent eddies cause pressure to drop further and enable cavitation growth. Experiment also shows another effect of cavitation interaction with the flow (not strictly turbulence), as highest α fluctuations are observed in the closure region and not in the region of highest pressure fluctuations, which is upstream. This makes for a strong argument not to use cavitation models which do not include α transport equation.

To make the presentation of turbulence cavitation interactions complete and also add some additional pointers for following description of numerical simulations, some experimental findings where shear induced cavitation was observed should be also mentioned. Lately, such cavitating sheared flows were studied in [62] and [63]. The two works complete each other as they present results obtained with use of PIV-LIF method [62] and X-rays [63] concerning the same sheared flow. The LIF-PIV method was used to obtain mean and fluctuating velocity fields of liquid phase in a two phase flow while X-rays were used to obtain information about mean value of α and its fluctuations. It was found that successive vaporisations and condensations inside turbulent area generated additional velocity fluctuations due to strong density changes. Compared with non cavitating flow, the vorticity thickness was considerably increased. Velocity fluctuations increased with increased cavitation. It was also noted that velocity fluctuations in longitudinal direction were higher, therefore the turbulence cannot be assumed as isotropic. This is also confirmed with the changed shape of vortices (ellipsoidal form) when cavitation developed. However, higher longitudinal fluctuations are typical in sheared flows. Nevertheless, this phenomena is interesting since article [62] argues that increase in longitudinal fluctuations was also observed in other experiments in cavity collapse region. Therefore cavitation collapse could lead to increased longitudinal fluctuations, especially as the X-ray measurements in this experiment revealed that cavitation structure along the flow direction included vaporisation and collapse events at once. Interestingly as well, although velocity fluctuations increased with increased cavitation, turbulent diffusion remained practically unchanged. As it can be seen, this experiment provided a lot of insight into turbulence cavitation interactions and some of the conclusions regarding numerical modelling of phenomena will be introduced in the following section. Because of the possibilities that are now available with the use of X-rays and PIV, other experiments and methods were also developed in the last years. Amongst these, the experiments done by our laboratory can be mentioned. These are based on using high intensity X-rays to observe cavitating flow in the before introduced venturi with $18° - 8°$ converging-diverging part. The X-rays in this case take also the role of illumination for PIV and do not serve only for measurements of α. Moreover, PIV is performed by looking simultaneously at solid particles and also bubbles formed in cavitation region, therefore data about velocities in both phases and α is available at the same time. At the moment, experimental data is still under analysis and only mean velocities in liquid and vaporous phase were obtained. More about the experiment can be found in [64, 65].

As it can be seen from the described experimental work, turbulence cavitation interactions have wide field of effects and studies. Main interactions can be defined as pressure drop caused by turbulent fluctuations, which causes vaporisation in center of turbulent eddies and also in the shear layer. Bubbles which are entrapped or convected with the eddies after cavity collapses or is shed, remain in eddies because of this lower pressure in them. On the other hand, they cause higher vorticity and turbulent fluctuations. Experiments where re-entrant jet causes cavitation cloud shedding are often used to validate simulations. However, capturing turbulence cavitation interactions in them is a very difficult thing.

3.2 Numerical resolution of considered interactions

Great majority of models and especially the models which are most practically used nowadays are based on single phase model group. These models are also the focus of here given presentation of numerical resolution of turbulence cavitation interactions. Moreover, the emphasis is on simulations using RANS approach, for same reasons.

Since cavitating flows are per rule also turbulent, thus difficult to simulate directly, RANS approach is a preferred choice. Some of the earliest cavitation models in the group of single phase models were actually used in simulations with inviscid assumption or Euler approach. Examples are [15, 17]. However, as this approach lacks the influence of viscous dissipation while there exist many RANS turbulence models, simulations with them were studied and gained more and more attention. Regarding the use of RANS approach, it has to be admitted that it causes low frequency separation between the modelled and computed scales. However its use in cavitating flow simulations is argued as appropriate on the basis of it being well known that RANS equations can be applied for flows where the time scale of mean flow unsteadiness is considerably larger than turbulence characteristic time scale. In most unsteady cavitating flows, the frequency of cavitation shedding or oscillations is less than 100 Hz, which makes cavitation characteristic time scale also much lower than time scale of mean flow unsteadiness [25]. However, RANS models are also known to have many shortages and introduce many issues. Most notably, they tend to over predict the eddy or turbulent viscosity μ_t, which affects the development of flow unsteadiness in a reducing manner. This was also noted to be the main issue when the models were applied to cavitating flow simulations with different cavitation models. Not surprisingly, the use of different cavitation models on the other hand revealed that differences between cavitation models in predicting cavitation occurrence affects the obtained flow results in considerable manner as well.

The issue of RANS models over predicting turbulent viscosity in cavitating flow simulations and cavitation models being able or unable to capture different aspects of cavitation occurrence can be very well seen in many articles. The purpose of this section is not to present many of them as a good illustration of the problem can be given on the basis of only three works. These are [66, 25, 26]. In them, flow in before pointed out venturi test sections was considered and it nicely revealed issues raised with the use of different RANS and cavitation models. The presence of before mentioned re-entrant jet and consequent cloud shedding with important turbulence cavitation interactions, not to mention the available experimental results, makes the two test cases very suitable for study of turbulence and cavitation models abilities. This also gives the reasons why they are considered as the benchmark cases for proposed cavitation and turbulence models.

In [66], cavitating flow in venturi with $18° - 8°$ converging-diverging part is considered. Cavitation was modelled with a simple barotropic model from [17], also shown on figure 1.6. Study considers the effect of using different turbulence models on ability to predict cavitation cloud shedding and therefore capture re-entrant jet. Standard and modified $k - \varepsilon$ RNG models were compared with performance of $k - \omega$ model without and with inclusion of compressibility effects. The results can be put into two groups. One concerns standard $k - \varepsilon$ RNG and $k - \omega$ models, while the other includes results with both modified models. The use of standard models shows that they tend to over predict turbulent viscosity in the mixture zones (where liquid and vapour are present), which then causes quasi stable sheet cavitation and no appearance of re-entrant jet. Simulated cavity is more than 50% shorter than the one measured in experiments and α is over predicted. Figure 1.11, taken from [66], presents the obtained results with these two models.

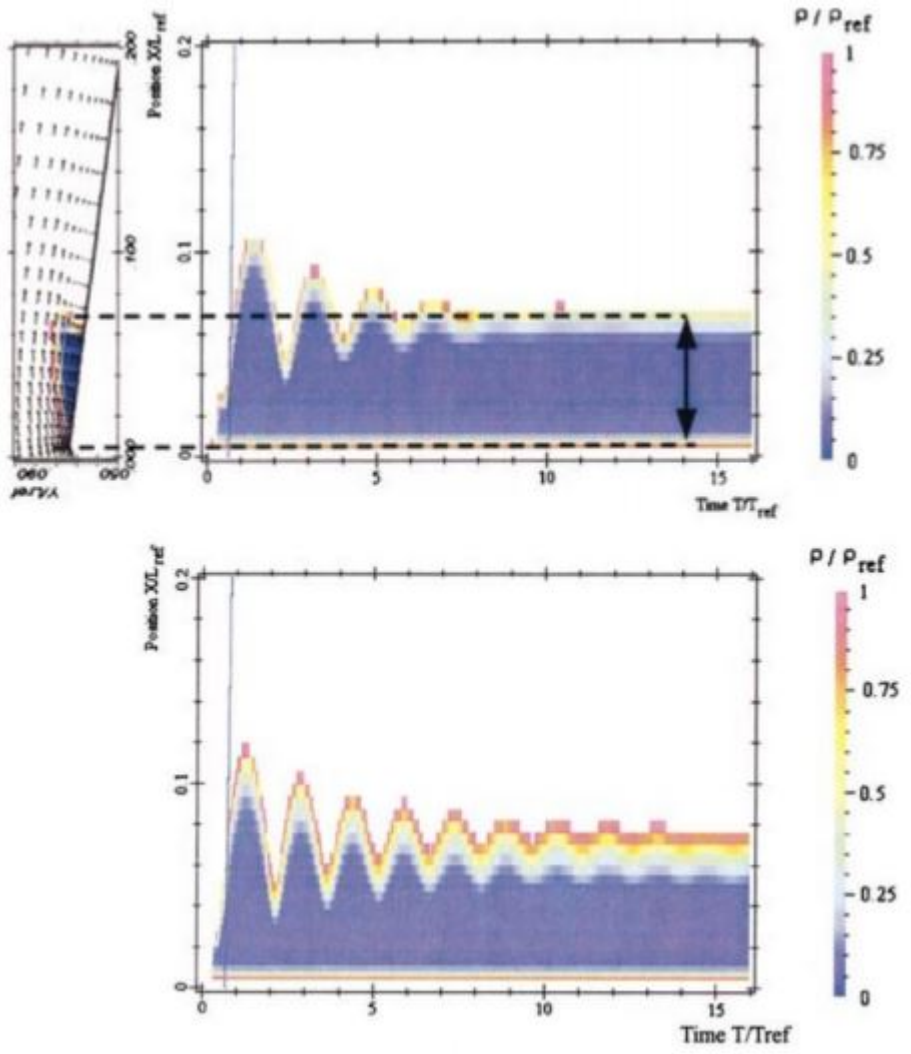


Figure 1.11: Results of cavitating flow obtained with use of standard $k-\varepsilon$ RNG model (above) and $k-\omega$ model (below). What graphs depict is explained through comparison given on the upper graph (taken from [66]).

Interestingly, in [66] it is mentioned that global level of dissipation induced by turbulence models does not seem to point towards inability to simulate the unsteady cavitating flow behaviour. Nevertheless, the problem of too high turbulent viscosity in the mixture regions needs to be addressed and the two used turbulence models achieve this by emphasising the effect density changes have. A limiter, otherwise known as Reboud's limiter, is applied to $k-\varepsilon$ RNG model, making it a modified $k-\varepsilon$ RNG model. For the case of $k-\omega$ model, the mentioned modification to account for compressibility effects is used. Since Reboud's limiter shows the effect caused in both cases in a simple manner and is used also with other turbulence models, it seems appropriate to present it. The turbulent viscosity μ_t in $k-\varepsilon$ RNG model is defined with equation (1.9), where $C_\mu = 0,085$. The limiter proposes to decrease μ_t by changing the mixture density ρ with function $f(\rho)$ and modified turbulent viscosity definition follows in (1.10). Function $f(\rho)$ is given with equation (1.11) where coefficient n has to be much higher than unity. The values of this function for $n = 10$ in dependence of ρ are shown in graph on figure 1.12.

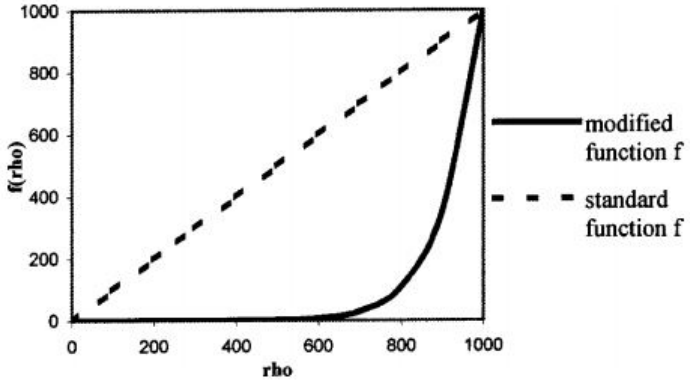


Figure 1.12: Values of function $f(\rho)$ or Reboud's limiter for $n = 10$ (taken from [66]).

$$\mu_t = \rho C_\mu k^2/\varepsilon \tag{1.9}$$

$$\mu_t = f(\rho) C_\mu k^2/\varepsilon \tag{1.10}$$

$$f(\rho) = \rho_v + \left(\frac{\rho_v - \rho}{\rho_v - rho_l}\right)^n (\rho_l - \rho_v) \tag{1.11}$$

It is clearly shown that values of $f(\rho)$ are much smaller than those of ρ in the mixture range. This in return diminishes the turbulent viscosity. Same effect is in the $k-\omega$ model which accounts for compressibility effects obtained in a more complex manner. Le Favre averaging is applied to standard $k-\omega$ model equations, resulting in additional turbulent dissipation term, meaning it increases turbulent dissipation. The term affects the regions of variable density, hence it diminishes the turbulent viscosity there. Both models were found to much improve simulations and produce similar results. These are shown for both models on figure 1.13.

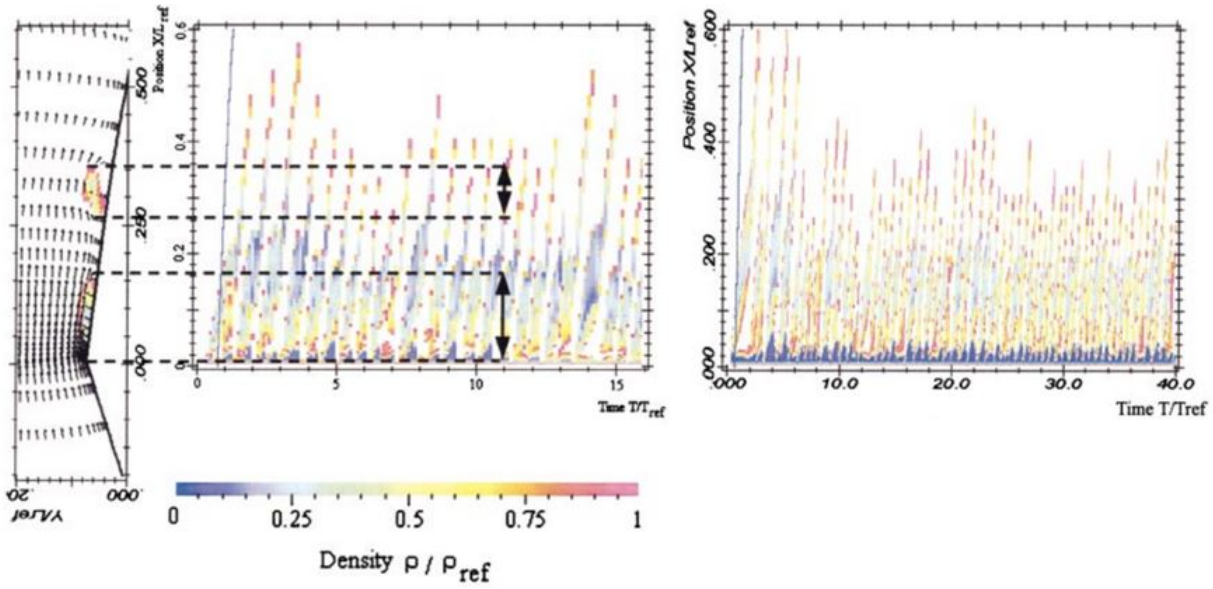


Figure 1.13: Results of cavitating flow obtained with use of modified $k-\varepsilon$ RNG model (left) and $k-\omega$ model (right) (taken from [66]).

It can be seen that the unsteady and quasi periodic nature of cavitating flow is now captured. The frequency of cavitation oscillations is well captured, also with fluctuations around the mean frequency, which are observed in experiment as well. Moreover, cloud shedding and their convection downstream is noted, which is contrary to before mentioned argument against the use of barotropic models which do not feature α transport equation. α is also better predicted, with lower amplitudes than before. Comparison of mean velocity and α profiles and their fluctuations shows good agreement with experimental results and also presence of re-entrant jet. It can therefore be concluded that density changes or with them compressibility effects induced by cavitation demand specific inclusion into turbulence models, otherwise simulations can fail to capture important flow effects.

It was also noted that not only inclusion of compressibility effects in turbulence models leads to better cavitating flow prediction, but also certain features of cavitation models. A clear example of this is given in [25], where cavitating flow in the venturi with $4,3° - 4°$ converging-diverging part was simulated. The flow in this geometry is quasi stable therefore steady and unsteady computations were performed. Only unsteady ones are described here since higher dependence on the choice of models was revealed in them. $k-\omega$ SST model was used in these simulations with two different barotropic models. One is the simple barotropic model, used also in before mentioned work. The other is barotropic model, based on using stiffened gas equations of state. As already mentioned in description of barotropic models, the simple model produced better results. The model using the stiffened gas equations of state namely failed to capture the quasi stable sheet cavitation with re-entrant jet. The simple model on the other hand provided good results for both velocity profiles and α values. The results are interesting to compare with the previously mentioned work, since the $k-\omega$ SST model had no limiter or modification applied. It is therefore obvious that not only turbulence models but also cavitation models play an important role in capturing flow characteristics. Regarding this work, it can be added that although computations at lower cavitation numbers included a large cavitation structure which broke off from the attached sheet, this structure was not convected with the flow. This is different to results in [66] and supports before given argument against the use of models which directly link pressure and vapour volume fraction.

The two barotropic models from before mentioned works were in [26] adapted to include also α transport equation. This recent study focuses on comparing performance of different turbulence and also cavitation models with calculations of quasi steady cavitating flow in venturi with $4,3° - 4°$ converging-diverging part. For comparison of turbulent models, simple barotropic model is used (now with α transport equation), while turbulence is modelled with $k-l$, Spalart-Allmaras and Jones-Lander $k-\varepsilon$ models, all of which had the Reboud's limiter applied. Results show that all models capture the re-entrant jet (not the case without limiter) and produce similar results. Pressure distributions upstream the cavity collapse is similar to experimentally measured one for all models, higher differences can be noted downstream. Although all models predict the peak pressure fluctuations at correct spot, their amplitude there and downstream is over estimated and differs considerably between the models. All models enable creation of vortical cavitation clouds from shear layer. These clouds are then convected downstream, which is the consequence of using the mentioned transport equation. It was concluded that the influence of turbulence models with the applied limiter is weak as similar flow dynamics is obtained with them. α is found to be over predicted in recirculating region and Spalart-Allmaras seems to give best results regarding this. This model was also used to perform an estimation of cavitation model influence, for which it was used with the two mentioned barotropic models. Similar flow dynamics with vapour cloud shedding is observed in this case too. Very similar results can be observed

in the region of attached cavity sheet, while differences become more apparent downstream. It was found that the stiffened gas model produced better mean α values in a certain region, but under predicted the recirculating flow in that same region. It also produced higher values of α compared to the simpler model while at the same time the values on the wall were found to be better. Better estimation of pressure fluctuations follow from its use as well. It can be noted that the same model produced much better results with the use of α transport equation than in the previously considered work, where it failed to capture re-entrant jet. Indeed, the use of Reboud's limiter might have something to do with these better results too.

The three presented works are focused on the flow in shown venturi sections. It should be stated that turbulence and cavitation models revealed different ability to capture flow characteristics, including turbulence cavitation interactions, in other geometries too. Often, the Reboud's limiter was the key to obtaining better results [66]. However, the last presented work gives an impression that qualitatively equal results can be obtained with certain modifications to cavitation and turbulence models. Indeed this is possible and some work showing this can be seen in [18] and [67]. The first considers simulations on a hydrofoil, where cavitation model source terms are tuned to give same maximum values for creation or destruction of vapour. In the second work, an optimization strategy was applied to tune the empirical coefficients in three different cavitation models in order to better asses their performance in same flow regime. Nevertheless, even if the quality of results with different models can be put on similar level, it has to be acknowledged that such modelling can still be significantly improved. For instance, above presented works clearly show different estimation of pressure fluctuations, hence velocity and density fluctuations are also different. Even mean flow characteristics are different, as it is easily noted that mean velocity profiles express differences between various cavitation and turbulence models, not to mention also mean α values. Therefore improvements in cavitation and turbulence modelling are needed.

Indeed, here presented results all deal with RANS approach for modelling turbulence. As it was not so recently stated in [68], the improvement of results with the use of Reboud's limiter does not explain why this modification is essential for simulations of unsteady cavitation. It only shows that turbulence is different in the pure liquid and mixed phases region. The goal should be therefore to better understand what causes the change, and the use of a limiter or other modification hardly gives more information about this. Such modifications are not really directly connected with the events in the flow, especially since they are based on single phase approach, while rich two phase interactions are present in the flow. Furthermore, a case can be made that such modifications are applied on the level which ensures that simulations satisfy globally observed flow behaviour, such as cloud shedding, re-entrant jet presence, cavity length and so on. A case for this is made in experimental investigation mentioned before and presented in [62]. The LIF-PIV measurements of velocity field in the liquid combined with X-ray measurements of α raised enough data to try and predict turbulent viscosity. Three different predictions of μ_t were proposed, on the basis of multiplying Reynolds stress tensor from LIF-PIV measurements with density from X-ray measurements which had Reboud's limiter applied or not. Results show that with lower cavitation numbers discrepancies in predicted μ_t become considerably higher betweeen the cases where Reboud's limiter was applied and not. The results are for illustrative purposes shown on figure 1.14.

It should be admitted that the presented results come from a case of shear induced cavitation, however they make a strong argument that turbulent viscosity is in usual RANS simulations of cavitating flows deliberately made lower in order to obtain better flow behaviour on global scale.

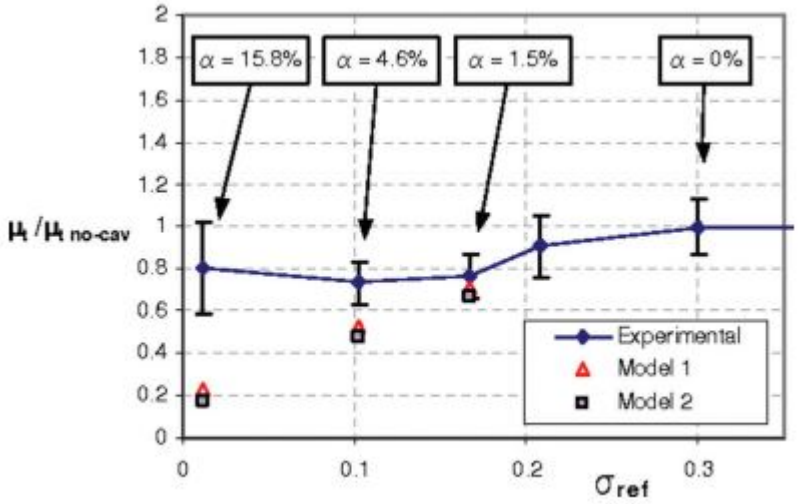


Figure 1.14: Results of μ_t prediction on the basis of experimentally measured velocity and density fields. Model 1 and 2 had Reboud's limiter applied. (taken from [62]).

Furthermore, it could be argued that because of RANS and its inability to well capture turbulence cavitation interactions, even some of the global flow characteristics never really fit completely with the experimentally measured ones. As mentioned before, mean velocity profiles differ when different models are used etc, hence need for improved turbulence modelling is obvious. An answer to this is in the increased use of LES approach, which became more feasible with increased computational power. An early attempt at performing LES simulations of cavitating flows can be seen in [69], where barotropic model was used with a simple Smagorinsky LES model to simulate cavitation on a hydrofoil. Lift, surface pressure and noise were well captured. A more extensive presentation of LES abilities to capture turbulence cavitation interactions can be seen in more recent works. An example is presented in [70], where cavitating flow in the presented venturi with $18° - 8°$ converging-diverging part is considered. Simulations were performed using dynamic Smagorinsky LES model with cavitation model using transport equation for α. Cavitation model is a bit unclear, as it seems to be a mixture of Singhal and barotropic model. Moreover, results cannot be compared with experiments in same geometry as higher cavitation numbers were used in simulations and no periodic cavitation behaviour was noted (contrary to experiment). Additionally, vaporisation pressure was increased in order to ensure cavitation occurrence. Nevertheless, results show much improved ability of using LES in capturing turbulence cavitation interactions. Irregular shedding of small scale vapour structures with creation of hairpin like vortices is observed near cavity closure region, which is in accordance with experiments in similar cavitating flow behaviour (without cloud shedding), notably with [60]. Collapse of cavitation structures was found to clearly produce vorticity and increase turbulence downstream. The presence and importance of baroclinic torque was noted. Upstream, vapour formation at the throat was revealed to suppress velocity fluctuations. The reason is that vortices with vapour are formed in the shear region at the throat and show tendency to oppose vortex stretching. Namely, when vortices strech, pressure in their centre is supposed to drop, but in this case cavitation growth would result from it. Some similar and additional effects of cavitation on turbulence are also noted in another recent study presented in [71]. There, the Schnerr-Sauer model was used with WALE LES model to simulate cavitation on a twisted hydrofoil. Cavitation importantly differs from the case in [70] not just because of different domain geometry but also because unsteady, periodic cloud shedding

was simulated. Results show good agreement of cavitation behaviour with the one observed in experiments. Moreover, interactions between cavitation and vortices are discussed on the basis of analysing vorticity transport equation related to vortex stretching, expansion or contraction and baroclinic torque. It was again shown that cavitation promotes vortex production and flow unsteadiness. Since periodic cloud shedding is present, it was found that cavitation appearance accelerates vortex stretching. This is not objecting to results in [70] since it does not refer to same region of cavitation or cavitation behaviour. Cavitation also increases vortex dilatation and leads to an increase in the baroclinic torque. Last one became important source of vorticity generation. Vortex dilatation and baroclinic torque in cavitating flow cases also increase to same magnitude as vortex stretching, while they were not present for non cavitating flow.

It should be acknowledged that all conclusions regarding turbulence cavitation interactions obtained with the presented LES simulations were not confirmed with equally extensive comparison with experimental results as in the case of described RANS simulations in the two venturi geometries. Nevertheless, it can be seen that LES offers many improvements, although there are still many terms which are modelled. Importantly, use of le Favre averaging because of density variations raises additional questions about turbulence modelling in case of cavitation. Moreover, argument is raised in [72] that LES simulations still do not account for cavitation which occurs in fine elementary vortices as these are too small and therefore captured in sub grid scale. Thus it is only logical to consider performing also DNS simulations, in which abilities and impact of cavitation models or the approach on which they are based (homogeneous mixture, single phase) can be better estimated.

3.3 On present Direct Numerical Simulations of cavitation

DNS simulations are becoming more and more interesting also for simulations of multiphase flows. However, not many attempts at simulating hydrodynamic cavitation with single phase modelling can be noted. Even less attempts are done for simulations with interface tracking methods.

Probably the most comparable studies to before presented RANS and LES simulations using single phase approach are the studies published in [72] and [37]. In [72], shear induced cavitation is simulated and vortical structures between an incompressible and cavitating flow regime are compared. Cavitation is modelled with the use of modified form of before mentioned first proposed empirical model, presented in [16]. The calculated flow with obtained vortical structures is consistent with those in previous experimental and theoretical studies. Cavitation appears to be clearly captured in the vortex cores, where low pressure regions are present. Its effect on the vortex and in general turbulence appearance is consistent with the findings of before mentioned LES study of cavitating flow in a venturi [70]. Upstream, where cavitation appears and develops, it was found that it decreases turbulent intensity compared to non cavitating flow case. Two arguments for this are given. Firstly, this is a consequence of pressure in vortices being limited or kept at vaporisation pressure. Therefore pressure fluctuations are damped and turbulence intensity is lowered. The second argument is based on cavitation modulation of vortices. This follows from the first argument and is similar to the one given before in LES simulations saying that vortex stretching is damped as higher pressure fluctuations or pressure drops would cause additional vaporisation. Moreover, if additional vaporisation in a vortex does occur, it leads to vortex weakening. On the other hand, turbulence intensity was observed to increase downstream, where cavities collapse. This is also in agreement with before presented results.

In [37] a submerged jet cavitation is simulated. It uses equal cavitation model and fluid properties as in [15], where one of the first models using Rayleigh-Plesset equation is presented.

This raises some questions, since the phase transfer is directly connected with pressure change and therefore important convective effects are not included. Nevertheless, velocity, vorticity and pressure fields are compared for cases of non cavitating and cavitating flows (two cavitating and one non cavitating case observed). Again, strong interaction between cavitation and turbulence is observed with cavitation forming in cores of primary vortex rings and vicinity of streamwise vortex tubes. However, higher and faster minimum pressure fluctuations were observed in cavitation here, opposing to the results from previously described simulations. It is argued that this is a consequence of alternating growth and collapse of bubbles inside cavitating regions in very short time periods. Periodic shedding of vortex rings also helped the jet to alternate between cavitating and non cavitating regime, causing such fluctuations. Cavitation was also found to accelerate the jet, but suppress its growth and velocity fluctuations where vapour is present. This agrees with the before mentioned effects of cavitation on vortex structures. Furthermore, vorticity transport equation was applied and led to similar conclusions as LES simulations in [70, 71]. Decrease in magnitude of vortex stretching is observed, which is similar to [70], while presence of non-zero dilatation and baroclinic torque in the main cavitation regions is noted. The latter is worth noting, as pressure and phase transfer are directly connected in these simulations, yet baroclinic torque was not zero. Regarding vorticity it was also noted that transverse vorticity in the cavitating regions is increased due to vapour formation, while streamwise vorticity is weakened. Further downstream, cavitation causes increase in velocity fluctuations.

The two presented DNS simulations and their results, though interesting, can hardly be applied to other hydrodynamic cavitation cases presented here, since they stress cavitation, strongly induced by sheared flow. Other published DNS simulations also, at least to our knowledge, do not describe typical hydrodynamic cavitation cases, such as flow in a venturi, over a hydrofoil etc. Maybe an example of closest simulations would be the work published in [73], where Euler-Lagrange approach was developed to enable DNS or LES simulations in which bubbles injected into turbulent boundary layer are tracked, while their behaviour can be modelled and interaction with the flow observed. The interaction is however done with one way coupling applied, meaning that only effect of liquid onto the bubbles is included in equations. A similar work, where one way coupling is used, is shown in [74]. There, single bubble behaviour in a vortex is observed. DNS with interface tracking is also used to simulate single bubble development and results are compared with the one way coupling approach. DNS with interface tracking of larger amount of bubbles is also used in [12], which is a work already mentioned in section 2.1. However, Euler equations (inviscid approach) are used to describe the two phases and the axisymmetric assumption is used, making the problem essentialy 2D. This was still applicable as intention was not to observe global fluid movement in a certain direction but pressure and shock waves propagation and effects in bubbly flow. Collapse of bubbles was also simulated in order to see how bubble presence can help decrease cavitation erosion. Interestingly, the interface tracking method was in [13] (also mentioned before) compared to single phase modelling. Behaviour and effects of bubbles in a jet exposed to high energy input (supplied by proton pulses) was observed. The energy input was simulated by a sudden increase in internal energy. As written in section 2.1, the actual movement of the jet is not in question but rather its change in shape because of cavitation occurrence. It was found that the two approaches give very similar results. Similar evolution of the two phase domain, change of the jet shape and velocity of its surface was noted. The interface tracking does however account for more effects. Again, such simulations can hardly be compared to usual hydrodynamic cavitation simulations. To conclude with the presentation of performed DNS simulations, the use of single phase modelling to simulate cavitation around a circular cylinder can be seen in [75]. The purpose of performing such simulations was to predict cavitation noise. Therefore the

two phases were assumed compressible, with their densities changing only in regard to pressure. This enabled observation of pressure waves and noise prediction. Barotropic model was used together with transport equation for α, similarly to LES simulations in [70]. The reason why these simulations were not mentioned before is in the laminar flow conditions which were used in them and 2D computational domain. The simulations are as such not performed to study turbulence cavitation interactions, but strictly the noise prediction. There is also a connected work to this in [76], which had same simulations performed at higher Re number and in a 3D domain. However, the focus is still on noise prediction and no turbulence characteristics are reported or studied.

Although there are not many attempts to simulate usual hydrodynamic cavitating flows, the reported DNS simulations show that DNS with single phase modelling presents an interesting and potentially fruitful tool for development and study of cavitation simulations. To conclude with this section, it should be however also mentioned that there exists a doubt about appropriateness of referring to such simulations as DNS simulations. Questions about this are often raised and are well justified, as DNS, by the name, would require resolving of all scales and phenomena in the flow. Contrary to this, use of single phase modelling to describe phenomena which includes sharp interfaces between the phases and multiple flow structures means there are many simplifications of the flow present. Only interface tracking methods could therefore provide real DNS simulations of cavitating flows. But as it was shown, these are highly impractical, hence referring to simulations with single phase modelling and no turbulence models as DNS simulations is used and seen in literature. There are also other reasons why this can be acceptable. For instance, results of such simulations show single phase modelling can still lead to good accuracy compared to interface tracking [12]. Moreover, all turbulent scales, even the smallest vortices, are captured in them. This is one of the limits which LES approach cannot overcome, despite offering much more detailed results than RANS approach. It should be however also acknowledged that the smallest scales are questionably well simulated with single phase modelling as many effects between two phases (surface tension and drag effects, slip velocity etc) are neglected. But it is actually important to define what the actual drawbacks and deviations from the reality of exactly these simplifications are. With simulations, referred to as DNS simulations, this is possible without intrusion of errors or simplifications coming from other models, such as RANS and LES. Finally it could be said that describing such simulations as cavitating flow simulations applying DNS approach (on behalf of not using turbulence models) would be more accurate. But for the sake of simplicity and correspondence with available literature on the subject, the name of DNS simulations of cavitating flows is retained in this work.

Chapter 2

Goals of this work

As it can be seen from the focal point of theoretical background chapter, the main interest of this work is to study and improve understanding of hydrodynamic cavitation with the turbulence cavitation interactions present in it through the use of cavitation models based on single phase models group. This thesis is actually a part of a bigger project, where new experimental techniques are proposed in order to gain better insight into hydrodynamic cavitation and obtain better description of the flow. These experiments are based on the use of X-rays which supply a way to measure vapour volume fraction and also an illumination for PIV analysis. PIV analysis can be done in both phases, which raises further information about velocities in liquid and vapour. Experiments were briefly mentioned in section 3.1 of previous chapter, where the reference for them is also given. The experiments are not thoroughly described as this is not the objective of this thesis. However, their settings are. The experiments were performed in a venturi test section with mentioned geometry of $18° - 8°$ converging-diverging part. The height and the width of the inlet channel was 5 mm. The experiments are therefore performed on a millimetric scale, which is commanded by the size of the X-ray beam. The test section is shown on figure 2.1. The observed flow, intended for simulations, has mean velocity at the inlet of $\bar{u} = 6,7\ m/s$ and the Reynolds number $Re = 16700$ (based on channel half height). The cavitation number is $\sigma = 0,34$ (pressure $p = 19,5\ kPa$ measured at venturi inlet).

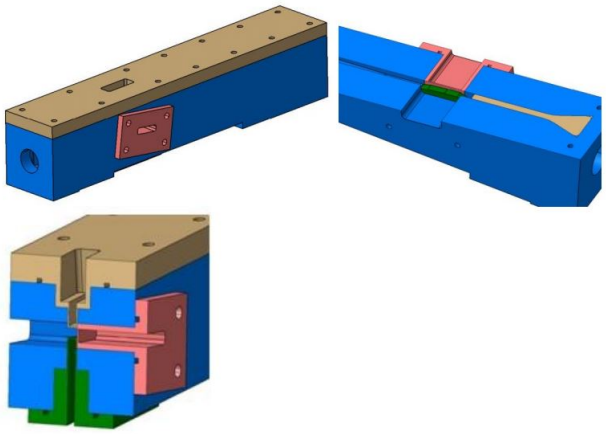


Figure 2.1: The used test section(Taken from [64]).

As experiments enable obtaining a lot of previously unavailable data, a numerical tool was also needed to be a counter part of the experimental work and, together with experimental

data or alone, raise data for analysis and further development of cavitation models. The need for such a tool comes from different observations. The presently available experimental data is heavily based on statistical average data for velocity and vapour volume fraction. Most often used experiments in venturi geometry [42, 43, 44] are an excellent example of this. Such data is appropriate for validation of RANS simulations, although serious limitations are imposed in this case already. Articles describing these experiments reveal that assumptions had to be made in order to gain estimations about certain effects, like mass and momentum balance inside cavities, also interactions between the phases. The used measuring techniques, relying on optical probes, simply did not supply enough data about the flow globally. It thus follows there is a serious lack of important information to assess the abilities of LES and DNS approach. However, with the experiments performed in this project and also other newer experiments, where PIV or LIF-PIV methods and also X-rays were used [60, 62, 63], more information about the flow fields is obtained. Therefore LES and DNS simulations of cavitating flows started to get suitable database with which they can be validated. But where there seem to be many simulations nowadays using LES approach and single phase modelling, simulations using DNS approach are still scarce. Especially when it comes to simulating hydrodynamic cavitation. Indeed, the first two works mentioned in section 3.3 of chapter 1 seem to be the closest examples to the often performed cases of hydrodynamic cavitation. But they focus on shear induced cavitation and their results are hardly applicable to usually studied hydrodynamic cavitation cases. Moreover, computationally they seem to not be so expensive, as they were done on $384 \times 192 \times 120$ [72] and $120 \times 100 \times 100$ [37] points in x, y and z directions respectively. To give a comparison, the reported LES in [70] used $412 \times 64 \times 32$ points and ensured cavitation with higher vaporisation pressure and therefore lower Re number, while the LES on a twisted hydrofoil [71] used approximately 3×10^6 points. If the computational expenses are roughly studied by estimating that DNS simulations need three times more refined grid per direction than those with LES models (based on comparison of suggested Δx^+ and Δz^+ values in [77]), it quickly follows that a DNS simulation of a hydrodynamic cavitation in a venturi or on a hydrofoil would require far greater computational expenses. Moreover, there exist different groups of cavitation models using single phase modelling. Each has some better or worse features than the other, which was also shown in section 3.2 of previous chapter with reported cases where different cavitation models were compared. As is well known and also shown in that same section, the coupling of cavitation models with RANS turbulence models leads to the mentioned issues with modelling turbulent or eddy viscosity. Furthermore, even use of LES models cannot alleviate all issues as for instance cavitation still cannot be captured in sub grid scale vortices, although cavitation interaction with vortices is one of the fundamental aspects of turbulence cavitation interactions. Hence it is hard to estimate which cavitation models or group of models give best description of cavitating flows. Or which characteristics should be included in modelling of cavitation. Consequently, it is also difficult to propose improvements for either RANS or LES models. But a good cavitation model, validated with DNS, can be expected to raise also improvements of turbulence models (with yielding a database from which models can be proposed etc). This brings us to the point that as many ways for single phase modelling exist and DNS simulations are computationally very expensive, the task of improving either cavitation or turbulence models can be made easier with a numerical tool, suitable for fast DNS simulations. As the methods which are used for these are also often applied in LES simulations, the desired tool is also capable of fast LES simulations.

The development of such a tool is also the goal of this thesis. The decision for it is importantly based on the existence of a code in our laboratory, suitable for fast DNS and LES simulations of incompressible flow, called MFLOPS-3D. For simplicity, the aspect of LES simulations will be

rather not stressed. The code was used for various simulations of turbulent flow with adverse pressure gradient. The geometry of the domain used in these simulations is similar to the pointed out venturi geometries as it was simply a channel with a bump on the lower wall, representing the upper, low pressure side of an air foil. It also has same height ratio between the inlet and the throat as the $18° - 8°$ converging-diverging part venturi. This geometry can be seen on figure 2.2. Turbulent flows at friction Reynolds number around $Re_\tau = 650$ were simulated successfully. It was therefore decided that the code will be adapted also to the simulations of cavitating flows, meaning that an algorithm for cavitating flow simulations had to be introduced into it.

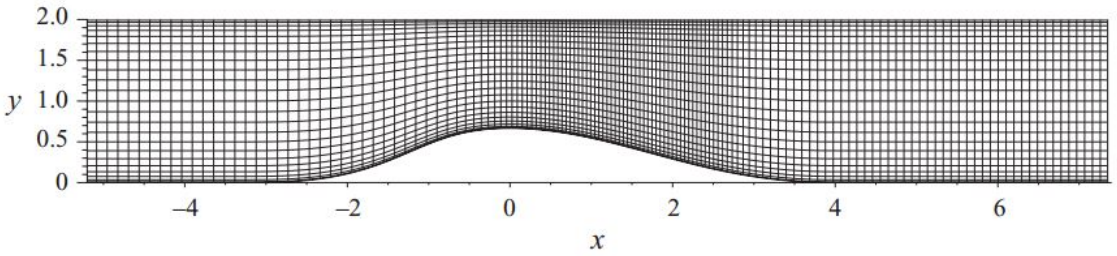


Figure 2.2: Geometry used in DNS simulations of adverse pressure gradient with MFLOPS-3D (taken from [78]).

The purpose of this thesis is to present how the adaptation was done and what it demanded. Indeed introduction of an algorithm for cavitating flow simulations to a code should be a simple task nowadays because of vast experiences gained in doing this over the last two decades. However, when it comes to a code for fast DNS simulations such as MFLOPS-3D, additional limitations in solving the system of equations are raised. The techniques used in this code to enable faster simulations impose the need to have a constant matrix with which the vector of unknowns is multiplied. In other words, the left hand side of a system of equations being solved has to be constant. This is not a particular characteristic of MFLOPS-3D only but is present in some other codes for fast DNS simulations which apply similar numerical methods as well. It also leads to many complications in proposing a stable algorithm for simulations of cavitating flows. Therefore the goals of the work in the thesis had to be adapted and focus was switched from aiming at performing DNS cavitating flow simulations to developing a tool suitable for them (and also for LES). The algorithm was successfully developed and it introduces some novel approaches. Results of its development is also, to our knowledge, first development and use of a verification procedure based on the Method of Manufactured Solutions for codes which deal with cavitating flow problems and utilize single phase models. Moreover, along the way of algorithm's development, other issues with MFLOPS-3D code were revealed as well. These are a consequence of various factors. The most important is the adaptation of the original MFLOPS-3D code to discretization with finite differences in all three directions, in order to overcome the limitations imposed by the use of spectral methods. This caused some issues which are still not satisfactory resolved and sadly, make the performance of DNS simulations at the moment impractical. All of these problems have to be consequently also put to light. The thesis structure is therefore as follows:

- MFLOPS-3D code for incompressible flow simulations is presented in a thorough manner. The presentation is focused on the adapted version of the code in order to overcome spectral methods limitations. The reason is in the fact that this code was the starting point of developing a new algorithm. Original code is also described, but in much less detail.

- Then, new algorithm for cavitating flows is presented. The governing equations and chosen assumptions are discussed at first. An explanation why the algorithm used in the original code can represent a good basis for the new one is given, together with the known issues which threaten its successful adaptation to cavitating flow simulations. This is then followed by general presentation of how certain steps are done in new algorithm, where explanation of additional issues, encountered because of specific characteristics of MFLOPS-3D code, is given. These are fundamentally equal to those in some other codes which use similar numerical methods. Solutions for them are depicted and finally the presentation of all versions of the developed algorithm follows.
- Verification procedure based on the use of Methods of Manufactured Solutions is explained and the test case for the new algorithm is given.
- Verification and performance results are given and discussed for the old, incompressible flow code and the new one with new algorithm.
- Setting up real flow simulations is discussed and issues encountered at performing them are explained. The remedies, applied or suggested, are discussed. Some issues are noted already in the performed verification cases and are therefore pointed out before.
- Conclusions from the performed work are given and needed future activities are defined.

Chapter 3

Used numerical methods, theoretical background and MFLOPS-3D description

This chapter is devoted to the introduction of numerical methods forming the basis of MFLOPS-3D code. These methods also give a frame for the proposed DNS simulations. Description of the methods is accompanied with reasons why a certain method is used. In such a manner, an overview of the MFLOPS-3D code, used for later development of new algorithm and code for cavitating flow simulations, is also given.

1 Incompressible flow governing equations

The governing equations for incompressible flow could be considered as one of the basic fluid dynamics equations since they describe one of the most fundamental flows, incompressible flow. These equations are represented by Navier-Stokes continuity and momentum equations, which are given in their original form in equations (3.1) and (3.2) respectively [79].

$$\rho\left(\frac{\partial\vec{v}}{\partial t}+(\vec{v}\cdot\nabla)\,\vec{v}\right)=-\nabla p+\nabla\cdot\left(\mu(\nabla\vec{v})+\mu(\nabla\vec{v})^{T}\right)-\frac{2}{3}\nabla\left(\mu(\nabla\cdot\vec{v})\right) \tag{3.1}$$

$$\frac{\partial\rho}{\partial t}+\nabla\cdot(\rho\vec{v})=0 \tag{3.2}$$

Since the incompressible flow features constant density ρ, equations (3.1) and (3.2) can be simplified to equations (3.3) and (3.4). Here, one of the most important aspects of incompressible flows is that constant density, as shown in equation (3.4), gives us zero velocity divergence. This fact is used to simplify the viscous terms from equation (3.1) to only one viscous term present in equation (3.3).

$$\rho\left(\frac{\partial\vec{v}}{\partial t}+(\vec{v}\cdot\nabla)\,\vec{v}\right)=-\nabla p+\mu\Delta\vec{v} \tag{3.3}$$

$$\nabla\cdot\vec{v}=0 \tag{3.4}$$

Non-dimensional form of equations is often used in incompressible flows since the effects in the flow depend only on the factors included in Reynolds number. The MFLOPS-3D code uses a similar approach as normalisation of the governing equations is done by division with density. This only affects the momentum equations, hence the momentum equations solved in MFLOPS-3D

code can be written as equation (3.5). In it, Re stands for the ratio between density and viscosity. Variable Re otherwise stands for Reynolds number, which is here for channel flow defined using channel half height. In MFLOPS-3D, Re can be used in such manner since the incompressible flow simulations are done imposing a unit mean flow velocity in the channel and unit half height of the channel. This is a usual practice in LES or DNS simulations of incompressible channel flows. It therefore follows that Reynolds numbers of channel flows are directly set by ratio between density and viscosity. Pressure gradient term otherwise seems not affected, although this is not so. The reasons why and how it is actually treated will be seen in the following section.

$$\frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla)\, \vec{v} = -\nabla p + \frac{1}{Re} \Delta \vec{v} \tag{3.5}$$

2 Fractional step approach and projection methods

One can use different methods to develop an algorithm to solve the presented governing equations of the flow. The method used in the MFLOPS-3D code comes from the class of fractional step methods, which are often referred to in literature also as projection methods [80]. Since this naming is more general [80, 81], it is also used in this work. The reason why MFLOPS-3D code uses a projection method is that in most cases, the governing equations as presented before and which deal with primitive variables, are solved with such methods or more precisely with methods based on fractional step approach [82]. They enable solving Navier-Stokes equations of incompressible flow in a simple sequence of decoupled elliptic equations for velocity and pressure at each time step. This makes such methods very efficient for large scale numerical simulations [80]. Finally, as it is mentioned in [81], projection methods, which fall in the group of fractional step methods, are also better suited to simulations of unstable flows because they demand only one solution of the proposed equations forming the algorithm, instead of multiple iterations over all of them. Consequently it is no surprise they are commonly encountered in DNS and LES simulations [77].

In this part, the logic behind projection methods is described, as a good knowledge of the subject is fundamental to the understanding of demands one faces when performing incompressible flow simulations. Moreover, as it will be later shown, projection methods were in the past also successfully used to develop algorithms for cavitating flow simulations, therefore their understanding is crucial from this aspect as well. Description given here relies hugely on the articles [82, 80, 83], which are also advised literature if the reader wants to find more about the subject.

2.1 Projection methods definition

Projection methods were introduced by Chorin and Temam [84, 85]. Rigorous mathematical definition says they are based on the observation that first terms on both sides of equations (3.3) or (3.5) form Hodge decomposition [82]. A projection as written in (3.6) can be obtained, where $\mathbf{P}$ is the operator which projects a vector field onto the space of divergence-free vector fields with appropriate boundary conditions [82].

$$\frac{\partial \vec{v}}{\partial t} = \mathbf{P}\left(-(\vec{v} \cdot \nabla)\vec{v} + \frac{1}{Re}\Delta \vec{v}\right) \tag{3.6}$$

What this means in simpler words is that usually, the method introduces first an approximation to the momentum equations (3.5) in order to determine the velocity $\vec{v}^*$ or a provisional, predictor

velocity. This velocity is not divergence free. The equation for it can be generally written as in (3.7).

$$\frac{\vec{v}^* - \vec{v}^n}{\Delta t} + \nabla q = -\left((\vec{v}\cdot\nabla)\vec{v}\right)^{n,n-1} + \frac{1}{Re}\Delta\vec{v}^* \tag{3.7}$$

$$\mathbf{B}(\vec{v}^*) = 0 \tag{3.8}$$

First order backward difference formula is used for time derivative in (3.7), where the only other implicitly treated term beside $\vec{v}^*$ in the time derivative is the viscous term. This choice follows from the fact that in MFLOPS-3D only these two terms can be treated implicitly. Moreover, implicit treatment of viscous terms and more explicit treatment of other convection terms is a common choice in DNS simulations of wall bounded flows [86]. This is based on ensuring satisfactory representation of present frequencies in the flow. Therefore all following momentum equations for $\vec{v}^*$ utilise same approach. Variable q in (3.7) represents some approximation of the pressure p which is a matter of choice. On the other hand, non linear term has to be treated explicitly in a certain manner, therefore $(n, n-1)$ superscript is used, n depicting previous, last resolved time level. This means that we can treat the equation as effectively linear. The equation (3.8) represents the need to have defined boundary conditions for $\vec{v}^*$. Once a solution for $\vec{v}^*$ is known, one can proceed with discounting (3.7) from original momentum equation (which uses same temporal discretization). This leads us to an elliptic equation, referred to as the projection equation and written generally as (3.9).

$$\vec{v}^* = \vec{v}^{n+1} + \Delta t \nabla \Phi^{n+1} \tag{3.9}$$

This equation is solved for Φ using the divergence free condition for real velocity $\vec{v}^{n+1}$ from equation (3.4), which leads us to a Poisson equation. If pressure p is directly obtained from this equation, that is from Φ, one can refer to the projection method also as a pressure-Poisson method [82]. But in general, Φ represents an intermediate and pressure related variable, through which, by using equation (3.9), we project or transfer obtained velocity $\vec{v}^*$ to the divergence-free velocity $\vec{v}^{n+1}$. This last procedure is actually given with (3.6) and represents the core of the projection method.

The last step in the method is the calculation of pressure. This can also be omitted in some methods, as will be seen later. However, the pressure can be determined from (3.9) and (3.7), based on the way q and Φ are defined. Generally, the pressure is defined as in (3.10).

$$p^{n+1} = q + \mathbf{L}(\Phi^{n+1}) \tag{3.10}$$

$\mathbf{L}$ represents the connection between Φ and p. An important aspect to this is also to realize that Φ, with respecting the divergence-free condition of real velocity, gives us a pressure which respects this condition. As it is shown in [83, 82, 80], the choice of q, boundary conditions for $\vec{v}^*$ and $\mathbf{L}$, is of big importance when accuracy of the method is considered. This will attempted to be shown here through the presentation of most interesting versions of projection methods for us.

2.2 Some versions of projection methods

There are three basic groups of projection methods. These are the pressure-correction methods, the velocity-correction methods and the consistent splitting methods [80]. As the MFLOPS-3D code uses those from the first group, only some methods from it will be mentioned here. The description of these follows the explanation in [80], although the topic is also well covered in [82]. The main difference between the two is that in [82], Crank-Nicolson method is used for temporal discretization, while in [80] viscous terms are treated only implicitly, as given in (3.7). As this approach is used in MFLOPS-3D code, work in [80] is used in the explanation here too.

2.2.1 The non-incremental pressure-correction scheme

This is the simplest pressure correction scheme, originally proposed by Chorin and Temam [80]. If first order backward difference formula is used again for time derivative discretization, the method results in (3.11), which is just slightly different from (3.7):

$$\frac{\vec{v}^* - \vec{v}^n}{\Delta t} - \frac{1}{Re}\Delta\vec{v}^* = f(t^{n,n-1}) = -\left((\vec{v}\cdot\nabla)\vec{v}\right)^{n,n-1} \tag{3.11}$$

The term $f(t^{n,n-1})$ always includes all remaining terms which are treated explicitly. Here, this is only the non-linear term. Zero value is chosen for q, giving the only difference to 3.7 and resulting in no pressure term used in solution of $\vec{v}^*$. This is also the origin of the name (non-incremental) for this group of projection methods. Based on this and neglection of the difference between viscous terms in (3.11) and starting Navier-Stokes momentum equations (3.5), the following form of projection equation is obtained:

$$\frac{\vec{v}^{n+1} - \vec{v}^*}{\Delta t} = -\nabla p^{n+1} \tag{3.12}$$

This and all other following methods propose the following boundary conditions for $\vec{v}^*$:

$$\vec{v}^* \cdot \vec{n} = \vec{v}^{n+1} \cdot \vec{n} \tag{3.13}$$

It it, $\vec{n}$ represents the normal vector on the boundary. Because of this, equation (3.12) leads to zero pressure gradient in normal direction on the boundaries. This induces a numerical boundary layer which prevents the scheme to be fully first order accurate in L^2 norm for pressure [80].

2.2.2 The standard incremental pressure-correction scheme

These schemes are based on using an explicit value of pressure for q in (3.7). Although such approach can increase the accuracy of the scheme, it can as well cause the accumulation of pressure errors in time, which is especially present in parallel computations [82, 87]. The scheme introduces following equation for $\vec{v}^*$ if 2^{nd} order backward difference formula is used for the time derivative:

$$\frac{(3\vec{v}^* - 4\vec{v}^n + \vec{v}^{n-1})}{2\Delta t} - \frac{1}{Re}\Delta\vec{v}^* = f(t^{n,n-1}) = -\nabla p^n - \left((\vec{v}\cdot\nabla)\vec{v}\right)^{n,n-1} \tag{3.14}$$

Here, value of pressure from previous time step p^n is used as q. After discounting (3.14) from (3.5) and neglecting viscous terms difference, we obtain the following projection equation:

$$\frac{3(\vec{v}^{n+1} - \vec{v}^*)}{2\Delta t} = -\nabla(p^{n+1} - p^n) \tag{3.15}$$

Neglecting the difference between viscous terms is more plausible here because both velocities are closer to one another than in the case of non-incremental method. This is the consequence of the choice for q [80, 82]. The boundary condition for $\vec{v}^*$ is same as before and with (3.15) leads to the constant and zero value of the pressure change gradient normal to the boundaries. Which means that the pressure gradient normal to the boundary does not change in time. Hence a numerical boundary layer is introduced again, making the scheme 2^{nd} order accurate in L^2 norm only for velocity but not fully second order accurate also for pressure [80].

It is interesting to note that this method is essentially the same as the one mentioned under Bell, Collela and Glaz in [82], the difference being only in the time discretization and choice of q.

2.2.3 The rotational incremental pressure-correction schemes

This schemes are an answer to the problem which the boundary condition for $\vec{v}^*$, equation (3.13), introduces to the definition of pressure in schemes presented before. This is the artificial von Neumann boundary condition, which makes pressure gradients have constant values in normal directions on the boundaries. The scheme retains same equation for projection velocity as before, equation (3.14). The important difference is that it introduces variable Φ instead of directly using pressure in the projection equation. This variable was already mentioned in the introduction to this part, but not used until here. The projection equation is written with it as:

$$\frac{3(\vec{v}^{n+1} - \vec{v}^*)}{2\Delta t} = -\nabla \Phi^{n+1} \tag{3.16}$$

Here, Φ follows from the difference between (3.5) and (3.14), in which one does not neglect viscous terms and applies the observation that $\nabla \times \nabla \times \vec{v}^* = \nabla \times \nabla \times \vec{v}$ [80]. Φ therefore follows as:

$$\Phi^{n+1} = p^{n+1} - p^n + \frac{1}{Re} \nabla \cdot \vec{v}^* \tag{3.17}$$

This is actually the first time that the pressure equation is written and solved with all the terms mentioned in (3.10). It represents also a consistent pressure boundary condition, improving the accuracy of pressure solution. According to [82], the method enables velocity and pressure to be second order accurate, while in [80], it is argued that for pressure the convergence rate of only 3/2 is achievable generally. The information in [80] is more precise, since it refers to general domain geometries, while [82] gives such conclusions only on the basis of calculations in periodic channel. An explanation where the decrease in accuracy comes from in general geometries will be given in the chapter considering verification of original and new code.

2.2.4 The Kim and Moin scheme

This scheme is important since it is the first one that introduced a consistent boundary condition for projection velocity $\vec{v}^*$ [83, 82]. Without it, the solution can have considerable numerical errors [83]. The reason for this is that one has to define values of $\vec{v}^*$ on a boundary in all directions. And while the equality of $\vec{v}$ and $\vec{v}^*$ in a normal direction to the boundary is a useful and in a way even required choice (since if satisfies essential mass conservation on continuous level, which will be focused on later) the same is not needed and actually better not applied for other two directions. However, this is used in the schemes considered before. As a solution, Kim and Moin used the projection equation, in which they approximated unknown function Φ^{n+1} with the value from

previous time step. The equation for $\vec{v}^*$ boundary condition obtained from this is (if a projection equation like (3.16) is used):

$$\vec{v}^* = \vec{v}^{n+1} + \frac{2\Delta t}{3}\nabla\Phi^n \tag{3.18}$$

While $\vec{v}^{n+1} = \vec{v}^*$ is applied in normal direction on the boundary, the other two components are prescribed with (3.18). Therefore (3.18) can be written as follows [82]:

$$\tau \cdot \vec{v}^* = \tau \cdot \left(\vec{v}^{n+1} + \frac{2\Delta t}{3}\nabla\Phi^n\right) \tag{3.19}$$

The τ stands for the tangential direction vector on the boundaries. Using this boundary condition enables second order accuracy for velocities also in non-incremental pressure-correction schemes [82]. Otherwise such accuracy for velocities can be achieved only if incremental schemes are used. The complete procedure from which the given equation follows is given and more thoroughly explained in [83].

The originally proposed Kim and Moin scheme is otherwise almost identical to rotational scheme described before. It features same equation for predictor velocity (3.14), just without the explicit pressure (therefore with $q = 0$) [82, 83]. It also uses 2^{nd} order implicit Crank-Nicolson time stepping affecting viscous terms. The scheme introduces same projection equation as equation (3.16). This means that Φ includes all the differences between (3.5) and predictor velocity equation, except the time derivative. Also, while Φ is calculated, pressure itself does not appear at all. This is appropriate only for incompressible flows, where liquid variables do not change because of pressure. Therefore one does not need to know the exact pressure, value of Φ suffices to determine velocity at new time level. Nevertheless, the authors in [83] recognize that in order to obtain accurate pressure, the equation in the form of (3.10) has to be used. They use the connection between p and Φ given in description of rotational scheme. The scheme is reported to have 2^{nd} order accuracy for velocity and pressure in [82]. But as given in previous section, the order of accuracy for pressure is generally 3/2.

2.2.5 Projection method as it is used in MFLOPS-3D code

The method which is used in the MFLOPS-3D code is based on Kim and Moin method. Non-incremental pressure-correction scheme is chosen and retained from Kim and Moin method as the use of $q = p^n$ in parallel computations can lead to accumulation of pressure errors in time [82, 87]. The second order backward differencing in time is preferably used (third order backward differencing is also available) with the viscous terms treated implicitly, giving the equation (3.20) for $\vec{v}^*$:

$$\frac{(3\vec{v}^* - 4\vec{v}^n + \vec{v}^{n-1})}{2\Delta t} - \frac{1}{Re}\Delta\vec{v}^* = f(t^{n,n-1}) = -\left((\vec{v}\cdot\nabla)\vec{v}\right)^{n,n-1} \tag{3.20}$$

The method, like the one in 2.2.3, does not directly solve for pressure, but at first obtains Φ. The projection equation is therefore already shown with equation (3.16). Pressure is defined after solution for Φ is known with equation (3.21).

$$p^{n+1} = \Phi^{n+1} - \frac{1}{Re}\nabla\cdot\vec{v}^* \tag{3.21}$$

Definition of pressure as in equation (3.21) enables the pressure to have higher order of accuracy, as mentioned in the description of rotational scheme. But in order to ensure this, the $\vec{v}^*$

boundary condition comes from the Kim and Moin scheme and is equal to (3.18) or (3.19). The actual order of accuracy found for this method agrees well with the one given in [80] for rotational incremental scheme, although non incremental scheme is used here. Results proving this will be given in the mentioned chapter about verification.

Before continuing, two additional points should be addressed. The first is the naming of the projection methods used in the following text. The incremental or non-incremental pressure-correction schemes will be for the sake of simplicity referred to as pressure incremental or non-incremental projection methods. The second point is the question opened at presentation of governing equations. This refers to the pressure gradient term treatment in the non-dimensional NS momentum equations (3.5). As pointed out, this term appears to not be affected by division with density. This is not the case. The term itself is affected, but three reasons exist why this does not need to be marked with a change in ∇p term in (3.5). Firstly, the solution algorithm uses gradient of Φ and not of pressure to obtain velocity. Secondly, pressure gradient term is not included in predictor velocity $\vec{v}^*$ solution, since pressure non-incremental method is used. Therefore the term itself does not need to be changed since pressure as physical variable does not play an important role in incompressible flow simulations. This is accompanied by the third reason, which comes simply from the fact that solutions are obtained for normalised NS equations. For the unit mean velocity and channel half height, one does not need to know exact density and viscosity to set Re number of the flow, only their ratio. Therefore division of pressure gradient with a certain density would stand out from other terms and make whole simulation results in a way less general. Which brings out the final point, that results can be projected to a certain channel size and material properties (giving same Re number together with mean velocity). In this case it applies that if one would desire to know pressure as physical variable, the results obtained with equation (3.21) simply need to be multiplied by density.

3 Discretization methods

This part of the theoretical background deals with the discretization used in MFLOPS-3D code. Same discretization techniques are also used in cavitating flow simulations. It is generally very important to know the discretization techniques used in any kind of simulation one does. But when one regards DNS simulations, these techniques are even more important, since those which enable higher order of accuracy are able to catch more details on coarser grids. Higher accuracy therefore means they help lower the amount of CPU power needed to perform such demanding simulations [86].

3.1 Temporal discretization

The code enables use of 2^{nd} or 3^{rd} order backward differencing scheme for time derivatives. 2^{nd} order is used by default. The 3^{rd} order scheme, though available, is generally avoided since it was shown to make calculations unstable. In this work, only 2^{nd} order scheme will be used. Besides bad experience with MFLOPS-3D and 3^{rd} order backward scheme, an important support for this decision is also mentioned in [80]. There, it is firstly mentioned that projection method using pressure-correction scheme described in section 2.2.1 cannot be more than 1^{st} order accurate since it has irreducible splitting error of order $O(\Delta t)$. Therefore even higher order schemes for time derivatives do not improve its overall order of accuracy. Same applies to the scheme described in section 2.2.2, only that the order of accuracy in debate is at most 2^{nd} (possible for velocity). Similar conclusions that increase in time derivative order of accuracy does not increase the overall

order of accuracy can be also drawn for other pressure-correction schemes presented. It is also mentioned in the same article that in some cases, third order accurate scheme for time derivative can sometimes give third order accuracy for velocity, but only under certain conditions. If these are not respected, projection method with such time derivative can become unstable or can even be unconditionally unstable. Interestingly, these findings apply well with the performance of the third order accurate time derivative scheme in MFLOPS-3D.

A method which should also be mentioned in this part is the one step Adams-Bashforth method used to advance some explicit terms in time. The method is equal to forward Euler method and is first order accurate. Terms treated with it are described with the superscripts $n, n-1$. Such terms are most notably non linear terms, where this method helps having better approximation of these important, yet explicitly treated terms. The method approximates or predicts the value of a certain term ψ as shown in equation (3.22). A two step second order Adams-Bashforth method is also available but it showed to cause some stability issues therefore the one step method is preferred.

$$\psi^{n,n-1} = \psi^n + \left(\frac{d\psi}{dt}\right)^n \Delta t = 2\psi^n - \psi^{n-1} \tag{3.22}$$

3.2 Spatial discretization

Spatial discretization used in the code bases on structured (regular) and collocated grid. Methods used for it deserve a special explanation, especially since problems were encountered with the used discretization method and solutions for them differ on their nature. Some problems were connected with Runge phenomena [88] while mostly the used discretization is believed to impose issues with compatibility condition. Since collocated grid is used, some odd even decoupling problems were also noted, though they were not found to pose same impact as others. This problems will be referred to in later chapters. The main point of this part is rather to present the used spatial discretization techniques and how they are implemented into the MFLOPS-3D code.

Compact finite differences are used in MFLOPS-3D for spatial discretization. Before going to the description of this method and how it is implemented into the code, it should be said that the MFLOPS-3D code in its actual original form used compact finite differences only in streamwise, x, direction. For z or spanwise direction the code used spectral Fourier expansion with N_z modes as this direction was set to be periodic. In wall normal, y, direction, the code used pseudo-spectral Chebyshev collocation method. From the use of spectral Fourier expansion and Chebyshev method the code also got its name, MFLOPS-3D, meaning a "Multidomain FLOw Pseudo-Spectral 3D solver". The name was retained here for simplicity despite the change done to use compact finite differences in all three directions. More about the code with such discretization methods can be found in [89, 90], while examples of its use can be found also in [91, 92, 78, 93].

The decision to switch from spectral discretization methods to only compact finite differences was a consequence of limitations which spectral methods implied. The code was parallelised only in z direction, with number of sub domains or processors used dependent on number of Fourier modes N_z. This imposed a limitation on the size of computational cases which could be performed. Moreover, the spectral methods also impose limitations on the computational domains, especially on boundary conditions and use of more complex geometries [94, 95, 96, 97]. With such restrictions, one can simulate only simple incompressible flow cases, while limitations are therefore considerably more severe for cavitating flow simulations. Nevertheless, spectral methods are used extensively in DNS and LES simulations as they offer highest accuracy (uniformly for all

wavenumbers) and accordingly, also result in no to very little artificial numerical dissipation. This becomes increasingly important at higher Re numbers. Namely, in order to perform valid DNS simulations, at least most of the dissipation must be accurately captured [86]. At higher Re numbers, physical dissipation can quickly get very small and even smaller than numerical dissipation imposed by many discretization methods. Spectral methods offer best performance regarding this and thus can require much lower computational demands (coarser grids) than other discretization methods. There exist many examples of DNS and LES simulations with them, mostly of turbulent channel flows, such as [98, 99, 100, 101]. On the other hand, they were used, despite the issues imposed on boundary conditions, also for duct turbulent flow simulations. Examples are [102, 103].

Compact finite differences are close to the mentioned abilities of spectral discretization methods. They offer spectral like accuracy and can compete with spectral methods also in terms of numerical dissipation. At the same time they importantly use smaller stencils of points. Moreover, they enable simulations in more complex geometries and also do not suffer restrictions on boundary conditions, which are so problematic for spectral methods [94, 81, 87, 97]. They enable parallelisation in all directions as well. Because of this, they were noted as an interesting alternative to spectral methods [94, 104, 97]. Their use in DNS and LES simulations is therefore increasing, examples can be seen in [105, 95, 97, 104]. Consequently they were also applied for spatial discretization in all three directions in MFLOPS-3D. Up to now published results obtained with MFLOPS-3D code using compact finite differences can be seen in [106]. It should however be mentioned that compact finite differences seem to be an active field of research, with different proposals for their use in case of projection methods. Examples are in works like [107, 104, 105]. Moreover, examples of using compact schemes in all three computational directions are scarce. For instance [95, 97] use such an approach. Therefore the chosen discretization approach is not an ordinary one and the presence of issues mentioned at the start of this section is in a way not surprising.

In the following sections, the compact finite differences will be presented by first a general presentation of the theory behind them. This will also give a general derivation for first and second derivatives. After, description of how these schemes are applied to non uniform grids is given. Finally, compact finite difference schemes as used in MFLOPS-3D code are presented through their stencils. The description relies in its most part on work presented in [94].

3.2.1 Compact finite differences, general description

Compact finite difference schemes are in the literature also reffered to as Padé schemes [81, 94], Hermitian methods or Hermitian finite differences [104]. Simply described, these methods apply values of a variable f and its derivatives f' in a number of points to obtain a system of equations, from which those very derivatives can be calculated. Each of these equations can be derived with the use of polynomial fitting and a very nice example of how a fourth order compact finite difference scheme can be obtained is shown in [81]. Same example is shown below as it gives an important view for understanding how factors used in compact schemes are obtained. Assuming we have a uniform mesh on x coordinate, we can write a polynomial of degree four at node i, as given in equation (3.23).

$$f = a_0 + a_1(x - x_i) + a_2(x - x_i)^2 + a_3(x - x_i)^3 + a_4(x - x_i)^4 \tag{3.23}$$

The coefficients from a_0 to a_4 can be found by applying the polynomial to three variables and two derivative values. Since we are interested in derivative in x_i, we can simply write a first

derivative of (3.23) at x_i, giving us equality in (3.24).

$$f' = a_1 \tag{3.24}$$

This is fairly simple and shows that polynomial is also just a Taylor series, where coefficients a_1 to a_4 are the derivatives of order corresponding to the subscript and divided by the corresponding factorial. But in order to obtain the desired compact finite difference from it, one has to write also expressions of (3.23) and (3.24) in x_{i-1}, x_{i+1}. If we then sum the values of two derivative expressions and discount the values of variable expressions, we have a system of two equations, from which a scheme given in (3.25) follows.

$$\frac{1}{4}f'_{i-1} + f'_i + \frac{1}{4}f'_{i+1} = \frac{3}{4h}\left(f_{i+1} - f_{i-1}\right) \tag{3.25}$$

Such an equation can be written for all points, exception those near and on domain boundaries (how these are treated will be included in the explanation later). This gives us a tridiagonal system of equations which can be solved to obtain the values of the derivatives. In such a logic, compact finite difference schemes apply smaller stencils of points and still achieve higher orders of accuracy mentioned before. In this case, this is the fourth order, since the polynomial or Taylor series used in (3.23) is fourth order polynomial and terms up to fourth order were eliminated, except the ones with first derivative. To achieve higher order of accuracy, higher degree polynomials and more points or larger stencils of points can be used. With them we obtain the expressions composed from variable and derivative values in those points, which then gives the compact difference schemes with wanted order of accuracy. As can be seen, this depends on the highest polynomial or Taylor series term we are able to match and eliminate [94]. Since a lot of schemes can be developed, a useful way to present them is to have a general expression for them, and an example is given in equation (3.26). However, it has to be stressed that a uniform mesh is needed to write such a generalization.

$$\beta f'_{i-2} + \alpha f'_{i-1} + f'_i + \alpha f'_{i+1} + \beta f'_{i+2} = c\frac{f_{i+3} - f_{i-3}}{6h} + b\frac{f_{i+2} - f_{i-2}}{4h} + a\frac{f_{i+1} - f_{i-1}}{2h} \tag{3.26}$$

The coefficients β, α, a, b, c are the coefficients which follow from the mentioned matching or elimination of Taylor series terms. Of course, elimination does not apply to the terms featuring the derivative we are trying to obtain (in previous case a_1). Since equation (3.26) is written for a uniform mesh, it is also possible to define connections between the coefficients for a certain order of accuracy. Examples for fourth and eight order accuracy are given in (3.27) and (3.28) respectively [94].

$$a + 2^2 b + 3^2 c = 2\frac{3!}{2!}\left(\alpha + 2^2\beta\right) \tag{3.27}$$

$$a + 2^6 b + 3^6 c = 2\frac{7!}{6!}\left(\alpha + 2^6\beta\right) \tag{3.28}$$

The connections between coefficients also allow a choice to omit some coefficients. For example, β can be either used or omitted, thus giving us a pentadiagonal or tridiagonal schemes. Furthermore, different schemes can be proposed by using or omitting different coefficients and also by choosing values of some. Examples of such different schemes are given in [94].

The presented logic to obtain compact finite difference schemes for first derivative is used also in other areas where these schemes are applied. One can derive such schemes also for higher order

derivatives and even for filtering and interpolation purposes [94, 97]. Furthermore, they can be used on staggered or collocated grids, example is [95]. In the case of MFLOPS-3D code, these schemes are also used for the second derivatives. On a uniform mesh, a general compact scheme for second derivative is given with equation (3.29).

$$\beta f''_{i-2} + \alpha f''_{i-1} + f''_i + \alpha f''_{i+1} + \beta f''_{i+2} = c\frac{f_{i+3} - 2f_i + f_{i-3}}{9h^2} + b\frac{f_{i+2} - 2f_i + f_{i-2}}{4h^2} + a\frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} \tag{3.29}$$

The coefficients β, α, a, b, c follow again from matching of Taylor series terms in order to eliminate all except those with the second derivative. And equations which give connections between them for different orders of accuracy are also available. For the sake of comparison, the connections for fourth and eight order accuracy are given in (3.30) and (3.31) respectively.

$$a + 2^2 b + 3^2 c = 2\frac{4!}{2!}\left(\alpha + 2^2\beta\right) \tag{3.30}$$

$$a + 2^6 b + 3^6 c = 2\frac{8!}{6!}\left(\alpha + 2^6\beta\right) \tag{3.31}$$

It is obvious that connections are almost identical to those for first derivative scheme. And as in the case of first derivative, different schemes can be obtained with inclusion or omitting of certain coefficients. Examples of such schemes are again given in [94].

3.2.2 Compact finite differences near the boundaries

In the previous section, the compact finite differences are defined generally and for the case of points in the centre of a domain or in a periodic domain. It is obvious they use symmetrical stencils and are therefore central schemes. However, computational domains are usually limited, thus it is necessary to derive also compact finite difference schemes in points near or on the boundaries. Forward and backward schemes are used for such tasks. For example, for point $i = 1$, forward difference scheme can be used. Equation (3.32) gives a general scheme for first derivative and equation (3.33) for second.

$$f'_1 + \alpha f'_2 = \frac{1}{h}\left(af_1 + bf_2 + cf_3 + df_4\right) \tag{3.32}$$

$$f''_1 + \alpha f''_2 = \frac{1}{h^2}\left(af_1 + bf_2 + cf_3 + df_4 + ef_5\right) \tag{3.33}$$

Similar equations as (3.32) and (3.33) can be written for point $i = n$ at the end of a domain, just by using backward scheme. Similar logic to this is used also for second or penultimate points in a domain, where one has more options what to chose, since presence of at least one point on the other side of the point in interest makes it possible to not have purely forward or backward schemes. Once one gets to the point which has enough points on both sides to satisfy the chosen central scheme, the rest of the equations in the system of the equations can be written in same manner, that is, with the chosen central scheme. But attention should be given that the stencils of derivative values on the left hand side of the system of equations form diagonal matrices. This is a consequence of usually applied solving procedures (based on LU decomposition). Stencils of variable values on the right hand side of the system of equations should then be chosen to complement demands for certain order of accuracy.

3.2.3 Compact finite difference schemes on non uniform grids

As it was mentioned, the schemes presented before are developed for uniform grids. These have very limited use in simulations, hence schemes adapted for non uniform meshes have to be developed. There are generally two ways to develop such schemes and they will be briefly presented here.

The first possibility is to transform the non uniform mesh into uniform, on which use of previously shown schemes is possible. Usually, the mesh change is done by applying Jacobian transformation. Examples can be seen in in [97, 104]. The logic in this approach is that if X is considered to be transformed coordinate and x the physical one (on a non unform grid), the first and second derivatives on physical coordinates can be obtained with the use of equations (3.34) and (3.35).

$$\frac{\partial f}{\partial x} = \frac{\frac{\partial f}{\partial X}}{\frac{\partial x}{\partial X}} \tag{3.34}$$

$$\frac{\partial^2 f}{\partial x^2} = \frac{\frac{\partial^2 f}{\partial X^2} - \frac{\partial f}{\partial x}\frac{\partial^2 x}{\partial X^2}}{\left(\frac{\partial x}{\partial X}\right)^2} \tag{3.35}$$

Each of the derivatives on the right hand side of equations above can be evaluated with the compact finite difference schemes as presented before [97]. This method exhibits limitations in the sense that the non uniform mesh has to be sufficiently smooth in order to well define $\frac{\partial x}{\partial X}$ and $\frac{\partial^2 x}{\partial X^2}$ and thus calculate them without a big loss in accuracy [97, 104]. It is because of this that such approach has not been used commonly in cases of unsteady flow and DNS simulations.

The other way, which is also used in MFLOPS-3D code, is to develop the schemes specifically for certain non uniform mesh. This can be done in different ways, which are all similar in the fact that they are based on matching and eliminating terms in Taylor series. Therefore they follow the presented procedure in 3.2.1, but for the case of non uniform mesh. Consequently, if they are obtained for uniform grids, they also give same results as schemes presented before. Such a procedure is used for instance in [97] and [96]. Since the approaches on how to derive the schemes in this way can vary, only the method used in MFLOPS-3D code will be presented here.

The approach used in MFLOPS-3D code is based on the choice of a stencil of points, for which the Taylor series for derivative and variable values will be written. From these equations one then tries to eliminate all terms except those which feature the desired derivative. If a case for fourth order accurate first derivative in central points is considered and Taylor series for values of variable and its first derivative are taken in points $i-1, i, i+1$, we get the system of six equations as given in (3.36) − (3.41).

$$f(i-1) = a_0 + \frac{\partial f}{\partial x}(x_{i-1} - x_i) + \frac{\partial^2 f}{\partial x^2}\frac{(x_{i-1}-x_i)^2}{2!} + \frac{\partial^3 f}{\partial x^3}\frac{(x_{i-1}-x_i)^3}{3!} + \frac{\partial^4 f}{\partial x^4}\frac{(x_{i-1}-x_i)^4}{4!} \tag{3.36}$$

$$f(i) = a_0 + \frac{\partial f}{\partial x}(x_i - x_i) + \frac{\partial^2 f}{\partial x^2}\frac{(x_i-x_i)^2}{2!} + \frac{\partial^3 f}{\partial x^3}\frac{(x_i-x_i)^3}{3!} + \frac{\partial^4 f}{\partial x^4}\frac{(x_i-x_i)^4}{4!} \tag{3.37}$$

$$f(i+1) = a_0 + \frac{\partial f}{\partial x}(x_{i+1} - x_i) + \frac{\partial^2 f}{\partial x^2}\frac{(x_{i+1}-x_i)^2}{2!} + \frac{\partial^3 f}{\partial x^3}\frac{(x_{i+1}-x_i)^3}{3!} + \frac{\partial^4 f}{\partial x^4}\frac{(x_{i+1}-x_i)^4}{4!} \tag{3.38}$$

$$f'(i-1) = \frac{\partial f}{\partial x} + \frac{\partial^2 f}{\partial x^2}(x_{i-1} - x_i) + \frac{\partial^3 f}{\partial x^3}\frac{(x_{i-1}-x_i)^2}{2!} + \frac{\partial^4 f}{\partial x^4}\frac{(x_{i-1}-x_i)^3}{3!} \tag{3.39}$$

$$f'(i) = \frac{\partial f}{\partial x} + \frac{\partial^2 f}{\partial x^2}(x_i - x_i) + \frac{\partial^3 f}{\partial x^3}\frac{(x_i-x_i)^2}{2!} + \frac{\partial^4 f}{\partial x^4}\frac{(x_i-x_i)^3}{3!} \tag{3.40}$$

$$f'(i+1) = \frac{\partial f}{\partial x} + \frac{\partial^2 f}{\partial x^2}(x_{i+1} - x_i) + \frac{\partial^3 f}{\partial x^3}\frac{(x_{i+1}-x_i)^2}{2!} + \frac{\partial^4 f}{\partial x^4}\frac{(x_{i+1}-x_i)^3}{3!} \tag{3.41}$$

The Taylor series above are in contrast to equation (3.23) given with derivatives and factorials, which are in (3.23) represented with coefficients. As the first derivative in point x_i is wanted, all terms without $\frac{\partial f}{\partial x}$ should be eliminated. Since (3.40) gives $f'(i) = \frac{\partial f}{\partial x}$, and a scheme with $f'(i)$ in unit form is desired, the elimination of all other terms should be done in a way to result in such a scheme. As we actually look for factors to multiply equations (3.36) − (3.41) to achieve this and at the same time we have to separately treat terms according to the derivative they feature, it is practical to write a system of equations in matrix form as given below. In it, the distances between the points are given as $x_{n1} = x_{i-1} - x_i$ and $x_{p1} = x_{i+1} - x_i$, while obvious $0 = x_i - x_i$ is also used.

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & x_{n1} & 0 & x_{p1} \\ x_{n1} & x_{p1} & \frac{x_{n1}^2}{2!} & 0 & \frac{x_{p1}^2}{2!} \\ \frac{x_{n1}^2}{2!} & \frac{x_{p1}^2}{2!} & \frac{x_{n1}^3}{3!} & 0 & \frac{x_{p1}^3}{3!} \\ \frac{x_{n1}^3}{3!} & \frac{x_{p1}^3}{3!} & \frac{x_{n1}^4}{4!} & 0 & \frac{x_{p1}^4}{4!} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

In this form, the rows of the matrix on the left represent the terms from the system of equations (3.36) − (3.41) which include same derivatives or terms of same order. That is also why the first row has first two columns empty. Additionally, it could be also said that certain column features all the terms which are used in a certain Taylor series equation. And that the last three columns represent equations (3.36)−(3.38), while the first two stand for (3.39) and (3.41). Obviously there are only five equations out of six included in the matrix on the left side. The reason is that equation (3.40) contributes its term into the right hand side vector value −1. This is a consequence of the mentioned equality $f'(i) = \frac{\partial f}{\partial x}$ and the wish to have a scheme with $f'(i)$ in unit form. With the solution for the vector on the left hand side, coefficients $\alpha_1 - b_3$ are obtained, which then define compact finite difference scheme on non uniform mesh. Contrary to uniform grids, where for instance coefficients α_1 or α_2 are given with single α as symmetry across the point in question x_i is expressed, the coefficients here are generally different. Coefficients b_1, b_2 etc also include in them directly the distances between the grid points, which is not the case in before considered uniform grid example. In limiting case of uniform mesh and presented system of equations, one obtains first derivative compact scheme with $\alpha_1 = \alpha_2 = \frac{1}{4}$ and $b_1 = -b_3 = \frac{3}{4dx}$, where dx is the distance between two points. This corresponds to the example given in 3.2.1 (scheme in (3.25)). It also complies with the need that the methods for non uniform grids should in the cases of uniform grids converge to same schemes.

Although the example given here only illustrates the procedure for first derivative with fourth order accuracy on central points in a mesh, it can be applied to various stencils of points and with them for various orders of accuracy. Moreover, it can also be used for definition of schemes for second derivative or for other purposes where compact schemes are used. Finally, its use is not limited only for central points on a mesh and symmetric stencils of points. It can be applied also to the points near or on the boundaries of a domain, where forward or backward schemes are used. Since coefficients for each point in a stencil are different, the general schemes which follow from the procedure are different than general schemes given before. For the first derivative, the equation for central schemes (3.26) is changed to (3.42), while the forward or backward schemes at the boundaries, before described with equation (3.32), follow (3.43). Second derivative have general schemes defined in same manner. For clarity, it is worth to mention that coefficients α_i and b_i in presented schemes are not equal for schemes in different points. Same notation is used only for simplicity.

$$\beta_1 f'_{i-2}+\alpha_1 f'_{i-1}+f'_i+\alpha_2 f'_{i+1}+\beta_2 f'_{i+2} = b_1 f_{i-3}+b_2 f_{i-2}+b_3 f_{i-1}+b_4 f_i+b_5 f_{i+1}+b_6 f_{i+2}+b_7 f_{i+3} \quad (3.42)$$

$$f'_1 + \alpha_1 f'_2 = b_1 f_1 + b_2 f_2 + b_3 f_3 + b_4 f_4 \quad (3.43)$$

3.2.4 Compact finite difference schemes as used in MFLOPS-3D code

The purpose of this section is to complete the presented theory about compact finite difference schemes with the definition of their use in MFLOPS-3D. This includes the available orders of accuracy for certain derivative, chosen stencils for schemes in central and border points and solution techniques to obtain their coefficients as well as solutions of a system of equations, composed from resulting compact schemes. These solutions are finally also the desired derivatives.

The MFLOPS-3D code enables a choice between various orders of accuracy for derivatives. 2^{nd}, 4^{th}, 6^{th} and 8^{th} order accurate first and second derivative schemes are available. Both first and second derivative schemes use same stencils of points, the size of which depends on the chosen order of accuracy and position of the point for which a scheme is defined. If the mesh has n points in a certain direction, only the schemes for points $\{3; n-2\}$ have symmetric stencils or can be considered as central schemes. They also always exhibit the stated order of accuracy. Other four points use complete or partial forward or backward schemes and stencils of points corresponding to them. Consequently the order of accuracy in them can be lower than in the central points. An overview of stencils for both first and second derivatives for all grid points is given in Table 3.1. Overview also accounts for different order of accuracy, where classification bases on central schemes order of accuracy. Referring to this table and also stencil of points for a scheme in general, an additional distinction has to be made. Stencil of points for a certain scheme can actually be divided into implicit and explicit part. The implicit part or stencil refers to the stencil of points from which derivatives are included in a certain scheme (left side of a scheme equation), except the desired derivative. The explicit stencil of a scheme is represented on the other hand by points from which variable values are used (right side of the same equation). The implicit and explicit stencils of points are in Table 3.1 depicted with *impl* and *expl* notations.

Table 3.1: Implicit and explicit stencils of compact schemes as used in MFLOPS-3D for various points in the mesh and for available orders of accuracy.

order of accuracy	8^{th}		6^{th}		4^{th}		2^{nd}	
point	impl	expl	impl	expl	impl	expl	impl	expl
1^{st} and n	3	6	1	6	1	4	0	4
2^{nd} and n-1	1	5	2	6	1	5	0	4
3:n-2	4	5	2	5	2	3	0	3

It can be seen from Table 3.1 that the 2^{nd} order accurate schemes are fully explicit. They only feature the desired derivative value on the left hand side of any scheme equation. Schemes for first and last points on a mesh are clearly completely forward or backward. The schemes for second points are also only forward or backward if the implicit stencil value equals one. Otherwise they are partially forward or backward as they feature one derivative from opposite direction too. As mentioned, the order of accuracy stated in Table 3.1 applies only to points $\{3 : n-2\}$. For the case of schemes mentioned under 2^{nd} and 4^{th} order accuracy it was noted that same order applies

also for boundary points. Therefore these two groups of schemes are globally 2^{nd} or 4^{th} order accurate. Convergence tests showing order of accuracy of first derivatives for different points or groups of points are given in the Appendix for the schemes listed under 4^{th} and 8^{th} order of accuracy. Second derivatives are also given, but for slightly different schemes. Following section dealing with mono domain solver gives the explanation on this subject. The schemes listed as 4^{th} order accurate were also used generally in calculations done in this work, since they provide global accuracy of such order.

In order to obtain coefficients β_i, α_i, b_i for the schemes in all grid points (defined with stencils as given in Table 3.1), a matrix system like the one in previous section is constructed from Taylor series for each grid point. The solutions (coefficients) are obtained from it by a call to LAPACK library routine *dgesv* [108]. This solves a real system of linear equations using LU decomposition with partial pivoting and row interchanges.

Finally, once the coefficients for the schemes in all points are obtained, the derivatives of a certain variable can be defined. This is done in a certain direction separately for each grid line. Reason is that system of equations is formed with the use of obtained compact schemes for the points on one grid line. The system is for the case of first derivative presented with equation (3.44), where vector $[f]'$ represents the desired values of first derivatives in a grid line, vector $[f]$ the known variable values in same line, while rows in matrices $\mathbf{A}$ and $\mathbf{b}$ include the implicit and explicit coefficients of schemes in corresponding grid points.

$$\mathbf{A}[f]' = \mathbf{b}[f] \tag{3.44}$$

System of equations is formed in same manner for second derivative and it can be presented by at most a pentadiagonal matrix system. This is based on the fact that the implicit point stencils, which form the matrix $\mathbf{A}$, do not include more than four points. On the other hand, matrix $\mathbf{b}$ is directly multiplied with vector $[f]$, resulting in a vector on the right hand side (RHS). The pentadiagonal form is also the reason why schemes in table 3.1 are such and why the order of accuracy is not globally equal in general. As mentioned before, diagonal form of matrices is preferred for solving reasons. Pentadiagonal system can be effectively solved with the use of an adapted LU method. In this case, Tridiagonal Matrix Algorithm (TDMA) adapted to the pentadiagonal matrix is applied. The solution of derivative values is therefore direct and fast. More about such algorithm can be found in [109].

3.3 Non-linear term discretization

Considering discretization techniques, it is also important to define how the non linear term in momentum equations (3.1) is described. The importance of non linear term does not lie only in the fact that for practical solutions of Navier-Stokes equations we need to treat it explicitly or semi explicitly. The way the term is treated also defines if flow quatities, notably kinetic energy, will be conserved. Experience showed that if kinetic energy is not conserved, simulations of incompressible flow need to have some numerical dissipation, otherwise they become unstable [110].

The ability of non linear or a general $\psi(x, y, z, t)$ term to be conservative is first defined with the way in which the term is included in differential form of equations, which are also used in this case (since we are using finite difference approach). If a term can be written in a divergence form, shown generally in (3.45), it can have this ability and we say it is conservative a priori [110].

$$\nabla \cdot \psi(x, y, z, t) \tag{3.45}$$

Why is the divergence form of a certain term conservative is shown in [110] on a thorough example. Here, only a simple illustration is given. If we take for instance continuity equation (3.2) for incompressible flow, in which the time derivative equals zero, velocity divergence then equals zero. This is also a needed condition in order to conserve mass in incompressible flow simulations and already points to why a term in divergence form is considered conservative. A complete explanation is obtained if one performs volume integral of a term in divergence form and then makes the use of Gauss theorem. In case of continuity equation, this gives equation (3.46).

$$\int_V \nabla \cdot \vec{v}\, dV = \oint_S \vec{v} \cdot \vec{n}\, dS = 0 \tag{3.46}$$

Equation (3.46) shows that the divergence form enables known connection between the volume derivative, which considers whole domain, and surface integral, which considers only fluxes in or out of a domain. It means that if a term can be written in divergence form, it will be able to resemble the influence of inflow or outflow in a domain, thus conserve certain variable. As it will be seen later, a condition as written here, is only the first and analytical demand, which needs to be assured in order to satisfy conservation laws. Because of the analytical nature, it can also be referred to as continuous demand [107]. Condition should also be satisfied discretely, but this will be considered later. For the moment, only this analytical demand will be put into use for the non linear term in question.

If equation (3.7) is considered, one can see that the non linear term is not given in divergence form. In fact, the form in (3.7) follows from the use of continuity equation (3.2) and is in [110] referred to as advective form. Such a form is not conservative a priori. It is conservative only if continuity equation (3.4) is satisfied. Since this is also the goal in incompressible flow simulations and ensured by projection methods, this advective form of non linear term seems to be appropriate for use. However, another form is better for use and also shows better characteristics than original divergence form, which would be present in (3.7) without the use of continuity equation in it. This is the skew-symmetric form, which can be written as given in equation (3.47). This form is a combination of divergence and advective form. It can also be written otherwise, for instance, as a combination of divergence form and continuity equation [110]. The form presented here is also the form used in MFLOPS-3D code.

$$((\vec{v} \cdot \nabla)\vec{v}) = \frac{1}{2}\nabla \cdot (\vec{v}\vec{v}) + \frac{1}{2}\left((\vec{v} \cdot \nabla)\vec{v}\right) \tag{3.47}$$

This form shows its advantages if transport equations for a square of a certain velocity v_i or kinetic energy K are considered. Transport equation for square of velocity ${v_i}^2$ is given in (3.48) and follows from Navier-Stokes momentum equation (3.1) for i^{th} component being multiplied with v_i. The i subscript at non linear, pressure and viscous terms stands for the i^{th} component of the terms (certain direction).

$$\rho\left(\frac{\partial {v_i}^2}{\partial t} + v_i\left((\vec{v} \cdot \nabla)\,\vec{v}\right)_i\right) = -v_i \nabla p_i + v_i\left(\nabla \cdot \left(\mu(\nabla\vec{v}) + \mu(\nabla\vec{v})^T\right) - \frac{2}{3}\nabla\left(\mu(\nabla \cdot \vec{v})\right)\right)_i \tag{3.48}$$

If non linear term $(\vec{v} \cdot \nabla)\vec{v}$ in (3.48) is written with (3.47), it follows that the product of velocity component v_i with i^{th} component of the nonlinear term can be written in divergence form given in (3.49) [110]. This means that the skew-symmetric form of the non linear term is a priori conservative in velocity square equation. The derivation is analogous for the case of kinetic

energy equation, which is just a summation of equation (3.48) for all three velocity components. Hence the skew-symmetric form of non linear term is also a priori conservative regarding kinetic energy. This is also the reason why is was chosen in MFLOPS-3D code.

$$v_i \left(\frac{1}{2} \nabla \cdot (\vec{v}\vec{v}) + \frac{1}{2} \left((\vec{v} \cdot \nabla) \vec{v} \right) \right)_i = \nabla \cdot \left(\frac{{v_i}^2}{2} \vec{v} \right) \tag{3.49}$$

Although the non linear term in skew-symmetric form is conservative a priori for kinetic energy equation, continuity has to be conserved for the form to be conservative in momentum equation, like for the case of using advective form. However, the skew-symmetric form has to be also spatially discretized correctly in order for conservation properties to hold completely. As already mentioned, the divergence form presents only analytical or continuous condition, but not also discrete. In [110], it is mentioned that for collocated grid systems as used in MFLOPS-3D (in [110] such configuration is referred to as regular grid system) the skew-symmetric form is conservative in kinetic energy equation without some special treatment in discretization when simple second or fourth order central differences are used. Advective or divergence form on the other hand demand adjustments in their derivative definitions (interpolation of variables and derivatives is used) otherwise they do not conserve kinetic energy nor any of square velocity components. No specific results are mentioned about skew-symmetric form performance when compact finite differences are used, but two predictions can be made nevertheless. Firstly, in [110] it is mentioned that non uniform grids raise further questions about conservation properties of different non linear term forms. No discrete operators, which would be fully conservative and at the same time keep the order of accuracy, exit. One can therefore opt to keep the operators unchanged and retain same order of accuracy but have worse conservation properties or vice versa. There exists also a middle way, where operators can be adjusted with weights, which keeps the order of accuracy and enables conservation with errors of same order. It can be said that the first option, where the order of accuracy of derivatives is preserved, is chosen in MFLOPS-3D, as the derivative operators have no special treatment applied to improve conservation properties on non uniform grids. Secondly, in [111], a related article to [110], but dealing with forms for non linear term in cases of compressible flows, it is mentioned that no fully conservative schemes for the non linear term exist when compact finite differences are used. The reason is in the forms for this term (divergence, advective and skew-symmetric) being not commutable between each other when such discretization is used, even if discrete continuity is satisfied. Meaning that they do not add up to the same result even if the basic demand for this, satisfaction of continuity, is ensured. However, the skew-symmetric form is in [111] mentioned to be the only stable one at the inviscid limit when compact finite differences are used. Therefore the two predictions which can be done for the use of skew-symmetric form in MFLOPS-3D are that the conservation of kinetic energy is ensured up to certain order of accuracy and stable simulations are still possible, which cannot be said for the other two forms.

To conclude the discussion about non linear term form, it can be reminded that one step Adams-Bashforth method is used in order to approximate this explicitly treated term in equation (3.7) to the new time level $n+1$. The term is therefore written as given in equation (3.50).

$$\left((\vec{v} \cdot \nabla) \vec{v} \right)^{n+1} \approx \frac{1}{2} \nabla \cdot (\vec{v}\vec{v})^{n,n-1} + \frac{1}{2} \left((\vec{v} \cdot \nabla) \vec{v} \right)^{n,n-1} \tag{3.50}$$

4 Solving techniques

DNS simulations of turbulent flows are by definition three dimensional. As all turbulent scales have to be resolved in them, this leads to high computing demands. Therefore the codes specifically developed for DNS simulations usually feature specific solving techniques to enable higher performance and shorter computational times. These, like spectral methods, which are often used for spatial discretization in such simulations, can limit the flexibility to perform various simulations. In case of solving techniques, the limitations are usually shown regarding simulations in various geometries and flow configurations.

The MFLOPS-3D code aims to enable high computational performance by using a unique mix of mono and multi domain solving techniques. Fast direct solver is used to solve a system of equations on the level of mono or sub domains. This mono domain solver is then coupled with a non overlapping multi domain method featuring iterative solver, enabling parallelization of the code in three directions. Furthermore, the multi domain method demands the mono domain solver to be used only twice per time step and has therefore potential to further improve the computing performance. Both mono and multi domain methods include approaches which form the mentioned unique mix, to our knowledge not used in any other code. The purpose of this section is to present these mono and multi domain solving techniques. At first, the mono domain solver is presented for two types of equations which are solved in MFLOPS-3D. Then, the multi domain method explanation and its coupling with mono domain solver is given. Finally, application of boundary conditions for a solved system of equations is explained.

4.1 Mono domain solver

Considering projection methods described in section 2.2, one can see that there are only two variables which need to be solved using a system of equations. These two are $\vec{v}^*$ and Φ. It is possible and common to use a solver which shares same basis to solve for both variables. In order to have such an universal solver, similarities between equations for $\vec{v}^*$ and Φ have to be found. If the projection equation (3.16) is multiplied with divergence, the divergence-free constraint is implied and a Poisson equation is obtained. And if (3.20) is reorganized, it forms Helmholtz-like equation with a non-zero right-hand side (RHS in the following text). Both equations are equal in the fact that they feature a laplacian operator of the implicit, sought values on the left hand side (LHS in the following text). They are given as (3.51) for $\vec{v}^*$ and (3.52) for Φ.

$$\left(\Delta - \frac{3Re}{2\Delta t}\right)\vec{v}^* = \left(\frac{-4\vec{v}^n + \vec{v}^{n-1}}{2\Delta t} + ((\vec{v}\cdot\nabla)\vec{v})^{n,n-1}\right)Re \tag{3.51}$$

$$\Delta\Phi = \frac{3}{2\Delta t}\nabla\cdot\vec{v}^* \tag{3.52}$$

The systems of equations following from these two equations can be solved with essentially the same approach, especially since Poisson equation is just a form of Helmholtz equation with zero constant on LHS. The mono domain solver in MFLOPS-3D solves such systems by applying first eigendecomposition of discrete Laplacian, which then leads to diagonalisation of the whole LHS matrix [112]. To show this approach and also to first just generally present the mono domain solver and its characteristics, a basic one dimensional case, given in (3.53), should be considered. In it, matrix $\mathbf{B}$ stands in the place of Laplacian, where factor a represents possible constant like the one present in (3.51). Matrix $\mathbf{I}$ stands for identity matrix. One dimensional case is the basis of

the used eigendecomposition as Laplace operator has to be actually composed of three matrices. This is a consequence of compact finite differences.

$$(\mathbf{B} - a\mathbf{I})u = D \tag{3.53}$$

If the matrix $\mathbf{B}$ is diagonalisable, we can decompose it into eigenvectors and eigenvalues as presented in (3.54). In this equation, $\mathbf{H}$ is a matrix whose i^{th} column represents i^{th} eigenvector of matrix $\mathbf{B}$ and $\mathbf{\Lambda}$ is the diagonal matrix, composed of corresponding eigenvalues. $\mathbf{H}^{-1}$ is the inverse of $\mathbf{H}$.

$$\mathbf{B} = \mathbf{H}\mathbf{\Lambda}\mathbf{H}^{-1} \tag{3.54}$$

With $\mathbf{B}$ decomposed, the complete matrix on the LHS of (3.53) can be diagonalised as the procedure in equations (3.55) – (3.59) shows.

$$\left(\mathbf{H}\mathbf{\Lambda}\mathbf{H}^{-1} - a\mathbf{I}\right)u = D \tag{3.55}$$
$$\left(\mathbf{\Lambda}\mathbf{H}^{-1} - \mathbf{H}^{-1}a\mathbf{I}\right)u = \mathbf{H}^{-1}D \tag{3.56}$$
$$\left(\mathbf{\Lambda}\mathbf{I} - a\mathbf{I}\right)\mathbf{H}^{-1}u = \mathbf{H}^{-1}D \tag{3.57}$$
$$\bar{u} = \mathbf{H}^{-1}u \quad \text{and} \quad \bar{D} = \mathbf{H}^{-1}D \tag{3.58}$$
$$\left(\mathbf{\Lambda}\mathbf{I} - a\mathbf{I}\right)\bar{u} = \bar{D} \tag{3.59}$$

Final result of the procedure above is equation (3.59), which represents a diagonal system of equations solved directly for $\bar{u}$. In order to obtain solution for u, the $\bar{u}$ is multiplied with $\mathbf{H}$. As written, this approach is the basis of mono domain solver in MFLOPS-3D. It enables one to find solutions for $\vec{v}^*$ and Φ directly and therefore more accurately than an iterative approach. It also enables one to find solutions faster. The reason for this is the fact that Laplacian Δ is a function of geometry only. Added to this, the a in (3.53) is in incompressible flows a constant as given in (3.51). Both of them are therefore constant in time and consequently the eigendecomposition of Laplacian and diagonalisation of (3.51) or (3.52) has to be done only once, at the start of a simulation. Then, mono domain solver only has to directly solve the system of equations, which leads to a faster and more efficient solution procedure [104]. The eigenvectors in $\mathbf{H}$, inverse eigenvectors in $\mathbf{H}^{-1}$ and eigenvalues in $\mathbf{\Lambda}$ are obtained with a standard algorithm from LAPACK (*dgeev* routine is used for eigenvalues and eigenvectors while *dgetrf* and *dgetri* are used to inverse eigenvectors) [108].

However, Laplacian operator is three dimensional and the attribution from each dimension is presented by a separate term which forms its own matrix on the LHS. Therefore the eigen-decomposition has to be done separately for each term, making the diagonalisation procedure more complicated than what was presented in (3.55)–(3.59). Moreover, inclusion of all three dimensions means one needs to take into account that the unknowns u and RHS are represented by a rank-3 tensor, while the 2^{nd} order derivative operators are 2D matrices, each affecting values in the tensor of unknowns in different direction. Therefore even more attention has to be paid to the way in which diagonalisation is done. MFLOPS-3D code uses LAPACK procedures, which enable easier handling of such matrix-tensor operations (*dgemm* routine is used for this). Only the final result of diagonalisation in 3D is presented here, while the procedure is in a bit more detail given in the Appendix.

For the 3D diagonalisation, one should first write the usually encountered Helmholtz equation (before given with (3.53)) as defined in (3.60). In it, variables $D_x^{(2)}$, $D_y^{(2)}$ and $D_z^{(2)}$ represent the

matrices composed of coefficients for discretized 2^{nd} derivatives in Laplacian operator, σ is the constant value, represented before by a in (3.53), and U is the tensor of variables we solve for. F on RHS is rank-3 tensor following from known values in the domain. Then, operations involving eigendecomposed $D_x^{(2)}$, $D_y^{(2)}$ and $D_z^{(2)}$ in their given or transposed form are used, finally leading to the desired diagonal system of equations. This is given as defined in equations (3.61) and (3.62).

$$D_x^{(2)}U + D_y^{(2)}U + D_z^{(2)}U - \sigma U = F \tag{3.60}$$

$$\left(\mathbf{\Lambda}_x + \mathbf{\Lambda}_y + \mathbf{\Lambda}_z - \sigma\right) S_x^{-1} U \left(S_y^{-1}\right)^T \left(S_z^{-1}\right)^T = S_x^{-1} F \left(S_y^{-1}\right)^T \left(S_z^{-1}\right)^T \tag{3.61}$$

$$\left(\lambda_{x,k} + \lambda_{y,l} + \lambda_{z,m} - \sigma\right) \bar{u}_{k,l,m} = \bar{f}_{k,l,m} \tag{3.62}$$

Equation (3.61) represents the diagonalized system of equations, where $\mathbf{\Lambda_x}$, $\mathbf{\Lambda_y}$ and $\mathbf{\Lambda_z}$ stand for diagonal matrices of eigenvalues from $D_x^{(2)}$, $D_y^{(2)}$ and $D_z^{(2)}$. S_x^{-1}, S_y^{-1} and S_z^{-1} are the inversed matrices of their eigenvectors, with the superscript T meaning the transposed form. The following equation (3.62) gives an equation for a variable solution in a certain point, where the $\bar{u}_{k,l,m}$ and $\bar{f}_{k,l,m}$ are point values from multiplications of tensors U and F with inverse eigenvectors, while values $\lambda_{x,k}$, $\lambda_{y,l}$ and $\lambda_{z,m}$ represent eigenvalues in the corresponding points k, l, m. Solution for whole tensor U follows appropriate multiplication of tensor of $\bar{u}_{k,l,m}$ solutions with eigenvector matrices (*dgemm* routine is used). As mentioned, the diagonalisation presented here gives only the final step, which provides also the solution for the sought variables U. The diagonalisation procedure is otherwise better presented in the Appendix.

Although the mono domain solver presented here enables faster, direct solutions of equations' systems, it also imposes an important limit. This is the need to have the factor a in (3.53) or σ in (3.60) equal for all points in the domain. Therefore a or σ cannot be a vector or a matrix. As it will be seen in the case of cavitating flow simulations, it is often desired to have them used as variables with different values in the domain, but in such a case, the eigendecomposition would be more complicated and would have to be done for each time step separately. This would make the whole solution procedure including the multi domain solver slower.

It should be also mentioned that regarding original MFLOPS-3D code, which used spectral approach, the diagonalisation was not done in the same manner. In that code, only y direction second derivative was diagonalised, and since spectral Fourier expansion was used in z, the system was transformed into a series of one-dimensional Helmholtz-like equations in x. These were efficiently solved using the pentadiagonal structure of the matrices, as the second derivative in x was defined with 4^{th} order compact scheme [89, 90]. The uniqueness of the presented mono domain solver is however not just in comparison with previous MFLOPS-3D code version, but also in comparison with what is usually applied in the codes. The reason for this is that to our knowledge, there are hardly any codes which use compact finite differences and eigendecomposition in all three directions, raising such a direct solver. An example of similar approach, that is, eigendecomposition in three directions, can be found in [112]. Discretization is however done with spectral methods. Otherwise works in [104, 87] seem to offer closest examples to here presented approach. Both works feature compact finite differences in two directions and solver using eigendecomposition applied to corresponding terms in Laplace operator. Hence equations analogous to (3.62) are used to obtain solutions in points, but with one less eigenvalue. Their derivation is also simpler than the one demanded here. Furthermore, in [87], calculations are 2D, while in [104], 3D domain is used. Fourier extensions are used in spanwise direction there. As a consequence, equation (3.62) is in a point applied according to the amount of wave numbers. Other examples

where direct solvers were applied with compact schemes were found, but LU decompsition was used instead of the here presented approach. These are works in [105] and [107]. Both use compact schemes in only two directions. It was therefore concluded that the mono domain solver as presented here is indeed a unique feature of MFLOPS-3D code.

4.2 Multi domain solver

In order to connect the solutions from sub domains to the whole computational domain, the MFLOPS-3D code uses an approach based on influence matrix technique. This is in the literature referred to also as Schurr complement matrix [106] and continuity influence matrix [87, 113]. This technique enables one to obtain continuous solutions on the sub domain interfaces without the need for overlapping and numerous uses of mono domain solver. It is also known to have good scalability and accuracy [113]. Solutions obtained with it also respect the divergence free condition in incompressible flows. Influence matrix technique represents the core of the multi domain solver and will be here described with the one and two dimensional example, much like in [114]. Similar explanation is also given in [113]. One and two dimensional cases are given as the explanation is with both more complete.

The one dimensional case gives the basis on which the influence matrix technique works. Suppose we have a 1D domain with $y \in [d, e]$, where a general form of a system of linear equations is given as (3.63). In it, $\mathbf{G}$ represents coefficient matrix resulting from discretization, V is the vector of unknowns and RHS the right hand side vector. Dirichlet boundary conditions are chosen with $V(d) = g_d$ and $V(e) = g_e$.

$$\mathbf{G}V = RHS \tag{3.63}$$

The domain $[d, e]$ can be split into two domains, namely $[d, c]$ and $[c, e]$. In order to use influence matrix to form multi domain method, one first needs to find the solution of the systems of equations given by (3.63) in both sub domains for the case of unit disturbance on the interface. Therefore two systems of equations, given with (3.64) and (3.65), have to be solved. They are referred to as first solution in influence matrix technique, which is also denoted by the subscript 1. The superscripts, on the other hand, define the domain number. It can be said that the first solution gives the response of the system of equations in both sub domains to the unit disturbance on the interface.

$$\mathbf{G}V_1^I = 0 \text{ with } V_1^I(d) = 0 \text{ and } V_1^I(c) = 1 \tag{3.64}$$

$$\mathbf{G}V_1^{II} = 0 \text{ with } V_1^{II}(c) = 1 \text{ and } V_1^{II}(e) = 0 \tag{3.65}$$

Contrary to the first solution, the second solution is obtained by solving the two systems of equations coming directly from (3.63) and with homogeneous boundary condition (0 values) on the interface. The two systems of equations are given in (3.66) and (3.67). The mentioned Dirichlet boundary conditions are used on the corresponding domain boundaries.

$$\mathbf{G}V_2^I = RHS \text{ with } V_2^I(d) = g_d \text{ and } V_2^I(c) = 0 \tag{3.66}$$

$$\mathbf{G}V_2^{II} = RHS \text{ with } V_2^{II}(c) = 0 \text{ and } V_2^{II}(e) = g_e \tag{3.67}$$

Having obtained first and second solutions in both sub domains, the fact that the system of equations (3.63) is a linear one gives the basis on which to continue. As the intention is to have a continuous solution over the interface, equal values on both sides of the interface should be found first. Applying the mentioned linearity, equations (3.68) and (3.69) can be written for the points on both sides of the interface and should have equal value. The constant γ multiplying the first solution values is used to ensure this.

$$\text{domain I: } V^I = V_2^I + \gamma V_1^I \tag{3.68}$$

$$\text{domain II: } V^{II} = V_2^{II} + \gamma V_1^{II} \tag{3.69}$$

However, γ is not known. In order to find it, the equality of the first derivative across the interface is imposed, which ensures also the continuity of the solutions. This leads to the equation (3.70), which gives the solution for γ.

$$\gamma = \frac{\left(\frac{\partial V_2^{II}}{\partial y} - \frac{\partial V_2^{I}}{\partial y}\right)}{\left(\frac{\partial V_1^{I}}{\partial y} - \frac{\partial V_1^{II}}{\partial y}\right)} \tag{3.70}$$

As γ is now known, and with it also the value of V on the interface, the final solution of (3.63) in both domains is found by solving (3.66) and (3.67) with γ imposed instead of homogeneous boundary condition on the interface.

Although the presented $1D$ case shows the basis of influence matrix use, it does not actually include the influence matrix as used in $2D$ or $3D$ cases. The main difference is that in the latter, more points are present on an interface. Therefore the first solution is more complex, which greatly affects the solution of interface values given before by a simple equation (3.70). A $2D$ case is used here to show this. The equation (3.63) is considered again, but in a domain with $x \in [a, b]$ and $y \in [d, e]$. Dirichlet boundary conditions are applied in this case too. These conditions are for x and y given in (3.71) and (3.72).

$$V(x, d) = f_d(x) \ ; \ V(x, e) = f_e(x) \tag{3.71}$$

$$V(a, y) = f_a(y) \ ; \ V(b, y) = f_b(y) \tag{3.72}$$

If the domain is divided into two sub domains at $y = c$, where $d < c < e$, the first solution, now given with systems in (3.73) and (3.74), has to be done N times, where N is the number of interface points without the edge or boundary points (on which V is known). Each time (3.73) and (3.74) are solved, all points on the interface except the point in interest m have 0 values. This is denoted with δ_m variable. In such a way, the effect of unit disturbance for each point on the interface is obtained for the system of equations (3.63) in both sub domains.

$$\mathbf{G}V_1^I = 0 \text{ with } V_1^I(x, d) = V_1^I(a, y) = V_1^I(b, y) = 0 \text{ and } V_1^I(x, c) = \delta_m \tag{3.73}$$

$$\mathbf{G}V_1^{II} = 0 \text{ with } V_1^{II}(x, e) = V_1^{II}(a, y) = V_1^{II}(b, y) = 0 \text{ and } V_1^{II}(x, c) = \delta_m \tag{3.74}$$

Having first solution, one can proceed to the second solution, now given with (3.75) and (3.76). This is done in same manner as in $1D$ case, namely only once.

$$\mathbf{G}V_2^I = RHS \text{ with}$$
$$V_2^I(x,d) = f_d(x),\ V_2^I(a,y) = f_a(y),\ V_2^I(b,y) = \ f_b(y) \text{ and } V_2^I(x,c) = 0 \tag{3.75}$$

$$\mathbf{G}V_2^{II} = RHS \text{ with}$$
$$V_2^{II}(x,e) = f_e(x),\ V_2^{II}(a,y) = f_a(y),\ V_2^{II}(b,y) = \ f_b(y) \text{ and } V_2^{II}(x,c) = 0 \tag{3.76}$$

What remains to be done at this point is to obtain interface values. It is here where another main difference between $1D$ and 2 or $3D$ cases appears. As linearity of the system of equations (3.63) still holds, equations (3.68) and (3.69) can be used again. But since there are multiple points on the interface and a unit disturbance in a certain point affects all points differently, finding the values of V on the interface is more complicated. Equations like (3.68) and (3.69) are written for each point on the interface, but with an important difference that γ, V_2^I and V_2^{II} are now vectors, composed from values in interface points, and $\mathbf{V}_1^I$ or $\mathbf{V}_1^{II}$ are matrices. In these, each column from first to last represents a unit disturbance in a corresponding point. Hence simple equations (3.68) and (3.69) become systems of equations as given in (3.77) and (3.78).

$$\text{domain I: } V^I = V_2^I + \mathbf{V}_1^I\gamma \tag{3.77}$$

$$\text{domain II: } V^{II} = V_2^{II} + \mathbf{V}_1^{II}\gamma \tag{3.78}$$

Like in $1D$ case, we use the fact that the values V on both sides of the interface are equal and continuity of first derivative across the interfaces has to be satisfied. Equation (3.79) follows from this derivative equality. R_r in it is a vector of differences between first derivatives of second solution as defined in (3.80). $\mathbf{R}$ is similar to R_r in including the differences between first derivatives, but since it refers to these differences regarding first solution, it is a matrix with N columns. Its definition is given in (3.81).

$$\mathbf{R}\gamma = R_r \tag{3.79}$$

$$R_r = [R_r^1, R_r^2, ...R_r^i, ..., R_r^N]^T \text{ where } R_r^i = \left(\frac{\partial V_2^{II}}{\partial y} - \frac{\partial V_2^I}{\partial y}\right)_i \tag{3.80}$$

$$\mathbf{R} = [R_1, R_2, ...R_i, ...R_N] = \frac{\partial \mathbf{V}_1^I}{\partial y} - \frac{\partial \mathbf{V}_1^{II}}{\partial y} \tag{3.81}$$

$\mathbf{R}$ is also the influence matrix, on which the multi domain technique bases. In order to obtain γ, this matrix needs to be inversed. The solution for γ including the inversed $\mathbf{R}$ is given in (3.82) and it is easily seen that it is comparable but much more complex than (3.70) used in $1D$ case.

$$\gamma = \mathbf{R}^{-1}R_r \tag{3.82}$$

With the known γ one can now obtain the final solution for V by solving (3.75) and (3.76) again, but this time with values of γ imposed on the interface instead of the homogeneous boundary condition.

The presented influence matrix technique forms the multi domain method used in MFLOPS-3D. It can be noted that it sets a limitation for sub domains to have same number of interface points across an interface. Which then constraints amount of points to be used in a certain sub domain. Following comparison of the equation (3.63) and equations (3.51), (3.52), the LHS of equations solved in MFLOPS-3D is constant. Consequently the first solution has to be done only once, at the start of a calculation, and the influence matrix can then be formed. This is very useful, as obtaining the first solution is a computationally heavy procedure, even though direct mono domain solver is used for it. The same mono domain solver (which is in MFLOPS-3D used for all mentioned solutions in sub domains) is then used only twice to obtain a certain solution in whole domain. That is, once in order to obtain the second solution and then finally to obtain the good solution after interface values are known. Combined with the good scalability and accuracy of influence matrix technique [113], this is the key on which the possibly better computational performance could be obtained compared to techniques which use overlapping sub domains (like Schwartz). The two features are also the reason why the technique was chosen to pursue fast DNS simulations in MFLOPS-3D.

However, the actual performance of the code is determined by the solution of equation (3.82), that is, by the solution of interface values with the influence matrix (this will be often referred to as influence matrix solution). The influence matrix size depends on the size of the computational domain and number of sub domains. The more sub domains one uses and the more points that they have, the bigger the influence matrix will be. And typically, the influence matrices will be big, sparse and non-symmetric. Furthermore, influence matrix is in MFLOPS-3D applied as 3D multi domain method, to interfaces in all three directions. This is an important and unique feature as to our knowledge, other codes use influence matrix at most as 2D multi domain method. Examples are the referred works in [87, 113, 114]. In [87], it is mentioned that an extension to 3D is supposed to be done with assumption of one homogeneous direction with spectral or pseudo-spectral methods used in it. This would therefore reduce a 3D problem to a series of 2D ones, meaning the influence matrix technique can remain a 2D multi domain method. Work in [113] is an example of such an extension (although an indirect one, since spectral methods are used in all directions). Same cannot be done in this case, as compact finite differences are used in all three directions, for which the reasons were given in section 3.2. Consequently here presented and used influence matrix technique is a unique multi domain method. However, its application in 3D makes influence matrix $\mathbf{R}$ dimensions increase considerably, meaning that it becomes too huge to be inversed directly. Direct inversion is on the other hand the approach used in [87, 113].

It therefore follows that solution for γ cannot be obtained directly following equation (3.82). Instead, an iterative approach is applied using system in equation (3.79). Which one, depends on the boundary conditions. Since system of equations for $\vec{v}^*$ from (3.51) uses Dirichlet boundary conditions, Krylov solver with GMRES (Generalized Minimal Residual Method) is used. In the case of solving for Φ with (3.52), where homogeneous von Neumann conditions are used, Krylov solver with hierarchical GMRES is applied. The reason for this comes from the fact that with such boundary conditions, the solution for Φ represents Poisson-Neumann problem, which is well known to be a singular problem [87, 104]. Issues with null eigenvalues appear, which are in the case of using influence matrix for multi domain method not revealed in usual manner [87, 113]. This is also the reason why the issues of Poisson-Neumann problem were skipped in the presentation of direct mono domain solver, although they are known to otherwise pose considerable difficulties [87, 104, 107]. The Poisson-Neumann problem singularity is namely not an issue anymore for mono domain solver when the influence matrix technique is used. This follows from the shown explanation of this technique, where it is clear the mono domain solver has to always deal with

Dirichlet boundary conditions imposed on interfaces. Only sub domains on domain boundaries have also von Neumann boundary conditions imposed, but just on those boundaries. Nevertheless, the issues caused by Poisson-Neumann singularity are still present and in this case transferred to the level of influence matrix [87, 113]. This will be discussed later in more detail as it had two consequences. Firstly, it affected the development of the algorithm for cavitating flow simulations and secondly, it was found to cause lower performance already for the starting version of the MFLOPS-3D code used in this work. However, it can be said that if the problem is well handled, the solution can be defined only up to an arbitrary additive constant [87, 107]. Which means that the absolute value of pressure cannot be obtained by default when such boundary conditions are used. This is also consistent with the influence of pressure in incompressible flows, since only the pressure gradient or difference is important in them.

The singularity of Poisson-Neumann problem and influence matrix zero eigenvalues caused by it are in the case of direct influence matrix inversion in [87, 113] treated by ensuring that zero eigenvalues divide zero values. Same is not done with here used iterative solver. Both mentioned solvers for (3.79), one for $\vec{v}^*$ and other for Φ, are implemented into the code using PETSc toolkit, a suite of different procedures for solving various systems of equations [115]. Both use block Jacobi preconditioners and incomplete LU factorization as a method to treat the blocks. The reason is that the influence matrix $\mathbf{R}$ is built from blocks as there are multiple interfaces in computational domain [113]. Since solutions for γ are obtained iteratively, relative convergence criteria is applied (*ksp_rtol* option to define relative decrease in residual norm [115]) and set to be 10^{-8} for $\vec{v}^*$ case and 10^{-5} for Φ. MFLOPS-3D code offers three options to handle issues caused by singularity of Poisson-Neumann problem and null eigenvalues, which affect the influence matrix solution for Φ. The first leaves PETSc to deal with these issues, second fixes a value in one of interface points and the last one introduces a small perturbation in place of constant a in equation (3.53). First option was used as it was found to offer fastest and most accurate results.

Typically, the solutions for $\vec{v}^*$ components demand only a few iterations of the Krylov solver (often only one or two). The solution for Φ was found much more demanding, which is also the reason for chosen lower convergence criteria. Number of its iterations varies greatly and depends on the size of the computational system and the calculated case itself. This is also the main point of before mentioned lower performance of MFLOPDS-3D code caused by Poisson-Neumann problem. Performance analysis of this multi domain method with examples showing how number of iterations changes in different cases and comparison with mono domain calculations is given in section 5 of chapter 6.

To conclude, two points can be written. One is that as always, multi domain method demands communication between sub domains. Following the influence matrix use, each sub domain is assigned to one processor. The communication between them is handled by classical message passing library (MPI). The other is that from the given description and reasons, it follows the multi domain method used in MFLOPS-3D is a very specific tool, and like the mono domain solver, not often used in other codes. Hence it was also said in the introduction that both form a unique mix.

4.3 Boundary conditions application

The boundary conditions are in MFLOPS-3D code implemented through the mono domain solver with the use of compact finite difference schemes. With the use of non overlapping influence matrix technique this causes that the interface values for first, second and final solutions, presented in previous subsection, are applied in the same manner as boundary conditions. Where boundary conditions can be either Dirichlet or von Neumann type, the interface values are always applied as Dirichlet boundary conditions, thus also evading the Poisson-Neumann problem singularity on the level of mono domain solver.

In order to illustrate how the boundary conditions are implemented, the discretization of Laplace operator terms should be a bit more explained. These terms or second derivatives can be each viewed as **B** matrix which was mentioned in the subsection about the mono domain solver and used to give an example for diagonalisation in one direction. The implementation of boundary conditions is done at the level of this matrix, which is obtained for a certain term in Laplace operator in same manner for all three directions. Compact schemes are used to define second derivatives in all points where the variable values are unknown, that is, in all points except on boundaries. If there are N points in a certain direction, $N-2$ schemes are written, where a general scheme for point i (on non uniform grid) is given with equation (3.83). Equation follows from before given general equation for first derivative scheme (3.42), with less points in the implicit and explicit stencil listed here. The schemes for points next to interfaces are forward or backward and not central as in the shown example. These forward or backward schemes do not include interface points in the implicit stencils of points, they include them only in the explicit ones. If the schemes obtained on a line are then put into a system of equations as shown in section 3.2.4 with equation (3.44), matrix form given in (3.84) can be written. Both matrices used in it have $N-2$ rows, but **A** which multiplies implicit stencils of points has $N-2$ columns while **b** has N columns as it multiplies explicit stencils. As written, these include also interface values. Coefficients forming **A** and **b** can be referred to as implicit and explicit coefficients, like their corresponding stencils of points.

$$\alpha_1 f''_{i-1} + f''_i + \alpha_2 f''_{i+1} = b_1 f_{i-2} + b_2 f_{i-1} + b_3 f_i + b_4 f_{i+1} + b_4 f_{i+2} \tag{3.83}$$

$$\mathbf{A}[f]'' = \mathbf{b}[f] \tag{3.84}$$

The above given matrix formulation is then used to define certain discretized second derivative term in Laplace operator. The definition of schemes for one line is actually enough to define a global term for certain second derivative in Laplace operator on the level of sub domain, which is a consequence of distances between neighbouring points in a certain direction in sub domain being equal in all mesh lines. How this is done will be explained in the following chapter. The definition of the term or mentioned matrix **B** for a certain direction finally follows from inversion of **A** and its multiplication with **b**. Since matrix **b** has N columns, vector $[f]$ includes also values on the boundaries of the domain. It is here where the boundary conditions are applied. Or interface values, since this approach is used by mono domain solver. The boundary conditions are applied by multiplication of values in first and last columns in **B** with f_1 or f_N and inclusion of these products on the right hand side of the system of equations to be solved. This is done for all lines in certain direction. It should be noted that because of this, the diagonalisation shown in subsection 4.1 is in practice only performed on **B** which does not include the first and last columns. Correspondingly for 3D eigendecomposition, the matrices for discretized second

derivative terms D_x^2, D_x^2 and D_z^2 are also eigendecomposed without their first and last columns included.

The given description only explains how Dirichlet boundary conditions are applied, since vector $[f]$ includes values of variable on boundaries or interfaces, but not also values of first derivatives. However, there is no limit why this vector should not include also the values of first derivatives. Which would mean that von Neumann boundary conditions can be applied in the same manner as Dirichlet boundary conditions. The inclusion of derivative values in the mentioned vector is done with help of compact schemes. The schemes for second derivatives can be written not only with variable values included on the explicit, right hand side, but also with the values of first derivatives. Namely, for a certain derivative of order K, explicit side can include at most derivatives of $K-1$ order. To obtain such schemes, the system of equations (3.36) – (3.41), from which the implicit and explicit coefficients are obtained, must include Taylor series for variables, first and second derivatives. Example for a system of equations for a forward scheme is given with equations (3.85) – (3.89). The series for second derivatives are written for two points next to the boundary as implicit stencils cannot include boundary points. Taylor series for variables and first derivatives give explicit coefficients. It should be stressed that in order to include von Neumann conditions in compact schemes, Taylor series for first derivatives are written only for points on boundaries. Which is also the reason why in equations (3.85) – (3.89) there is only equation (3.87) which shows such a series.

$$f''(2) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^3 f}{\partial x^3}(x_2 - x_2) + \frac{\partial^4 f}{\partial x^4}\frac{(x_2-x_2)^2}{2!} \tag{3.85}$$

$$f''(3) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^3 f}{\partial x^3}(x_3 - x_2) + \frac{\partial^4 f}{\partial x^4}\frac{(x_3-x_2)^2}{2!} \tag{3.86}$$

$$f'(1) = \frac{\partial f}{\partial x} + \frac{\partial^2 f}{\partial x^2}(x_1 - x_2) + \frac{\partial^3 f}{\partial x^3}\frac{(x_1-x_2)^2}{2!} + \frac{\partial^4 f}{\partial x^4}\frac{(x_1-x_2)^3}{3!} \tag{3.87}$$

$$f(2) = a_0 + \frac{\partial f}{\partial x}(x_2 - x_2) + \frac{\partial^2 f}{\partial x^2}\frac{(x_2-x_2)^2}{2!} + \frac{\partial^3 f}{\partial x^3}\frac{(x_2-x_2)^3}{3!} + \frac{\partial^4 f}{\partial x^4}\frac{(x_2-x_2)^4}{4!} \tag{3.88}$$

$$f(3) = a_0 + \frac{\partial f}{\partial x}(x_3 - x_2) + \frac{\partial^2 f}{\partial x^2}\frac{(x_3-x_2)^2}{2!} + \frac{\partial^3 f}{\partial x^3}\frac{(x_3-x_2)^3}{3!} + \frac{\partial^4 f}{\partial x^4}\frac{(x_3-x_2)^4}{4!} \tag{3.89}$$

After obtaining implicit and explicit coefficients, compact scheme including first derivative on the boundary can be formed. For the presented case, the scheme is given with equation (3.90).

$$f_2'' + \alpha_1 f_3'' = b_1 f_1' + b_2 f_2 + b_3 f_3 \tag{3.90}$$

The presented scheme gives an example for a forward scheme. Same approach is taken to form backward scheme and include derivative values on the boundaries on the opposing end of a mesh line. There are at least four schemes in each mesh line which have explicit terms taken also from boundaries or interfaces. If von Neumann boundary conditions are used, they are written in presented manner. Since the schemes used to implement von Neumann boundary conditions differ from others only in inclusion of first derivatives, matrices $\mathbf{A}$, $\mathbf{b}$ and following $\mathbf{B}$ are formed in the same manner as before which makes the application of von Neumann boundary conditions for a system of equations to be the same as application of Dirichlet boundary conditions.

The whole application of boundary conditions on the mono domain solver level through presented approach finally also means the explicit and implicit stencils of points, which define the second derivatives in Laplace operator, are not exactly equal to those which define general first and second derivatives (given in table 3.1). They are given in table 3.2, where they distinguished, as before, according to their order of accuracy in central schemes. The main difference from general ones is that the implicit stencils never include the boundary points, meaning that there are also no compact schemes defined in those points. Therefore these are also skipped in the notation

in the first column. The convergence tests results showing the order of accuracy for such second derivatives are given in appendix for the schemes listed under 4^{th} and 8^{th} order of accuracy.

Table 3.2: Implicit and explicit stencils of compact schemes as used in MFLOPS-3D for second derivatives in Laplace operator for available orders of accuracy.

order of accuracy	8^{th}		6^{th}		4^{th}		2^{nd}	
point	impl	expl	impl	expl	impl	expl	impl	expl
2^{nd} and n-1	3	6	1	6	1	4	0	3
3^{rd} and n-2	1	5	1	5	1	5	0	3
4:n-3	4	5	2	5	2	3	0	3

5 Mapping

When finite differences are used for spatial discretization, issues emerge how to perform simulations in various geometries. Finite difference methods are not as flexible in this aspect as finite volume methods, and an approach has to be chosen on how to deal with simulations of flow in various geometries. One option is to write the equations to be solved in curvilinear coordinates. The other is to change or map the geometry of interest into a rectangular geometry, in which usual Cartesian coordinate system can be used. This method is referred to as mapping and is also applied in the MFLOPS-3D code for simulations of flows in various geometries. The purpose of this section is to describe it and its effects on the solution procedure.

When considering mapping, one needs to define two sets of coordinates. One represents physical coordinates, while the other represents coordinates on rectangular transformed mesh. Physical coordinates are here barred (eg $\bar{x}$), while transformed or Cartesian are given without a bar (eg x). Connections between two sets of coordinates are given with mapping equations. In MFLOPS-3D the only coordinate which is actually transformed is $\bar{y}$, therefore the mapping equations can be generally given as in (3.91).

$$t = \bar{t} \; ; \; x = \bar{x} \; ; \; y = f\left(\bar{y}(\bar{\eta_1}(\bar{x}), \bar{\eta_2}(\bar{x}))\right) \; ; \; z = \bar{z} \tag{3.91}$$

It can be noted that $\bar{y}$ changes because of $\bar{\eta_1}$ and $\bar{\eta_2}$, which are functions of $\bar{x}$. $\bar{\eta_1}$ and $\bar{\eta_2}$ define lowest and highest local limit of physical domain. In a venturi, $\bar{\eta_1}$ would describe the lower and $\bar{\eta_2}$ the upper wall. $\bar{\eta_1}$ and $\bar{\eta_2}$ are depicted also on figure 3.1 where the transformation from a venturi to rectangular geometry can be seen together with presentation of physical and transformed coordinates. It can also be noted that direction of $\bar{x}$ is not everywhere same as direction of x but it strictly follows the geometry of the channel. Yet the two coordinates are taken as equal in equation (3.91). The reason is in alignment of mesh lines in y or $\bar{y}$, therefore positions of points in $\bar{x}$ or x coordinates are same.

With known connections between transformed and physical geometry it is possible to define derivatives in physical coordinates through the derivatives on mapped, Cartesian coordinates. This is shown in equations (3.92) to (3.94).

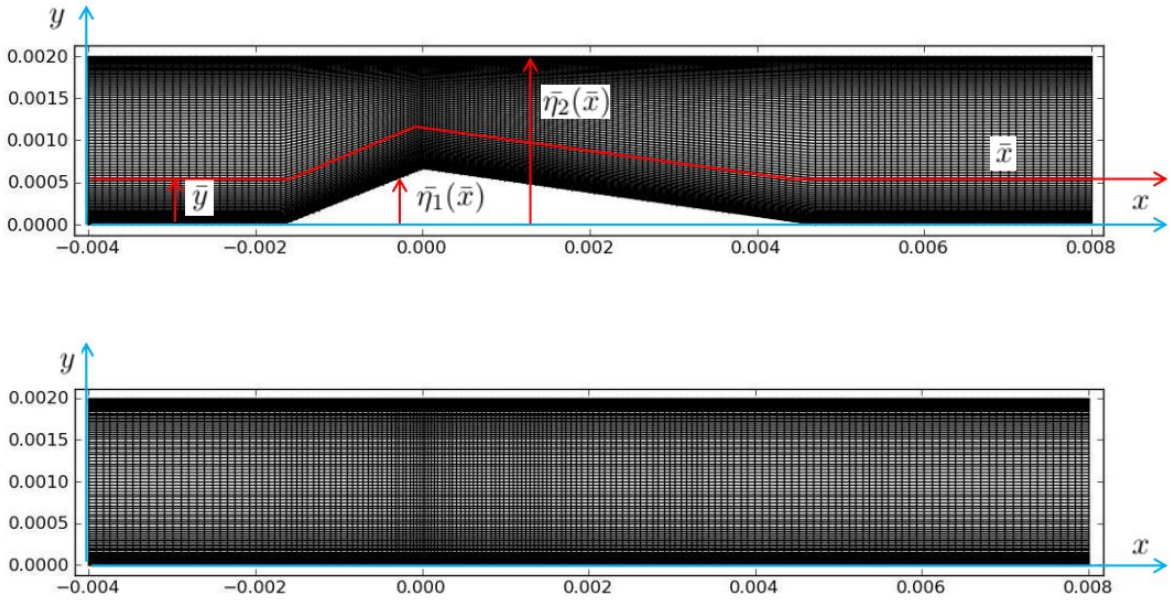


Figure 3.1: The venturi geometry above is mapped into rectangular geometry below. Blue coordinate lines represent the Cartesian or mapped coordinates and the red ones show the physical ones. Physical coordinates are for illustration purposes not shown in their origin, which is the same as the one for Cartesian coordinates.

$$\frac{\partial}{\partial \bar{x}} = \frac{\partial}{\partial x}\frac{\partial x}{\partial \bar{x}} + \frac{\partial}{\partial y}\frac{\partial y}{\partial \bar{x}} \tag{3.92}$$

$$\frac{\partial}{\partial \bar{y}} = \frac{\partial}{\partial x}\frac{\partial x}{\partial \bar{y}} + \frac{\partial}{\partial y}\frac{\partial y}{\partial \bar{y}} \tag{3.93}$$

$$\frac{\partial}{\partial \bar{z}} = \frac{\partial}{\partial z} \tag{3.94}$$

The ability to write derivatives in physical coordinates through derivatives in Cartesian coordinates enables direct use of finite differences as defined in MFLOPS-3D. Moreover, since the derivative values are given for physical coordinates, variables do not need any transformation. The solutions obtained with the code are therefore final solutions. This however demands that the second derivatives have to be changed as well, especially as they form the Laplace operator. They are for $\bar{x}$ and $\bar{y}$ coordinates given in (3.95) and (3.96). It should be mentioned that equations (3.92)−(3.96) have a lot of terms which include derivatives of mapped coordinates in regards to physical ones. These are here not specifically defined but follow from knowing the connections given in (3.91).

$$\frac{\partial^2}{\partial \bar{x}^2} = \frac{\partial^2}{\partial x^2}\left(\frac{\partial x}{\partial \bar{x}}\right)^2 + \frac{\partial^2}{\partial y^2}\left(\frac{\partial y}{\partial \bar{x}}\right)^2 + 2\frac{\partial x}{\partial \bar{x}}\frac{\partial y}{\partial \bar{x}}\frac{\partial^2}{\partial x \partial y} + \frac{\partial^2 x}{\partial \bar{x}^2}\frac{\partial}{\partial x} + \frac{\partial^2 y}{\partial \bar{x}^2}\frac{\partial}{\partial y} \tag{3.95}$$

$$\frac{\partial^2}{\partial \bar{y}^2} = \frac{\partial^2}{\partial x^2}\left(\frac{\partial x}{\partial \bar{y}}\right)^2 + \frac{\partial^2}{\partial y^2}\left(\frac{\partial y}{\partial \bar{y}}\right)^2 + 2\frac{\partial x}{\partial \bar{y}}\frac{\partial y}{\partial \bar{y}}\frac{\partial^2}{\partial x \partial y} + \frac{\partial^2 x}{\partial \bar{y}^2}\frac{\partial}{\partial x} + \frac{\partial^2 y}{\partial \bar{y}^2}\frac{\partial}{\partial y} \tag{3.96}$$

The given definitions of second derivatives are not directly used for Laplacian operator in the solver. To explain why, it should be first mentioned that the divergence and Laplacian operator definitions also change when mapping is used. They can be generally given as in (3.97) and (3.98).

$$\bar{\nabla} = \nabla + \vec{G}_\eta \tag{3.97}$$

$$\bar{\Delta} = \Delta + L_\eta \tag{3.98}$$

The goal in writing the operators in such manner is that $\vec{G}_\eta$ and L_η are the operators including the terms which appear additionally to the Cartesian derivatives in different mapped geometries. Therefore they present the connection between the two grids. If the mapping equations in (3.91) are considered, $\vec{G}_\eta$ and L_η are defined as given in (3.99) and (3.100)−(3.103), respectively. It can be seen that many terms from long definitions in equations (3.95) and (3.96) disappear. This is a result of alignment of y, $\bar{y}$ mesh lines and coordinates x, $\bar{x}$ being independent coordinates. For instance, all terms bar the second one in (3.96) are zero.

$$\vec{G}_\eta = \left(\frac{\partial}{\partial y}\frac{\partial y}{\partial \bar{x}}, \frac{\partial}{\partial y}\left(\frac{\partial y}{\partial \bar{y}} - 1\right), 0 \right) \tag{3.99}$$

$$L_\eta = L_{\eta,x} + L_{\eta,y} + L_{\eta,z} \tag{3.100}$$

$$L_{\eta,x} = \frac{\partial^2}{\partial y^2}\left(\frac{\partial y}{\partial \bar{x}}\right)^2 + 2\frac{\partial x}{\partial \bar{x}}\frac{\partial y}{\partial \bar{x}}\frac{\partial^2}{\partial x \partial y} \tag{3.101}$$

$$L_{\eta,y} = \frac{\partial^2}{\partial y^2}\left(\left(\frac{\partial y}{\partial \bar{y}}\right)^2 - 1\right) \tag{3.102}$$

$$L_{\eta,z} = 0 \tag{3.103}$$

The L_η operator is especially important for the solver and solution procedure. The reason is the presented combination of mono and multi domain solution methods. These at first demand constant left hand side of a system of equations. And secondly, applied eigendecomposition in mono domain solver, done separately per each direction, requires each of Laplace operator terms to include only discretization in certain direction and its multiplication by unity. The L_η operator violates both demands and has to be therefore treated explicitly. It then follows that the definition of L_η has a very important benefit. It separates the constant, cartesian parts of the Laplacian operators from others and therefore enables direct use of existing solver in MFLOPS-3D. But the consequence of Laplacian operator being split into two parts is now that the equations to obtain $\vec{v}^*$ solution, equation (3.51), and Φ solution, equation (3.52), have to be adapted and are given in (3.104) and (3.105), respectively.

$$\left(\Delta - \frac{3Re}{2\Delta t}\right)\vec{v}^* = \left(\frac{-4\vec{v}^n + \vec{v}^{n-1}}{2\Delta t} + ((\vec{v}\cdot\nabla)\vec{v})^{n,n-1} + \left((\vec{v}\cdot\vec{G}_\eta)\vec{v}\right)^{n,n-1}\right)Re - L_\eta \vec{v_e}^* \tag{3.104}$$

$$\Delta\Phi = \frac{3}{2\Delta t}\left(\nabla\cdot\vec{v}^* + \vec{G}_\eta\cdot\vec{v}^*\right) - L_\eta\Phi_e \tag{3.105}$$

The introduction of L_η to the right hand side and explicit treatment of Φ and $\vec{v}^*$ (as Φ_e and $\vec{v_e}^*$) in them means that the algorithm used in the MFLOPS-3D code performs iterative steps to obtain solutions of these two variables. Thus the whole solution procedure becomes longer. The convergence check which is used in MFLOPS-3D code for these iterations is not a usual one. It

was namely found that the most effective convergence check, in order to not perform too many iterations and still obtain accurate solutions, is a check where highest relative differences between results of two iterations should be smaller than size of a time step. The convergence criteria is mathematically presented with equation (3.106).

$$\frac{|\Phi - \Phi_e|_{max}}{|\Phi|_{max}} < \Delta t \tag{3.106}$$

As use of L_η on the RHS of equations enables their LHS to remain constant, it also allows for global definition of Laplacian Δ terms in each sub domain with the use of only one mesh line in each direction. The reason is simply in the fact that Δ represents second derivatives on mapped, Cartesian mesh, and the neighbouring points on all lines for a certain direction in such mesh have always same distances between each other.

Mapping has one other important effect on calculations. It also demands adaptation of boundary conditions. Since the solver operates in Cartesian coordinates but the solutions of variables are still given in physical coordinates, the boundary conditions should be given for physical coordinates as well. In case of velocity boundary conditions this does not demand some specific approach. Only derivatives of intermediate variable Φ, used for boundary conditions of velocities in tangential directions on the boundaries, are written with derivatives given in (3.92) − (3.94). And same has to be respected for the derivatives in the advection boundary condition on the domain outlet. This boundary condition will be given in the following chapter.

On the contrary, the boundary conditions for Φ are not so simple to apply in mapping. Before continuing, it has to be firstly stated that MFLOPS-3D code was mainly used for venturi type geometries. The following explanation is given with this in mind. The use of von Neumann boundary conditions for Φ is based on equality assumption between $\vec{v}^*$ and $\vec{v}$ in normal direction on the boundaries. Because the solver solves the system of equations using Cartesian coordinates it also demands von Neumann boundary conditions to represent the values in normal direction in Cartesian coordinates. As a result, values for Φ gradient in Cartesian coordinates have to be adjusted to ensure the mentioned equality (or zero value of Φ gradient) in normal directions on the boundaries. The normal gradient on inlet and outlet is usually given simply with $\frac{\partial}{\partial \bar{x}}$ as inlet and outlet are preferably aligned with y while the same does not always apply with $\bar{x}$ and x near these boundaries. The von Neumann boundary condition for x direction in the solver is therefore given with derivative which follows from equations (3.107) and (3.108).

$$\frac{\partial \Phi}{\partial \bar{x}} = \frac{\partial \Phi}{\partial x}\frac{\partial x}{\partial \bar{x}} + \frac{\partial \Phi}{\partial y}\frac{\partial y}{\partial \bar{x}} = 0 \tag{3.107}$$

$$\frac{\partial \Phi}{\partial x} = -\frac{\partial \Phi}{\partial y}\frac{\partial y}{\partial \bar{x}} \tag{3.108}$$

Because the lower and upper walls are usually not aligned with x coordinate it is very difficult to write the von Neumann boundary condition for y direction. Some mathematical development has to be done to find the general expression for a derivative in the normal direction on these two boundaries and to extract from it the value in y direction. This general expression is given with (3.109), where $\bar{\eta}$ can be either lower or upper local limit of physical domain, depending on the boundary for which the derivative is defined.

$$\frac{\partial \Phi}{\partial y} = \frac{\frac{\partial \bar{\eta}}{\partial \bar{x}}\frac{\partial \Phi}{\partial x}}{\frac{\partial y}{\partial \bar{\eta}} - \frac{\partial y}{\partial \bar{x}}\frac{\partial \bar{\eta}}{\partial \bar{x}}} \tag{3.109}$$

Direction z does not demand any changes as mapping is not applied in it. Therefore the Cartesian derivative in it is equal to the derivative in physical coordinate and they are both zero.

It can be noted that changes done to von Neumann boundary conditions are based on using the derivative values of Φ variable itself. In order to make this possible, Φ variable in them is used explicitly. However, as iterations to obtain converged solution for Φ are done because of operator L_η, these values are constantly updated. Same holds for the values of $\vec{v}^*$ variables which are used in the advection boundary condition gradients.

Mapping is here presented in the manner as used in the scope of this thesis. For an additional explanation of its use, one car look into [89].

6 Conclusions

This chapter presents the MFLOPS-3D code, the numerical tool with which the desired DNS simulations of cavitating flows are to be performed. As it can be seen, the code differs from its actual initial version, where spectral and pseudo-spectral methods were used. These were replaced by compact finite differences in order to gain more flexibility when it comes to simulating flow in different geometries used in incompressible and cavitating flow experiments. Compact finite differences are the obvious choice as they offer accuracy and numerical dissipation similar to that of spectral methods. Because they are used in all three directions, the solution techniques were adapted as well. On the level of sub domains, eigendecomposition in all three directions is used, raising a direct mono domain solver. As it is shown, other codes were found to use at most eigendecomposition with compact finite differences in two directions. However, this does not mean there are actually no codes using such direct mono domain solver. But it seems to us there are none which apply it with compact finite differences in three directions and are supposed for use also in larger scale and more practical computational problems (various geometries).

For simulations on larger scales the codes need to be also parallelized. Use of compact finite differences encouraged the use of influence matrix technique as multi domain method since technique was shown to offer good scalability and accuracy. Importantly, the technique is applied as 3D multi domain method, which has not been remarked in any other code. The closest examples are works in [87] and [113] which use influence matrix as 2D multi domain method. Because of this difference, direct solver for influence matrix solutions used in those two works was not an option any more. The influence matrices namely became too big to be inversed, therefore iterative solvers were implemented. Performance of the code is set by performance of these solvers, which will be shown in a later chapter dealing with verification and performance results. It was nevertheless already hinted that performance is at the moment not as good as desired, for which the issues with Poisson-Neumann problem solution are responsible.

The code was used is spite of this as a tool to develop new algorithm suitable for codes which offer fast DNS simulations. One reason is that improving performance of the code was not the subject of this thesis. But it is a subject of ongoing research in our laboratory, where another thesis is currently conducted involving work on this problem. An attempt to improve the performance was done in the scope of this thesis nevertheless and will be shown in a later chapter. The second reason for using the code is the mentioned flexibility to perform simulations in various geometries and also on larger scales, enabled by the use of compact finite differences in all three directions and implemented mono and multi domain solvers. This ability is very important for cavitating flow simulations. The third reason is that though the code features a unique mix of mono and multi domain solvers, it still imposes same constraint as other codes which aim at

enabling fast DNS simulations with higher flexibility. Examples of such codes are mentioned works in [87, 113]. They both use a direct mono domain solver and influence matrix technique as multi domain method. It is because of the latter that the mentioned constraint is imposed. This constraint is the demand for a constant LHS of a certain solved system of equations. With it, the heaviest solution step, the described first solution in section 4.2, is done only once. Which then makes faster parallel simulations with influence matrix technique possible. The need for constant LHS also reversibly means that fast direct solver is the obvious choice on mono domain level.

Where the constant LHS does not present an issue for incompressible flows, it does present additional problems when cavitating flow simulations are considered. Therefore the use of MFLOPS-3D code to develop a new algorithm suitable for fast DNS simulations means that the algorithm is not specific only to this code, but can be applied to other codes meant for fast DNS as well.

Chapter 4

Development of the new algorithm for cavitating flow simulations

This chapter presents development of the new algorithm for cavitating flow simulations. As pointed in previous chapter, the algorithm was developed to be suitable for codes which enable fast DNS and offer higher flexibility regarding computational domain size and shape. New governing equations are firstly introduced with the explanation of their choice and the reasons why they can be solved with projection methods. This is followed with the presentation of the new algorithm, where the algorithm is at first presented through main equations that form projection method in it. This presentation shows encountered demands and their solutions in order for the used equations to be solved in appropriate way for existing codes which offer desired DNS simulations. Then, a more precise presentation of the algorithm versions follows, since these include different iterative loops and approaches, making them impossible to be explained at once.

As the whole algorithm development presentation is quite detailed, while MFLOPS-3D code was used as a basis for it, the end conclusions include a table to give a summary of the changes done in order to adapt the used code to cavitating flow simulations. As written, the applied methods for new algorithm can be used in other codes as well.

1 The governing equations

The chosen system of governing equations describing cavitating flows is a very well known and often used one as it forms the basis of single phase cavitation models explained in chapter 1, section 2.2. It is given with the Navier-Stokes momentum and continuity equations for homogeneous mixture phase and additional transport equation for vapour volume fraction α which includes source term S. This term governs creation and destruction of the gaseous phase. The NS momentum and continuity equations are given in (4.1) and (4.2) respectively. It can be seen that the continuity equation is used to simplify the momentum equation while otherwise, all terms from NS equations are included. The α transport equation is given in (4.3).

$$\rho\left(\frac{\partial\vec{v}}{\partial t}+(\vec{v}\cdot\nabla)\,\vec{v}\right)=-\nabla p+\nabla\cdot\left(\mu(\nabla\vec{v})+\mu(\nabla\vec{v})^{T}\right)-\frac{2}{3}\nabla\left(\mu(\nabla\cdot\vec{v})\right) \tag{4.1}$$

$$\frac{\partial\rho}{\partial t}+\nabla\cdot(\rho\vec{v})=0 \tag{4.2}$$

$$\frac{\partial \rho_v \alpha_v}{\partial t} + \nabla \cdot (\rho_v \alpha_v \vec{v}) = S \tag{4.3}$$

Since homogeneous mixture can be composed of multiple phases, it was decided to only include liquid and vapour phase and not include also non condensable gases, as is done for example in [33]. This then determines that the density and viscosity of the mixture phase are defined with equations given in (4.4). The subscripts l and v stand for liquid and vapour.

$$\rho = \alpha_v \rho_v + (1 - \alpha_v)\rho_l \ ; \ \mu = \alpha_v \mu_v + (1 - \alpha_v)\mu_l \tag{4.4}$$

1.1 The reasons for such governing equations

It was mentioned in section about numerical simulations of cavitating flows 2 in chapter 1 that the reasons why single phase cavitation modelling was chosen in this work will be given later. As the governing equations have been defined, it seems correct to refer to these reasons here. The decision for such governing equations is based on different results and observations. As one can see, the shown governing equations do not just correspond to single phase cavitation modelling, but also assume included phases to be incompressible and make use of additional transport equation for α. Therefore in total three groups of reasons should be given to explain why such cavitation modelling is applied.

1.1.1 Reasons for single phase modelling

Reasons for this kind of modelling are shortly given already when different cavitation modelling approaches were introduced. The fact is that nearly all of cavitating flow simulations are today performed with the use of such approach. Single phase modelling can be found in simulations performed in academic and industrial area, from simulations of simple geometries to complete turbines. Reason is in lower computational costs as only one set of governing equations needs to be resolved while still good physical description of cavitation is obtained. The proof for this can be easily seen in the vast amount of cases and models available for single flow modelling and lack of them in the case of two flow modelling. The lack is even more pronounced in the case of heterogeneous modelling.

It therefore follows that single phase modelling is the most practical approach to perform cavitating flow simulations. This is especially important for this work as hydrodynamic cavitation problems are the only ones considered. Using other approaches would mean adding a lot of computations to already very cumbersome DNS simulations. For instance, trying to follow two separate phases or maybe even all cavitation structures which appear in the flow with them would make DNS simulations practically impossible to be performed in the considered flow configurations. Finally, chosen approach towards DNS simulations also gives the most straightforward way to apply possible new observations and conclusions to nowadays most practically performed simulations of cavitating flows since it shares same cavitation modelling with them.

1.1.2 Reasons for inclusion of α transport equation

Before mentioning the reasons for inclusion of this equation, it should be stated that the here developed algorithm can be adjusted to accommodate also cavitation models, such as many barotropic ones, which directly apply phase transition description into NS governing equations. Therefore the algorithm could be applied to all cavitation models which use single phase modelling

and incompressible assumption of phases, although the focus is in this work given only to the models which include α transport equation.

The reasons why α transport equation was applied are in the generally better abilities it enables for cavitation modelling. Firstly, the equation introduces its convective character to description of cavitating flow. This allows inclusion of effects inertial forces have on cavities, leading to better description or modelling of elongation, detachment and drift of cavitation structures [5]. Secondly, models which directly link pressure change to α change and do not use α transport equation are not capable to account for baroclinic torque. This effect was in chapter 1 mentioned to be observed in different experiments and also simulations, but not explained. It is the effect caused by differences in gradients of pressure and density which lead to production of vorticity described with equation (4.5). This was found to be present in cavitating flows where experiments such as [60] show higher vorticity production in closure region of cavitation, where vortices with small (micro) bubbles in their center are formed. The importance of this vorticity inclusion in simulations was stated in [5]. Since ρ and p gradients are in models which omit α transport eqaution parallel, they lead to zero values and effect of baroclinic torque [5]. On the other hand, models with inclusion of α transport equation can accommodate for this effects. Moreover, it can be also restated that experimental findings in [53] show that locations of highest α and p fluctuations do not coincide. The models with direct link between these two variables cannot be expected to well predict such effects.

$$\omega = \nabla \frac{1}{\rho} \times \nabla p \tag{4.5}$$

Finally, an important reason to include the equation in question is also that the models which utilise it exhibit better stability, as found in [27].

1.1.3 Reasons for incompressible treatment of phases

The choice to treat phases as incompressible seems to be nowadays a bit too conservative regarding the newest experimental results and also numerical work. Recent experiments, where [116] has to be pointed out, show proof of a shock induced cavitation collapse. This hugely contributes to the explanation of cavitation cloud shedding, where up to now the described re-entrant jet was the main explanation mechanism. In numerical area, simulations as [117] show ability to capture this shock induced mechanism. Moreover, since development of inducers and pumps for rocket engines demands observation and prediction of cavitation in cryogenic liquids, where thermal effects are more pronounced and could even lead to cavitation retardation, a lot of work has been done in order to include or model thermal effects in cavitation. Examples of such cavitation models can be found in [3, 31, 39, 24, 25] and others. Nevertheless, it was for this work decided that the phases will be treated as incompressible, with constant densities and viscosities. This choice follows from past experience and results obtained with such modelling, chosen cavitating flow to be simulated and some theoretical results concerning influence of thermal effects on cavitation bubble behaviour. Finally, the fact that original MFLOPS-3D code was developed for incompressible flow simulations also had important part in the decision. In the following text, these reasons will be further discussed.

Firstly, the choice for incompressible treatment of phases was preferred because of ability of original code to simulate incompressible flow. It was decided that it is thus better to start with incompressible phases assumption, develop a new code and algorithm, and then, if all is performing in a wanted manner, possibly introduce methods for compressible phases treatment. Secondly,

as mentioned in chapter 2, the flow configuration or configurations to be simulated are based on experiments performed in the scope of a larger project. These are also presented in [64, 65]. In them, water at cca 20 $°C$ was used. This is close to more practical occurrence of hydrodynamic cavitation, encountered in water pumps, turbines, ship propellers etc. And importantly, cavitation at such temperatures and also geometries was up to now very successfully simulated with here chosen approach. Which is furthermore also used in many cavitation models, as can be seen in presentation of single phase models, where this characteristic is intentionally pointed out. Additionally, the statement about correctness of incompressible treatment at such conditions can be found in different literature. For example, in [3], it is mentioned that temperature variation is too small for cavitation occuring in cold water therefore it can be neglected. A more thorough explanation why thermal effects and compressibility can be neglected in such conditions is given in [8] on the basis of bubble dynamics. Same is stated also in [4]. In [14], incompressible treatment is supported also for the case of using fuel as a liquid.

Importantly, the flow configurations to be used in planned DNS simulations also support the choice for incompressible treatment of phases from one other point of view. Using DNS approach, it is preferred that the flow does not have very high Reynolds numbers. Higher Reynolds numbers namely mean higher turbulence and the need to use more refined grid. If the DNS simulations are to be performed and various models to be used and compared, it is better if the Reynolds number is kept as low as possible, in order to have lower CPU costs. Conditions applied in experiments to which planned DNS simulations are to be compared respect this, as cavitation was observed at Re as low as $Re = 16700$. This is favourable towards the choice of incompressible phase treatment because cavitation which occurs in such conditions is prone to be sheet cavitation or cloud cavitation with re-entrant jet being the mechanism of cloud shedding. Which means that shock governed cavitation collapse is avoided, therefore such cavitation can be successfully simulated with assumption of incompressible phases, which was also shown numerous times with various cavitation models.

Finally, the theory and also some experiments used to develop cavitation models based on bubble dynamics can be introduced to support the choice to neglect thermal effects and treat phases as incompressible as well. This theory is the mentioned explanation in [8], where the Schnerr-Sauer model development is described. Since the intention in this work is not to develop a new model, the discussion from [8] is here only roughly restated. To begin, the before mentioned Rayleigh-Plesset equation has to be introduced in one of its most often used form. The equation's development is not considered here but can be found in many sources such as [1, 7]. Figure 4.1 illustrates the role of this equation, which is description of single cavitation bubble behaviour in a pressure field. In it, R is the bubble radius, dR/dt bubble growth velocity, r is the distance from bubble centre, while p_b, T_b or p_{sys}, T_{sys} are pressure and temperature in bubble or its surroundings, respectively. The Rayleigh-Plesset equation in consideration is given with equation (4.6).

$$R\frac{d^2R}{dt^2} + \frac{3}{2}\left(\frac{dR}{dt}\right)^2 = \frac{p_b(t) - p_{sys}}{\rho_l} - \frac{2\psi(T_{sys})}{\rho_l R} - 4\frac{\mu_l}{\rho_l R}\frac{dR}{dt} \tag{4.6}$$

First term on the LHS gives the acceleration of bubble surface while the second one includes its velocity. The first term on the RHS describes the difference between the pressure inside and outside of the bubble. The pressure inside and outside is assumed uniform. The second term gives the effect of surface tension ψ while the last one includes viscosity effects. If bubble dynamics based models, described in paragraph 2.2.2.3, are considered, it can easily be noticed how much more complex this equation is compared to the one used in them. The discussion in [8] actually gives the reasons why the other equation, here restated with equation (4.7), can be used instead

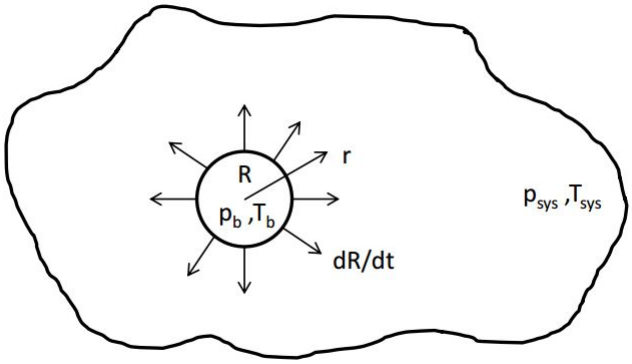


Figure 4.1: Graphical illustration of Rayleigh-Plesset equation considerations.

of equation (4.6), and not just why thermal and compressibility effects can be neglected.

$$\frac{3}{2}\left(\frac{dR}{dt}\right)^2 = \pm\frac{\mid p_v - p \mid}{\rho_l} \tag{4.7}$$

As it can be seen, equation (4.7) includes only velocity of bubble surface and pressure difference terms to model bubble behaviour. At the same time, it assumes pressure in a bubble to be constant and equal to vaporisation pressure p_v. This is in contrast with equation (4.6), where pressure in a bubble p_b changes with time and is under influence of temperature changes. It is through these pressure changes where compressibility of vapour phase and thermal effects are included. In [8], an analysis of the effect the inclusion of compressible vapour has on the bubble growth phase is done through comparison of complete Rayleigh-Plesset equation (4.6) with equation (4.7), which is combined with the so called Plesset-Zwick equation. The combination of these two equations is done in order to find the limiting temperature of the liquid at which the thermal effects in bubble growth can be neglected. It was found that the combined equations follow the behaviour of Rayleigh-Plesset equation very well, the only questionable area is the initial bubble growth where acceleration and surface tension play important role. Since the time interval where these two are important is very short and usually much shorter than computational time steps used, the error can be neglected. Then, the use of combined equations was compared with the use of equation (4.7). It was concluded that for water temperatures below 30 $°C$, the thermal effects can be safely neglected when one considers bubble growth. Same was found for bubble collapse phase, where it is shown that the velocity defined by (4.7) follows the velocity defined by (4.6) at the initial collapse phase very well on average. The only difference is the very end phase with following rebound. Which is at temperatures below 30 $°C$ not pronounced. Hence the incompressible treatment of phases for simulations including cold water was justified.

Since the discussion in [8] gives also the reason why the viscosity, surface tension and bubble acceleration effects can be neglected for usual cavitating flow simulations, it is considered appropriate to restate these findings here as well. It is argued that viscosity and surface tension effects can be neglected on behalf of much bigger influence the pressure difference term has compared to them. For acceleration, the explanation is not so simple. Acceleration effects on bubble growth are said to be neglected on the basis that typically, time steps used in simulations are longer or on the same order as the time in which acceleration takes place ($10^{-6} - 10^{-4}$ s, depending on pressure difference). Moreover, after the end of acceleration, bubble is for a longer period governed by a velocity which is defined with pressure difference, that is, equation (4.7). Therefore

neglecting acceleration for bubble growth seems plausible. For bubble collapse, the situation is a bit different. Velocity change in time is with inclusion of acceleration much different from the one defined by (4.7) on instantaneous level. But on average, the two velocities give very similar values for bubble size change, which is also the reason for retaining equation (4.7) for collapse phase as well.

1.2 Solving cavitating flow governing equations with projection methods

Before going to the presentation of the algorithm for solving the system of presented governing equations, it seems appropriate to introduce some points why projection methods in general offer a good basis for development of the wanted algorithm. And together with this also show known issues when such a development is considered and how to solve them. As it was mentioned in chapter 3, projection methods were in the past successfully adapted to the solution of governing equations for cavitating flow simulations. More precisely, they were adapted also for a solution of exactly the same system of governing equations as presented with equations (4.1)−(4.3). Very thorough examples of the adaptation, which also serve for the following explanations, are given in [8, 4]. In these two works, the projection method adaptation builds on SIMPLE (Semi Implicit Method for Pressure Linked Equations) algorithm, originally developed by Patankar and Spalding [8, 118]. Although this algorithm bases on the use of projection methods, it has to be said that it also significantly differs from the algorithm in MFLOPS-3D code. The reason is that there exist different versions of projection methods. Importantly as well, different numerical methods can be applied with them. The projection method used for SIMPLE algorithm is therefore not exactly the same as the one used in MFLOPS-3D. Or the ones which are suitable for this code (essentially methods as those in section 2.2 of chapter 3). Furthermore, other algorithms devised from the original SIMPLE algorithm have also been adapted to the solution of here proposed system of governing equations. An example is adapted SIMPLEC algorithm available in code Saturn [119, 120]. SIMPLEC is originally an improvement of SIMPLE algorithm [81]. Therefore it should be shown why and how projection methods as the one in MFLOPS-3D code can also be used in the case of here proposed governing equations. To do this, it is first necessary to spare some explanation about the SIMPLE algorithm. Other algorithms devised from it are not included since they use essentialy same approach. Here, the algorithm will not be presented thoroughly, since it is well documented in many sources, such as [81, 118, 8, 4]. Instead, a rather more general description of it will be given and used to make the explanation.

When comparing SIMPLE algorithm and its improved versions with the projection methods (and following algorithms) as used in MFLOPS-3D code, it is first noted that this algorithm is in the literature usually applied with finite volume methods. Therefore it operates with equations written in integral and not differential form. Such approach will also be used here, meaning that the equations concerning SIMPLE algorithm are given in integral form (except the pressure correction equations). The algorithm proceeds in same manner as the one in MFLOPS-3D. At first, momentum equation for predictor velocity $\vec{v}^*$, obtained from NS momentum equation, is solved. The NS momentum equations for real and predictor velocity are given for x direction or component in (4.8) and (4.9).

$$A_P^u u_P^{n+1} + \sum_l A_l^u u_l^{n+1} = Q_u^{n+1} - \left(\frac{\partial p^{n+1}}{\partial x}\right)_P V_P \tag{4.8}$$

$$A_P^u u_P^{m*} + \sum_l A_l^u u_l^{m*} = Q_u^{m-1} - \left(\frac{\partial p^{m-1}}{\partial x}\right)_P V_P \tag{4.9}$$

These two equations deserve some explanation, since it is hard to see how differential or even integral forms of NS momentum equations are organized in such notation. The basic idea, as it is mentioned in [81], is that values of velocities and pressure coming from centers of finite volume cells, for which these equations are written, are included in terms with subscript P. The summation term includes implicitly treated velocity values from neighbour cells, which come from discretization of the equations. Term Q depicts source term, which hides in itself explicitly treated terms from NS momentum equations and, if required, body force or other linearized terms that can depend on actual velocity u^{n+1} or other variables at current time level. Last term represents the pressure gradient, multiplied with the cell volume V_P. Coefficients A_P^u and A_l^u on the other hand are products of multiple coefficients coming from time discretization, density, viscosity and size of cell surfaces. Regarding the superscripts n and m, it should be said that superscripts with n denote time level, while those with m denote the iteration level. Therefore it is worth to notice that equation (4.9) is meant to be solved in iterative steps, where terms with $m*$ superscript include implicitly treated variables, while terms with $m-1$ superscript are based on the use of values from previous iteration. However, over these iterative steps, which are required by SIMPLE algorithm, variables converge to the point where equation (4.9) equals (4.8), thus all variables are considered to be at the current time level $(n+1)$. This, of course, is only a very rough explanation of the equation for predictor velocity and if the reader wishes to obtain more detailed view into the subject, explanations in [8, 81] are a recommended read. But for the use in the scope of this text, the given explanation suffices. What should only be added, is that when predictor velocity equation from SIMPLE algorithm is compared with the same equation from algorithm or projection methods as used in MFLOPS-3D, it can be noted that it is very hard to make a distinction which terms from NS momentum equations contribute to certain terms in equation (4.9). In fact, all terms except the pressure gradient and transposed viscous term can contribute to the implicitly treated variables. This is not the case in projection methods as used in MFLOPS-3D, where only time derivative and complete viscous term contribute to implicitly treated terms. Which has also important consequences in the following step. But it does not mean that such momentum equation is more implicit than the one in MFLOPS-3D code. In fact, it is hard to compare this characteristics of both equations and since it does not affect the explanation, this view remains here only mentioned.

The next step in SIMPLE algorithm is similar to projection equation step in MFLOPS-3D algorithm. An equation is proposed, in which one can use the known value of velocity divergence $\nabla \cdot \vec{v}$ or continuity equation constraint to get the pressure or an intermediate variable, which respects this constraint and is then also used to get the desired velocity field from predicted velocity $\vec{v}^*$. The algorithm therefore in this regard still proceeds in same manner as the one in MFLOPS-3D. Which is also expected, as projection equation is the main point on which projection methods work. But where this equation in case of projection methods as used in MFLOPS-3D comes from the difference between momentum equations for $\vec{v}$ and $\vec{v}^*$, its derivation in the case of SIMPLE algorithm takes another approach. At first, corrections of velocity $\vec{v}'$ and pressure p' are proposed as shown in equations (4.10) and (4.11).

$$\vec{v}^m = \vec{v}^{m*} + \vec{v}' \tag{4.10}$$

$$p^m = p^{m-1} + p' \tag{4.11}$$

The corrected values are actually the proposed values for velocity $\vec{v}^{n+1}$ and pressure p^{n+1}. If equations (4.10) (considering x component) and (4.11) are used in (4.9), it follows that one part can be written as (4.8) and cancels, while the remainders give connection between velocity correction for certain component and pressure correction gradient in the corresponding direction. This results in equation (4.12), which is the projection equation of SIMPLE algorithm. It should be noted that this equation includes two forms, where the only difference is that the summation term is simplified into barred term. The projection equation of projection methods as the one in MFLOPS-3D code (equation (3.16)) can be seen as an equivalent, since the difference between $\vec{v}^{n+1}$ and $\vec{v}^*$ gives exactly the correction $\vec{v}'$ while Φ can be considered as variable equal or only including pressure correction p' in itself (depending if the method is pressure incremental or non incremental version). However, in contrast to projection method in MFLOPS-3D, where one arrives at the equation for Φ via applying divergence directly to projection equation, SIMPLE algorithm first implements the new connection into continuity equation through the use of (4.10). This gives equation (4.13), which is contrary to other equations for SIMPLE algorithm not given only for one direction. Hence it is given in differential form and it should be observed that $\bar{\vec{v}}'$ is the vector composed from barred variables from equation (4.12).

$$u'_P = -\frac{\sum_l A^u_l u'_l}{A^u_P} - \frac{V_P}{A^u_P}\left(\frac{\partial p'}{\partial x}\right)_P = \bar{u}'_P - \frac{V_P}{A^u_P}\left(\frac{\partial p'}{\partial x}\right)_P \tag{4.12}$$

$$\nabla\cdot\left(\frac{\rho V_P}{A^u_P}\nabla p'\right)_P = \left(\nabla\cdot(\rho\vec{v}^{m*})\right)_P + \left(\nabla\cdot\left(\rho\,\bar{\vec{v}}'\right)\right)_P \tag{4.13}$$

Equation (4.13) is equivalent to Poisson equation (3.52) in MFLOPS-3D, although it is obtained from continuity equation. However, equation cannot be directly solved, since it features two unknown variables. Beside p' also $\bar{\vec{v}}'$ in the last term, which makes it impossible to be solved in such form. SIMPLE proposes a solution for this by neglecting the last term in (4.13) and therefore solving for p' as given with equation (4.14). This is also the main source of difference between projection method used in SIMPLE algorithm and projection methods as the one used in MFLOPS-3D. The consequence is that (before mentioned) iterative steps, including equations (4.9) and (4.14), have to be performed before final velocity $\vec{v}^{n+1}$ and pressure p^{n+1} are obtained. After each solution of equation (4.14), velocity and pressure are updated with the use of (4.10) and (4.11). Needed velocity corrections $\vec{v}'$ for this are obtained from (4.12) with neglected $\bar{\vec{v}}'$. Although the convergence characteristics of SIMPLE algorithm can be well affected because of this neglected term, the algorithm still produces correct solutions. An explanation why is given in [118]. Improved versions of SIMPLE algorithm deal with improving convergence issues by somehow accounting for the neglected term, but the overall approach used in them is the same.

$$\nabla\cdot\left(\frac{\rho V_P}{A^u_P}\nabla p'\right)_P = \left(\nabla\cdot(\rho\vec{v}^{m*})\right)_P \tag{4.14}$$

As shown, both versions of projection method, either the ones used in SIMPLE algorithm and its derivations, or the ones similar to the projection method in MFLOPS-3D, share a lot of common points. To conclude or shortly put, they all propose a predictor velocity which is then

corrected into the sought velocity by applying an equation in which the pressure field is forced to respect the continuity equation or velocity divergence constraints. The way in which this equation is derived differs between the methods, but its meaning and effect is the same. This, with the fact that projection methods as used in SIMPLE algorithm have been successfully adapted to solution of the concerned system of cavitating flow governing equations, is the reason for the decision to adapt the projection method used in MFLOPS-3D code for use in cavitating flows. But before going onto the presentation of adaptation, it is better to show how this adaptation is done in the case of SIMPLE algorithm to shed some light onto known problems when such adaptation is performed.

When considering simulations of cavitating flows with SIMPLE algorithm, the most important change which needs to be done is adaptation of pressure solution. There are also other changes, such as introducing equations for definition of variable density and viscosity on cell faces, but these are here omitted as they concern only factors A_P^u and A_N^u. The first change for the pressure solution actually comes from simulations of flows with variable density in general. As it is well shown in [8], the presented pressure correction solution with equation (4.14), to which density time derivative has to be added because of variable density, causes high numerical errors. These can be seen if a two phase flow with sharp transition between the phases but no phase transfer is concerned. Equation (4.14) with the added time derivative as in [8] is shown in (4.15). The solution for pressure and velocity following from it, though preserving continuity, does not preserve sharp transition between the phases. The reason for this is traced to the inability of (4.15) to ensure also velocity divergence constraint, since it is based on the implementation of velocity correction directly into whole continuity equation.

$$\nabla \cdot \left(\frac{\rho V_P}{A_P^u} \nabla p' \right)_P = (\nabla \cdot (\rho \vec{v}^{m*}))_P + \frac{\rho^{t+\Delta t} - \rho^t}{\Delta t} V_P \tag{4.15}$$

Since the problems with equation (4.15) come from using whole continuity equation and we know that the considered flow, although multiphase, is still divergence free (since there is no phase transfer), a remedy for the encountered problem is very simple. Instead of using the whole continuity equation, the velocity correction $\vec{v}'$ is simply used in the divergence free constraint $\nabla \cdot \vec{v}^{n+1} = 0$. This, with still neglecting the $\vec{\bar{v}}'$, results in the equation for pressure correction (4.16).

$$\nabla \cdot \left(\frac{V_P}{A_P^u} \nabla p' \right)_P = (\nabla \cdot \vec{v}^{m*})_P \tag{4.16}$$

Solving for pressure and then following velocity correction with the use of (4.16) ensures that obtained velocity and pressure respect divergence constraint, which eliminates the mentioned numerical errors. Equation (4.16) also makes the solution for pressure correction in SIMPLE very similar to Poisson equation for Φ in MFLOPS-3D as the divergence constraint is used in both cases directly. Thus both projection methods become even more similar. Same equation is then also used as a basis for the pressure solution in the case of cavitating flows simulations, giving additional reason for use and adaptation of the projection method in MFLOPS-3D for such simulations. However, in them the divergence of velocity is not equal to zero any more, since we have phase change between water and vapour. This would normally pose problems, since we use projection method to find a pressure which satisfies certain known divergence of velocity. Luckily, the presented system of governing equations (4.1)−(4.3) offers another connection between velocity and pressure through divergence. This follows from discounting liquid volume fraction α_l equation from the given vapour volume fraction α_v equation (4.3). The transport equation for α_l

can be simply written with the use of connection given in (4.4), while its source term is the same as in (4.3), only with opposite sign. Connection between source term S and velocity divergence follows from this deduction as given in (4.17).

$$\nabla \cdot \vec{v} = S \left(\frac{1}{\rho_v} - \frac{1}{\rho_l} \right) \tag{4.17}$$

As the source term depends greatly on the pressure field it offers a direct connection between the pressure and velocity divergence. This feature is used with great benefit when solving for pressure correction, although it posed huge issues at first. The equation for pressure correction, following from (4.16) and utilising connection in (4.17), is given with equation (4.18). Notice that the source term in it is treated explicitly. This is necessary, since the current pressure field is not known. What this means is that also the current divergence of velocity is not known. And when one solves for p' or $\vec{v}'$ with such an equation, simulations show large stability problems. These are well explained in [8]. Briefly, the issues are caused by the use of explicit divergence value and include high oscillations in results for p' between iterations, which grow with the amount of iterations performed and lead to divergent results.

$$\nabla \cdot \left(\frac{V_P}{A_P^u} \nabla p' \right)_P = (\nabla \cdot \vec{v}^{m*})_P - S_P^{m-1} \left(\frac{1}{\rho_v} - \frac{1}{\rho_l} \right) \tag{4.18}$$

The solution for this issue actually opened the possibility to use SIMPLE algorithm and with it also projection methods in general to perform cavitating flow simulations with the chosen governing equations. Since the source term changes according to pressure change or correction, it is possible to approximate or linearise the new source term as the sum of its explicit value and partial derivative in respect to pressure. The source term can then be written with equation (4.19). If this is put into equation (4.18), we obtain the final equation for pressure correction calculation in the case of cavitating flow simulations, equation (4.20).

$$S^m = S^{m-1} + \frac{\partial S}{\partial p} p' \tag{4.19}$$

$$\nabla \cdot \left(\frac{V_P}{A_P^u} \nabla p' \right)_P + \left(\frac{\partial S}{\partial p} p' \right)_P \left(\frac{1}{\rho_v} - \frac{1}{\rho_l} \right) = (\nabla \cdot \vec{v}^{m*})_P - S^{m-1} \left(\frac{1}{\rho_v} - \frac{1}{\rho_l} \right) \tag{4.20}$$

This equation eliminates the problem of not knowing the source term and with it the divergence of velocity in advance. As mentioned, the dependence of source term on the pressure is used here with great benefit. However, it is important to notice that the equation demands the explicit part of the source term to be defined with pressure from previous iteration, hence S^{m-1} notation. If pressure from previous time step is used, instabilities occur again, since such an approach violates the logic of SIMPLE algorithm. Namely, pressure and velocity corrections in it should tend to reach zero value with iterations, which effectively also means that equations for $\vec{v}^*$ and $\vec{v}^{n+1}$, (4.9) and (4.8), become the same. This is not possible if the source term is not updated with the pressure from previous iteration and same problems occur as those when equation (4.18) is used.

The presentation of the suitability of projection methods for solution of governing equations (4.1)−(4.3) is with this final equation (4.20) concluded. However, as it will be seen in the following explanation of the new algorithm, the presented approaches which resolve the problems with ensuring the pressure correction equation to respect the divergence constraints (and lead to equations (4.16) and (4.20)) are not directly applicable to the MFLOPS-3D code. But they offer a strong reference point of what needs to be done and respected when such an adaptation is desired.

2 The new algorithm

The new algorithm development is presented in this section. Since the algorithm is based on the projection method used in original MFLOPS-3D code, it is at first presented through equations that form it in cavitating flow case. Equations for predictor velocity and Φ are given first. As Φ or pressure solution proved to be very difficult and the steps used to resolve it in SIMPLE algorithm are not directly applicable in the case of MFLOPS-3D code, its explanation will be more thorough. Included in it are also solutions for real velocity. The presentation of Φ solution is then followed by presentation of the last step in the algorithm, the solution of newly introduced α transport equation (4.3) and the connections between material variables (4.4). Finally, as the algorithm uses a lot of iterative steps and can include different approaches towards pressure solution, the explanation is made complete with a more precise presentation of its developed versions. An important remark considering these is that the algorithm can use two versions of projection method, pressure incremental or non incremental. The differences concerning them are given together already in the presentation of the equations composing projection method.

2.1 Predictor velocity $\vec{v}^*$ solution

Introduction of complete Navier-Stokes momentum equations with variable ρ and μ included and the fact that $\nabla \cdot \vec{v}$ cannot be taken as zero changes the predictor velocity equation (3.20) into (4.21).

$$\rho\left(\frac{3\vec{v}^* - 4\vec{v}^n + \vec{v}^{n-1}}{2\Delta t} + (\vec{v}\cdot\nabla)\,\vec{v}\right) = -\nabla p + \nabla\cdot\left(\mu(\nabla\vec{v}^*) + \mu(\nabla\vec{v}^*)^T\right) - \frac{2}{3}\nabla\left(\mu(\nabla\cdot\vec{v}^*)\right) \quad (4.21)$$

ρ and μ in the above equation are for the moment considered known, therefore they have no superscript, although this is otherwise not so. It should be noted that contrary to (3.20), equation (4.21) features also pressure gradient term. This is included only in the pressure incremental version of the algorithm, while the non incremental version omits it. The same holds also for all following equations presenting $\vec{v}^*$ solution. In order to solve equations for u^*,v^* and w^* with the described solvers in MFLOPS-3D, equation (4.21) must be first reorganized into Helmholtz-like equation as (3.51). Since solvers demand constant LHS, this reorganization is not so simple as it was in the case of incompressible flow, because μ and ρ are not constant. It also results in an increased use of explicitly treated terms. Two ways of reorganization were applied. In the first, μ and ρ are included as:

$$\rho = \alpha_v(\rho_v - \rho_l) + \rho_l \quad \text{and} \quad \mu = \alpha_v(\mu_v - \mu_l) + \mu_l \quad (4.22)$$

Equations come from including $\alpha_v + \alpha_l = 1$ connection in (4.4). Then, the $\vec{v}^*$ terms that include only constant ρ_l and μ_l are put to the LHS while all other terms are left on the right or treated explicitly. The terms that at first feature $\vec{v}^*$, but are multiplied with non-constant values, have $\vec{v}^*$ replaced with a prediction for real velocity $\vec{v}^{n,n-1}$ obtained with Adams-Bashrofd extrapolation (shown in section 3.1, chapter 3). With this, we obtain (4.23), from which equations for u^*,v^* and w^* follow.

$$\begin{aligned}\left(\Delta - \frac{3\rho_l}{2\Delta t\mu_l}\right)\vec{v}^* =& \frac{3\alpha(\rho_v-\rho_l)\vec{v}^{n,n-1}}{2\Delta t\mu_l} + \frac{\rho}{\mu_l}\left(\frac{-4\vec{v}^n+\vec{v}^{n-1}}{2\Delta t} + ((\vec{v}\cdot\nabla)\vec{v})^{n,n-1}\right)\\ &+ \frac{\nabla p}{\mu_l} - \frac{1}{\mu_l}(\nabla\mu)\cdot(\nabla\vec{v}^{n,n-1}) - \alpha\frac{\mu_l-\mu_v}{\mu_l}\Delta\vec{v}^{n,n-1}\\ &- \frac{\nabla\mu}{\mu_l}\cdot(\nabla\vec{v}^{n,n-1})^T - \frac{\mu}{3\mu_l}\nabla(\nabla\cdot\vec{v}^{n,n-1})\\ &+ \frac{2}{3\mu_l}(\nabla\mu)(\nabla\cdot\vec{v}^{n,n-1})\end{aligned} \tag{4.23}$$

This equation proved to be inappropriate for obtaining $\vec{v}^*$. Moreover, it presents an inappropriate basis for the projection method, as equations for Φ and $\vec{v}^{n+1}$ should follow directly from its form. The reason is in the mentioned separation of terms and following large number of explicitly treated terms. Therefore the second way of reorganization was needed. The key to this was use of the Concus and Golub method, described in [121]. The method proposes an iterative scheme to enable the use of fast direct solvers, applicable only to Helmholtz equations with constant coefficients, in cases of Helmholtz equations with non constant coefficients. Which would be exactly the type of Helmholtz equation solved here, if variable LHS coefficients were allowed. As was shown in previous chapter, this is ultimately not possible because of the chosen multi domain method. It then follows that Concus and Golub method provides the ideal tool to apply the solvers in MFLOPS-3D to the solution of new equation for $\vec{v}^*$. Especially since it is in [121] shown to compete well with best performing iterative solving techniques used otherwise to solve Helmholtz equation with non constant coefficients. The way in which this method works is based on the following principle. If we have an equation like (3.53), but with non-constant a, we can add to the both sides some constant a_1. (3.53) can be written as (4.24).

$$(\mathbf{B} - (a+a_1)\mathbf{I})u = D - a_1\mathbf{I}u \tag{4.24}$$

The equation is mathematically still the same. But because practically we do not have u that we search for on the RHS, an explicit value u_e has to be used. If a term is then transferred to the RHS and given the explicit u_e, we obtain (4.25), which can be solved with the direct mono domain solver. Then, iterating for u is performed, until converged u is obtained. Thus we can say that equation (4.24) and with it also (3.53) with non-constant a is solved.

$$(\mathbf{B} - a_1\mathbf{I})u = D - (a_1-a)\mathbf{I}u_e \tag{4.25}$$

In the case of (4.21), the method, together with the fact that we need to obtain a Helmholtz like equation, results in (4.26).

$$\begin{aligned}\left(\Delta - \frac{3\rho_l}{2\Delta t\mu_l}\right)\vec{v}^* =& \rho\frac{-4\vec{v}^n+\vec{v}^{n-1}}{2\Delta t\mu} + \frac{\rho}{\mu}((\vec{v}\cdot\nabla)\vec{v})^{n,n-1} + \frac{\nabla p}{\mu}\\ &- \frac{1}{\mu}(\nabla\mu)\cdot(\nabla\vec{v}^{n,n-1}) - \frac{1}{\mu}(\nabla\mu)\cdot(\nabla\vec{v}^{n,n-1})^T - \frac{1}{3}\nabla(\nabla\cdot\vec{v}^{n,n-1})\\ &+ \frac{2}{3\mu}\nabla\mu(\nabla\cdot\vec{v}^{n,n-1}) + \left(\frac{3\rho}{2\Delta t\mu} - \frac{3\rho_l}{2\Delta t\mu_l}\right)\vec{v}_e^*\end{aligned} \tag{4.26}$$

$\vec{v}_e^*$ is the explicit value of $\vec{v}^*$ and comes from previous iteration or, initially, time step. ρ_l and μ_l are chosen to form the constant on the LHS as majority of the computational domain has

$\alpha_v = 0$ and the ratio between ρ and μ does not change much with α in case of water being the liquid. This equation has less explicit terms than (4.23) and proved to give good basis for the projection method. On the other side, it does increase the computation time as some iterations have to be performed to match $\vec{v}_e^*$ with $\vec{v}^*$ in the sub domains where cavitation is present. Value of 10^{-8} as highest absolute difference in certain point is set as convergence criteria. Amount of iterations is however not high and will be presented in a later chapter dealing with performance analysis.

To repeat, ρ and μ are considered as known in the equations presented above. This is of course not the case and is only used here for simplification of the presentation. ρ and μ are otherwise treated explicitly, the exact treatment will be presented later. Additionally, the skew symmetric form for the non linear term is not used here in the same manner as before, since that form is only valid for incompressible flow [110]. Therefore another skew symmetric form is used and given in following paragraph. The presentation of $\vec{v}^*$ solution here is also missing definition of boundary conditions. These are practically identical to those used in incompressible flow, but are given in the following section since some further explanations have to be first included.

2.1.1 Skew symmetric form of non linear term

In section 3.3 of chapter 3, it is shown that the purpose of using skew symmetric form for non linear term is ensuring conservation of kinetic energy, which the form conserves a priori. The form also conserves momentum if continuity equation is satisfied. The whole conservation property of the form in incompressible flow is based on the fact that terms which are written in divergent form are conservative, since continuity equation is practically represented only by divergence of velocity. Using the volume integral as in equation (3.46), it is shown that inflow and outflow effects are captured in the divergence form, thus conservation of properties it includes is said to be ensured a priori.

The same rules do not apply also to the compressible flow case. In it, continuity equation is zero only as a whole and same goes for the application of volume integral to it. Therefore the time derivative has to be included in the development of a conservative form with which the non linear term is written. Which consequently means that the complete convective part or convective terms of Navier-Stokes momentum equation (terms on LHS of equation (4.1)) have to be considered if momentum and kinetic energy are to be conserved. These terms are for clarity given in equation (4.27), since the NS momentum equation (4.1) gives their shorter form, obtained with the use of continuity equation. The theory given in [111] was used in order to find a good form for these terms and with it the non linear term.

$$conv = \frac{\partial \rho \vec{v}}{\partial t} + \nabla \cdot (\rho \vec{v} \vec{v}) \tag{4.27}$$

As it is mentioned in [111], the convective terms form given in (4.27), called also divergence form of convective terms (the form used in equation (4.1) is called advective form), is momentum conservative a priori. That is, even if continuity equation is not satisfied. But since the goal is to ensure conservation of kinetic energy, skew-symmetric form is preferred. This, as in the case of incompressible flow, conserves momentum if continuity is conserved and is able to conserve kinetic energy a priori. Different ways exist in which the form can be written. The form chosen for use here has same basis as the one used in incompressible flow code as it is composed of an arithmetic average of divergence and advective forms. It is given in equation (4.28) where the time derivative terms are written separately from other, spatial derivative terms.

$$\frac{\partial \rho \vec{v}}{\partial t} + \nabla \cdot (\rho \vec{v} \vec{v}) = \frac{1}{2}\left(\frac{\partial \rho \vec{v}}{\partial t} + \rho \frac{\partial \vec{v}}{\partial t}\right) + \frac{1}{2}\left(\nabla \cdot (\rho \vec{v} \vec{v}) + \rho \vec{v} \cdot \nabla (\vec{v})\right) \tag{4.28}$$

Since the skew-symmetric form now includes also time derivatives, this affects their discretization in equation for $\vec{v}^*$. The two terms can be written with 2^{nd} order backward difference scheme as shown in equation (4.29). Comparing such time discretization scheme with the ones used in [111], the form can be referred to also as semi discrete convection scheme. It turns out that the constant term on the LHS of equation (4.26) and terms introduced by Concus and Golub method remain unchanged on behalf of them corresponding to terms with the same density in both time derivatives in equation (4.29). Nevertheless, the remaining part of the time derivative in equation (4.26) has to be adjusted, as well as the non linear term. Equation for $\vec{v}^*$ is therefore finally written as shown with (4.30). The superscript $n, n-1$ shows that Adams-Bashforth method is still used.

$$\frac{1}{2}\left(\frac{\partial \rho \vec{v}}{\partial t} + \rho \frac{\partial \vec{v}}{\partial t}\right) = \frac{1}{2}\left(\frac{3\rho\vec{v}^* - 4\rho^n \vec{v}^n + \rho^{n-1}\vec{v}^{n-1}}{2\Delta t} + \rho \frac{3\vec{v}^* - 4\vec{v}^n + \vec{v}^{n-1}}{2\Delta t}\right) \tag{4.29}$$

$$\begin{aligned}\left(\Delta - \frac{3\rho_l}{2\Delta t \mu_l}\right)\vec{v}^* =& \frac{1}{4\Delta t \mu}\left(-4\vec{v}^n\left(\rho + \rho^n\right) + \vec{v}^{n-1}\left(\rho + \rho^{n-1}\right)\right) + \frac{1}{2\mu}\left(\nabla \cdot (\rho \vec{v} \vec{v}) + \rho \vec{v} \cdot \nabla (\vec{v})\right)^{n,n-1} \\ &+ \frac{\nabla p}{\mu} - \frac{1}{\mu}(\nabla \mu) \cdot (\nabla \vec{v}^{n,n-1}) - \frac{1}{\mu}(\nabla \mu) \cdot (\nabla \vec{v}^{n,n-1})^T - \frac{1}{3}\nabla(\nabla \cdot \vec{v}^{n,n-1}) \\ &+ \frac{2}{3\mu}\nabla\mu(\nabla \cdot \vec{v}^{n,n-1}) + \left(\frac{3\rho}{2\Delta t \mu} - \frac{3\rho_l}{2\Delta t \mu_l}\right)\vec{v}_e^*\end{aligned} \tag{4.30}$$

The use of skew-symmetric form for the convective terms of NS momentum equation does not yet guarantee that kinetic energy will actually be well conserved. As mentioned already for incompressible flow case, conservation properties depend not only on the form of equations used but also on discretization schemes. Regarding this, the question of actual performance of the skew-symmetric form is still valid, especially since compact finite differences are used, which were in section 3.3 of chapter 3, following [111], already said to lack existence of fully conservative schemes. Moreover, no special steps for ensuring or improving conservation properties on non uniform grids were applied, as also stated in 3.3. Still, the predictions about performance of skew-symmetric form given in that section are valid also for this case. Especially as [111] shows that compact finite differences were applied on a collocated grid (referred to as regular in [111]) with same temporal locations of variables used in time derivatives as in MFLOPS-3D. Which is important as all convective terms are now considered in a certain form and not only the non linear ones. As mentioned before, skew-symmetric form of convective terms was the only one stable at the inviscid limit when compact finite differences were used. It has to be admitted that compact schemes were not the same as used in MFLOPS-3D. While grid point values are used in the explicit stencils of compact schemes in MFLOPS-3D, mid point values were used in [111], which is said to be a more stable choice. Nevertheless, the mentioned performance of skew-symmetric form with compact schemes in similar spatio-temporal discretization as in MFLOPS-3D supports before given predictions about skew-symmetric form and is a strong point for use of the here presented form in $\vec{v}^*$ equation.

2.2 Solution of intermediate variable Φ, pressure and real velocity

The solution for the intermediate variable Φ and pressure in the new algorithm significantly differs from the solution in the case of incompressible flow. Many issues have been encountered and solved while developing this part of the algorithm, which is therefore considered as the most important part of the new algorithm. Moreover, correct pressure solution is in the case of cavitating flows, one could say, even more important as in the case of incompressible flows. The reason lies in the governing effect the pressure has on cavitation. In the governing equations, this most notably concerns the source term S which governs creation and destruction of gaseous phase. As such, the solution also deserves a more detailed explanation, covering the reasons for encountered problems and their resolutions. At first, two possible forms of projection equation are introduced together with the connections between Φ and p. Then, general issues of solving for Φ are given, to which additional issues presented by the requirements of the fast DNS solver are added. Finally, treatments which solve all these issues are presented and the final tools to obtain Φ solution are given.

2.2.1 Possible forms of projection equation and $\Phi - p$ connections

Projection equation is, as in the case of incompressible flow, obtained on the basis of differences between momentum equation for real velocity (4.1) and predictor velocity (4.26). The equations are considered to be both written in form of Helmholtz equation as given in (4.26) and to use same time discretization and explicit treatment of certain terms. It is also considered that an equality of $\vec{v}_e^*$ and $\vec{v}^*$ exists, thus the Concus and Golub method impact is omitted. If the equation for $\vec{v}^*$ is then discounted from the equation for $\vec{v}^{n+1}$, equation (4.31) is obtained. Explicit pressure gradient term is present in it if pressure incremental version of the algorithm is considered. Exact treatment for this explicit term will be shown later, but for the explanation here, the value from time level n is used. This represents first order accurate approximation in time. One step Adams-Bashforth method given in (3.22) was also used but found to cause unstable simulations.

$$\frac{3\rho}{2\Delta t}(\vec{v}^{n+1} - \vec{v}^*) = -\nabla p^{n+1} + \nabla p^n + \mu\Delta(\vec{v}^{n+1} - \vec{v}^*) \tag{4.31}$$

Since cavitating flow simulations require results for pressure field, it was at first tried to use equation (4.31) to form an equation for direct pressure solution. This was found to be impossible for both versions of the algorithm, since 3^{rd} derivatives are obtained from viscous terms once divergence is applied to (4.31). These were found to cause highly unstable simulations as using higher derivatives is always difficult and the errors that follow from solution procedure tend to be increased when such derivatives are applied. Therefore an intermediate variable Φ was reintroduced. As in the case of incompressible flow, this variable replaces the right hand side of equation (4.31), hence the projection equation can be written as (4.32). Equation (4.32) is also the first form of projection equation which can be used and equation, on which the following explanation of the algorithm development will be based.

$$\frac{3\rho}{2\Delta t}(\vec{v}^{n+1} - \vec{v}^*) = -\nabla\Phi \tag{4.32}$$

Inclusion of intermediate variable Φ demands also creation of $p - \Phi$ connection. For this, the equality of equations (4.31) and (4.32) is used to write equation (4.33). This gives connection between p and Φ in divergence form, from which the gradient operator has to be eliminated to obtain desired equation. From an example for this elimination in the case of incompressible flow,

given in [80], it was proposed that the elimination of the gradient should give us equation (4.34). A proof had to be done to show that this proposed equation is correct. The proof is based on same logic as the one given in [80] for the incompressible flow case and is given in the Appendix. With it, this important connection is established and use of intermediate variable Φ in case of cavitating flows is enabled.

$$\nabla\Phi = \nabla p^{n+1} - \nabla p^{n} - \mu\Delta(\vec{v}^{n+1} - \vec{v}^{*}) \tag{4.33}$$

$$p^{n+1} = \Phi + p^{n} + \mu\nabla\cdot(\vec{v}^{n+1} - \vec{v}^{*}) \tag{4.34}$$

The derived connection is not important just because it gives a stable basis for accurate pressure solution, but also because it provides the equation for real velocity solution and defines the boundary conditions for $\vec{v}^{*}$. Both uses of this equation are done equally to the incompressible flow case. Velocity $\vec{v}^{n+1}$ can be obtained directly from equation (4.32) once Φ is obtained. And the boundary conditions for $\vec{v}^{*}$ in tangential directions use explicit value of Φ, while the value of $\vec{v}^{*}$ in normal direction to the boundary is equal to $\vec{v}^{n+1}$.

It should be also mentioned that although the presented connection in (4.34) is valid generally, its use differs in the cases of pressure incremental and non incremental versions of the algorithm. In the case of pressure non incremental version the connection obviously skips term p^{n}. The use of this connection in the case of incremental version has a more important difference. Namely, the viscous terms are omitted, thus the connection simplifies to the point where Φ is equal to pressure change. This is, as in the incompressible flow case, applicable because of $\vec{v}^{*}$ being closer to $\vec{v}^{n+1}$ if incremental version of the algorithm is used [82]. The decision to omit the viscous terms in (4.34) is importantly also based on observations that their use in pressure incremental version of the algorithm did not give an improvement or even actually made results worse.

Another projection equation form was created additionally to presented equation (4.32). This equation was derived on the basis that equation (4.32) carries a resemblance to the first proposed pressure correction equation in SIMPLE algorithm, equation (4.15). As (4.15), the proposed equation (4.32) also features the product of density and velocity. Equation (4.15) proved to be unsuitable for multiphase flows, since it failed to ensure zero velocity divergence condition in flows without phase transfer. In the same manner, it also fails to ensure any other velocity divergence constraint, including connection between velocity divergence and the source term. It can be argued that this is the consequence of additional density time derivative used in it. But the fact is also that there is a known functional dependence or description of velocity divergence available, while the divergence of the product $\rho\vec{v}$ is an unknown. And since the solution for variable Φ follows from Poisson equation, which is obtained with applying divergence to the projection equation (4.32), the divergence of this product is also present in the solution for Φ. Finite difference approach enables us to easily apply product rule for derivatives and hence split the divergence in two parts (as will be shown later), but similar issues as those presented in the case of SIMPLE algorithm were still feared. Therefore another form of the projection equation which does not feature mentioned product was desired. This seemed difficult to obtain, since the requirements of the solvers in MFLOPS-3D demand us to have a system of equations which feature Laplacian operator on the LHS. And not an operator which is multiplied with a certain value or variable or even hides a division by it. It was at first considered that forming a projection equation without the $\rho\vec{v}$ product included would give exactly such a result. But as it was found, it is possible to form projection equation where the density is excluded by simply introducing another intermediate variable. More precisely, equation (4.32) can simply be divided by density

and the gradient of Φ divided by this density can be written as gradient of another intermediate variable Φ_2, which is also shown in equation (4.35).

$$\frac{3}{2\Delta t}(\vec{v}^{n+1} - \vec{v}^{*}) = -\frac{\nabla\Phi}{\rho} = -\nabla\Phi_2 \tag{4.35}$$

A connection between Φ_2 and p has to be established before this equation can be used, like in the case of equation (4.32) for introduction or variable Φ. If the division as in (4.35) is used also in (4.33) or (4.31), the connection in differential form, given with equation (4.36), is obtained. As in the case before, the connection between Φ_2 and p without gradient operators is at first proposed and given in (4.37). Its proof is, similarly to (4.34), based on the logic shown in [80] and given in the Appendix.

$$\nabla\Phi_2 = \frac{(\nabla p^{n+1} - \nabla p^{n})}{\rho} - \frac{\mu}{\rho}\Delta(\vec{v}^{n+1} - \vec{v}^{*}) \tag{4.36}$$

$$\Phi_2 = \frac{(p^{n+1} - p^{n})}{\rho} - \frac{\mu}{\rho}\nabla\cdot(\vec{v}^{n+1} - \vec{v}^{*}) \tag{4.37}$$

As it will be seen later, this second proposed projection equation results in a Poisson equation which is very similar to the equation for pressure solution in SIMPLE for multiphase flows, equation (4.20). Interestingly, both equations do not seem to feature density directly. Where the density is in the case of the last presented projection equation hidden in Φ_2, it is in equation (4.20) hidden in coefficients A_P^u.

However, this second projection equation surprisingly did not give an improvement over the first proposed equation in the tests that were performed. Therefore it is in the scope of this work only mentioned, but results with it are not shown. Nevertheless, it is included in the new code as it might still offer some improvements over the first equation.

2.2.2 Poisson equation for Φ and issues encountered when solving it

The two projection equations presented before are used to define two possible Poisson equations with which the solution for Φ is obtained. The first projection equation, equation (4.32), can give two mathematically equal Poisson equations, shown in (4.38) and (4.39), after the divergence is applied to it. Both equations are shown since they are considered to be counterpart of the problematic pressure solution equation (4.15) in SIMPLE algorithm. As it was mentioned in previous section, use of $\rho\vec{v}$ product divergence can be considered problematic to ensure the defined velocity divergence condition. The Poisson equation shown in (4.38) features exactly this product and is therefore unwanted. Simple use of product rule for derivatives changes it to (4.39). This equation now directly includes the velocity divergence and correct solution of Φ should ensure that velocity divergence condition is also respected. This was also observed in tests. Therefore equation (4.39) presents a suitable Poisson equation for Φ solution, even if it is shares similarities with the problematic equation (4.15).

$$\Delta\Phi = -\frac{3}{2\Delta t}\left(\nabla\cdot\left(\rho\vec{v}^{n+1}\right) - \nabla\cdot(\rho\vec{v}^{*})\right) \tag{4.38}$$

$$\Delta\Phi = -\frac{3}{2\Delta t}(\vec{v}^{n+1} - \vec{v}^{*})\cdot\nabla\rho - \frac{3\rho}{2\Delta t}\left(\nabla\cdot\vec{v}^{n+1} - \nabla\cdot\vec{v}^{*}\right) \tag{4.39}$$

Contrary to (4.32), the second presented projection equation, equation (4.35), results in only one possible Poisson equation. This is given in equation (4.40). The equation is more compact

than first proposed Poisson equation (4.39). If compared with equation (4.20) in SIMPLE algorithm, where it should be considered that the velocity divergence is replaced by linearised source term, it is clear that the two equations are very similar. Although it was not found to offer better results than (4.39), it is considered that it might still prove more appropriate, especially since it features less explicitly treated terms. Why this is so will be explained in the section considering α transport equation solution.

$$\Delta\Phi_2 = -\frac{3}{2\Delta t}\left(\nabla\cdot\vec{v}^{n+1} - \nabla\cdot\vec{v}^{*}\right) \tag{4.40}$$

Both given Poisson equations can be seen only as the basis on which the Φ solution is formed. They both include velocity divergence which has to be replaced by source term using connection from equation (4.17). Moreover, the equation (4.39) also features terms with velocity $\vec{v}$ and density. The values of these are unknown prior to knowing Φ solution and have to be treated explicitly. This all resulted in many issues which needed to be overcome and the solution for Φ became very different from the one used in the original algorithm and code. Some of the issues were known in advance. These are the issues which are explained in section 1.2 with the adaptation of SIMPLE algorithm to cavitating flow simulations. Briefly put, the issues include first the problem of using the divergence of $\rho\vec{v}$ product (and density time derivative) and more importantly, the issue of not knowing the actual source term prior to pressure solution. The $\rho\vec{v}$ divergence problem actually proved to be avoided in our code without difficult treatment, as mentioned before. The source term issue on the other hand was found to be the completely different and much more demanding in the case of our code than in the case shown in [8, 4] with SIMPLE algorithm. The solution mentioned in these sources and shown in section 1.2 also proved to be of limited help. The reason why is in the solvers used in MFLOPS-3D code. As stressed, these demand constant LHS of the Poisson equation, therefore the source term cannot be simply linearised and have its dependence on pressure change treated implicitly. Consequently, issues with stability appeared and led to different proposed procedures. With the use of Poisson equation (4.39) it was also noted that the use of explicit density and velocity poses some issues, but nowhere near so profound as the inability to treat S in same manner as in [8]. Therefore the solution of S issues was the most difficult part of the development. They were also made even bigger with the presence of compatibility condition, which caused issues as it was at first strongly imposed by the use of homogeneous von Neumann boundary conditions for Φ, like for incompressible flow. Since the compatibility condition was further found to be possibly also affecting performance of the original and consequently also here developed new MFLOPS-3D code in profound manner, it requires better understanding and is explained in a bit more detail in the following section. However, the focus is mainly given to issues affecting new algorithm and with it also new code.

2.2.2.1 Compatibility condition constraint

As mentioned in section about the multidomain solver 4.2 in chapter 3, the solution for Φ with von Neumann boundary conditions represents Poisson-Neumann problem, which is well known to be a singular problem [87, 104, 122]. More precisely, the matrix representing the Laplace operator in such a case is singular as it has zero eigenvalues [104, 87]. Consequently two cases are possible. Either no solution can be found or infinitely many solutions exist. The difference between the two is in the satisfaction of compatibility condition [87, 107, 123, 124]. This condition follows from the use of von Neumann boundary conditions and imposes certain demands on the velocity fields through the used projection equation [105, 124]. To explain it, one must at first consider surface

integral of the used von Neumann boundary conditions. Since homogeneous zero gradient value is imposed everywhere, the value of integral is considered zero. Through the projection equation, the surface integral of the difference between velocities $\vec{v}$ and $\vec{v}^*$ has to be zero as well. All this is shown with equation (4.41) and should be satisfied with the equality of the two velocities in normal direction, which is also chosen as the boundary condition (and from which the zero gradient for Φ follows).

$$\oint \nabla\Phi \cdot d\vec{S} = \oint \frac{-3}{2\Delta t}(\vec{v}^{n+1} - \vec{v}^*) \cdot d\vec{S} = 0 \tag{4.41}$$

However, this does not yet mean that the compatibility condition is satisfied. The surface integral is namely connected with the Poisson equation through the use of Gauss theorem. Therefore compatibility condition means that equality in equation (4.42) and consequently also in (4.43) has to be satisfied.

$$\oint \nabla\Phi \cdot d\vec{S} = \int \Delta\Phi dV = 0 \tag{4.42}$$

$$\oint \frac{-3}{2\Delta t}(\vec{v}^{n+1} - \vec{v}^*) \cdot d\vec{S} = \int \frac{-3}{2\Delta t}(\nabla \cdot \vec{v}^{n+1} - \nabla \cdot \vec{v}^*)dV = 0 \tag{4.43}$$

This makes the compatibility condition more difficult to ensure. If only the surface integrals were considered, this would be easily ensured, since Dirichlet boundary conditions for velocities mean that surface values of velocities are directly defined and equal. But the need to have volume integral of Poisson equation also equal to zero demands that the difference between the volume integrals of $\nabla \cdot \vec{v}^{n+1}$ and $\nabla \cdot \vec{v}^*$ is zero. In the case of incompressible flows, where $\nabla \cdot \vec{v}^{n+1} = 0$, one then has to ensure that the volume integral of $\nabla \cdot \vec{v}^*$ is zero. Considering Gauss theorem, connection in equation (4.44) follows. Essentially same connection was already shown in section 3.3 of chapter 3, where it was used to raise the argument about conservation properties. Conforming to that, since $\vec{v}^*$ equals $\vec{v}^{n+1}$ in normal direction on the boundaries, this simply means that mass in the domain has to be preserved or mass inflow has to equal mass outflow.

$$\int \nabla \cdot \vec{v}^* dV = \oint \vec{v}^* \cdot d\vec{S} = 0 \tag{4.44}$$

Although this demand seems very simple and basic, there exist quite some problems in ensuring it and a lot of literature is devoted to this topic. The reason lies in the fact that for a real flow simulation, one knows only the velocity on one side of the domain, which is usually inlet. Velocity on the other side, outlet, has to be defined in a certain manner in order to impose some boundary conditions. Very often this is done by using convective boundary condition, like given with equation (4.45) [81, 105, 87], which is also used in the case of MFLOPS-3D. In it, U_c stands for convection velocity and $\vec{n}$ for the vector in the normal direction on the outlet. Such a definition can quickly lead to differences between mass flow on inlet and outlet, thus to the volume integral in (4.44) not equal to zero. In order to avoid this, different approaches can be used. They all have common goal which is to ensure the needed equality of mass inflow and outflow. An example, where this is well discussed, can be found in [105].

$$\frac{\partial \vec{v}}{\partial t} + U_c \frac{\partial \vec{v}}{\partial \vec{n}} = 0 \tag{4.45}$$

If this requirement is satisfied and with it also compatibility condition as given above, the Poisson equation can be solved, but infinitely many solutions exist. As mentioned in the explanation about the multidomain solver, the solution can be defined only up to an arbitrary additive constant. This can be simply explained with the fact that if equation (4.42) could be integrated to obtain Φ, the surface integral with the zero gradient shows that whatever constant C can be added to the solution of Φ and the equation (4.42) would still hold. This characteristic of Poisson-Neumann problem can also explain behaviour of the solution procedure for Φ. As it is mentioned in [107], some solvers, particulary iterative ones, can converge to a certain single solution without special treatment. But in some cases, the possibility of infinitely many solutions demands prescription of some reference value in a single point. This is often the case when direct solvers are used. Contrary to this, when compatibility is not ensured and hence solution does not even exist, the iterative solvers always exhibit non convergent behaviour while the direct ones sometimes cannot even obtain some solution. This information is important to consider in the case of MFLOPS-3D code, where a mix of direct (in mono domain) and iterative (for multi domain solution) solvers is used. As it is explained in section 4.2, chapter 3, and also in [87, 113], the use of influence matrix technique as multi domain method leads to the singularity of Poisson-Neumann problem to be expressed thorugh multi domain solution. Therefore the issues connected with compatibility are revealed in influence matrix solutions. Since this is obtained iteratively, satisfaction of compatibility leads to convergent solutions. It is also mentioned that MFLOPS-3D allows for prescription of a value in one point, which can help to improve convergence in certain cases. However, if compatibility condition is not ensured, the use of influence matrix technique leads to two issues. Firstly, it was observed that the convergence becomes worse the more the compatibility is violated. Here, it has to be mentioned that some small tolerance in violating compatibility always exists. Hence it is still possible to obtain valid solutions if compatibility is slightly violated, but the price is paid in the increased amount of iterations for influence matrix solutions. This is also in accordance with the behaviour of iterative solvers mentioned in [87]. However, not ensuring or violating compatibility more and more then leads to the presence of non convergent solutions for Φ and with them also invalid results. This issue of convergence problems is not surprising. The same can not be said for the second issue. It was namely also observed that failure to satisfy compatibility in the case of using influence matrix technique actually leads to non continuous gradients over the interfaces between sub domains. This problem is also mentioned in [87] and is particular to the use of influence matrix as multi domain method.

The last mentioned problem was also encountered constantly when the homogeneous von Neumann boundary conditions were used in the case of solving cavitating flow governing equations. The discontinuous gradient of Φ (and consequently also of all other variables) coming from not well ensured compatibility is often very hard to see in the case of incompressible flows. Some examples will be shown in a later chapter. But it can be on the other hand very pronounced in the case of cavitating flow simulations. This problem can be at first very interesting to notice, since equality of $\vec{v}^{n+1}$ and $\vec{v}^{*}$ and with it also satisfaction of equation (4.41) are imposed with von Neumann boundary conditions. And by the Gauss theorem, one could say that requirements (4.42) and (4.43) are then satisfied if previously mentioned problem of inflow and outflow equality is ensured. This is a false conclusion and the reason for it hides in the fact that $\nabla \cdot \vec{v}^{n+1}$ is not zero but depends on the value of source term S. Which also means that equation (4.44) does not apply anymore. Furthermore, S depends on pressure and therefore, as mentioned, is not known in advance. Which finally means that there exists an unknown mass imbalance. This can be clearly seen if one considers a volume integral with non zero $\nabla \cdot \vec{v}^{n+1}$, which then translates into a non zero surface integral, showing the inequality of mass inflow and outflow. And this can only

be defined after S is known. Which consequently means that equality of $\vec{v}^*$ and $\vec{v}^{n+1}$ in normal direction on the boundaries is generally not applicable any more. And with it also the zero value of Φ gradient in same directions. If such boundary conditions are nevertheless still imposed, this would cause inequality between presented surface and volume integrals. Where the surface integral would always impose equal, zero value, while the volume integral value depends on the source term. This leads to large violation of compatibility condition imposed by von Neumann boundary conditions for Φ and impossibility to find Φ solution. Therefore another boundary conditions have to be chosen if the problem of compatibility violation is to be avoided and solutions for Φ and then pressure are to be made possible.

It has to be also mentioned that the presented compatibility condition is only the continuous compatibility condition, which strictly deals with imposed boundary conditions [107]. Compatibility has to be furthermore satisfied also on the discrete level, which is much more difficult to ensure. Discrete compatibility is considered as possibly a large issue in the MFLOPS-3D code also, affecting its performance, and some work was done at the end of this thesis on it. However, understanding continuous compatibility is sufficient for explanation of the new algorithm, therefore the issue of discrete compatibility is left to be dealt with later on in the thesis.

2.2.3 Solution of the Poisson equation issues

Poisson equations used to find Φ solutions are given with equations (4.39) and (4.40). Here, these two equations are rewritten with source term included instead of $\nabla \cdot \vec{v}$, giving equations (4.46) and (4.47), as this form of equations is considered as a basis for the following explanation.

$$\Delta\Phi = -\frac{3}{2\Delta t}(\vec{v}^{n+1} - \vec{v}^*) \cdot \nabla\rho - \frac{3\rho}{2\Delta t}\left(S^{n+1}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right) - \nabla \cdot \vec{v}^*\right) \tag{4.46}$$

$$\Delta\Phi_2 = -\frac{3}{2\Delta t}\left(S^{n+1}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right) - \nabla \cdot (\vec{v}^*)\right) \tag{4.47}$$

2.2.3.1 Relaxation of compatibility constraint

As explained in the previous section, the boundary conditions for Φ cannot be homogeneous von Neumann conditions. The question is, which boundary conditions should replace them. Ideally, Dirichlet boundary conditions could be used, but in reality we cannot know the pressure values on all boundaries. At best, pressure can be known or correctly imposed in some points. It is because of this reason the von Neumann boundary conditions are more accurate choice. Although they can impose some limitations on accuracy of calculated pressure, especially the possible numerical boundary layer, approaches as the one in section 2.2.3 of chapter 3 were introduced to treat this and enable good pressure solutions to be found also on boundaries. Hence it was known that whatever other conditions would be imposed, it would mean a drop in accuracy for pressure on the boundaries. The usual mixed boundary conditions, where von Neumann conditions are imposed on all boundaries except the outlet, were chosen. On the outlet, simple constant value for Φ is prescribed. Such an approach is used in all other codes, where same modelling of cavitating flows is used, eg [38, 120, 41]. There are even codes, which use such boundary conditions also for cases of incompressible flows. But it should be mentioned, that where boundary conditions for Φ were chosen in MFLOPS-3D, other codes use conditions for p. In both cases, the pressure solution is hampered at the outlet, since constant values are imposed

on it, thus lowering the accuracy of all solutions near it. Hence a longer outlet area is required to lower this impact.

The use of mixture boundary conditions solves the problem of compatibility constraint by relaxing it, but not also completely removing it. It relaxes the compatibility since the pressure on the outlet has its value defined instead of its derivative. Since there is no zero derivative on the outlet, there can be difference between the proposed velocity $\vec{v}^*$ and finally calculated velocity $\vec{v}$ on the outlet. Hence there can be difference between mass inflow and outflow, which is caused and determined by the source term. And naturally, the surface or volume integrals in equations (4.42) and (4.44) are not zero any more, but depend on the calculated source term. However, compatibility constraint is not completely removed since there are still two directions in which only von Neumann boundary conditions are imposed. As it is mentioned in [87], von Neumann boundary conditions produce one zero eigenvalue per direction when the eigendecomposition of Poisson equation is performed. Effectively this means there are two null eigenvalues, one for y and z direction, when mixed boundary conditions are used. The presence of these two can still impose some issues with compatibility and affect solutions or solver performance. This was also noted for the influence matrix solutions for Φ, where the use of mixed boundary conditions did not lead to the expected decrease in iterations of the solver, which will be shown in a later chapter considering verification and performance tests. Regarding the iterative solver for influence matrix, it should be mentioned that for the case of mixed boundary conditions, same solver was chosen as for predictor velocity (Krylov solver with GMRES).

The solution of compatibility constraint to enable cavitating flow calculations took quite a lot of time, although it can be considered to be a well known problem with established solutions when flows with mass transfer effects are considered. But in exchange, there were some interesting proposals also developed. Two of them are mentioned here, but not described in detail, since they are not the subject of the thesis yet they could be interesting for use in some flow cases. In the first, the von Neumann boundary conditions were kept. The problem of volume integral of Poisson equation not being zero was cleared by first defining the value of this volume integral V_P, as given with equation (4.48). Then, the value was divided by volume of the whole domain Vol and the result was discounted from the Poisson equation, as shown with equation (4.49). In this manner, the volume integral of Poisson equation is zero and a solution can be found even if compatibility would be otherwise violated. Such solution is actually used in some codes for incompressible flow, when the compatibility is not satisfied by chosen methods.

$$V_P = \int \left(-\frac{3}{2\Delta t}(\vec{v}^{n+1} - \vec{v}^*) \cdot \nabla\rho - \frac{3\rho}{2\Delta t}\left(S^{n+1}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right) - \nabla \cdot \vec{v}^* \right) \right) dV \tag{4.48}$$

$$\Delta\Phi = -\frac{3}{2\Delta t}(\vec{v}^{n+1} - \vec{v}^*) \cdot \nabla\rho - \frac{3\rho}{2\Delta t}\left(S^{n+1}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right) - \nabla \cdot \vec{v}^* \right) - \frac{V_P}{Vol} \tag{4.49}$$

The other proposal is also based on using homogeneous von Neumann boundary conditions, but not in all pressure computations. In verification tests performed, which are described in chapters 5 and 6, it was observed that even if homogeneous von Neumann boundary conditions are used, the first solutions for pressure are still accurate on the boundaries. It is after some time steps that solutions become so affected by the error caused by compatibility condition that they are also inaccurate on the boundaries. Therefore a mix of two solvers for Poisson equation was proposed. At first, equation (4.46) is solved using homogeneous von Neumann boundary conditions. Then, obtained Φ solutions on the boundaries are imposed as Dirichlet boundary conditions and final solution for Φ is raised, together with an updated source term. Solution for

Φ is in both proposals otherwise found in iterative steps, during which the velocity $\vec{v}$ and source term S are updated. For the first solution, explicit values of S are used.

Interestingly, both proposals gave similar improvement over the simple use of homogeneous von Neumann conditions. There were actually many tests performed with such homogeneous conditions and quite some procedures proposed, which also form the basis on which the following final procedure was developed. All of such procedures of course worked if the source term was kept at low values. With the two proposals, the source term could be higher, but still not without limits. This was achieved only when compatibility condition was relaxed with the use of mixed boundary conditions. Nevertheless they are both interesting since they give an option to have not very low source term included in simulations where homogeneous von Neumann conditions are used and hence more accurate pressure solutions can be obtained at the boundaries.

2.2.3.2 Source term and velocity update

The relaxation of compatibility condition was one of the two main problems which had to be resolved in order to successfully solve for Φ in cavitating flow case. The other was the need to keep the LHS of Poisson equation constant, which meant inability to directly treat change in source term implicitly as is done in the shown SIMPLE algorithm example. There was also the drawback of using explicit density and velocity $\vec{v}$ in equation (4.46) but this was found to not be very problematic. As mentioned before, a lot of solution procedures were proposed in order to resolve the issues with source term. To explain how the final solution procedure form for Φ was obtained, equation (4.46) will be used, since it was found to give better results and also includes the need to update velocity $\vec{v}$. This is not present in the case of equation (4.47). Equation (4.46) should for this purpose be considered with added notation for explicit terms, resulting in equation (4.50). Here, known superscripts $n, n+1$ depict the time level, while superscripts k and $k+1$ are introduced in order to differentiate between different iteration levels. This is needed since the obvious approach to the problem is performing iterating sweeps to obtain actual values of otherwise explicitly treated variables.

$$\Delta\Phi^{k+1} = -\frac{3}{2\Delta t}(\vec{v}^{n+1,k} - \vec{v}^{*}) \cdot \nabla\rho^{n} - \frac{3\rho^{n}}{2\Delta t}\left(S^{n+1,k}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right) - \nabla \cdot \vec{v}^{*}\right) \tag{4.50}$$

For the moment, density ρ in equation above is considered as taken from previous time level, since the α transport equation is the last step in the algorithm and cannot be used until pressure p^{n+1} and with it S^{n+1} are defined. As can be predicted from previous description, this dependence between p and S was in the core of here described problem. Since the LHS of Poisson equation has to be constant, it was at first tried to perform iterations around (4.50) and update $\vec{v}^{n+1,k}$, $p^{n+1,k}$ and consequently $S^{n+1,k}$ after each iteration. The velocity update applies projection equation (4.32) as equation (4.50) is used, while pressure update is then based on the use of equation (4.34). The source term was, after definition of $p^{n+1,k}$, updated by using new pressure in its equation. The iterations proved to be too unstable whenever this update was considered and even if pressure was under relaxed. Velocity updates on the other hand improved the results. Hence the first algorithms were proposed with only velocity update being done during solution for Φ. This imposes a lot of potential stability issues, since the source term used would be always completely explicit. It was noted that simulations are stable only if S is small and with it also the α range, which agrees with observations about problems imposed with the use of explicit S in [8], mentioned in section 1.2. Therefore another solution was needed.

When performing various validation tests, which will be described in chapters 5 and 6, it was observed that ability to have some valuable prediction for the new source term (and therefore not using completely explicit S) gives some improvement of the stability. And the closer such treatment of the source term is to the ideal implicit treatment shown in section 1.2, even if explicit values are used in it, the better are the results. This finding was then used to form the first sufficiently stable solution procedure for Φ. When making the prediction for S to be used in equation (4.50), it was found that a proposal for S in form of linearised source term, as used in [8] and given with equation (4.19) in section 1.2, is not enough. Since the tests performed also included variable α in time and place, and the source term is generally not just a function of pressure but also of α, it was noted that even better proposal for it can be obtained if the linearisation is done completely, that is, also in regards to α. This complete linearisation of source term is shown with equation (4.51). However, because iterations in question are done only around Poisson equation for Φ and do not include also solution for α, the linearisation of S in regards to α can be of little help, since $d\alpha$ and with it the additional term would, on the first look, be always the same. But as the point is to have as good prediction for S as possible, where only Φ can affect this prediction, it was then tried to express $d\alpha$ as a function of pressure change. This would mean that $d\alpha$ is then also a function of Φ. If this can be done, it can be used to achieve two objects. On one hand it would make a better prediction for the source term and hence better stability when solving the Poisson equation. On the other, it would increase the presence of Φ on the RHS of equation (4.50), hence make for a possibly better tool when performing iterations around this equation. The $d\alpha$ can indeed be written as a function of pressure change dp. In order to achieve this, the α transport equation (4.3) has to be written with the use of completely linearised source term from equation (4.51). Moreover, the divergence term in it has to be split by using derivative product rule so that the resulting velocity divergence term can be replaced by source term. The remaining convection term can then be written together with the time derivative as material derivative. Hence equation (4.52) is obtained. In it, α value has for the moment no superscript, but it is treated explicitly (values are taken from the last solution).

$$S^{n+1} = S^n + \frac{\partial S^n}{\partial p}dp + \frac{\partial S^n}{\partial \alpha}d\alpha \tag{4.51}$$

$$\frac{d\alpha}{dt} + \alpha\left(S^n + \frac{\partial S^n}{\partial p}dp + \frac{\partial S^n}{\partial \alpha}d\alpha\right)\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right) = \frac{S^n + \frac{\partial S^n}{\partial p}dp + \frac{\partial S^n}{\partial \alpha}d\alpha}{\rho_v} \tag{4.52}$$

The goal of having equation above is to express $d\alpha$ from it. Which means that $d\alpha$ is chosen as value defined from material derivative. This was found to give good results and was therefore kept. Overall, the use of such approach can be also seen as using complete α change which was predicted using the known local changes in time and changes because of convective effects, which makes for a more effective source term prediction. Trying to express $d\alpha$ only from the time derivative would on the other hand make the development and also use of the final expression more cumbersome. The expression for $d\alpha$, following from rearrangement of terms in equation (4.52), is given in equation below.

$$d\alpha = \frac{S^n + \frac{\partial S^n}{\partial p}dp}{\frac{1}{K} - \frac{\partial S^n}{\partial \alpha}} \quad ; \quad K = \Delta t\left(\frac{1}{\rho_v} - \alpha\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right)\right) \tag{4.53}$$

The $d\alpha$ as defined with (4.53) was then used in completely linearised source term in equation (4.50) in order to achieve mentioned better prediction for S. An important aspect here is also the connection between dp and Φ, because of which the second mentioned possibility for improved

solution procedure also exists (increase of Φ presence on RHS). The connection follows from the $p-\Phi$ connection and is clear in the case of pressure incremental version of the algorithm, since Φ in it gives directly dp. In the non incremental version, the connection is more difficult to be defined. For it, equation (4.34) is used without the p^n, since non incremental version is considered. But as the pressure difference is sought, p^n finds its place in the equation through being deduced from both sides, which gives the equation (4.54). This is not yet the final expression for $dp-\Phi$ connection in this version. Completely linearised S can be used instead of $\nabla\cdot\vec{v}^{n+1,k+1}$, and together with $d\alpha$ as expressed above, this results in additional terms which include dp. If this are put together, the final expression for dp is found and given in equation (4.55).

$$dp = p^{n+1,k+1} - p^n = \Phi^{k+1} + \mu\nabla\cdot(\vec{v}^{n+1,k+1} - \vec{v}^*) - p^n \tag{4.54}$$

$$dp = \frac{\Phi^{k+1} + \mu\left(\left(S^n + \frac{\partial S}{\partial\alpha}\frac{S^n}{\frac{1}{K}-\frac{\partial S}{\partial\alpha}}\right)\left(\frac{1}{\rho_v}-\frac{1}{\rho_l}\right) - \nabla\cdot\vec{v}^*\right) - p^n}{1 - \mu\frac{\partial S}{\partial p}\left(\frac{1}{\rho_v}-\frac{1}{\rho_l}\right) - \mu\frac{\partial S}{\partial\alpha}\frac{\frac{\partial S}{\partial p}}{\frac{1}{K}-\frac{\partial S}{\partial\alpha}}\left(\frac{1}{\rho_v}-\frac{1}{\rho_l}\right)} \tag{4.55}$$

In this definition of dp one has to pay attention to Φ which is defined with $k+1$ superscript. This is so, because this equation is build from using equation (4.54), which assumes that actual Φ and with it also velocity $\vec{v}^{n+1,k+1}$ are known. But since the equation is used in prediction of source term value, Φ in it is treated explicitly (taken either from previous time step or iteration level).

With these improvements, that is, introduction of completely linearised S where $d\alpha$ is expressed as a function of dp and therefore Φ, first stable simulations were obtained. It was finally possible to have source terms with higher amplitudes, hence higher mass transfer, and also α which varied from $0-1$ in whole domain, even instantaneously. However, the presented improvements only enabled this after introduction of under relaxation for Φ. Which made simulations much longer, as besides already performing iterations around equation (4.50), even more iterations were now needed since the convergence of Φ was by default slower. Therefore it was decided that although the algorithm is able to give good results, even additional improvement of it has to be done.

The idea to include and then also increase the presence of Φ on the RHS led to the final proposed solution for Φ. This was to introduce Concus and Golub method also to this case. Since linearised source term can be now in both incremental and non incremental version of the projection method expressed as function of Φ, the Poisson equation can be written effectively as Helmholtz equation with non constant coefficients. Momentum equations for $\vec{v}^*$ present same type of equations and Concus and Golub method was used to obtain their solutions successfully. Therefoire it was tried also in solving for Φ, in order to make the solution more implicit and thus improve convergence. With it, originally considered Poisson equation is ultimately changed into Helmholtz equation with a constant coefficient on the LHS. And of course, with the same constant also on the RHS. This constant is given with σ_p and Helmholtz equation resulting from its addition in the case of using (4.50) is given with (4.56). σ_p has to be chosen carefully, since it determines implicitness of the procedure and can as such also importantly affect convergence characteristics. This effect of the constant in Concus and Golub method is well shown in [112] and the choice for σ_p also follows there observed results. As Φ can be with completely linearised source term included on a greater level on RHS, one has many options for its proposal. It was found that the best performance is achieved when σ_p is based only on the part of source term which comes from linearisation in regards to pressure, $\frac{\partial S}{\partial p}$. The reason is that this part is, depending

on chosen cavitation model source term, largely or solely a function of α. Hence, it is possible to form σ_p which fits it reasonably well over whole domain, as the possible range of α is always kept between $0-1$ (which defines also the possible range of ρ and μ). Contrary to this, the range of pressure, which largely defines $\frac{\partial S}{\partial \alpha}$, cannot be predicted. Moreover, $\frac{\partial S}{\partial p}$ term has greater effect on whole source term change and therefore also Φ solution. The best results with such σ_p are possible if constant value for α in it is well adjusted to certain instantaneous range of α in the domain. Here, a question arises, since α is normally zero in large part of the domain, but use of zero value would actually make σ_p also zero in any cavitation model. And this would mean no improvement with Concus and Golub method for the cavitation regions, where the more implicit Φ solution procedure is needed. But if the constant α is chosen with higher values, in order to make solution more implicit for the cases of regions with higher α being present, the procedure would suffer from too many iterations or maybe also worse stability in regions with α closer to zero. It was finally found that simple choice of middle value of $\alpha = 0,5$ gives best results over all possible α ranges in the whole domain, therefore σ_p was defined with it. In order to give a general form of σ_p expression, it should be considered that $\frac{\partial S}{\partial p}$ is multiplied also with coefficient depending on the time derivative discretization $\left(\frac{3}{2\Delta t}\right)$, ratio of densities $\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right)$ and most importantly by dp. σ_p has to be in the end multiplied with Φ and not dp, in order to actually implement Concus and Golub (from here on CG) method. And Φ comes from this pressure change. In the case of pressure incremental version of the algorithm, the equality $dp = \Phi$ directly gives the form for σ_p in equation (4.57). Superscript $0,5$ in it depicts the use of $\alpha = 0,5$ value.

$$(\Delta + \sigma_p)\,\Phi^{k+1} = -\frac{3}{2\Delta t}(\vec{v}^{n+1,k} - \vec{v}^{*}) \cdot \nabla \rho^n - \frac{3\rho^n}{2\Delta t}\left(S^{n+1,k}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right) - \nabla \cdot \vec{v}^{*}\right) + \sigma_p \Phi^k \tag{4.56}$$

$$\sigma_p = \frac{3\rho_{0,5}}{2\Delta t}\frac{\partial S}{\partial p}_{0,5}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right) \tag{4.57}$$

In the case of pressure non incremental version of the algorithm, Φ has to be first expressed from (4.55). This expression had to be adjusted, because it includes $\frac{\partial S}{\partial \alpha}$ terms in the denominator. This has unpredictable range coming from the presence of pressure in it. Therefore another expression for dp was proposed, where the problematic terms were put into numerator and had explicit values of dp applied. This expression is shown in (4.58). dp obtained from it is divided only by terms that have well predictable range and are more suitable for use in CG method. General form of σ_p for non incremental version is then given with expression (4.59).

$$dp = \frac{\Phi^{k+1} + \mu\left(\left(S^n + \frac{\partial S}{\partial \alpha}\frac{S^n + \frac{\partial S}{\partial p}dp}{\frac{1}{K} - \frac{\partial S}{\partial \alpha}}\right)\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right) - \nabla \cdot \vec{v}^{*}\right) - p^n}{1 - \mu\frac{\partial S}{\partial p}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right)} \tag{4.58}$$

$$\sigma_p = \frac{3\rho_{0,5}}{2\Delta t}\frac{\partial S}{\partial p}_{0,5}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right)\frac{1}{1 - \mu_{0,5}\frac{\partial S}{\partial p}_{0,5}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right)} \tag{4.59}$$

With introduction of CG method the convergence of iterative procedure for Φ improved considerably. Especially as no under relaxation of Φ results is needed any more. In fact, as such an improvement is caused by σ_p constant following only from $\frac{\partial S}{\partial p}$ term, there is no need to use completely linearised source term any more. It was also observed that accuracy and performance with CG method can be in pressure incremental version slightly improved if only linearisation of

the source term in regards to pressure is applied. The difference is on the other hand practically negligible in the pressure non incremental version. Therefore $\frac{\partial S}{\partial \alpha}$ term was actually skipped in the simulations from which the later presented verification results in chapter 6 follow. But the following presentation of the algorithm and its versions is nevertheless given with it included. This makes for a more complete summary and was chosen also because of the success the complete linearisation offered before CG method was implemented. Moreover, the algorithm proceeds in same manner and with same equations if S is linearised completely or only in regards to pressure. For the latter, only the $\frac{\partial S}{\partial \alpha}$ term needs to be set as zero in the shown equations, no other change is needed.

Furthermore, it was found that inclusion of CG method does not demand specific iterations for Φ only. This means that in the case of using equation (4.56), velocity $\vec{v}^{n+1,k}$ can be updated after each new solution for Φ. Inclusion of another iterative loop for CG method iterations, where only Φ is updated, actually gave no improvement in stability or accuracy of results, it only resulted in longer simulations. This possibility also contributed to much faster simulations with inclusion of CG method than with the first stable procedure without it. The velocity update applied during the iterations for Φ when equation (4.50) or (4.56) is solved, is done using the projection equation (4.32) and is given with the correct superscripts in (4.60). As stated before, the velocity update is not needed during iterations for solution of other Poisson equation, equation (4.47). Velocity is updated only after the converged Φ_2 (and with it also Φ) is obtained, with equation (4.61). However, solution of it includes same treatment of source term update issues with CG method. Only the $dp - \Phi$ connection is slightly different as it follows from equation (4.37) and not (4.34). Interestingly, σ_p is in this case same as shown here, for both incremental and non incremental version. This is also the reason why the inclusion of CG method for this other equation is omitted here. The convergence criteria for Φ is also the same for both equations. Converged Φ solutions are reached when the differences between two successive iterations are in each point smaller than 10^{-3}. This is higher than the chosen criteria for $\vec{v}^*$ solution. The reason is that stricter criteria was not found to improve results, only increase amount of iterations for Φ. This was especially unwanted since the issues with compatibility condition are still present for influence matrix solution. Same convergence criteria was also applied in first stable Φ solution procedure.

$$\vec{v}^{n+1,k+1} = \vec{v}^* - \frac{2\Delta t}{3\rho^n}\nabla\Phi^{k+1} \tag{4.60}$$

$$\vec{v}^{n+1,k+1} = \vec{v}^* - \frac{2\Delta t}{3}\nabla\Phi_2^{k+1} \tag{4.61}$$

Regarding the stability of solution procedure for Φ, it was shown that highest possible stability was achieved. This was done with performance comparison of here presented algorithm and its specific version where Dirichlet boundary conditions for Φ are used. Comparison test were done with verification method enabling the use of Dirichlet conditions for Φ and it was found the two versions express same stability. The verification method is presented in chapter 5 while the mentioned results will be discussed in chapter 6.

The point about stability can be used also to make another argument. The shown solution procedure with linearised source term and CG method applied reminds of the solution procedure shown with SIMPLE algorithm. Which is no surprise as same governing equations are being solved. But it can be seen that same results were achieved in a different, more complex manner, over development of various proposals. The argument could be also raised that for the case of pressure non incremental version, the source term does not need to be linearised. CG method

could be applied directly through the pressure included in S itself, as Φ represents a considerable part of p. This was tried, but results were not encouraging or even caused unstable simulations. It was concluded that this is caused by higher source term changes during iterations of CG method. Such application of CG method namely then operates with complete source term changes and not just with the changes which are smaller and applied to a good previous basis. Therefore the use of linearised source term and application of CG method through it is a better, more stable choice. For clarity, it can be said that such application is also the only one possible for the case of pressure incremental version, since $\Phi = dp$ in it.

The improvements regarding solution speed and stability found with the inclusion of CG method finally mean that the problem of ensuring source term update and keeping the LHS of Poisson equation constant was solved. With this, tools for successful solution procedure for Φ are given. There were actually two different procedures for Φ finally set, both possible in pressure non incremental and incremental version of the algorithm. These, in total four procedures, are presented after the presentation of the final step in the projection method, the solution for α.

2.3 Solution of vapour volume fraction α

The solution of the transport equation for α is the last step in the projection method and algorithm. With the obtained α^{n+1} from it, the updates of ρ^{n+1} and μ^{n+1} follow, which use equations (4.4). Although the equation can be used already to form $d\alpha$ expression in Φ solution and a procedure could be devised to extract α^{n+1} from $d\alpha$, it is better if a solution is done separately, after Φ is obtained. Strong reason for this is that $\vec{v}^*$ values used in solution for Φ were obtained with an explicit value of ρ. Which means a non wanted effect on an update and use of α^{n+1} during iterations for Φ. However, it is worth to mention that other algorithms, for instance those mentioned in [8, 120], can include solution for α as the first step, before solving for $\vec{v}^*$. This is referred to as α (and ρ, μ) update and makes practically no difference to having this solution included as the last step. In both cases, values of α, ρ, μ in solutions for $\vec{v}^*, \Phi, \vec{v}^{n+1}$ will follow from the use of velocities and source term from equal time or iteration levels.

It was found that α solution can be obtained in a quite simple manner, without use of a certain solver. Indeed the α transport equation (4.3) cannot be reshaped into a Helmholtz or Poisson equation, solved by the solver in the code, therefore another solver should be applied. For instance, time stepping ordinary differential equation integrator provided by PETSc was tried (TS, forward Euler [115]), but the obtained solutions were not acceptable. Since practically all variables in equation (4.3) are known, it was tried to find a solution for each point separately. This actually results in the transport equation solved in a similar manner as provided with explicit methods used for advection equations, such as Lax-Wendroff, Lax-Frierdrichs or upwind scheme [122]. However, as α transport equation is not a simple advection equation, it is preferred to relate to the devised solution in a different manner. Because the approach was found to work well and convergence results showed no notable effect on precision, it is retained in the code as α solution procedure. The procedure is based on use of equation (4.3), developed into equation (4.62). α^{n+1} in time derivative and second term can be expressed together, leading to equation (4.63), which is then solved iteratively for each point until converged solution is found. Hence also superscripts $k, k+1$, which determine terms updated during iterations. As source term is a function of α, it is updated although a good value for it is found already in Φ solution.

$$\frac{3\alpha^{n+1} - 4\alpha^n + \alpha^{n-1}}{2\Delta t} + \alpha^{n+1} S^{n+1} \left(\frac{1}{\rho_v} - \frac{1}{\rho_l} \right) + \vec{v}^{n+1} \cdot \nabla \alpha^{n+1} = \frac{S^{n+1}}{\rho_v} \tag{4.62}$$

$$\alpha^{n+1,k+1} = \frac{\frac{S^{n+1,k}}{\rho_v} + \frac{4\alpha^n - \alpha^{n-1}}{2\Delta t} - (\vec{v}^{n+1} \cdot \nabla)\alpha^{n+1,k}}{\frac{3}{2\Delta t} + S^{n+1,k}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right)} \tag{4.63}$$

The convergence criteria for α solution was set to be the same as for $\vec{v}^*$. Converged solution is reached when difference between two successive iterations becomes smaller than 10^{-8} in each point. Although such results are obtained after 30 iterations at most (if α varies greatly in the domain), it has to be considered that solution is obtained for each point separately and the solution procedure is therefore faster than in other cases where solvers in the code are used. α solution thus presents the fastest step in the algorithm. The amount of iterations is partially a consequence of resemblance with explicit methods used for solution of advection equation, but more importantly it is caused by relatively high under relaxation for $\alpha^{n+1,k+1}$. This is done with equation (4.64) after each solution of equation (4.63). Tests showed that factor $d = 0,3$ returns stable results for all possible ranges of α in a domain. Without under relaxation the solution is unstable.

$$\alpha^{n+1,k+1} = d\alpha^{n+1,k+1} + (1-d)\alpha^{n+1,k} \tag{4.64}$$

Another, more problematic issue with the stability of α solution was also encountered and was found not to be caused by use of equation (4.63). It was observed that issues come from the advective term in equation (4.62), more precisely from the gradient in the advective term $\nabla\alpha$. This expresses strong Runge phenomena [88] which can be explained with two points. First, compact finite difference schemes tend to express this phenomena strongly at their interval limits. Since stencils of compact schemes do not overlap between sub domains there can be oscillations caused by Runge phenomena present globally. Furthermore, this phenomena states that whenever polynomial approximations (on which compact schemes are also based [81]) are used to describe certain functions on an interval,the final result tends to show higher oscillations close to the edges of the interval if polynomials of higher orders are used [88]. Which means that the oscillations are worse with higher order schemes. Their effects are made even worse by the second point, which is that since the solver in the code is not used for α solution, the results are not guaranteed to have identical values and continuous gradients on the interfaces. Therefore discontinuous results on interfaces are obtained, as shown on figure 4.2. This makes the Runge phenomena even more pronounced over time steps or iterations and finally leads to completely unstable solution procedures. Even if some other solver and not equation (4.63) is used, the same problem occurs, since it is fundamentally conditioned by compact schemes. What is even more concerning than these issues appearing in α solution alone is that ρ and μ gradients, which actually feature α gradient, are used in all previous steps in the algorithm. Therefore all solutions become highly unstable.

Different approaches were used in order to resolve the issue. At first, ability to use different schemes for derivatives was implemented. This means that derivatives with different orders of accuracy can be used in the code for different variables. For example, derivatives with higher order accuracy were applied for velocity and pressure. On the other hand, 2^{nd} or 4^{th} order schemes were used for gradients of α_v, ρ or μ. This resulted in smaller discontinuities of α related variables on the boundaries. An example can be seen on figure 4.3, where the same case is shown as in figure 4.2, just that second order accurate scheme was used.

Nevertheless, the discontinuities were still present and grew in time. To get rid of them completely, a routine which forced the values on an interface to be equal was imposed. Routine simply forces the values on the interface to be equal to the average value of interface solutions

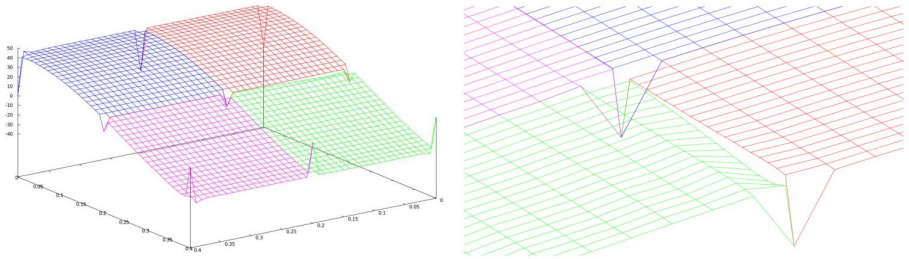


Figure 4.2: Discontinuities of density gradient in y direction on xy plane as a result of using the 8^{th} order compact schemes.

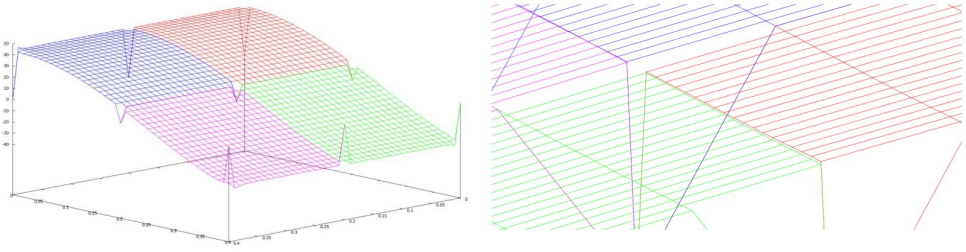


Figure 4.3: Smaller discontinuities of density gradient in y direction on xy plane as a result of using the 2^{th} order accurate compact scheme.

found in two neighbouring sub domains. Although the values on interfaces were made equal, the derivatives were not, which resulted in persistence of stability problems.

From this it followed that a solution should be done without compact schemes from MFLOPS-3D. The problematic derivatives in the advection term were therefore proposed to be changed with simple discretization schemes used in already mentioned explicit methods for advection equation. The main advantage of them is the simplicity to define interface derivatives which use data from next sub domain as only smaller amount of data needs to be communicated between processors (one or at most two values per a derivative in a point). Central difference schemes as used in Lax-Friedrichs method [122] treated the problem of value and gradient discontinuities, but caused effect of odd-even decoupling. Finally it was found that upwind schemes treat all of these issues and enable stable α solution. Second order upwind scheme was applied. The order of accuracy is smaller than provided by compact schemes, but convergence tests showed that this has only small effect on final result. The chosen scheme is adapted for the use on non uniform grids and follows from [125].

Implementation of such discretization scheme for advection term in α equation means that derivatives of α and related variables in other equations should be done in the same way in order to ensure compatibility of derivatives. Therefore all derivatives of ρ and μ used in $\vec{v}^*$ and Φ solution are done with the same second order upwind scheme. Which is believed also to be one of the reasons why projection equation (4.35) introducing Φ_2 did not produce as good results as first one, equation (4.32). When (4.35) is used, pressure is obtained after multiplication of Φ_2 with ρ. Because of the upwind scheme, this has lower order of accuracy which then directly affects accuracy of pressure solution and with it also velocity (through source term). The other reason is in the α errors on the boundaries of computational domain in calculations performed in this work. There, these errors can express sudden increase, which then again directly affects errors in pressure and velocities. This might not be connected only with the upwind scheme, but it does

show higher effect on results when Φ_2 is used. On the other hand, use of (4.32) leads to ρ and its gradients being present in Φ solution, which can result in issues with higher dissipation caused by this second order upwind scheme. It is assumed that if a higher order spatial discretization scheme was used for α, solutions obtained with the use of Φ_2 could offer an improvement over the solutions obtained with equation (4.32). Which is also one of the reasons why projection equation (4.35) is kept in the code.

2.4 Developed versions of the new algorithm

Previous section shows how the new algorithm for successful solution of cavitating flow is built in general. Differences between two main versions, pressure incremental and non incremental are also given. However, as the equations in presented steps use explicit ρ and μ, the solutions obtained with them depend also on the quality of these explicit terms. Tests showed that firstly satisfaction of continuity and secondly order of accuracy can be seriously hampered by this, therefore iterations around all previously presented steps were proposed. In this manner, updated ρ and μ after each α solution can be used in all steps of the projection method in the algorithm, from $\vec{v}^*$ to Φ solution. Since the introduction of such iterations means that there are two levels of iterations, one involving all equations and other which concerns each solved equation, they should be referred to as outer and inner iterations, respectively. The introduction of outer iterations makes simulations slower, but importantly improves satisfaction of continuity and order of accuracy. What is more, it was also found that algorithm can in both versions work in two different manners depending on the way in which pressure or Φ is solved. All in all this makes for four possible versions of the algorithm.

The difference between the two ways in which Φ is obtained depends on the way in which source term is treated. In explanation of the Φ solution procedure it was mentioned that Φ and $\vec{v}$ in equation (4.56) are updated during inner iterations for Φ. Which means that source term $S^{n+1,k}$ is also updated. However, if the reader did not observe before, it should be noted that only the terms following its total linearisation, given in equation (4.51), are updated. The basis, term depicted as S^n, stays the same during iterations for Φ. It was already mentioned that if this term is updated as a whole, the solution can become unstable. The reason was found to be in the higher changes which would be applied to the source term in this way. Although this was mentioned for the case of CG method implementation where constant σ_p is defined from complete S update, which is possible only for pressure non incremental version, the complete update of source term during CG iterations can be done in both versions of the algorithm. And results are the same whenever such an update of S is done, as again the changes are not always applied to a good previous basis. Which makes it much harder to find a good and converged Φ, resulting then in actual source term and velocity update. However, since outer iterations are also performed, it is possible to treat the basis for the source term in two manners. Either the basis is always the same and presented by source term value from previous time step, or it is updated during outer iterations and thus presented by source term obtained in last outer iterative step. The latter approach demands more attention to be correctly implemented. But since the source term basis is in it updated, it is closer to the SIMPLE algorithm for cavitation presented in [8, 4], while the other approach works in a more different manner.

The following sections give a better presentation of both pressure incremental and non incremental versions of the algorithm, where separate explanation is included for the two devised approaches in treatment of source term and Φ solution. The presentation is given as an overview of the algorithm steps with equations that are solved and also with a diagram showing these steps

graphically. Some definitions, which are equal in all versions, are mentioned at first. The used superscripts have same meaning as before, but have an additional superscript i used in order to depict the outer iteration level ($i+1$ is the current level, $i=1$ presents the first outer iteration). The presentation is based, in the same manner as explanation of devised Φ solution, on using equation (4.32) as the projection equation.

2.4.1 Equalities in all versions of the algorithm

As equations in all versions come from same origins, they share or demand some equal definitions. All versions of the algorithm start by prescribing boundary conditions and explicitly treated variables. Both of which depend on which outer iteration step is being done. In the first step, final values from previous time step, depicted with superscript n, are used to prescribe explicitly treated variables. In the second and following outer iterations, the values obtained in previous iteration are used for these variables, which are otherwise depicted with superscript $n+1,i$. In the first outer iteration this superscript therefore equals n, which is also illustrated in equation (4.66). Since there are also inner iterations performed in solutions for $\vec{v}^*$, Φ and α, it should be precised that these three variables and some of the variables or terms that depend on them and are changed during these inner iterations (like $\vec{v}$ in Φ) are at first also given values from previous time step or previous outer iteration. But they are then updated during their solution procedure, as written in previous sections. Their superscripts have k notation used instead of i, to depict this dependence on inner iteration level. This superscript equals n when $k=1$ and $i=1$. And if $k=1$ but $i\neq 1$, the superscript equals that of other explicitly treated variables, $n+1,i$. A slight exception among the explicitly treated variables is the non linear (NL) term. As given in section about NL term, this is now composed of time and spatial derivative terms (all convective terms from NS equations). Therefore not whole NL term has to be adapted, some parts which come from discretized time derivatives and therefore refer only to previous time levels are always the same. But parts which concern or use variables for the current time level are in the first outer iteration approximated by one step Adams-Bashforth method, while they are in all following outer iterations defined directly from values of obtained velocity and density in previous outer iteration, $\vec{v}^{n+1,i}$ and $\rho^{n+1,i}$. The used superscripts are the same as for other explicitly treated variables. Adams-Bashforth method is therefore used only for the NL term and only in first outer iteration.

A similar approach as for explicit variables is also used in defining boundary conditions. Those which are defined in same manner in all versions of the algorithm are boundary conditions for $\vec{v}^*$ and α. Both are of Dirichlet type. α has always the same known value on the boundaries prescribed, for all outer iterations. Therefore one definition of it suffices. Boundary conditions for $\vec{v}^*$ on the other hand are partially adapted for each outer iteration. Velocity in normal direction always equals the real velocity, as given with equation (4.65). Therefore it is always the same for outer iterations. Since values of velocity in tangential direction on the boundaries are prescribed with the use of projection equation, in which the values of Φ and ρ have to be taken as explicit, this boundary condition can be adapted during outer iterations. According to described treatment of explicit variables, values for Φ and ρ in this boundary condition are first taken from the previous time step results, while for second and later outer iterations they are based on the obtained results for these two variables. Therefore the boundary condition in tangential direction can be written with equation (4.66).

$$\vec{v}^{*,k+1}\cdot\vec{n}=\vec{v}^{n+1}\cdot\vec{n} \tag{4.65}$$

$$\vec{v}^{*,k+1} \cdot \vec{\tau} = \left(\vec{v}^{n+1} + \frac{2\Delta t}{3\rho^{n+1,i}} \nabla \Phi^{n+1,i} \right) \cdot \vec{\tau} \; ; \; if \; i = 1 \rightarrow \{n+1, i\} = \{n\} \tag{4.66}$$

Another equality used in all versions of the algorithm is, contrary to the definition of explicit variables and boundary conditions, present at the end of a certain outer iteration step. This is the continuity check, which determines if the simulation can proceed to the next time step. The continuity check is based on convergence of continuity equation. The convergence criteria is not chosen in same manner as for equations with inner iterations, where absolute difference of a value in each point during two successive inner iterations has to be lower than a prescribed limit. Instead, convergence uses volume integral of continuity equation residuals, as defined in equation (4.67).

$$R_{cont}^{i+1} = \int \left(\frac{3\rho^{n+1,i+1} - 4\rho^{n} + \rho^{n-1}}{2dt} + \vec{v}^{n+1,i+1} \cdot \nabla \rho^{n+1,i+1} + \rho^{n+1,i+1} S^{n+1,i+1} \left(\frac{1}{\rho_v} - \frac{1}{\rho_l} \right) \right) dV \tag{4.67}$$

It can be noted that the velocity divergence is replaced by source term, since these replacement is used over all steps of the algorithm. Also, the gradient of density is done with upwind scheme, since all α related variables use such spatial discretization. The convergence and with it continuity is said to be satisfied if the ratio cr between the current volume integral absolute value R_{cont}^{i+1} and highest volume integral absolute value in previous outer iterations, R_{cont}^{max} is smaller than 0.01. This ratio is also illustrated with equation (4.68).

$$cr = \frac{R_{cont}^{i+1}}{R_{cont}^{max}} \tag{4.68}$$

Such a continuity check was chosen as same levels of convergence cannot be reached with continuity equation as in equations with inner iterations, neither would same convergence criteria as used in those cases be very meaningful. The reason is that the other criteria gives a difference between two results in all points, while here it is much more important that continuity equation equals zero, locally and globally, for which the chosen convergence check gives a good estimate. The value of $0,01$ was chosen as it was observed that it gives a good estimation for continuity convergence and satisfaction in all performed simulation cases. Reaching smaller value was often very difficult and even impossible. In accordance with this, it was also found that 3 outer iterations are sufficient to achieve at least such order of continuity equation convergence. Performing more outer iterations did not improve continuity notably, even if the actual imposed limit was reached, which means that results were also not improved and only simulation time increased. Since the convergence of continuity was observed to be monotonic, and highest global order of accuracy for the algorithm was already reached after 3 outer iterations, another limit for outer iterations was also imposed by performing only 3 outer iterations. Finally this means that in all cases, at most 3 outer iterations are done, or 2 if the limit of $0,01$ is achieved already with the second outer iteration.

A similar continuity check is also done in commercial package Ansys Fluent, where a sum of absolute mass creation rate in all cells is taken as the estimation for a continuity equation residual. Usual convergence criteria set there is that this sum should drop to $0,001$ value of its highest value in first 5 outer iterations [38].

2.4.2 Pressure incremental version of the algorithm

2.4.2.1 Constant source term basis

This version of the algorithm was actually first stable version developed. After the definition of explicitly treated variables and boundary conditions, the first step, solution for $\vec{v}^{*,k+1}$, is here given with equation (4.69). The viscous terms on its RHS are put together for simplicity, since all variables in them are treated in same explicit manner. It can be seen that the source term is used to replace divergence of velocity in terms connected with the bulk viscosity. There are now two, since derivative product rule used on the bulk viscosity term enables summation of one resulting part with similar term in viscous terms. The replacement of $\nabla \cdot \vec{v}$ through S connection is done in order to have equality between the two terms better imposed.

$$\begin{aligned}\left(\Delta - \frac{3\rho_l}{2\Delta t \mu_l}\right)\vec{v}^{*,k+1} =& \frac{1}{4\Delta t \mu^{n+1,i}}\left(-4\vec{v}^{n}\left(\rho^{n+1,i}+\rho^{n}\right)+\vec{v}^{n-1}\left(\rho^{n+1,i}+\rho^{n-1}\right)\right)+\\ &\frac{1}{2\mu^{n+1,i}}\left(\nabla\cdot\left(\rho\vec{v}\vec{v}\right)+\rho\vec{v}\cdot\nabla\left(\vec{v}\right)\right)^{n+1,i}+\frac{\nabla p^{n}}{\mu^{n+1,i}}\\ &-\left(\frac{1}{\mu}(\nabla\mu)\cdot(\nabla\vec{v})+\frac{1}{\mu}(\nabla\mu)\cdot(\nabla\vec{v})^{T}\right)^{n+1,i}\\ &-\left(\frac{1}{3}\nabla S\left(\frac{1}{\rho_v}-\frac{1}{\rho_l}\right)-\frac{2}{3\mu}\nabla\mu S\left(\frac{1}{\rho_v}-\frac{1}{\rho_l}\right)\right)^{n+1,i}\\ &+\left(\frac{3\rho^{n+1,i}}{2\Delta t\mu^{n+1,i}}-\frac{3\rho_l}{2\Delta t\mu_l}\right)\vec{v}^{*,k}\end{aligned} \tag{4.69}$$

The use of constant basis for the linearised source term S^n has an important effect on this equation already. Although the source term used in it is updated during outer iterations, the pressure term is not. Instead, it equals the value from previous time level in all outer iterations. This is a consequence of using same basis S^n for linearised source term in Φ solution, which demands that Φ in this pressure incremental version of the algorithm equals complete pressure change during two time steps. Otherwise the source term cannot be completely updated to time level $n+1$ with its linearised form. And since projection equation, in this case equation (4.32), is obtained from the difference between momentum equations for $\vec{v}^*$ and original NS momentum equation, this demands that pressure p^n is used in equation for $\vec{v}^{*,k+1}$. For further illustration, it should be restated that Φ as the projection variable does not only have the connection with velocities $\vec{v}^*$ and $\vec{v}$, but also gives the equation to obtain pressure. The two connections are repeated below with equation (4.70), in which the role of p^n is also clearly shown.

$$-\nabla\Phi = \frac{3\rho}{2\Delta t}(\vec{v}^{n+1,i+1}-\vec{v}^{*,i+1}) = -\nabla p^{n+1,i+1}+\nabla p^{n} = -\nabla dp \tag{4.70}$$

With the converged solution for $\vec{v}^{*,k+1}$ the algorithm proceeds to the solution for Φ. The boundary conditions for this variable are the only ones which differ among the developed versions of the algorithm. Mixed boundary conditions are used, where only the Dirichlet conditions on the outlet are adjusted to the version of the algorithm. The von Neumann conditions on other boundaries are otherwise always equal to zero. In the case of this version, the pressure difference on the outlet has to be precised and must remain unchanged during outer iterations. A simple value of 0 Pa can be used, which also agrees with a constant value for pressure otherwise usually

imposed on the outlet. As it will be seen, this was not so in the performed verification tests, where pressure difference had to be specified for each time step. The solution for Φ is then obtained with the use of presented Helmholtz equation for Φ, equation (4.56). This is here rewritten with the suitable notation in equation (4.71).

$$\begin{aligned}(\Delta + \sigma_p)\,\Phi^{k+1} = &-\frac{3}{2\Delta t}(\vec{v}^{n+1,k} - \vec{v}^{*,i+1}) \cdot \nabla \rho^{n+1,i} \\ &- \frac{3\rho^{n+1,i}}{2\Delta t}\left(S^{n+1,k}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right) - \nabla \cdot \vec{v}^{*,i+1}\right) + \sigma_p \Phi^k\end{aligned} \tag{4.71}$$

After each solution of this equation, velocity $\vec{v}^{n+1,k}$ is updated with equation (4.60) written as (4.72). Source term update is done with equation (4.51) which does not need to be adjusted for use in this version of the algorithm, it is for simplicity just repeated in equation (4.73). The reason is that terms $\frac{\partial S^n}{\partial p}$ and $\frac{\partial S^n}{\partial \alpha}$ (if applied) remain unchanged during outer iterations, hence also the retained superscript n. Only dp and $d\alpha$ terms change, where pressure change is given directly as $dp = \Phi^{k+1}$, while $d\alpha$ follows from (4.53) in which only the α variable in K factor can be now better precised, resulting in (4.74). It should be stressed that the source term is not completely explicit in the first inner iteration for Φ^{k+1}. Instead, its value is predicted with the use of equation (4.73). Value for dp is represented by explicit Φ^k. Such approach was retained from important observation in section 2.2.3.2, which states that stability improved if a valuable prediction for source term was done.

$$\vec{v}^{n+1,k+1} = \vec{v}^{*,i+1} - \frac{2\Delta t}{3\rho^{n+1,i}} \nabla \Phi^{k+1} \tag{4.72}$$

$$S^{n+1,k+1} = S^n + \frac{\partial S^n}{\partial p} dp + \frac{\partial S^n}{\partial \alpha} d\alpha \tag{4.73}$$

$$d\alpha = \frac{S^n + \frac{\partial S^n}{\partial p} dp}{\frac{1}{K} - \frac{\partial S^n}{\partial \alpha}} \quad ; \quad K = \Delta t \left(\frac{1}{\rho_v} - \alpha^{n+1,i}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right)\right) \tag{4.74}$$

After convergence of Φ^{k+1} is reached, the final $S^{n+1,i+1}$ and $\vec{v}^{n+1,i+1}$ also follow and pressure $p^{n+1,i+1}$ is defined simply with addition of final Φ^{k+1} to p^n. The only solution left is solution of α^{n+1}, which is done with the given iterative procedure in section 2.3. The equation for α solution is repeated here in equation (4.75), where only superscripts for $\vec{v}$ need to be adjusted slightly. Source term is also updated with each α solution, but as there is no need for linearisation as in Φ solution, the update is done directly with application of $\alpha^{n+1,k+1}$ to $S^{n+1,k+1}$.

$$\alpha^{n+1,k+1} = \frac{\frac{S^{n+1,k}}{\rho_v} + \frac{4\alpha^n - \alpha^{n-1}}{2\Delta t} - (\vec{v}^{n+1,i+1} \cdot \nabla)\alpha^{n+1,k}}{\frac{3}{2\Delta t} + S^{n+1,k}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right)} \tag{4.75}$$

With the converged solution for $\alpha^{n+1,k+1}$ the solving steps in the algorithm are concluded. The final $S^{n+1,i+1}$ is also obtained, and α related variables ρ and μ are updated with the use of equation (4.4). Then, continuity check is performed and if convergence is satisfied, the simulation continues to the new time step. Otherwise, the obtained solutions are used to prescribe explicitly treated variables and whole procedure is repeated. This version of the algorithm is graphically presented in figure 4.4.

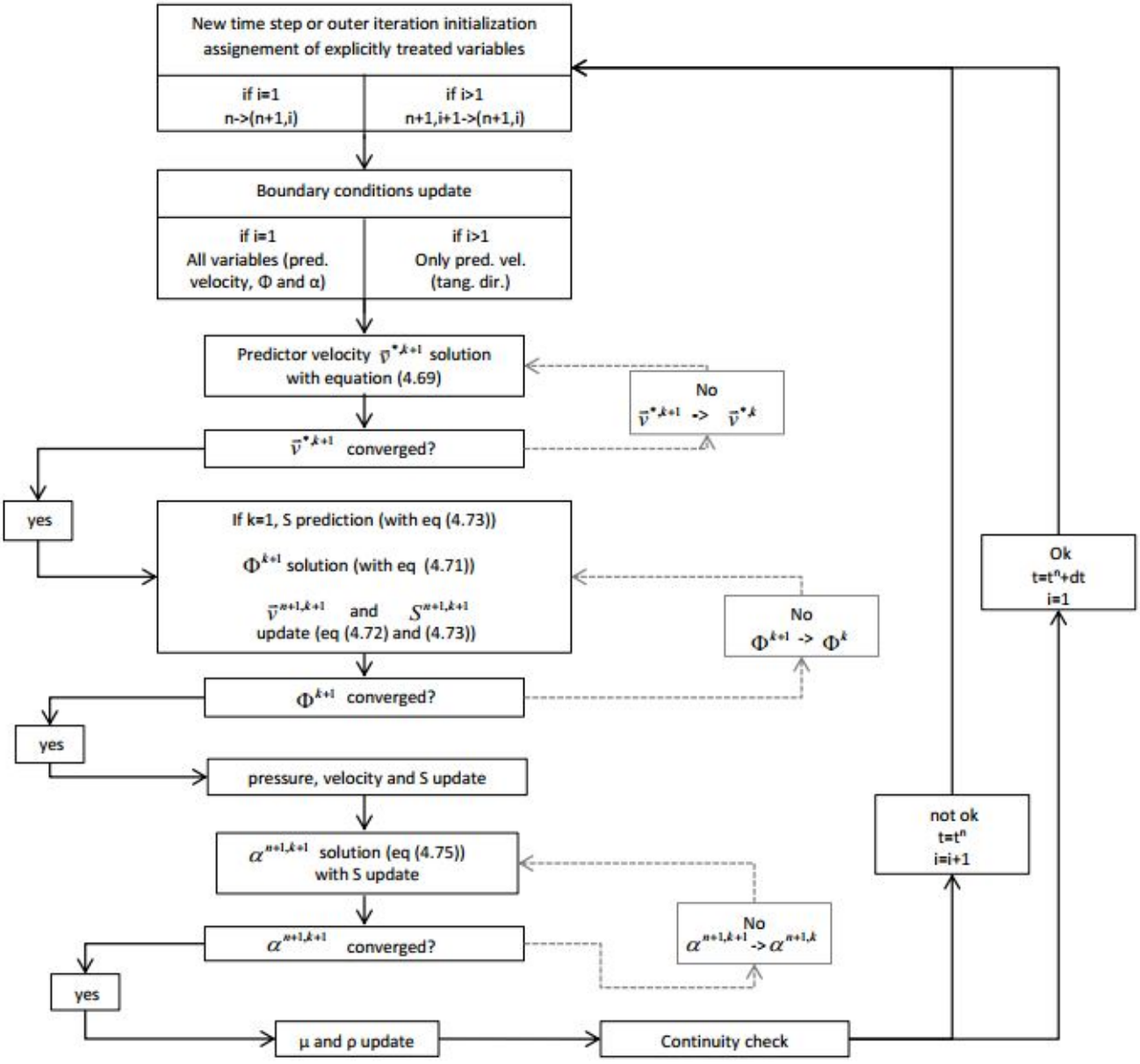


Figure 4.4: Graphical presentation of pressure incremental version of the algorithm with constant source term basis.

2.4.2.2 Updated source term basis

This second pressure incremental version of the algorithm has been developed as a possibly improved version of previously presented algorithm. The reason is in the similarities that this version has with the SIMPLE algorithm adapted to and widely used for cavitating flow simulations and presented in [8, 4]. These similarities will be pointed out here as well.

The first step, the solution for $\vec{v}^{*,k+1}$, is almost equal to the one in the previously described algorithm version. Equation for it is given in (4.76). Since the basis of linearised source term in Φ solution is in this version updated during outer iterations, the pressure in this equation is also updated. This is an important change, and follows from the same reasons as given before for keeping this same pressure term unchanged. In this case, as updated S basis is used, Φ does not present a complete pressure change between two time levels, except in the first outer iteration, where explicit variables are taken from previous time level. Otherwise Φ gives a correction for the obtained pressure, which is the main reason why this version resembles the mentioned SIMPLE algorithm. And also the reason why updated pressure has to be used in solution for $\vec{v}^{*,i+1}$.

$$\begin{aligned}\left(\Delta - \frac{3\rho_l}{2\Delta t \mu_l}\right)\vec{v}^{*,k+1} =& \frac{1}{4\Delta t \mu^{n+1,i}}\left(-4\vec{v}^{n}\left(\rho^{n+1,i}+\rho^{n}\right)+\vec{v}^{n-1}\left(\rho^{n+1,i}+\rho^{n-1}\right)\right)+\\ &\frac{1}{2\mu^{n+1,i}}\left(\nabla\cdot\left(\rho\vec{v}\vec{v}\right)+\rho\vec{v}\cdot\nabla\left(\vec{v}\right)\right)^{n+1,i}+\frac{\nabla p^{n+1,i}}{\mu^{n+1,i}}\\ &-\left(\frac{1}{\mu}(\nabla\mu)\cdot(\nabla\vec{v})+\frac{1}{\mu}(\nabla\mu)\cdot(\nabla\vec{v})^{T}\right)^{n+1,i}\\ &-\left(\frac{1}{3}\nabla S\left(\frac{1}{\rho_v}-\frac{1}{\rho_l}\right)-\frac{2}{3\mu}\nabla\mu S\left(\frac{1}{\rho_v}-\frac{1}{\rho_l}\right)\right)^{n+1,i}\\ &+\left(\frac{3\rho^{n+1,i}}{2\Delta t\mu^{n+1,i}}-\frac{3\rho_l}{2\Delta t\mu_l}\right)\vec{v}^{*,k}\end{aligned} \tag{4.76}$$

Since Φ has in this case a different role, its connection with pressure and velocity update also slightly changes and is defined with equation (4.77). In it, the dp^{i+1} depicts the pressure correction between outer iterations and not the pressure change between two time levels.

$$-\nabla\Phi = \frac{3\rho}{2\Delta t}(\vec{v}^{n+1,i+1}-\vec{v}^{*,i+1}) = -\nabla p^{n+1,i+1}+\nabla p^{n+1,i} = \nabla dp^{i+1} \tag{4.77}$$

After $\vec{v}^{*,i+1}$ solution the algorithm proceeds to the solution of this changed Φ variable. Attention should be given to the boundary conditions for it. In them, the logic that the first outer iteration gives a Φ^{k+1} which resembles the pressure change in time, while following outer iterations give only corrections for obtained pressure, should be respected. This results in the Dirichlet conditions being defined as pressure change in time for the first outer iteration and as zero for all other iterations. Of course, if a constant pressure in time is imposed on the outlet, a simple zero value can be always used. The solution for Φ^{k+1} is then obtained with equation (4.78), which is as such the same as equation (4.71) for previous version. The main difference between the two is therefore in the effect of Φ^{k+1} on velocity and source term update. Both can be now seen as correction of values obtained in previous iterations and not as complete update in time, which is another equality with aforementioned SIMPLE algorithm. While velocity update or projection is done with same equation as before, the source term update uses equation (4.79), where the use of updated basis is depicted. This updated basis also importantly means that the linearisation

terms have to use source term result from previous iteration, which is also shown in the equation. While the dp term has a different physical meaning it is still equal to Φ, as shown with equation (4.77). $d\alpha$ term is also changed and is given with equation (4.80).

$$\begin{aligned}(\Delta + \sigma_p)\,\Phi^{k+1} = &-\frac{3}{2\Delta t}(\vec{v}^{\,n+1,k} - \vec{v}^{\,*,i+1}) \cdot \nabla\rho^{n+1,i} \\ &-\frac{3\rho^{n+1,i}}{2\Delta t}\left(S^{n+1,k}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right) - \nabla \cdot \vec{v}^{\,*,i+1}\right) + \sigma_p \Phi^k\end{aligned} \tag{4.78}$$

$$S^{n+1,k+1} = S^{n+1,i} + \frac{\partial S^{n+1,i}}{\partial p} dp^{i+1} + \frac{\partial S^{n+1,i}}{\partial \alpha} d\alpha \tag{4.79}$$

$$d\alpha = \frac{S^{n+1,i} + \frac{\partial S^{n+1,i}}{\partial p} dp^{i+1}}{\frac{1}{K} - \frac{\partial S^{n+1,i}}{\partial \alpha}} \quad ; \quad K = \Delta t \left(\frac{1}{\rho_v} - \alpha^{n+1,i}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right)\right) \tag{4.80}$$

There is another, important difference regarding the source term update from the version presented before. Since an updated source term is proposed in first inner iteration for Φ^{k+1} already, attention should be given to mentioned new role of Φ. This variable is in first outer iteration equal to pressure change in time. Therefore a prediction for the source term with equation (4.79) in the first inner iteration here has to be defined with $dp = p^n - p^{n-1}$, since the final Φ^{k+1} from previous time step gives only a correction for pressure. And because Φ^{k+1} in second and following outer iterations has exactly the role of corrector, it is better that no prediction is used for the source term in the first inner iterations in these cases. Instead, the source term should be equal to $S^{n+1,i}$. The reason is best shown on the case of second outer iteration. There, prediction based on last Φ^{k+1}, which equals pressure change in time, would cause possibly too big change in source term as updated basis $S^{n+1,i}$ is already present in equation (4.79). Hence worse conditions for following inner iterations would be set and could lead to unstable simulations. Furthermore, it was observed that Φ^{k+1} magnitude drops during outer iterations monotonically, which confirms that predicting source term for $k = 1$ is suitable only in first outer iteration. Finally, the equation for Φ^{k+1} solution in first inner iteration when $i \neq 1$ is also affected by these issues. Since the constant σ_p is multiplied with explicit Φ^k coming from previous outer iterations and which thus has a higher magnitude than now expected Φ^{k+1}, it is more appropriate to use equation (4.81) than (4.78). The presence of σ_p on the LHS of it is not problematic because of the mentioned drop in Φ^{k+1} magnitude.

$$\begin{aligned}(\Delta + \sigma_p)\,\Phi^{k+1} = &-\frac{3}{2\Delta t}(\vec{v}^{\,n+1,k} - \vec{v}^{\,*,i+1}) \cdot \nabla\rho^{n+1,i} \\ &-\frac{3\rho^{n+1,i}}{2\Delta t}\left(S^{n+1,i}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right) - \nabla \cdot \vec{v}^{\,*,i+1}\right)\end{aligned} \tag{4.81}$$

Additionally to these differences to previously shown pressure incremental algorithm there is also the need to actually update pressure in each outer iteration. This is in former algorithm necessary only after the last outer iteration, but is changed here since updated pressure is used in $\vec{v}^{\,*,i+1}$ solution.

The last solution step is the same as in the previous version. Same are also the updates of source term and α related variables with new $\alpha^{n+1,k+1}$, as well as convergence check. Therefore this steps are here omitted. The graphical presentation of this algorithm version is given on figure 4.5.

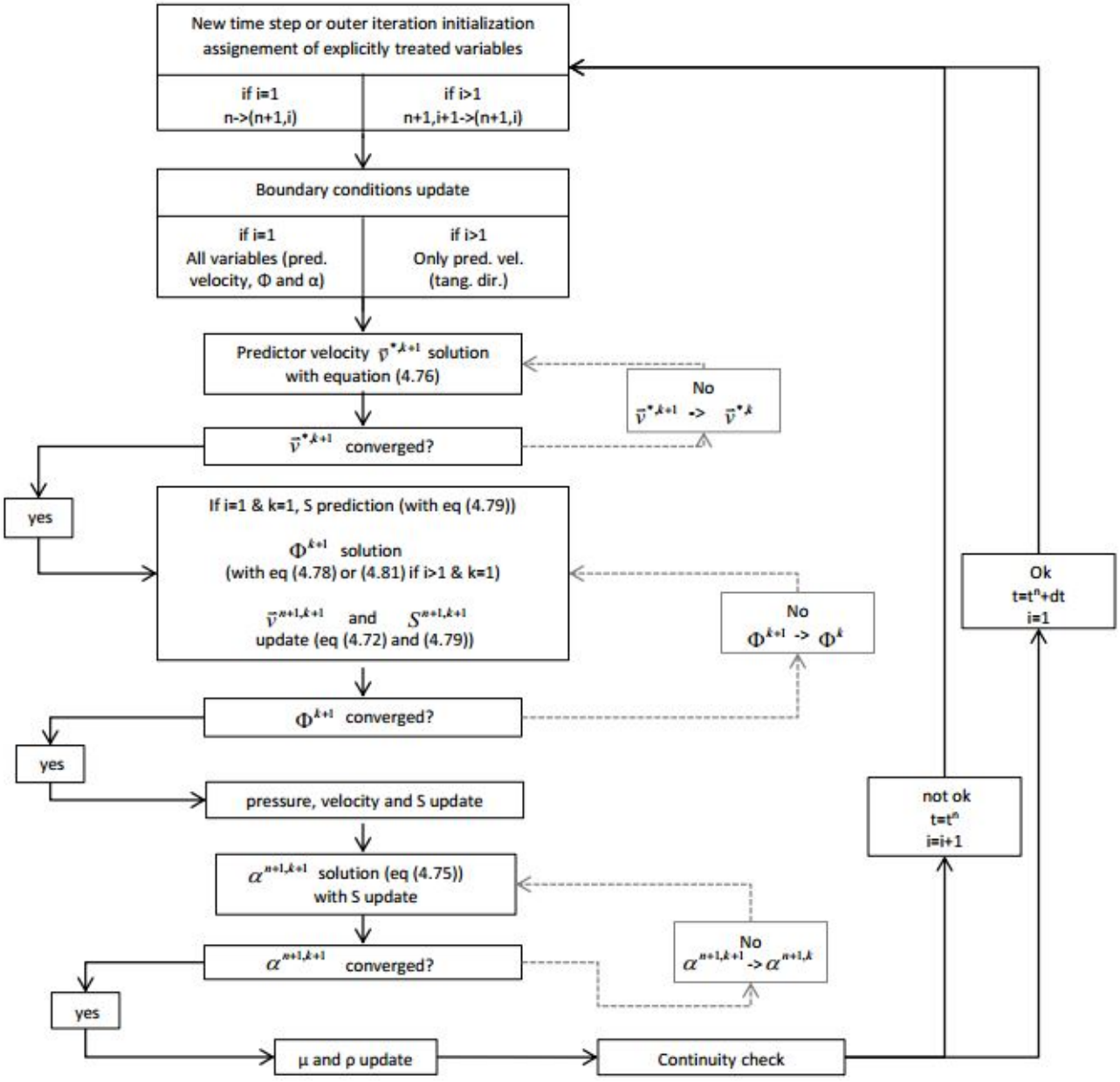


Figure 4.5: Graphical presentation of pressure incremental version of the algorithm with updated source term basis.

To conclude with the presentation of this version and with it also pressure incremental versions of the algorithm, another view on the difference between the two developed versions can be given. This is that the former version solves the governing equations until a converged pressure change in time is obtained, while the version given here searches for a Φ^{k+1} which only corrects the obtained variables from previous outer iteration. Therefore the magnitude of variable changes between outer iteration becomes smaller (tends to zero) in the case of this latter version, while in the previous it remains the same.

2.4.3 Pressure non incremental version of the algorithm

2.4.3.1 Constant source term basis

This version of the algorithm was initially thought of as the actual and only replacement of the algorithm for incompressible flow simulations. The equation which is used to obtain the solution for velocity $\vec{v}^{*,k+1}$ is given with (4.82) and is the same also for the latter presented version. The only difference from both previously shown equations for $\vec{v}^{*,k+1}$ is obviously the lack of pressure term.

$$\begin{aligned}\left(\Delta - \frac{3\rho_l}{2\Delta t \mu_l}\right)\vec{v}^{*,k+1} =& \frac{1}{4\Delta t \mu^{n+1,i}}\left(-4\vec{v}^{n}\left(\rho^{n+1,i}+\rho^{n}\right)+\vec{v}^{n-1}\left(\rho^{n+1,i}+\rho^{n-1}\right)\right)+\\ &\frac{1}{2\mu^{n+1,i}}\left(\nabla\cdot\left(\rho\vec{v}\vec{v}\right)+\rho\vec{v}\cdot\nabla\left(\vec{v}\right)\right)^{n+1,i}\\ &-\left(\frac{1}{\mu}(\nabla\mu)\cdot(\nabla\vec{v})+\frac{1}{\mu}(\nabla\mu)\cdot(\nabla\vec{v})^{T}\right)^{n+1,i}\\ &-\left(\frac{1}{3}\nabla S\left(\frac{1}{\rho_v}-\frac{1}{\rho_l}\right)-\frac{2}{3\mu}\nabla\mu S\left(\frac{1}{\rho_v}-\frac{1}{\rho_l}\right)\right)^{n+1,i}\\ &+\left(\frac{3\rho^{n+1,i}}{2\Delta t\mu^{n+1,i}}-\frac{3\rho_l}{2\Delta t\mu_l}\right)\vec{v}^{*,k}\end{aligned} \tag{4.82}$$

Φ in pressure non incremental versions of the algorithm does not represent only the pressure change. The absence of pressure term in $\vec{v}^{*,k+1}$ equation means that Φ-p connection, given here with equation (4.83), changes considerably. The mentioned pressure term is absent also from this connection while the previously missing viscous terms are now included as the difference between $\vec{v}^{n+1,i+1}$ and $\vec{v}^{*,k+1}$ is not negligible any more. This also raises question of imposing correct mixed boundary conditions for Φ on the basis that one can only provide pressure values on the outlet. It was found this is also sufficient and there is no need to include viscous terms, which can cause a lot of issues as they lead to inclusion of source term and $\nabla\cdot\vec{v}^{*,k+1}$ values on the outlet. The latter, though known in advance, poses issues because boundary values for $\vec{v}^{*,k+1}$ depend also on the use of explicit Φ. Nevertheless, although use of p values as outlet values for Φ is in such manner more appropriate, it can impact accuracy of the pressure solution at the outlet. This will be seen in later presented verification results in chapter 6.

$$-\nabla\Phi = \frac{3\rho}{2\Delta t}(\vec{v}^{n+1,i+1}-\vec{v}^{*,i+1}) = -\nabla p^{n+1,i+1}+\mu\Delta(\vec{v}^{n+1,i+1}-\vec{v}^{*,i+1}) \tag{4.83}$$

The solution for Φ uses same equation as previously shown versions of the algorithm. Equation is here repeated with (4.84). What is different about it is the more complex dp and therefore also

σ_p. Since the source term basis remains unchanged, equation (4.85) defines the linearised source term. The complex dp and (if used) $d\alpha$ applied in it are defined with equations (4.86) and (4.87), respectively.

$$\begin{aligned}(\Delta + \sigma_p)\,\Phi^{k+1} = &-\frac{3}{2\Delta t}(\vec{v}^{\,n+1,k} - \vec{v}^{\,*,i+1}) \cdot \nabla \rho^{n+1,i} \\ &-\frac{3\rho^{n+1,i}}{2\Delta t}\left(S^{n+1,k}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right) - \nabla \cdot \vec{v}^{\,*,i+1}\right) + \sigma_p \Phi^k\end{aligned} \tag{4.84}$$

$$S^{n+1,k+1} = S^n + \frac{\partial S^n}{\partial p} dp + \frac{\partial S^n}{\partial \alpha} d\alpha \tag{4.85}$$

$$dp = \frac{\Phi^k + \mu^{n+1,i}\left(\left(S^n + \frac{\partial S^n}{\partial \alpha}\frac{S^n + \frac{\partial S^n}{\partial p}dp}{\frac{1}{K} - \frac{\partial S^n}{\partial \alpha}}\right)\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right) - \nabla \cdot \vec{v}^{\,*,i+1}\right) - p^n}{1 - \mu^{n+1,i}\frac{\partial S^n}{\partial p}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right)} \tag{4.86}$$

$$d\alpha = \frac{S^n + \frac{\partial S^n}{\partial p}dp}{\frac{1}{K} - \frac{\partial S^n}{\partial \alpha}} \quad ; \quad K = \Delta t\left(\frac{1}{\rho_v} - \alpha^{n+1,i}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right)\right) \tag{4.87}$$

The use of CG method and the subsequent request to respect the correspondence between σ_p constant and dp term which defines it result here in a more cumbersome Φ solution procedure than in the case of pressure incremental versions. Especially if complete source term linearisation is applied. In this case, dp expression hides in itself an explicitly treated dp term in order to provide a suitable constant σ_p for CG method, as it was shown in section 2.2.3.2. This importantly affects how dp is set before and after Φ^{k+1} solution in case of complete source term linearisation. It was proposed that for $i = 1$ and $k = 1$, where the source term prediction and with it dp are required, the explicit value for dp is first set as $dp = p^n - p^{n-1}$. Then, expression (4.86) is used to obtain the actual dp used in following prediction of source term. With this, the correspondence between dp and σ_p is ensured. After the first Φ^{k+1} solution is found, the dp is calculated again with (4.86) where new Φ^{k+1} is used. In this case, the dp from previous definition with (4.86) is used as explicit dp. The source term can be then updated with equation (4.85) while pressure $p^{n+1,k+1}$ is obtained with equation (4.88). For following inner iterations, the explicit dp used in (4.86) after each Φ^{k+1} solution is defined by dp from previous inner iteration. Additionally, the first inner iterations in cases where $i \neq 1$ use the source term prediction which continues to use this previous dp. Solution procedure is simpler in cases where $\frac{\partial S}{\partial \alpha}$ term is not used. Equation (4.86) is directly applied to get first dp and predict the source term when $k = 1$ and $i = 1$. Procedure then runs in same manner as described for the completely linearised source term.

The update of velocity $\vec{v}^{\,n+1,k+1}$ which also follows each Φ^{k+1} solution is done with same equation as shown before for pressure incremental versions of the algorithm. After converged solution for Φ^{k+1} is found, final values for $\vec{v}^{\,n+1,i+1}$ and $S^{n+1,i+1}$ are defined, as well as $p^{n+1,i+1}$.

$$p^{n+1,k+1} = p^n + dp \tag{4.88}$$

The final step, α^{n+1} solution, and all following steps including continuity check, are the same as in previously presented two versions of the algorithm and therefore omitted here. This version of the algorithm is graphically presented on figure 4.6.

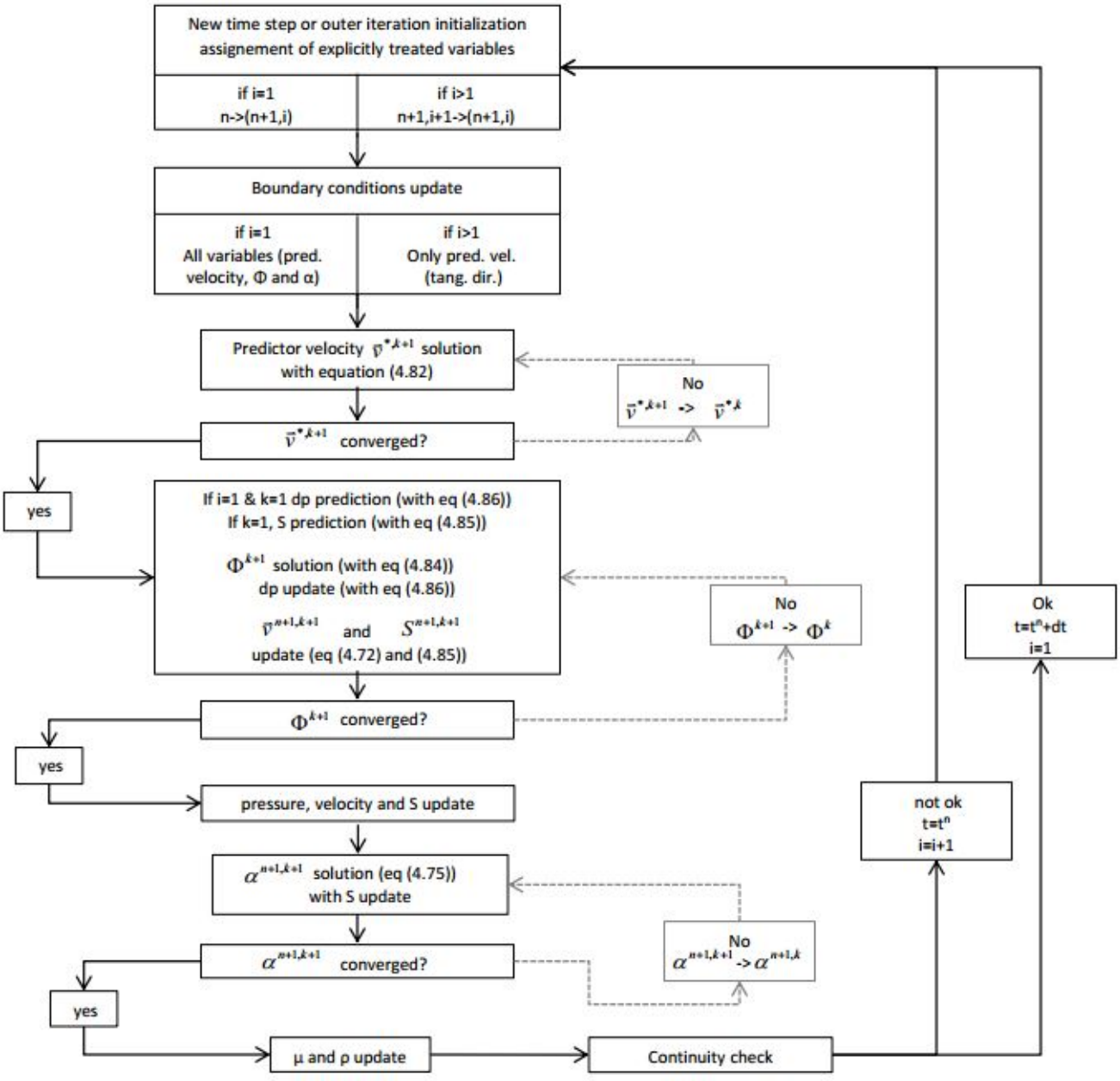


Figure 4.6: Graphical presentation of pressure non incremental version of the algorithm with constant source term basis.

2.4.3.2 Updated source term basis

This, last presented version of the algorithm is a mix of pressure non incremental version with constant source term basis and incremental version with updated source term basis. Since pressure term is excluded from $\vec{v}^*$ definition, it uses same equation for $\vec{v}^{*,k+1}$ solution as previously presented non incremental version, equation (4.82). The Φ^{k+1} solution also proceeds in the same manner as shown before, including the complicated way of updating dp. But the updated source term basis causes same effect as in the case of pressure incremental algorithm. First, the role of pressure change between iterations becomes same as in that version, meaning that equation (4.89) is used for source term update. As dp^{i+1} in it presents difference between two outer iterations and not time levels (except for the first outer iteration), the dp definition changes. Previous one, done with equation (4.86), has p^n term used in the numerator. The term is a consequence of adapting $\Phi - p$ connection to express dp, shown in equation (4.54). But since one now looks for pressure correction, p^n has to be replaced with $p^{n+1,i}$. As linearised source term is also changed, terms which represent it are adapted and expression (4.90) is finally obtained as the expression used for dp^{i+1}.

$$S^{n+1,k+1} = S^{n+1,i} + \frac{\partial S^{n+1,i}}{\partial p} dp^{i+1} + \frac{\partial S^{n+1,i}}{\partial \alpha} d\alpha \tag{4.89}$$

$$dp^{i+1} = \frac{\Phi^k + \mu^{n+1,i}\left(\left(S^{n+1,i} + \frac{\partial S^{n+1,i}}{\partial \alpha}\frac{S^{n+1,i} + \frac{\partial S^{n+1,i}}{\partial p} dp^{i+1}}{\frac{1}{K} - \frac{\partial S^{n+1,i}}{\partial \alpha}}\right)\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right) - \nabla \cdot \vec{v}^{*,i+1}\right) - p^{n+1,i}}{1 - \mu^{n+1,i}\frac{\partial S^{n+1,i}}{\partial p}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right)} \tag{4.90}$$

The updated source term basis importantly also demands that the prediction for the source term in the first inner iteration is done only for first outer iteration. Like in pressure incremental version, the pressure change dp^{i+1} magnitude decreases monotonically between outer iterations, therefore prediction for the source term in second and later outer iterations could lead to increased errors. However, the equation for Φ^{k+1} solution does not change as in pressure incremental version, where σ_p was left out for first inner iterations in such cases. The reason is in the magnitude of Φ, which importantly stays the same between outer iterations. On the other hand, the pressure change dp^{i+1} which follows the Φ^{k+1} solution, is for these cases done in a much simpler manner, with equation (4.91). The reason is in the use of non predicted source term in equation (4.84), which means all the linearisation terms equal zero and the complex update equation should not be used.

$$dp^{i+1} = \Phi^{k+1} + \mu^{n+1,i}\left(S^{n+1,i}\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right) - \nabla \cdot \vec{v}^{*,i+1}\right) - p^{n+1,i} \tag{4.91}$$

Apart from these changes for $k = 1$ and $i \neq 1$, the solution for Φ^{k+1} proceeds in the same manner as shown for previous pressure non incremental version. The equation used is also the same, and is therefore here not shown. The only difference left is in the final pressure $p^{n+1,k+1}$ definition, which has to respect the new role of dp^{i+1}, leading to the use of equation (4.92). All other following steps in the algorithm are same as before and are here omitted. The algorithm is graphically presented on figure 4.7.

$$p^{n+1,k+1} = p^{n+1,i} + dp^{i+1} \tag{4.92}$$

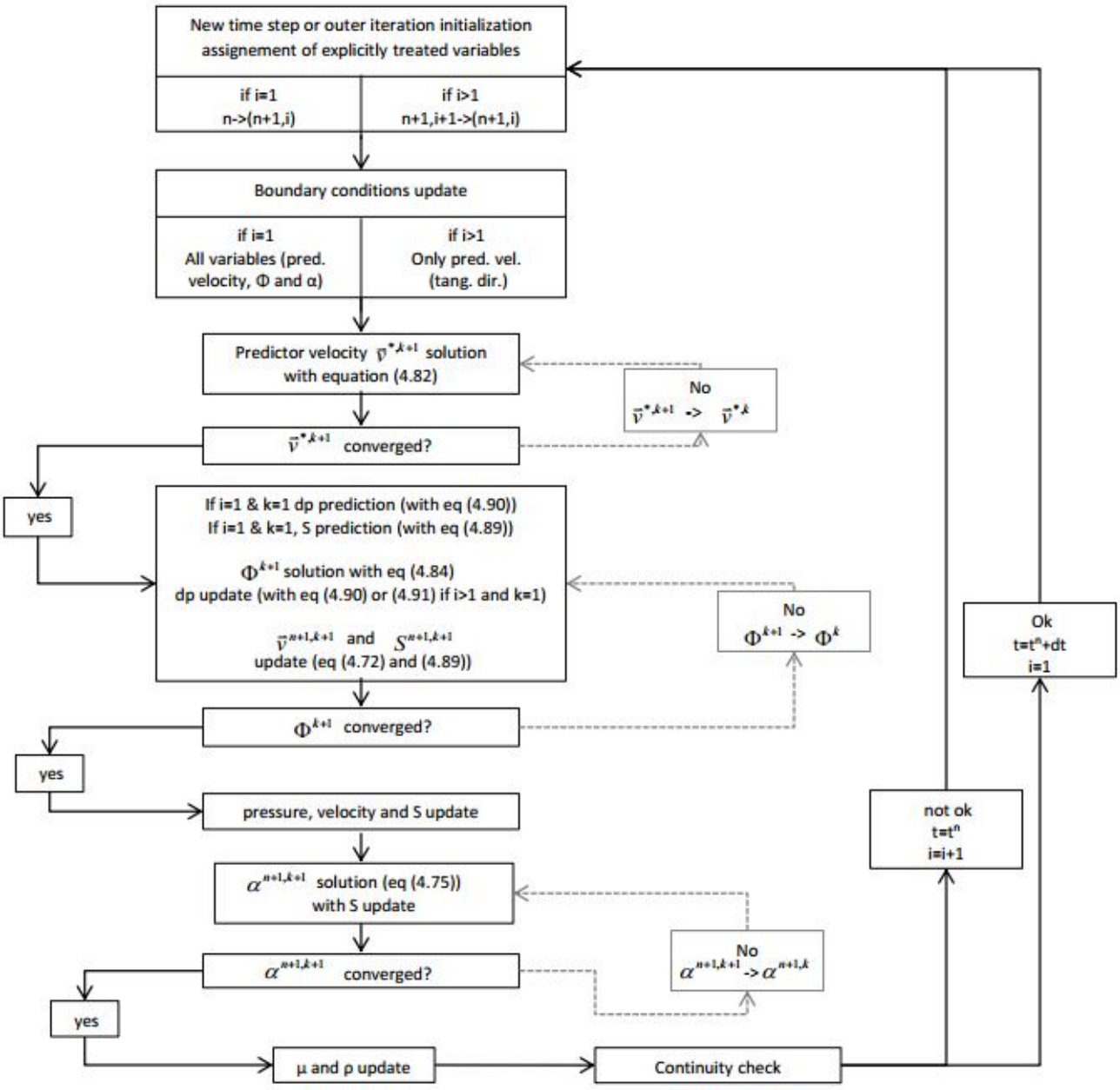


Figure 4.7: Graphical presentation of pressure non incremental version of the algorithm with updated source term basis.

With this version, the presentation of the developed versions of the algorithm are concluded. It is obvious that the pressure non incremental versions are more complex for use, on behalf of the dp update. However, the incremental and non incremental versions share same equalities when it comes to the meaning of dp. In cases where the source term basis is updated, this represents a correction for pressure, while with constant source term basis, the pressure change always depicts the obtained difference between two time levels. For the sake of simplicity and shortness, the four versions will be in the following chapters referred to as PICS (Pressure Incremental Constant Source term basis), PIUS (Pressure Incremental Updated Source term basis), PNCS (Pressure Non incremental Constant Source term basis) and PNUS (Pressure Non incremental Updated Source term basis).

3 Conclusions

This chapter presented development of the new algorithm for cavitating flow simulations with homogeneous mixture approach and single phase cavitation models. Algorithm is developed in a way to make it suitable for codes which enable fast DNS simulations and offer more flexibility regarding domain geometry. More precisely said, it enables the use of codes which apply either fast and direct Helmholtz solvers on the level of mono domains or influence matrix technique as multi domain method. Or both together, as is the case of MFLOPS-3D, where these solvers together with compact finite differences offer more liberty in choosing the geometry. This code with the new algorithm will be from here on often referred to as new code and the starting one as original MFLOPS-3D code.

Development of this new algorithm was quite demanding, because of the constraints imposed by original MFLOPS-3D code. Whole development and main characteristics of the new algorithm can be summarized in following points:

- The main key to the new algorithm is the use of CG method. This can compete well with fastest iterative solvers available for solving Helmholtz equations with non-constant coefficients and also enables direct application of influence matrix technique as multi domain method. It is therefore the needed tool for desired modification of MFLOPS-3D code and used for both $\vec{v}^*$ and Φ variables.

- Use of CG method is more demanding in case of Φ solution. An algorithm without using it for Φ solution was at first developed. For such case, completely linearised source term provides a stable solution. This algorithm is not suitable as it demands a lot of inner iterations to solve for Φ, causing high performance decrease.

- The observations from this algorithm were used in implementation of CG method for Φ solution. Although not being crucial, here given explanation includes completely linearised source term because of its advantages when no CG method was used.

- The constant introduced by CG method demands some attention in case of Φ solution. It is preferable it bases on a bounded variable. The term from linearisation of S with pressure was chosen as it can satisfy this.

- Four algorithm versions were developed since the use of CG method enables various approaches to solving governing equations and an algorithm as developed here was not used before. Versions differ on behalf of using the pressure gradient in $\vec{v}^*$ solution (pressure

incremental or non incremental versions) and updated or constant basis of linearised S. It was however not defined which version performs the best. This will be the topic of a later chapter, considering verification and performance tests.

- Algorithm presentation also includes a more detailed explanation of compatibility condition constraint, although focused only on continuous level (boundary conditions). This was done on one hand to better present the limitation which had to be relaxed in order to enable cavitating flow simulations. On the other hand, this also gives a basis for later given presentation of both new and original MFLOPS-3D code performance, where compatibility condition is believed to be the cause of some issues.

Connected with the CG method constant for Φ solution, there is one more important point about the algorithm which has to be addressed. Namely, the explanation of this constant derivation does not give an actual example of its application for a certain cavitation model. A universal example, based on the general form of source term, is rather given. For a particular constant derivation, models based on bubble dynamics or empirical relations, presented in section 2.2.2 of chapter 1, should be observed. It follows that forming the constant is easier for the empirical than bubble dynamics models. There, pressure remains present in the used linearised term as well, but to our help only in the denominator. This has an inverse effect as the observed source term change becomes bounded by highest and zero value. Which again enables us to find a good constant for CG method. However, additional care should be given to forming CG constant for both kinds of models if the source term is defined considerably different for vapour creation and destruction, leading to larger difference in its magnitude (at same $\mid p - p_v \mid$ values) for the two states. Observing results and CG constants in [121], a good strategy in this case would be to use value which is lower than here applied average value of $\frac{\partial S}{\partial p}$ in either vapour destruction or creation case.

Although the shown algorithm can be applied directly to the cases using α transport equation with a source term governing phase change, it or the logic in which it was created can be applied also to the cases where this transport equation is not used. For instance to barotropic models. Such work was not done here, but the statement is based on successful use of projection methods also for cavitating flow simulations where no α transport equation was used. This of course imposes a drawback as the heavily relied on connection between source term and velocity divergence is not given. Instead of it, another connection between velocity divergence and pressure is applied through the use of given phase change description. This can be used in order to raise a similar procedure to solve for Φ. Therefore it follows that the shown logic of the presented algorithm can be applicable to all single phase cavitation models.

Finally, a table to summarize all the changes between original and new MFLOPS-3D code, as mentioned in introduction to this chapter, is given with table 4.1. The methods enabling the new algorithm can be applied directly in other codes as well.

Table 4.1: Differences between original and new MFLOPS-3D code.

characteristic	original MFLOPS-3D code	new MFLOPS-3D code
temporal discretization	2^{nd} order backward difference scheme	2^{nd} order backward difference scheme
spatial discretization	compact finite differences	compact finite differences, 2^{nd} order upwind for α and related variables
mono domain solver	direct Helmholtz solver, based on eigen decomposition in 3D	direct Helmholtz solver, based on eigen decomposition in 3D
multi domain solver	iterative, Krylov solver	iterative, Krylov solver
projection method version	Kim and Moin, non incremental pressure-correction scheme	derivatives of Kim and Moin, non incremental pressure-correction scheme: a.) two non incremental schemes, directly derived from KM scheme b.) two incremental schemes, viscous terms in $\Phi - p$ connection skipped difference between two versions of each scheme:source term treatment
$\vec{v}^*$ boundary conditions	set by Kim and Moin scheme	set by Kim and Moin scheme
$\vec{v}^*$ solution	obtained in one step	obtained over iterative steps, demanded by CG method. CG method constant set by liquid properties and time derivative term
Φ boundary conditions	homogeneous von Neumann	mixed, with Dirichlet conditions on outlet set as pressure values
Φ solution	obtained in one step	multiple iterative steps, demanded by CG method. CG constant set by linearised source term
$\vec{v}$ solution	direct, obtained after Φ solution	direct, obtained after each Φ iteration
α solution	non existent	solution obtained in each point separately, in iterative steps
outer iterations after last projection method step	no	yes, at most 3

Chapter 5

Used code and algorithm verification approach

Previous chapter has a substantial part devoted to explanation of the issues, encountered when the new algorithm was being developed, and their solutions. It also mentions that observations lading to solutions followed from certain verification tests performed. These tests are based on the use of Method of Manufactured Solutions (MMS in the following text), which is a method widely used for code verification purposes [126, 127, 128, 129, 130]. This method was extensively used in the development of the algorithm and was a key to finding the solutions proposed in previous chapter. The reasons are in the low computational costs regarding the CPU power and also time which the method demands in order to verify that a solution procedure performs in the desired manner. Moreover, it also enables one to easily and precisely identify the source of issues and thus propose solutions faster [127, 130]. Computations performed with this method were also used to define which algorithm version has best performance regarding accuracy and stability. For the moment, these computations actually give the majority of results and proof about performance of the algorithm. Furthermore, it seems to us there is no example where this method was used for verification of proposed algorithms for cavitating flow simulations, despite the advantages it offers. Therefore the method and the tests have to be presented in more detail, which is also the role of this chapter. Firstly, the method is presented in order to give the principle which it utilises and its benefits. Then, the case developed for incompressible or original code tests is presented. This is followed with the case developed for cavitating flow governing equations and used in building and testing of the new algorithm. As mentioned, this seems to be the first case of such kind.

1 Method of Manufactured Solutions theory

A fundamental task which has to be done before the code can be generally used for flow simulations is to perform Verification and Validation tests. The difference between the two is that Verification is concerned with determination if equations are correctly solved while Validation refers to proving that correct system of equations is being solved for a certain problem [126, 127]. The Verification is therefore more a mathematical procedure, where the physics of the flow is not so much the main concern point, while the opposite holds for Validation. Verification of a code can also give the proof for consistency, since performing code verification shows if the error of the obtained solutions becomes smaller as more refined mesh and time step are used [127]. Which is exactly the demand of consistency [81]. However, two different kinds of Verification exist and one should be careful to

distinguish between them. The two are Code Verification and Verification of Calculations. The first is concerned with correct solving of the equations that the code uses to model the flow. The second is concerned with magnitude of error of a certain calculation, for which the exact solution is generally not known [126, 127, 130]. Code Verification is the type of verification which has to be done at first, since it generally shows if a code can solve chosen system of equations correctly. On the other hand, Verification of Calculations is verification procedure which should be used for every set of simulations in order to prove their consistency and estimate their error [126, 127]. The task of Code Verification has been performed in the scope of this work, since it is first needed to show that the developed algorithm is capable to correctly solve the chosen system of governing equations and to define overall order of accuracy. Moreover, the new algorithm and code had to be compared with the original one. The Method of Manufactured Solutions has been used for all these tasks. The reason is in the flexibility the method offers to check if the governing equations are solved correctly, possibilities one has with it to locate the issues with solution methods, its low demands for CPU power and time to perform verification and, importantly, its ability to perform Verification of order of accuracy. With this last ability, one does not only verify that code correctly solves the system of equations and is consistent, but also determines the order of accuracy of the code. This can then be compared to the theoretical order of accuracy, giving an additional tool to define possible errors in the code. Accoring to [127], Verification of order of accuracy done with MMS is the most rigorous and comprehensive code verification method.

The basis on which MMS stands is proposing or constructing some analytical flow expressions and with them flow cases which are then used for verification of a certain code. These analytical flow expressions do not need to satisfy the governing equations [127, 130], although they can satisfy flow constraints, such as divergence free condition in incompressible flows [128, 126]. The lack of satisfying the governing equations is the main difference between this method and Method of Exact Solutions (MES), which is also widely used for Code Verification. In MES, one first defines exact solutions for a certain set of governing equations. These solutions can be found in the literature or devised by developer and are then used in verification tests. An example of MES use would be 1D steady convection/diffusion equation, given in chapter 3.11 in [81] or a heat conduction problem mentioned in Appendix A in [127]. However, the main issue with MES is precisely the need to satisfy governing equations, which makes it difficult to have such exact solutions, especially those which can at once or as a set include all effects in the governing equations. This drawback can lead to inclusion of mistakes in the code even if verification has been performed [127]. MMS on the other hand gives one much more freedom in division of the test cases and hence also more complete tool for verification, as already mentioned. General principle of both MMS and MES can be also shown with a case involving equation (5.1), which is taken from [127]. In it, G is a differential operator, p represents variable in question while s is the source term.

$$Gp = s \tag{5.1}$$

In MES, one needs to first have source s known, for which a solution p has to be then found using analytical approach. In MMS, the variable p has to be at first precised. p is then submitted to operator G and the source s is defined. If equation (5.1) were governing equation of a certain case, this would mean that it is satisfied for both MMS and MES. However, the source term as used in MES is a part of the governing equation. Where in MMS, it usually stands for the imbalance which results from introduction of proposed p to the governing equation. Therefore it is a term which is in essence introduced to the governing equation, in order to make it work with the proposed p [126, 130]. Put in another way, one could imagine this term to be the forcing

field or body force, which forces the flow to follow defined analytical expression. Since definition and inclusion of these forcing or source terms into the governing equations is very easy for some proposed analytical flow expressions, it is obvious why MMS offers more versatile verification tool than MES.

However, even if obtaining source terms is easy, a lot of care should be given to proposing analytical flow expressions or solutions. Useful points for creation of analytical flow expressions and following source terms are given in [127]. Here, they are quickly stated, but interested reader is encouraged to look into this reference. First, the solution should be general enough, meaning that it enables all the terms in the governing equations have some notable value. In this demand, two further demands are hidden. One is the need for solution to have sufficient amount of non trivial derivatives, which makes it possible to include effect of higher order derivatives present in the governing equations. The other one is that these derivatives should be bounded by some small constant, which makes sure there are no too high space or temporal changes or even singularities included. These would at least hinder the ability to define the order of accuracy of the code. A demand which is connected with this last one in regards to both boundedness and order of accuracy is a demand to have smooth solution in time and space. Then, the solution should be proposed in a way to ensure flexibility in choosing domain limits but should also respect the constraints imposed by the code. Finally, the constraints and demands imposed by the flow or flow model should also be respected. There is one more demand which should be mentioned as it plays a crucial role in the suitability of proposed solutions and their source terms. Contrary to previous demands it does not come from [127] but from [126]. It states that the source terms added to the governing equations should not be orders of magnitude larger than the terms present in those equations. If so, the obtained numerical solution is mainly driven by the added source terms and hence makes verification questionable.

All of the demands are respected in the proposed analytical solutions and their following forcing terms in the case of incompressible flow. For the case of cavitating flow, all of the demands are respected too, but conditionally. Namely, some violations can follow if care is not taken. This will be discussed later. For this discussion and also generally, one demand should be a bit better referred to, as it is actually a very important subject in regards to the work presented here. This is that the cases should respect constraints imposed by the flow or flow model. In [127] this is illustrated by the need to devise a solution where a certain flow effect or characteristic is included. But as this and all other sources also mention, there is no need to have physical realism included in the devised solution. Therefore the need to have a certain flux included and tested to be correctly resolved can be done by describing it for example with a simple trigonometry function. Or if a flow has a certain quality which affects modelling and resolution of equations, this quality should also be included in the proposed solution, even if in some non-physical manner. It is actually here where the literature gets a bit unclear. For instance, codes developed for incompressible flow simulations usually have their algorithms built around the divergence free constraint, the signature quality of such flow. An example of such are the algorithms based on projection methods, which are presented in this work. However, it seems that MMS allows for the solutions to be proposed with this condition being violated. As can be seen in chapter 6.1 in [127], the proposed solution for incompressible flow code verification clearly does not respect the mentioned condition. It is true that the non zero divergence of velocity can be added as a source term to the continuity equation, as done otherwise in MMS, but the matter is that the algorithm built around the $\nabla \cdot \vec{v} = 0$ condition has to be then adapted to such case. Which raises the question about the quality of algorithm's verification, as one of the key points used by the algorithm is adjusted. Although MMS seems to make such kinds of verification tests possible,

some authors, as in [126, 128], devised and used solutions for incompressible flow code verification which respect the divergence free condition of the flow. Such approach was used in the scope of this work too. It is true that proposing analytical flow solutions is therefore more complex, as velocity divergence condition has to be ensured in them, but on the other hand, the verification is done in expectedly better way. This decision proved to be especially important for the use of MMS in verification of the new algorithm for cavitating flow simulations, where the divergence of velocity has to be connected with the cavitation source term S. Ensuring this connection without addition of any other source term to the continuity and α transport equation makes derivation of analytical expressions difficult, but in return it can be argued that the constraints imposed by the flow and flow model are better respected and thus make for more valuable verification. As mentioned at the beginning of this paragraph, some possible violations of certain other demands were noted and result from ensuring this connection. Some of them will be mentioned later in this chapter, while the rest will be revealed in the following one, where verification results are presented.

2 Implementation of MMS

The implementation of MMS to a certain code should be done with care. Not all codes are well suited to implement MMS, where their flexibility plays important role. There are two main areas which need to be observed. One deals with the boundary conditions, while the other concerns the inclusion of forcing terms.

Regarding boundary conditions and geometry, use of MMS offers us a lot of simplifications compared to real flow simulations. For instance, since analytical velocity expressions are known everywhere, there is no need to determine an inflow and outflow. And there is also no need to use domain which has walls on boundaries. All one needs to do is to simply use analytical expressions directly in velocity boundary conditions definition. In such manner, the often encountered issues and errors with prescription of outlet velocity conditions are avoided and excluded. Although this all seems easy, it is imperative that a code is flexible enough to implement such boundary conditions. Since this is not always the case, or if one wishes to include some specific boundary conditions, such as no slip, inflow/outflow etc, the solutions can be manufactured in a manner to satisfy this too [127, 130]. Examples of no slip boundary condition at the bottom of the domain can be found in [126].

The inclusion of forcing terms needs a bit more attention. Firstly, forcing terms are distributed, meaning they affect whole domain. As argued in [127, 130], not all codes are able to include such terms. Some commercial codes can quickly cause problems since the user has not a lot of options if the code comes without the possibility to include user defined distributed forcing terms. Secondly and also generally important, when implementation of forcing terms is done, the main consideration is to ensure that the way in which the forcing terms are developed is respected also in their inclusion in the code. Since MFLOPS-3D code is concerned in this work, example of this is here given for the case of this code. Example also shows in a bit more detail how forcing terms are devised. It is supposed that we have incompressible flow case with proposed velocities u, v, w in x, y, z directions respectively and pressure p. Three forcing terms are needed, one for each velocity field. Pressure does not need forcing, as it is obtained through projection step and is hence obtained directly on behalf of divergence free constraint and results for $\vec{v}^*$. To obtain the three forcing terms, the proposed analytical expressions are put in corresponding Navier-Stokes momentum equations. These are written with all terms on one side. This is so as the forcing

terms follow from an imbalance of governing equations when the proposed solutions are included in them. An example, applied also in the case of MFLOPS-3D, is shown with the used normalized Navier-Stokes momentum equations, here rewritten with equation (5.2), where $\vec{F}_t$ represents the vector with the three forcing terms included.

$$\frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla)\, \vec{v} + \nabla p - \frac{1}{Re} \Delta \vec{v} = \vec{F}_t \tag{5.2}$$

The forcing terms are then applied to the solution procedure. In MFLOPS-3D, the forcing terms are included in the computation of predictor velocity $\vec{v}^*$. This follows from the use of pressure correction projection methods. Since the system of equations used in this computation is given by Helmholtz type equations shown with (3.51), the forcing terms have to be deduced on the right hand side of them. And they have to be multiplied by same factors which multiply also other terms, in this case Re. Example for the inclusion in the case of MFLOPS-3D code for incompressible flow is given in (5.3).

$$\left(\Delta - \frac{3Re}{2\Delta t}\right) \vec{v}^* = Re \left(\frac{-4\vec{v}^n + \vec{v}^{n-1}}{2\Delta t} + \vec{v}^n \cdot \nabla \vec{v}^n - \vec{F}_t\right) \tag{5.3}$$

Example above includes equations for all three directions. What should also be mentioned is that since MFLOPS-3D uses finite difference approach, the forcing term values in $\vec{F}_t$ are computed for each point and then imposed directly into (5.3). This has to be adjusted in the case of codes which use finite volume approach.

To conlude the presentation of MMS implementation, it could be only added that the boundary conditions which are adjusted to perform MMS simulations are in the incompressible flow case only the boundary conditions for velocity. Analytical values for velocities are imposed on all boundaries in normal directions. In cavitating flow case, the conditions for Φ and α are also adapted. The mixed boundary conditions for Φ demand inclusion of analytical pressure values on the chosen Dirichlet boundary. α simply implements the analytical values on all boundaries.

3 Devised solutions for MMS

This section presents the analytical solutions devised to perform verification tests with MMS. Many analytical solutions were developed, but since there is no point in presenting all of them, only those which were used in verification tests to obtain order of accuracy and also response of the new algorithm to increased magnitude of certain equation terms are given. The equations are given separately for the cases of cavitating and incompressible flow tests. The forcing terms from them are obtained with *Maxima*, a tool for manipulation of symbolic and numerical expressions [131].

Before presenting the equations, effects of mentioned respecting of velocity divergence constraint in forming analytical solutions should be addressed in a bit more detail than before. In the case of incompressible flows, where the flow expressions have to ensure the divergence free constraint $\nabla \cdot \vec{v} = 0$, one is only concerned that the velocities are proposed in a way to satisfy this condition. The only other important variable, pressure, can be prescribed freely. Contrary to this, one must for cavitating flow solutions, if here chosen governing equations are considered, ensure that velocity divergence and source term S relation is respected. This relation is here repeated with (5.4). Moreover, as mentioned before and included in (5.4), source term is a function of pressure and α. Therefore it is important that this connections are included within it too.

$$\nabla \cdot \vec{v} = S\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right) = f(p, \alpha) \tag{5.4}$$

These connections make the creation of analytical solutions for cavitating flow very difficult. Instead of only ensuring that velocities produce zero divergence result, velocities, pressure, α and S have to be all connected. The task could be made simpler with inclusion of additional source terms for continuity and α transport equation, which is an option mentioned in the last paragraph of section 1. The source terms would be used to enforce inequalities caused by the divergence of velocity not being equal to S (which can then be freely constructed as some function of α and p). But as it was mentioned before, this would make the verification of the algorithm questionable. Especially since the algorithm presents a new approach towards solving presented cavitating flow governing equations, where connection given with (5.4) plays a crucial role in solution of pressure. Therefore respecting of the mentioned connections was retained and it was shown that although difficult, it is still possible to devise analytical solutions.

Furthermore, another point can be raised in regards to before mentioned physical realism of the devised analytical solutions. Although these solutions, despite respecting here discussed connections, do not follow physical realism, it seems that they can be used for tests in which responses of a code to certain conditions could be analysed. Here, it should be considered that in [130], it is argued that making conclusions about code's performance in real flow on the basis of MMS can be misleading. Nevertheless, as an algorithm and code, such as presented here, were never before used in real cavitating flow simulations, making some predictions about their abilities with computationally non demanding MMS tests seems a very attractive option. But for this, the analytical solutions should be devised in a manner which enables desired changes in theoretical flow behaviour. In case of cavitating flow, these would be for instance quick pressure and velocity changes, various levels of pressure or velocity magnitudes and presence of density and viscosity changes. The largest limitation is then one's ability to devise analytical flow expressions which offer such tests and still satisfy the imposed constraints of the flow.

3.1 Incompressible flow analytical solutions

The incompressible flow analytical expressions, used to create forcing terms, are given in equations (5.5)−(5.8). It can be noted that both velocities are a function of only one coordinate, which is on a first look a drawback. Moreover, velocity in z is set as zero, making the case essentially two dimensional. Cases where both u and v are functions of more than one coordinate and z is non zero are possible to be devised, but it was opted to use the presented equations. The reason for this was in the choice to have a case for incompressible flow which shares at least some similarities with the case devised for cavitating flow.

$$u = C\cos(gt)\cos(ay) \tag{5.5}$$

$$v = C\cos(gt)\sin(ax) \tag{5.6}$$

$$w = 0 \tag{5.7}$$

$$p = uv \tag{5.8}$$

The coefficients in the above equations are used to control the flow in the tests. Coefficient a controls the frequency of spatial oscillations while g is used for definition of oscillations in time. Constant C gives the amplitude of velocities and consequently also pressure. The forcing terms which follow from the presented analytical solutions are given in the Appendix. They were obtained in the manner as shown in previous section.

The above given equations respect the demands mentioned in presentation of MMS theory. With them, all the terms in the NS equations are present or non zero, expressions for variables are smooth and also give smooth non trivial derivatives. No singular points, discontinuities or locally high values are present and there are no limits regarding domain size. Finally, the amplitude of the source term in regards to other variables is also favourable. This is illustrated on figures 5.1, 5.2, which present development of certain terms in four chosen points over time, and on figures 5.3, 5.4, which show instantaneous values of terms in whole domain at two time levels. The shown terms are all terms from NS momentum equations for x and y directions. The equation or terms in z direction are zero as w is zero and are therefore not included. The used domain size is $\{-0,2;0,2\}$ m in both x and y directions. The four chosen points are located at $A(-0,1;-0,1)$, $B(0,1;-0,1)$, $C(0,1;0,1)$ and $D(-0,1;0,1)$. The values of the coefficients in proposed solutions are $a = 2\pi$, $g = \pi$ and $C = 0,55$. The domain and also coefficients are those which were chosen to perform Verification of order of accuracy for the original code and their choice was heavily based on the devised solutions and Verification of order of accuracy for the new algorithm for cavitating flows. Since the NS equations are for incompressible flow normalized with Re value, which is in MFLOPS-3D taken as a ratio between liquid density and viscosity, this value was set as $Re = 100$ and was also the same as used in the mentioned verification tests. The surface plots on figures 5.3 and 5.4 show instantaneous values at times $t = 0,25$ s and $t = 1,75$ s, respectively.

From the presented graphs it is clear the forcing terms do not exceed the amplitude of other terms. Since their amplitude is in the range of highest terms in NS equations, the question is if this is still acceptable or if the terms should be smaller. It can be seen in [126] that similar relation between the forcing terms and other terms from NS momentum equations was used, therefore it can be said that the proposed analytical solution is appropriate for code verification.

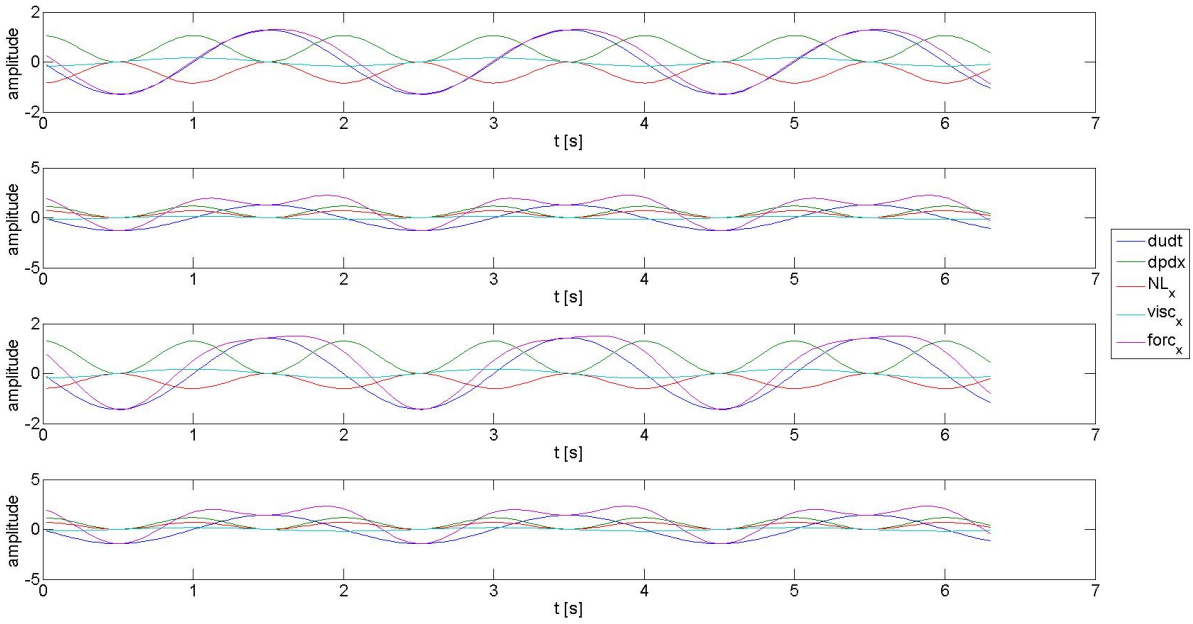


Figure 5.1: Time development of time derivative ($dudt$), nonlinear (NL_x), pressure ($dpdx$), viscous ($visc_x$) and forcing terms in points A, B, C and D (from top to the bottom) for x direction NS momentum equation.

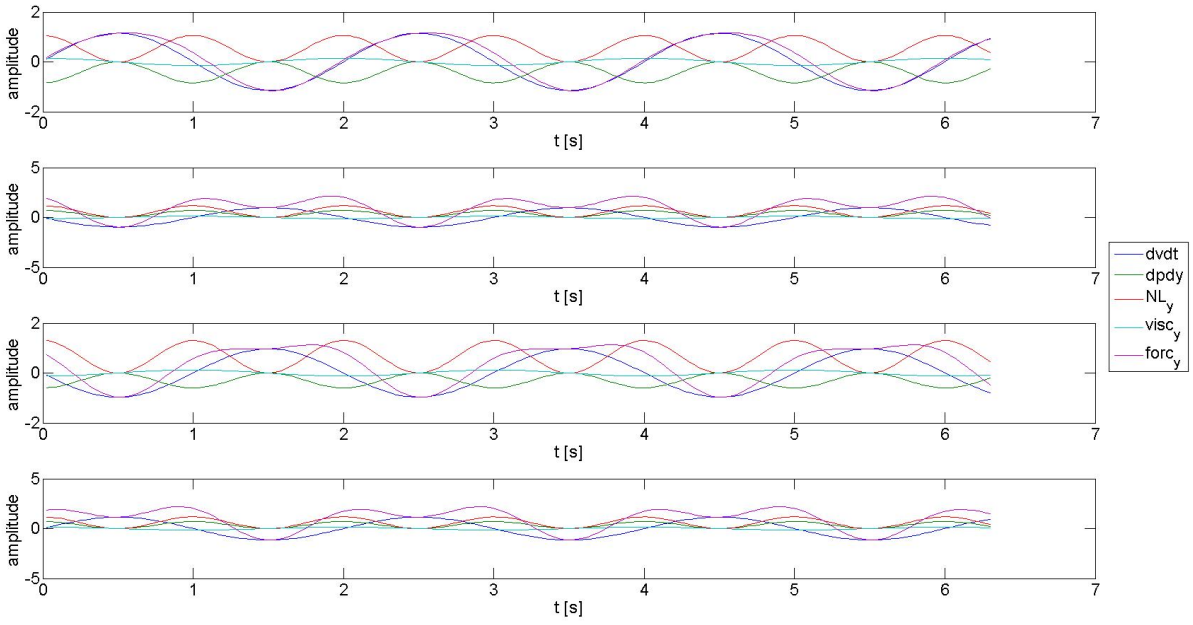


Figure 5.2: Time development of time derivative ($dvdt$), nonlinear (NL_y), pressure ($dpdy$), viscous ($visc_y$) and forcing terms in points A, B, C and D (from top to the bottom) for y direction NS momentum equation.

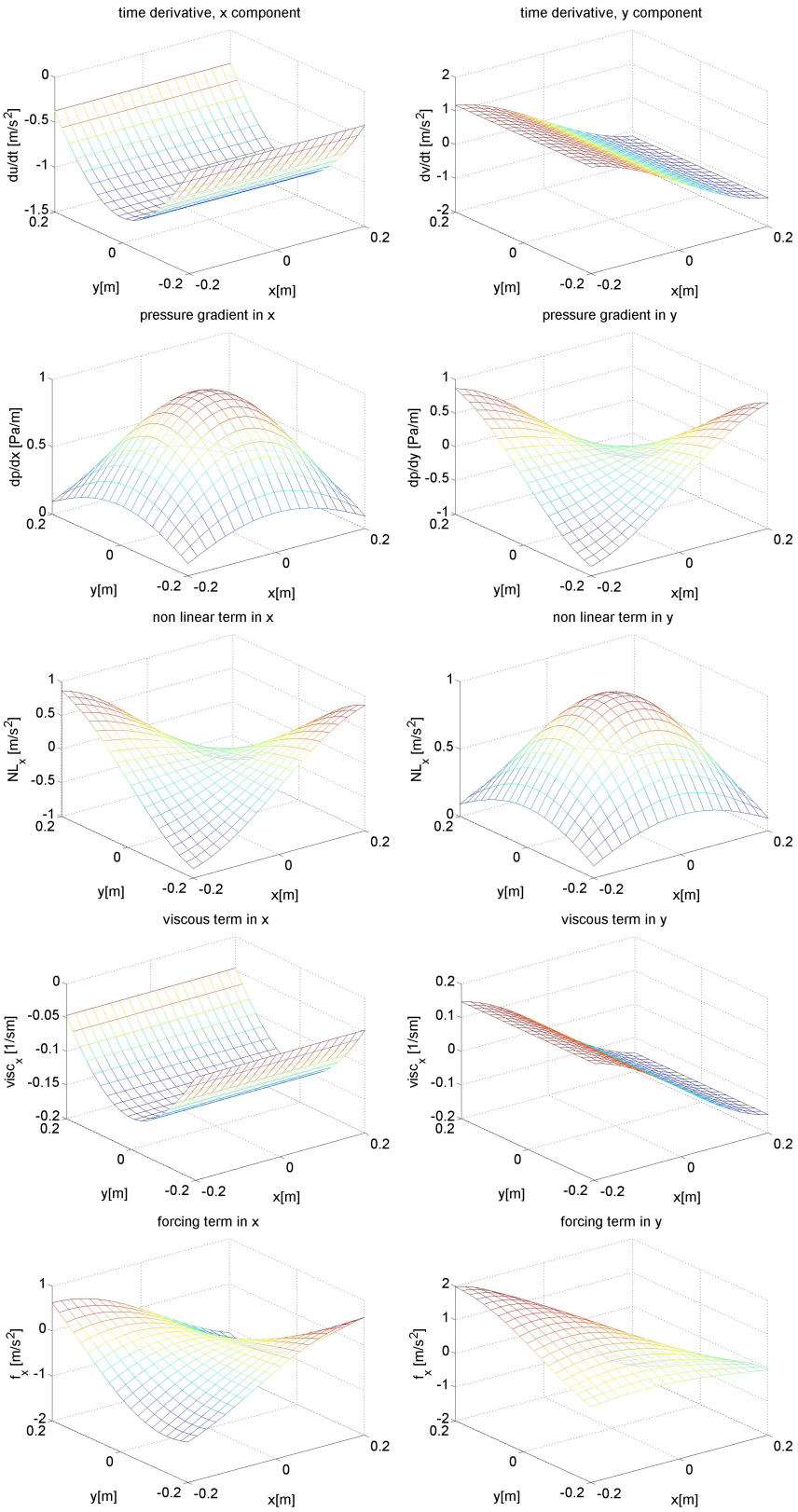


Figure 5.3: Instantaneous values of terms in NS momentum equations at $t = 0,25\ s$. Left column presents terms from equation in x direction, right one terms for y. Time derivative, pressure gradient, non linear, viscous and forcing terms are shown in rows from top to the bottom.

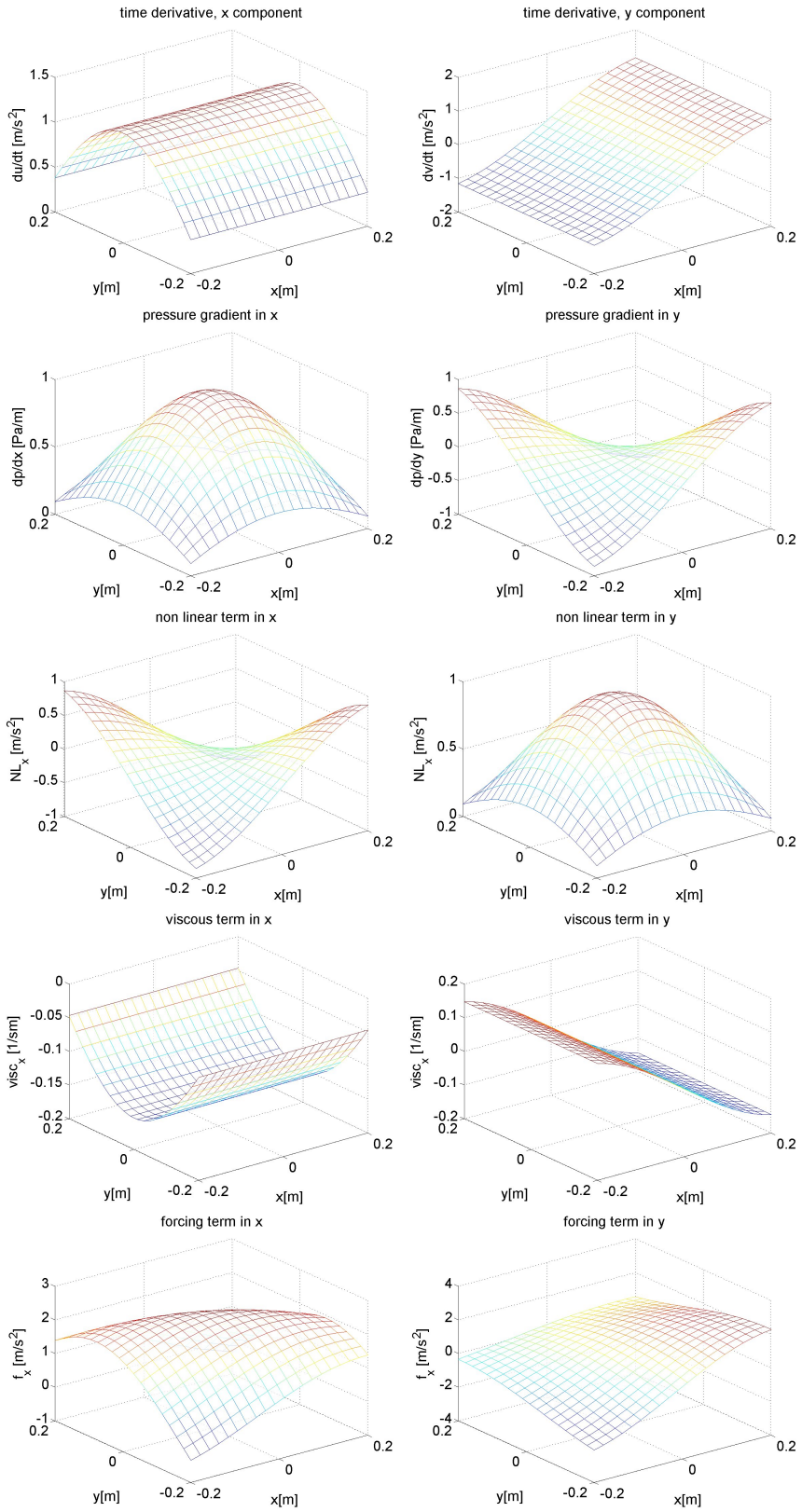


Figure 5.4: Instantaneous values of terms in NS momentum equations at $t = 1,75\ s$. Left column presents terms from equation in x direction, right one terms for y. Time derivative, pressure gradient, non linear, viscous and forcing terms are shown in rows from top to the bottom.

3.2 Cavitating flow analytical solutions

Development of solutions for the cavitating flow took much more time than development of solutions for incompressible flow case. The solutions presented in the previous section were actually developed after final solutions for cavitating flow were obtained, because of the mentioned decision to have some similarities between the two. The decision to respect the connection given by (5.4) and form the cavitating flow source term as a function of α and p led to many iterations of the proposed analytical equations. Finally, the solutions shown with equations (5.9)−(5.14) were found.

$$u = C\cos(gt)\cos(ay) \tag{5.9}$$

$$v = \left(1 - \frac{\rho_v}{\rho_l}\right)\frac{BC\cos(gt)\sin(gt)\cos(ax)\sin(ay)}{1 - \left(1 - \frac{\rho_v}{\rho_l}\right)\alpha} + \frac{Bg\cos(gt)\sin(ax)y}{1 - \left(1 - \frac{\rho_v}{\rho_l}\right)\alpha} \tag{5.10}$$

$$w = 0 \tag{5.11}$$

$$p = \frac{BCa\cos(gt)\sin(gt)\cos(ax)\cos(ay)}{\alpha\left(\left(1 - \frac{\rho_v}{\rho_l}\right)\alpha - 1\right)} + \frac{Bg\cos(gt)\sin(ax)}{\alpha\left(\left(1 - \frac{\rho_v}{\rho_l}\right)\alpha - 1\right)} \tag{5.12}$$

$$\alpha = B(\sin(ax)\sin(gt) + 1) + K1 \tag{5.13}$$

$$S = -\alpha p \rho_v \tag{5.14}$$

It can be seen that velocity u is the same as for incompressible flow case. But other terms are much more complex. The reason is in the way they were created in order to satisfy the mentioned connection and ensure $S = f(\alpha, p)$. As no similar examples were found in the mentioned sources and also other literature, it seems the presented solutions are the first of a kind. Therefore the way in which they were devised should be shortly explained, as codes which are based on same modelling of cavitating flows can now have verification tests performed with MMS since such analytical solutions are available. It was found that at first, source term form has to be precised. For simplicity, this was chosen to be direct product of α and p, as shown with (5.14). Then, α should be chosen. This is done as α transport equation can then serve for definition of other terms. For first, simpler analytical solutions which were developed, α was kept constant in time and one direction. With α transport equation this enabled predefinition of pressure solution too. Definitions of velocities then follow from the transport equation and $S - \nabla \cdot \vec{v}$ connection. Constant α in time and one direction is not a good choice for verification, since it hides possible issues with temporal resolution of α. But when α and also p are chosen to vary in time and space (and with them also S), the whole operation of obtaining complete set of analytical solutions becomes more complex. The approach taken in the scope of this work was then to not propose pressure but a velocity component (one is already set - w was always kept zero). To make the connection with S simpler, proposed component has a zero derivative in its direction. Hence the given u in (5.9). With the use of α transport equation and knowing that the only unknown part of $\nabla \cdot \vec{v}$ is $\frac{\partial v}{\partial y}$, while in S the pressure is not set, one can form an expression to obtain one of the two remaining unknowns. Then, *Maxima* or another symbolic operating tool can be used to obtain v component of velocity. The presented analytical solution was obtained in such manner. It should be also mentioned that α was chosen deliberately as a function of x and not y to make

manipulation with α transport equation easier. What is also important is that in its expression, B governs the amplitude of its oscillations and $K1$ sets the minimum amount α_{min} always present in the domain. This makes it possible to have α between 0 and 1 or adjust its range and minimum amount. These two coefficients are also the only two added compared to incompressible flow case, others have same meaning as before. Moreover, the source term has negative sign in order to resemble its positive nature when pressure drops below vaporisation pressure and vice versa. Therefore vaporisation pressure p_v is considered as set at 0 Pa. Regarding its form, it can be seen that it resembles source terms proposed by empirical cavitation models presented in section 2.2.2 of chapter 1. Also, very important aspect of this source term is that contrary to other forcing terms implemented by MMS, it does not need to be added to $\vec{v}^*$ or other equations to force the flow, but is obtained by the code itself if the solution algorithm solves the system of governing equations correctly. Which is equal to the real flow simulation cases. Comparing the devised analytical flow expressions here and those before for incompressible flow case, it can be seen that u velocities are the same. This is the consequence of the wish to have some similarities between the two cases in order to better compare performance of both algorithms. As expression for u is the simplest, the similarity was built on it. In the case of incompressible flow, this then conditioned also proposal for v, hence both expressions are simply functions of one coordinate.

The presented analytical solutions can fulfil the demands mentioned in the theoretical part, although some care must be taken. The main issue is that p can cause a division by 0 if settings permit $\alpha = 0$, therefore a singularity can be present. However, this can generally be avoided with having some $K1 \neq 0$, which is acceptable from the point that even actual cavitation models, where phase transfer is governed by the source term, demand some α_{min} be set in the domain, otherwise cavitation would never occur in simulations. The presence of α in denominator also affects definition of v, which can have higher amplitude than u, if the ratio between the densities is close to zero and α close to one. Again, the problem can easily be avoided by including some non zero $K1$ value. Finally, the amplitude of v compared to u can be also affected by presence of multiplication with y coordinate and factors B, C and g. Especially multiplication with y can make v and its derivatives much higher than u and its derivatives. Therefore domain limits and their effects need to be observed. Since this is a minor drawback and can be, together with all other potential problematic terms, well controlled, it was considered the solutions are nevertheless acceptable. One other possible violation of demands connected with possible pressure singularity exists and can be controlled with $K1$. This violation is discussed later, as it importantly affects the chosen case for Verification of the order of accuracy.

Another criteria for acceptance of the proposed solutions is the amplitude of terms in NS momentum equations compared to developed forcing terms. These amplitudes are shown in same manner as for incompressible flow and come from the proposed case for performing Verification of the order of accuracy. Its choice will be discussed in the next chapter. Here, only the values of coefficients used in analytical equations are given. These are $a = 2\pi$, $B = 0,5$, $C = 1$, $g = \pi$, $K1 = 0$. Same domain is used in this case as in the previous one. The difference from incompressible flow is therefore in value of C. This comes from the need to keep same CFL number, which will be referred to later. Material properties were chosen as $\rho_l = 10\ kg/m^3$, $\rho_v = 3\ kg/m^3$, $\mu_l = 0,05\ Pas$, $\mu_v = 0,01\ Pas$. This is not realistic, but as mentioned in [127, 130], the point of MMS is to show equations are solved correctly and not to be concerned about physics too much. This is the task of Code Validation. The material properties are therefore based on choices to evade the possible issues coming from high ratios between liquid and vapour properties yet to have notable difference between the two phases. The Re value as defined in incompressible flow case is here cca two times higher ($Re \approx 200$) for whole α range. Figures 5.5

and 5.6 show time development of all terms in four chosen points in NS momentum equations for x and y direction respectively. Surface plots of instantaneous values in the domain at same times as before ($t = 0,25\ s$ and $t = 1,75\ s$) are given on figures 5.7 and 5.8.

As it can be seen from those figures, the amplitude of terms increases highly in comparison to incompressible flow. The relation between them changes too, with pressure gradient and forcing terms becoming the highest in contrast to velocity time derivatives before. The amplitude also changes highly between the directions, where the terms from NS momentum equation for x direction reach higher values than those for y direction. It should also be noted that though viscous terms have relatively much smaller amplitude compared to other terms as in the incompressible flow case, their absolute amplitude is nevertheless higher than in that case. Although there are one to two less terms which have same amplitude as forcing terms, the proposed solutions were still taken to be appropriate for use. The main support for this is the high amplitude of pressure gradient terms, making the quality of pressure solution very important. Since pressure solution uses no forcing terms from MMS but needs to find correct S, this makes a very good basis for verification of the most important and novel part of the developed algorithm.

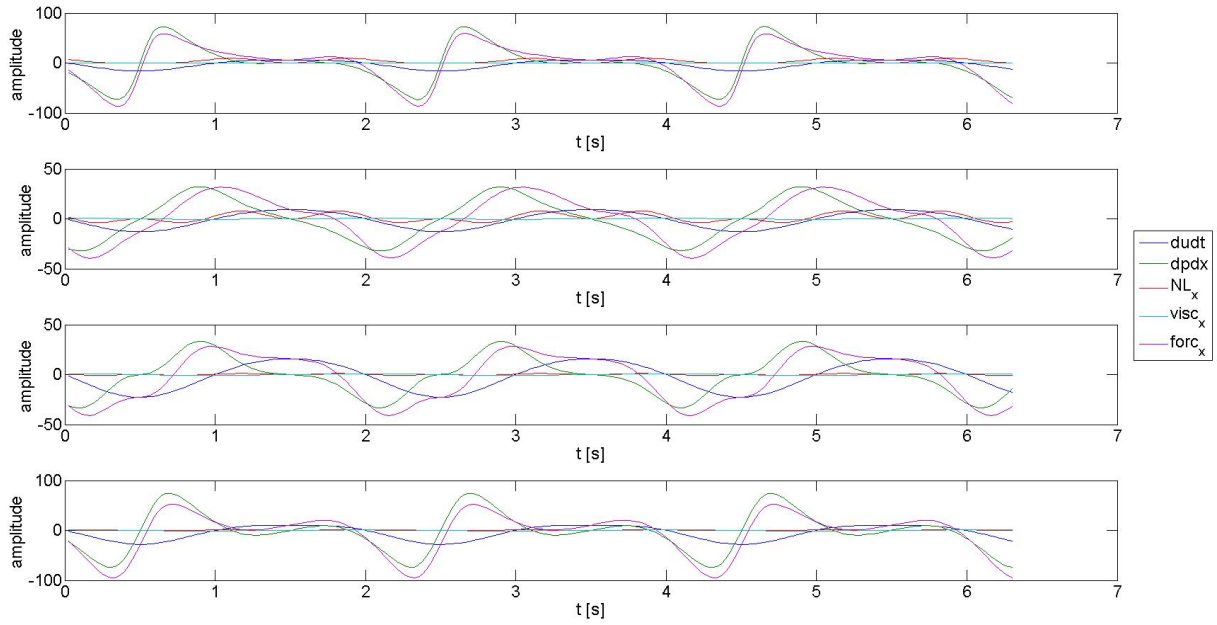


Figure 5.5: Time derivative ($dudt$), nonlinear (NL_x), pressure ($dpdx$), viscous ($visc_x$) and forcing terms in points A, B, C and D (from top to the bottom) for x direction NS momentum equation for cavitating flow.

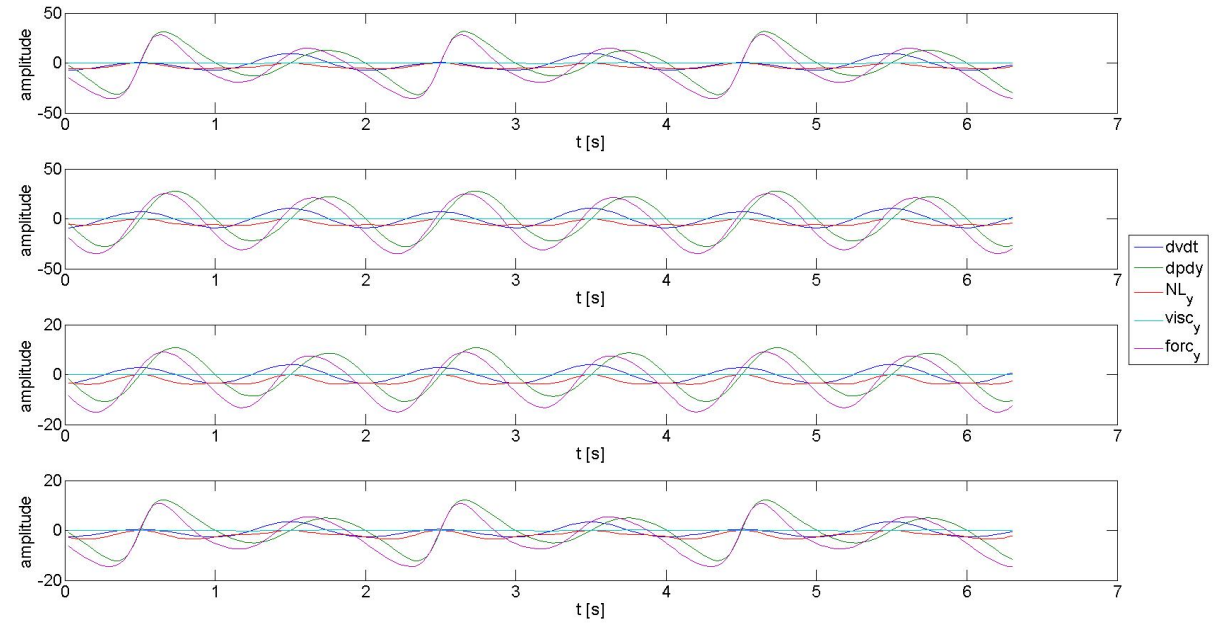


Figure 5.6: Time derivative ($dvdt$), nonlinear (NL_y), pressure ($dpdy$), viscous ($visc_y$) and forcing terms in points A, B, C and D (from top to the bottom) for y direction NS momentum equation for cavitating flow.

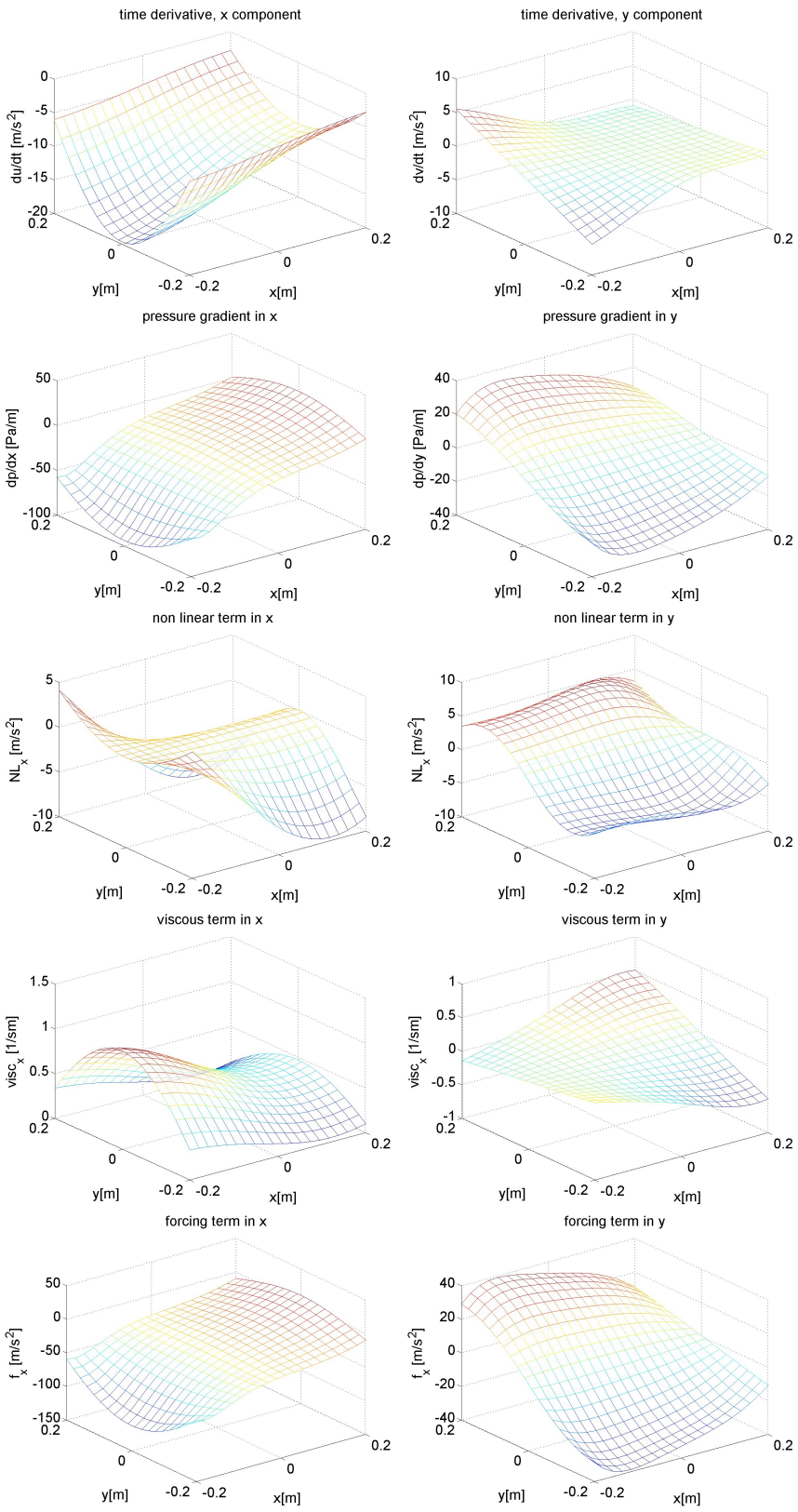


Figure 5.7: Instantaneous values of terms in NS momentum equations for cavitating flow at $t = 0.25\ s$. Left column presents terms from equation in x direction, right one terms for y. Time derivative, pressure gradient, non linear, viscous and forcing terms are shown in rows from top to the bottom.

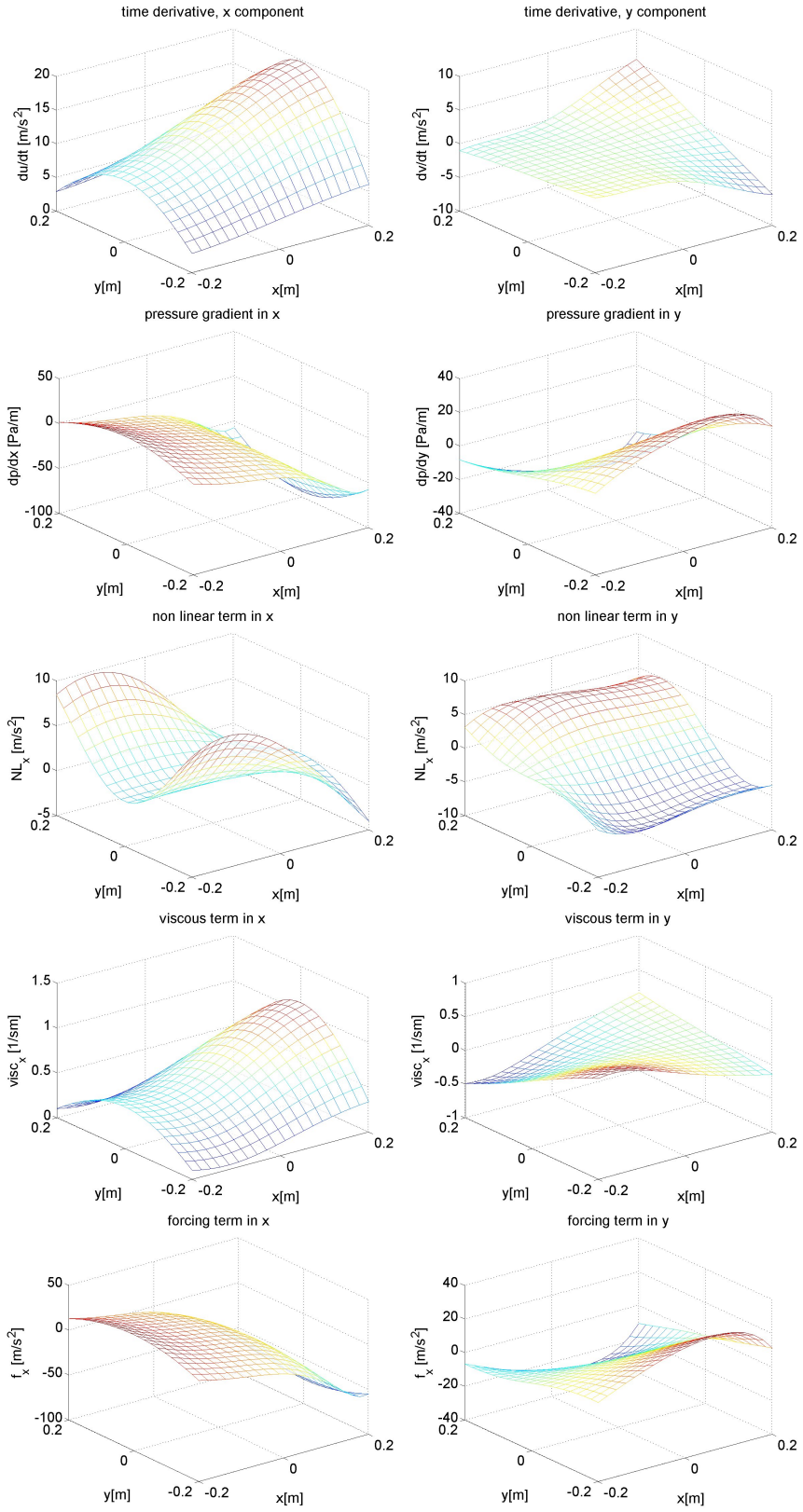


Figure 5.8: Instantaneous values of terms in NS momentum equations for cavitating flow at $t = 1.75\ s$. Left column presents terms from equation in x direction, right one terms for y. Time derivative, pressure gradient, non linear, viscous and forcing terms are shown in rows from top to the bottom.

4 Conclusions

The Method of Manufactured Solutions, used to develop the new algorithm and also verify it, was presented in this chapter. The shown analytical expressions for both incompressible and cavitating flow also form a basis for the next chapter, where Verification of the order of accuracy is given for the new and original algorithm and code.

There were actually many analytical flow cases developed, especially for the cavitating flow. These were all used in the development of the algorithm, for which it is difficult to describe how practical MMS turned out to be. The analytical expressions presented here were developed over different iterations of these cases and finally give a suitable tool for verification of the newly developed code and algorithm. Here, it is important to stress that to our knowledge, these expressions seem to be the first of the kind. Of course they belong to specific cavitation modelling, but with them, a new tool for verification of corresponding algorithms and codes for cavitating flow is available. This shows also great benefit for future development of such numerical tools as it brings the abilities of MMS to this area of simulations too. For instance, it was mentioned at the beginning of this chapter that MMS can enable one to find solutions for solving a certain flow problem easier and faster. This was also the experience in our case. Moreover, cavitating flow codes and algorithms were up to now often verified by simply applying them to perform real flow simulations for which satisfactory experimental results exists. As it was discussed in chapter 1, particularly its section 3, flows in venturi geometries with 18° or $4,3°$ converging and 8° or 4° diverging part were often used in such activities. Length of cavitation structures, appearance of re-entrant jet and frequency of cloud shedding are some examples of observed phenomena, used to verify and also validate certain algorithms and codes. Such tests to see if a certain solution procedure performs in desired manner are cumbersome from the point of needed time and computing power. Use of the developed MMS terms on the other hand offers a tool which can well improve this situation. Furthermore, the logic in which the shown analytical expressions were developed could serve to propose also analytical expressions suitable for other types of cavitation modelling. Which would even broaden the usefulness of this approach to develop and verify codes and algorithms for cavitating flow simulations.

To conclude the presentation of MMS method and make the presentation of the newly developed analytical solutions for cavitating flow complete, it can be said that the shown solutions also enable assessment of the new code's and algorithm's performance in various conditions. As it was mentioned before, it can be misleading to make predictions about performance in real flow conditions on the basis of MMS tests. But since nothing is known about performance of an algorithm as developed here in the case of real cavitating flow simulations, it would be somehow a wasted opportunity not to try and identify possible issues with the use of MMS already. Especially since such tests are computationally much less expensive than real flow simulations. Because of this it was decided that not only verification tests will be done with MMS but coefficients in developed solutions and ratios $\frac{\rho_l}{\rho_v}$ and $\frac{\mu_l}{\mu_v}$ will also be varied in order to see the algorithm's and code's response. And possibly make some predictions about behaviour or demands in real flow cases.

Chapter 6

Algorithm and code verification and performance tests

This chapter deals in large part with the activities concerning performing Verification of the order of accuracy (VOA from here on) of the original and new code, which includes all the proposed versions of the new algorithm. The MMS test cases used for this were then used for computational performance analysis of both codes (and algorithm versions) as well. The chapter is therefore divided into five parts:

- Chosen domain dimensions and factors for analytical flow expressions used in MMS are discussed at first. Criteria for VOA is also defined.
- Then, verification results concerning incompressible flow case are given. At first for the original code, as these define the reference point. After, verification results for the new code and algorithm in incompressible flow case are presented.
- Incompressible flow results are followed by VOA for the new code and all new algorithm versions in cavitating flow case.
- As MMS cavitating flow case was used to set some predictions about the new code and algorithm's response to certain increased effects, the results concerning MMS tests aimed at this are presented as well.
- Finally, performance analysis of the original and new code, including all new algorithm versions, is given.

Although the code is meant to be used in multi domain computations, results are given for both singe and multi domain cases. The reason is in the wish to verify the new algorithm works well in both cases and because some important differences were found between mono and multi domain computations. The noted differences are not always the same. The more important ones are also not a consequence of the algorithm or test case, but the multi domain solution itself. Although the differences are referred to and discussed in parts dealing with VOA, the main reason for those connected with multi domain solution is given in the last part of this chapter, dealing with computational performance analysis.

1 Test case choice

In description of the proposed analytical flow expressions for incompressible and compressible flow cases in sections 3.1 and 3.2 of previous chapter, the size of the domain and values of the coefficients used in performing VOA are already given. However, their choice was not explained, only the fact that they are based on the case for cavitating flow algorithm verification. These reasons are therefore given in the following section. Since mesh and time step are also important in performing VOA, the chosen meshing, mesh refinement and time steps are discussed afterwards.

1.1 Chosen domain dimension and factors

Choosing domain and factors in analytical flow expressions on the basis of cavitating flow case is obligatory as these expressions are much more complex than those for incompressible flow case. As mentioned in section where they are derived, they can also lead to violation of demands imposed to perform valid VOA. Expressions can quickly result in very uneven velocity and pressure fields, making it difficult to perform VOA without favouring a certain variable. Which makes a case for the before given factors and domain size.

For instance, estimation can be made that equations for u and v velocities can lead to better convergence results for u than for v, as the former has a simpler definition, which also makes its magnitude not vary over the whole domain equally. As mentioned before, v and pressure are dependant on α. Another issue is that v velocity includes a direct multiplication with y coordinate, making for increasingly higher amplitude than that of u with the size of the domain. On the other hand, u depends only on y but not on x coordinate, while v depends on both. Therefore same dimensions in x and y with a centre at zero are a straightforward choice. Same order amplitude of the two velocities is preferred, with ability to reach all values between zero and absolute amplitude. The choice of $a = 2\pi$ and domain size of $\{-0,2; 0,2\}$ m is the first step in ensuring this. With such a and domain size, spatial oscillations can reach almost all values between zero and unity (sin and cos functions in range $\{-0,4\pi; 0,4\pi\}$). Ensuring complete range of values available to spatial oscillations was avoided because of pressure singularity. If $x \approx \pm 0,5$ m is reached, pressure suffers from singularity effects, causing also much faster temporal changes from positive to negative values in vicinity of singularity. This can be avoided (singularity) or made slower (change) by using non zero $K1$, but it was opted not to apply it. The reason was in the past experience when the algorithm was developed, where $K1$ helped produce more stable source term update. However, as multiplication with the temporal function is applied, u can reach all values between zero and unity over time. On the other hand, presence of g, B and C factors, together with influence of α in the denominator, ensure that v also reaches values of same order of amplitude over time. Where the $B = 0,5$ was chosen to have highest range of α possible, C and g are left to achieve desired similarity between v and u. The $C = 1$ and $g = \pi$ return highest absolute v of cca $0,625$ m/s (depending on the time step used).

It can be noted that the focus in the domain size and factors choice is on the range and magnitude of velocities. Pressure was observed to ensure the singularity is avoided, its magnitude otherwise was not the primary concern. It was found however, that pressure itself reaches cca five times higher amplitudes than velocities, confirming it is a very important term in the solution, especially through its inclusion in the source term. Moreover, the fast changes in its sign at the limits of the domain in x are still present as $K1$ was not used. This is the possible violation mentioned but not discussed in section 3.2 of chapter 5, where the analytical flow expressions are derived. In order to discuss this, the demand that the cases should respect constraints imposed

by the flow or flow model should be recalled. This demand was already discussed as it presents an important decision chosen for development of analytical expressions (to ensure $\nabla \cdot \vec{v} - S$ connection). It also backs the presence of fast changes in pressure. The governing equations solved by the algorithm are chosen to correspond to single phase modelling of cavitation where α transport equation is used with models based on bubble dynamics or empirical relations. In such cases, the switch between cavitation growth and collapse is quick and happens when pressure crosses vaporisation pressure p_v because S sign changes with $p-p_v$ difference sign. The problem of this quick source term change regarding the issues it imposes on the stability of pressure solution is well discussed in [8] and also in this work. Therefore an issue with verification arises. On one hand, demands for good verification with MMS suggest that there should be no too high temporal or spatial changes of variables. On the other hand, the demands imposed by the flow model should be respected. In the scope of this work, a somehow middle way is used. The presence of fast pressure and source term sign change was retained but at the same time the change was still bounded and equal in each cycle. Analytically the change is not discontinuous, although numerically this cannot be ensured (not even with applying some value of $K1$, although the change is made lower and smoother). In such manner, one of the main challenges in performing cavitating flow simulations with such models is included in the verification procedure, but in a bounded manner. Finally, presence of such characteristic is considered to be welcome as it makes for a good test of stability for the proposed new Φ solution procedure. Therefore we reckon that inclusion of such characteristic provides a more thorough verification test. An example of pressure change during two time steps (at times $t = 0,49\ s$ and $t = 0,504\ s$) with corresponding change in source term is given in figure 6.1. This is also an illustration of the changes at longest time step used in VOA, therefore it also gives estimation of highest order of magnitude of this changes applied in VOA. The pressure field in it is very uneven, but it exhibits such characteristic only during short period when the illustrated changes happen. The following figure 6.2 shows this fast pressure change in one oscillation time in two limiting positions of the domain, where the change is the highest. It can be seen that in whole domain, two changes per cycle happen (both in different position).

With this, the justification of the chosen domain and coefficients is at the end. It can be only added that a test case with less pronounced change in pressure could be performed, but the focus was kept on the described one. The reasons are in the cumbersome verification procedure performed which finally demanded a lot of CPU hours and lack of time after VOA with the chosen coefficients was completed. Nevertheless, such a test case gives also a good basis for the later increase in various coefficients to see how the algorithm reacts to various other cases. As the VOA for incompressible flow algorithm was also performed and compared with results for the new algorithm, the choice for its coefficients and domain naturally followed from the compressible flow case. Moreover, the VOA for the new algorithm was also performed with incompressible flow case to verify that the order of accuracy in such case does not differ much from goal order of accuracy. The domain was therefore kept unchanged, hence also a was the same. Same goes for g. The value for $C = 0,55$ was chosen as it ensures same order of amplitude for velocities and similar CFL number as in the cavitating flow case. This further makes for a better comparison between the two codes. As CFL depends on the chosen mesh and time step, the reasons for such C factor will be made more obvious with the following section on the mesh and time step choice.

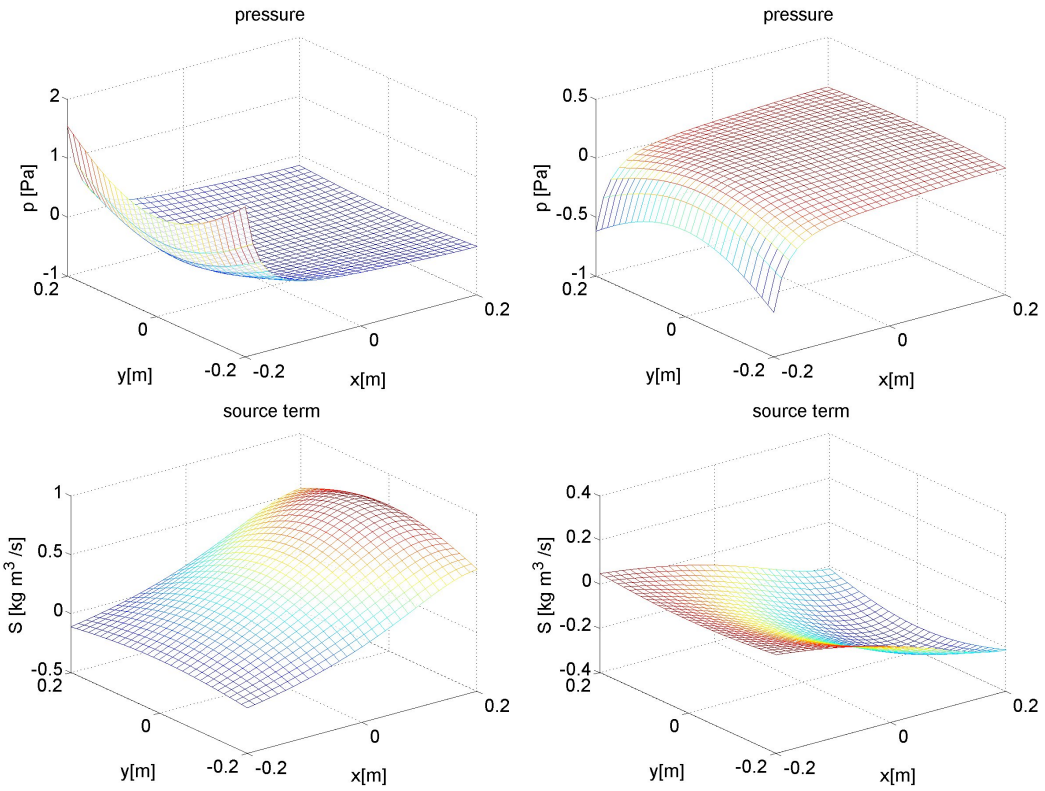


Figure 6.1: Change in pressure and source term during two time steps. Left columns gives the state at $t = 0,49\ s$ and the right at $t = 0,504\ s$.

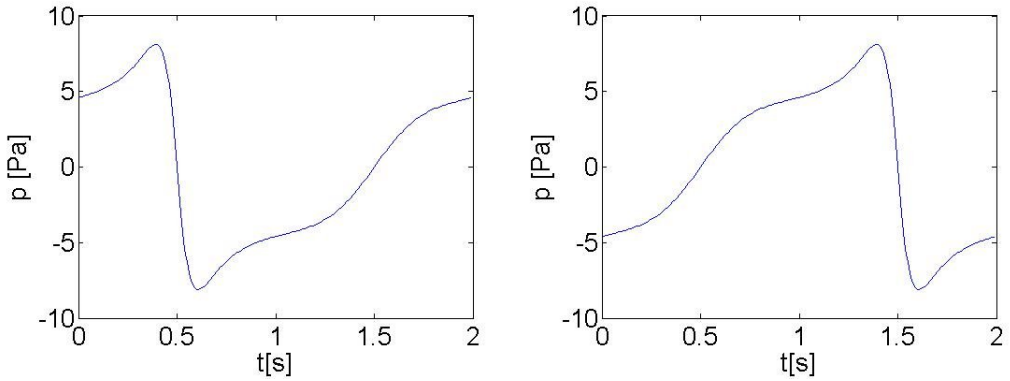


Figure 6.2: The two graphs show pressure change in one cycle in positions $x = -0,2\ m\ ;\ y = -0,2\ m$ (left) and $x = 0,2\ m\ ;\ y = 0,2\ m$ (right).

1.2 Chosen meshes and time steps

An important aspect for performing desired verification is also that the choice of domain and coefficients should be appropriate for gradual mesh refinement. The before described case was found to be suitable from this point as well. The reason is that even for the chosen dimensions and coefficients, which do not permit the spatial oscillations to reach complete range of values on their own and therefore present a relatively small computational domain, more than 30 points in x and y direction have to be used in order to find a stable solution. As the calculations are run with non zero z dimension, where the lowest amount of points is limited by stencils of compact finite differences (10 per direction), even such relatively small case leads to a cumbersome verification procedure, especially when single domain calculations are considered. The mesh has to be namely refined in all three directions, although the flow is essentially 2D, as the accuracy of solution in z can affect accuracy in other two dimensions as well. Nevertheless, same refinement level is not required. If a bigger domain or higher a value were chosen, with coefficients adapted to them, the verification procedure would probably become even more computationally demanding. Therefore the chosen domain and coefficients can be taken as appropriate from this points of view for performing VOA too.

Regarding the mesh, it was found that a non uniform mesh has to be used. Uniform mesh resulted in unstable simulations, the reason was found to be in the solution for α and use of second order upwind derivative in it. More precisely, main issue is in the accuracy near and on interfaces of sub domains or, in the case of mono domain computation, domain limits. Therefore it suffices to refine the mesh towards these locations, meaning that in multi domain cases, the mesh is equally locally refined in each sub domain, while in mono domain cases, only global refinement is used. The refinement in both cases uses same law for all directions, based on *tanh* function given in equation (6.1). In it, $x(i)$ is location of a certain point, x_a and x_b are the limits of the (sub) domain, while δ is used to govern the refinement and was set as $\delta = 1,1$. The $x(i)$ included on the RHS is the position of i^{th} point on uniform grid. This gives ratio between largest and smallest distances between grid points on a certain mesh of cca $\Delta x_{max}/\Delta x_{min} \approx 2,65$.

$$x(i) = x_a + 0.5(x_b - x_a)\left(1 + \frac{\tanh\left(\delta\left(2\frac{x(i)-x_a}{x_b-x_a} - 1\right)\right)}{\tanh(\delta)}\right) \tag{6.1}$$

The ratio $\Delta x_{max}/\Delta x_{min} \approx 2,65$ holds for all used meshes. The starting mesh had 41 points in x and y directions and 11 points in z. This was then gradually increased to the most refined mesh, which had 197 points in x and y and 41 points in z. The meshes used for VOA in cases of mono and multi domain tests had same amount of points, but as explained before, the mono domain cases had refinement applied only towards the limits. The used meshes for mono domain cases are included in table 6.1, while for multi domain cases, table 6.2 gives information about number of sub domains (and used processors), points in them and their corresponding ratios.

It can be seen that mono domain cases reach higher $\Delta x_{max}/\Delta x_{min}$ ratio than multi domain ones, but the Δx_{max} and Δx_{min} differences between them both are small. Importantly and contrary to expectations, it is obvious that the applied mesh refinement does not impose successive grid halving. Halving the grid to perform VOA is actually not advisable in this case, as compact finite differences are used. Regarding this, [127] argues that for discretization with finite differences, one should always use smooth grids. If successive grid halving is applied, non smooth grids are obtained and lead to possible decrease in observed order of accuracy. In [130] a more general advise is given, that successive grid halving is not required, only refinement suffices. Therefore it

Table 6.1: Meshes used for mono domain VOA computations.

Points in x and y	Points in z	Δx_{min} [m]	Δx_{max} [m]	ratio
41	11	0,00516	0,01373	2,66
61	17	0,00339	0,00916	2,70
81	21	0,00252	0,00687	2,73
121	27	0,00167	0,00458	2,74
161	33	0,00125	0,00344	2,75
181	37	0,00111	0,00305	2,75
197	41	0,00102	0,00280	2,76

Table 6.2: Meshes used for multi domain VOA computations.

Pts in x/y	Pts in z	Sub dom in x/y	Sub dom in z	Pts in x/y per sub dom	Pts in z per sub dom	Δx_{min} [m]	Δx_{max} [m]	ratio
41	11	2	1	21	11	0,00539	0,01369	2,54
61	17	2	1	31	17	0,00349	0,00914	2,62
81	21	4	1	21	21	0,00269	0,00684	2,55
121	27	4	1	31	27	0,00174	0,00457	2,63
161	33	5	1	33	33	0,0013	0,00343	2,64
181	37	6	2	31	19	0,00116	0,00305	2,63
197	41	7	2	29	21	0,00107	0,0028	2,62

can be concluded that the applied refinement of the grid is appropriate. Correctness of this claim was confirmed also in the performed VOA for incompressible flow code, where uniform grids were additionally used. Same amount of points were used in them as in the given tables.

Besides the number of points, the grids used in mono and multi domain computations share also same time step lengths. These were defined in order to keep similar highest instantaneous CFL numbers or CFL_{max} on all grids. The CFL_{max} is observed as the flow fields change in time between minimum and maximum values, therefore CFL also changes periodically. Important factor at setting the time steps is the fact that CFL in MMS calculations is less restricted than in real flows. Where real flow cases would demand CFL_{max} numbers for MFLOPS-3D in range $0,07-0,18$, the CFL_{max} numbers can be easily more than ten times higher when MMS is applied. The reasons for this can be found in the implicit nature and amplitude of forcing terms applied to equations for predictor velocities and on the other side lower amplitude of non linear and viscous terms. However, the oscillatory nature of CFL in these cases should be kept in mind. The presence of such high possible CFL_{max} numbers also means that much smaller instantaneous CFL numbers occur periodically as well. It was actually found that if CFL_{max} is below unity, results with MMS tend to be worse, in cases of both cavitating or incompressible flow. Presumably because of too low CFL numbers being present. On the opposite, the presented case for performing VOA for new algorithm was found to run in a stable manner even with $CFL_{max} > 4$. For the here discussed cases, $CFL_{max} \approx 2,5$ was used. The choice of time step lengths for each mesh was not focused on ensuring completely the same CFL_{max} number to occur in all performed calculations. This is also not of a big importance, as broad range of applicable CFL values shows computations are not sensitive towards this. Instead, the starting mesh was simply assigned time step length $\Delta t = 0,014\ s$. Then, two values of Δt were calculated for each mesh by assuming the local highest velocities in positions of Δx_{max} and Δx_{min} will not change from one mesh to the other. The two

Table 6.3: Used time steps for the chosen meshes and the corresponding CFL_{max}.

total pts in x/y	time step [s]	Multi dom compr	Mono dom compr	Multi dom incompr	Mono dom incompr
41	0,014	2,52	2,61	2,54	1,96
61	0,0093	2,55	2,68	2,69	1,99
81	0,007	2,41	2,73	2,55	2,02
121	0,004667	2,55	2,77	2,7	2,05
161	0,0035	2,63	2,79	2,69	2,06
181	0,00311	2,58	2,79	2,7	2,06
197	0,00286	2,58	2,8	2,67	2,07

Δt were therefore seen as time steps required to keep equal CFL_{max} in positions of Δx_{max} or Δx_{min}. Because of velocity assumption, they were both for each mesh calculated from relation given in (6.2), which follows from the definition of CFL number. On the basis of these two results, a time step for a certain mesh was chosen. This procedure was applied for the case of cavitating flow multi domain calculations and the chosen time steps were applied to corresponding meshes in mono domain calculations as well. And also for incompressible flow cases. The defined time steps for chosen meshes are given in table 6.3. The actual CFL_{max} noted in calculations is also given with them. This means that four groups of CFL values are given (for mono and multi domain cases of cavitating and incompressible flow). The CFL number calculated in simulations was defined for the case of 3D flow, although the flow is essentially 2D. Definition for it is given in (6.3).

$$\frac{\Delta t_1}{\Delta x_1} = \frac{\Delta t_2}{\Delta x_2} \tag{6.2}$$

$$CFL = \Delta t \left(\frac{|\, u \,|}{\Delta x} + \frac{|\, v \,|}{\Delta y} + \frac{|\, w \,|}{\Delta z} \right) \tag{6.3}$$

As it is shown, same meshes and time steps are used for compressible (cavitating) and incompressible flow cases. On the basis of the chosen time steps and wish to have similar CFL_{max} between the two flow conditions, the analytical flow expressions for incompressible flow were then assigned factor $C = 0,55$. The starting grid in multi domain case was used for its definition. It can be seen that for other multi domain cases, the CFL_{max} does not equal so well the one in compressible flow case. As the difference in CFL_{max} is not of such big importance, this was not altered. However, the chosen time steps cause bigger difference in the mono domain case. Since more interest was given to the multi domain cases and the difference is too big to be quickly adjusted while keeping these cases so similar regarding CFL_{max}, this was left unchanged.

1.3 Chosen simulation time and VOA criteria

As in other simulations, the ones done for VOA also demand that the simulation time is long enough to capture all temporal effects. Since analytical flow expressions involve temporal oscillations with $g = \pi$, it takes $t = 2\ s$ to simulate one cycle, which is also seen on pressure versus time graphs in figure 6.2. The chosen simulation time is $12,6\ s$. Quite some simulations were run also to $25,2\ s$, but results were the same, ruling out the problem of error accumulation.

For the definition of order of accuracy, it was decided that this will be defined for each variable using spatial and temporal averaged L_2 norm of results error. This error is in MMS easy to define, as analytical results for all variables are known. The order of accuracy of a certain variable is then defined as an overall order of accuracy since it includes temporal and spatial errors at the same time. Where spatial and temporal order of accuracy can be separately defined by separately refining grid and time step [127], the overall order of accuracy is defined with the simultaneous refinement of time step and grid spacings [126]. Here, this is enabled by adjusting the time step for each mesh to keep CFL number approximately equal. The orders of accuracy mentioned for different established incompressible flow projection methods in section 2.2 of chapter 3 are also given as overall order of accuracy for a certain variable, which makes for a practical comparison of original and new algorithm with them. They are not all given in L_2 norm. However, the most important ones, used for comparison here, are given in this norm and come from [80], which is to our knowledge the most thorough review on orders of accuracy of projection methods. From it, we conclude that the goal orders of accuracy we are aiming at are 2^{nd} for velocity and 3/2 for pressure. Regarding the pressure, 3/2 is the generally and conservatively expected order of accuracy, while 2^{nd} order of accuracy is possible.

The chosen L_2 norm of error is used without normalisation. It was at first considered to apply normalisation with analytical values for certain variable, but since these oscillate in time and space, divisions by zero were included. Therefore the averaging by number of points in the domain and performed time steps was the only treatment applied to results. This also makes comparison of results for a certain case possible. Since the order of accuracy is defined by convergence slope, the normalisation is actually not required for VOA. Same approach is also used in [127, 126]. The L_2 norm of error used is therefore defined as given with equations (6.4) and (6.5). Equation (6.4) gives the instantaneous L_2 norm result, where N_{pts} is the number of points in the domain, $\psi_{i,j,k}$ the obtained result for a certain variable in a certain point and $\psi^{ext}_{i,j,k}$ its corresponding exact value. This spatially averaged result is then summed over all time steps and divided by their number N_{ts} in (6.5) to give final value for L_2 norm.

$$L_{2,s} = \sqrt{\frac{\sum_i \sum_j \sum_k \left(\psi_{i,j,k} - \psi^{ext}_{i,j,k}\right)^2}{N_{pts}}} \tag{6.4}$$

$$L_2 = \frac{\sum_t L_{2,s}}{N_{ts}} \tag{6.5}$$

In order to define the order of accuracy for a certain variable, the L_2 results on a certain grid are plotted against the average grid spacing of that grid. Then, the slope of the convergence is obtained in a $log - log$ plot and its exponent gives the order of accuracy. Since same grid spacing is used for x and y directions with the flow examples being essentially 2D, using the grid spacing for x direction suffices for the definition of the order of accuracy for all observed variables (where w velocity component is of no interest). However, a question about the correctness of using average grid spacing on the essentially non uniform grid arises. Moreover, the used grids

do not have constant refinement ratio r between each other. Here, the before given claim that successive grid spacing is not required and that smooth grids are required can be seen as one support to perform VOA on the chosen grids. The other point is that the order of accuracy was not defined from observing the decrease in L_2 norm on two successive grids which would then give a sequence of observed orders of accuracy, as suggested in [127]. As mentioned, it was defined from the exponent of convergence slope, which is also an often used approach and directly gives an average order of accuracy. Moreover, the use of average grid spacing was confirmed to be acceptable through comparison of results for incompressible flow case obtained on uniform and non uniform grids. These reveal that the order of accuracy observed on two different grids, where the L_2 norms are plotted against same values of grid spacing, does not differ much.

The following two chapters discuss obtained VOA results. First, the incompressible flow case results are given, to establish the reference VOA results obtained with the original code. These results are then compared with the new code, to verify that order of accuracy is retained with the new algorithm too. After this, the results with the new algorithm in discussed cavitating flow case are presented. The results were all obtained with the use of 4^{th} order compact finite differences (and 2^{nd} order upwind for variables dependent on α). This was chosen as these schemes have an overall 4^{th} order of accuracy, even on the sub domain limits. The reported orders of accuracy from calculations come from fitting a power function curve to a certain convergence slope with least squares method. Then, the exponent of the curve p_s is obtained, defining the measured order of accuracy. The graphs showing convergence slopes do not include also the fitted curves, only enveloping curves with exponents close to the ones obtained from the fitted curves.

2 Verification of the order of accuracy for incompressible flow case

The VOA with the incompressible flow case discussed in this section has multiple purposes. These can be put into following groups:

- First, the VOA for the original, incompressible flow code gives the reference orders of accuracy for observed variables. These are here u, v velocities and pressure. Two additional goals are included in VOA of this code.
- One additional goal is to establish that VOA, done with using non uniform grids and convergence slopes drawn for average grid spacing Δx, gives correct results. Therefore VOA for original code is done on uniform and non uniform grid. Original code is suitable as only compact finite differences are used for spatial discretization in it. Since 4^{th} order accurate schemes were applied, which have such accuracy overall, the possible mixed order of accuracy effects are eliminated.
- The second additional goal, for which original code is used, is to compare differences between the mono and multi domain calculations and set reference points for them. These refer to both computational performance and VOA results. The first will be addressed in a later section dealing with performance of the code, while the second are addressed in this. It was namely found that in VOA, some differences between the mono and multi domain calculations are always expressed on most refined meshes. Therefore it is appropriate to present a reference case for them using the starting version of the code.
- Finally, VOA with the new code is done to establish that same order of accuracy is reached with new algorithm too. Two versions were used for this, the PIUS and PNCS.

Regarding the mono domain calculations, the new and old algorithms still proceed in equal manner as explained in previous chapters. The only difference is that the multi domain method is not used, meaning only direct solutions based on 3D eigendecomposition are applied. This also results in only one application of mono domain solver to obtain a solution for a certain system of equations, instead of two used in the multi domain cases. The chosen starting grid and consequent refined grids enable all tests to be performed also in a mono domain configuration, which is another supporting point for the domain and mesh choice. That is, the total amount of points in the most refined mesh (1591169 points) proved to not be too much for a mono domain simulation.

2.1 VOA of the original incompressible flow algorithm and code

As the tests regarding the incompressible flow code include two additional goals, this section is split into two parts. First reveals the results on uniform grid. It also includes mono and multi domain results and already reveals the important difference between them. The next part deals with the results obtained from the chosen non uniform grids and also gives the results from mono and multi domain computations.

2.1.1 Uniform grid results

The measured orders of accuracy for all variables correspond with the goal ones (2^{nd} for velocities and 3/2 for pressure). Figure 6.3 shows the obtained L_2 norm values for u and v velocity, while figure 6.4 shows same information for the pressure. The (power function) curves added to the figures have exponents close to the observed orders of accuracy while more explicitly defined orders of accuracy p_s, obtained from fitting curves to the convergence slopes, are given in table 6.4. Since the largest grid spacing is $\Delta x = 0,01\ m$, the logarithmic scale for grid spacings can conveniently start at this value. This holds also for the case of non uniform grids.

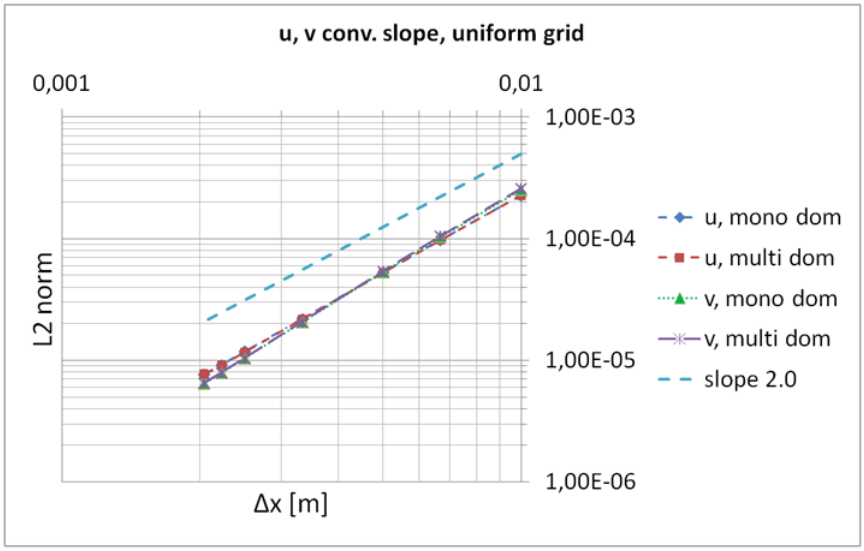


Figure 6.3: Convergence slopes for u and v velocities obtained in mono and multi domain calculations on uniform grids. The *slope* 2.0 line gives a curve with exponent corresponding to the measured order of accuracy of presented variables.

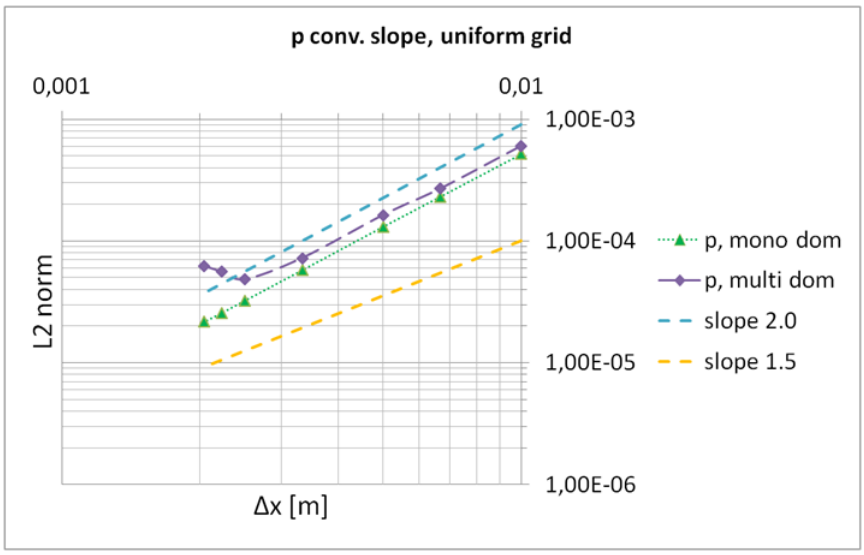


Figure 6.4: Convergence slopes for pressure obtained in mono and multi domain calculations on uniform grids. The *slope* 2.0 and *slope* 1.5 line give curves with exponents corresponding to the measured orders of accuracy.

Table 6.4: Orders of accuracy p_s defined from least squares method fitted power curves to sets of L_2 norms for observed variables.

case\p_s	u	v	p
mono dom	2,14	2,32	2,0
multi dom	2,14	2,33	1,55
multi dom limited	2,14	2,32	1,85

It can be seen that both velocities express same p_s in mono and multi domain calculations. Even their L_2 norms for same meshes coincide almost perfectly. The reported p_s show that the goal order of accuracy is actually surpassed in this case, especially for v. Same cannot be said for pressure, which shows considerable discrepancies between mono and multi domain calculations. Where the mono domain calculations show constant decrease in errors, the multi domain ones show increase for the most refined two meshes. This is the behaviour noted in all cases of multi domain calculations and hence demands a further explanation on what causes it. The reason for it comes from the influence matrix method used as multi domain method. This method and its iterative solution procedure, described in section 4.2 of chapter 3, does not converge in as good a manner for pressure as for velocity in this case. The differences are present in all computed cases, however they get problematic on bigger ones where more sub domains are included. The effects of this will be discussed later in the part considering performance of the code, while the reasons will be revealed here but better referred to in the following section presenting new algorithm results in incompressible flow case. For the purpose of VOA, the multi domain results should take this issue into account. This is done by defining p_s by fitting a power function curve to L_2 norms from all grids, except the two most refined ones. These results are included in table 6.4 in the last row ("multi dom limited" assignment refers to such results). It can be seen that the p_s obtained in such manner do not differ much from the ones obtained in mono domain cases. If only the first four grids are observed, pressure p_s is even closer to the one obtained in mono domain calculations ($p_s = 1,92$), which can also be observed from the slopes in figure 6.4. Nevertheless, choice to not account only for the two most refined meshes in curve fitting was retained.

To conclude this presentation, it can be said that the second order overall accuracy for velocity is confirmed. The mono domain cases show that pressure solution can reach same order of accuracy, surpassing the goal one. The multi domain cases, although troubled for most refined grids, can still reach the goal $p_s = 3/2$ (even with the results from most refined grids included). The results also confirm one other important factor skipped before. This is that all of the calculations in this case give solutions in the asymptotic range of convergence, which is a basic request to perform VOA.

2.1.2 Non uniform grid results

The results on non uniform grids, which are also used in later VOA of new algorithm, show that the convergence slopes are very similar to those obtained on uniform grids. These can be seen on figures 6.5 and 6.6, which, like before, show the convergence slopes and same curves added to them to illustrate the order of accuracy. The more explicitly defined orders of accuracy are given in table 6.5. These show that orders of accuracy are indeed very close, with slightly lower values obtained on non uniform mesh. The highest difference is for v velocity, where $p_s = 2,32$ on uniform mesh should be taken with some reserve. Actually, all results with $p_s > 2.0$ should be considered so, since second order backward differencing in time should limit the overall order

of accuracy for a certain variable to $p_s = 2.0$. Therefore it was confirmed that using average grid spacing values to plot the L_2 norms of results on non uniform grids is acceptable. If anything, the reported order of accuracy is rather lowered, which means that we are on the safe side and do not overestimate it.

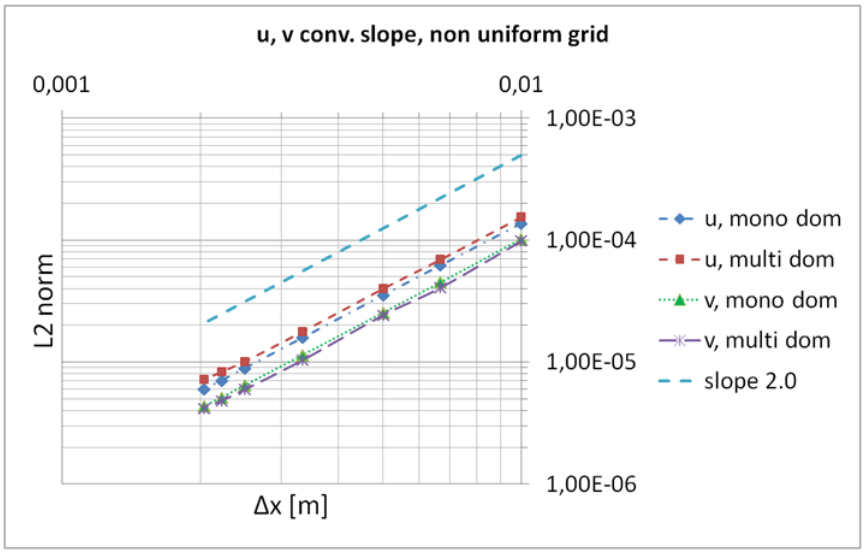


Figure 6.5: Convergence slopes for u and v velocities obtained in mono and multi domain calculations on non uniform grids.

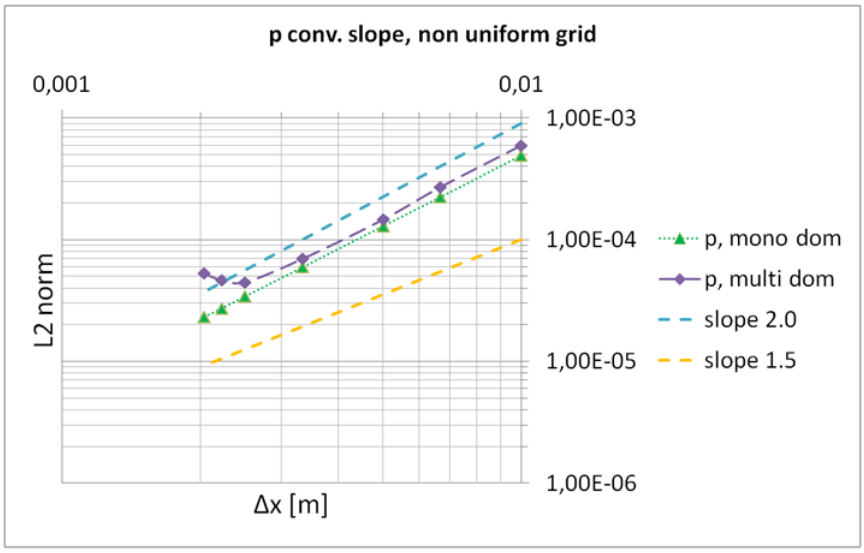


Figure 6.6: Convergence slopes for pressure obtained in mono and multi domain calculations on non uniform grids.

Table 6.5: Orders of accuracy p_s defined from least squares method fitted power curves to sets of L_2 norms for observed variables on non uniform grids.

case\p_s	u	v	p
mono dom	1,98	1,99	1,92
multi dom	1,94	1,99	1,63
multi dom limited	1,96	2,02	1,88

As in the case before, the multi domain results again exhibit the increase in errors when mesh is much refined. Interestingly, the results for u and v velocities on non uniform grids do not coincide with each other any more. Only the results for a certain component in multi and mono domain computations do. Here presented results again confirm before given conclusions about the original code and algorithm meeting (and surpassing in case of p) the goal orders of accuracy. A more graphic presentation of results at a certain instant can also be given. Figures 6.7 and 6.8 show results for u velocity and pressure at $t = 25,2\ s$, respectively. The results come from the starting, coarsest grids in both mono and multi domain cases. Their relative errors are also given. These are obtained with division of difference between calculated and exact results with highest absolute value of exact results. Division with local values of exact results leads to great spikes in spots where values are equal to zero, therefore it is impractical. Such definition of relative errors is used also for all the following results. The shown results demand some explanation as they feature certain characteristics not usually expected. In figure 6.7, the relative error is zero on the domain limits in x direction. This is a consequence of boundary conditions and there imposed equality between final and predictor velocity in normal directions. Since this is for u velocity not imposed on boundaries on y direction limits, the results there exhibit certain degree of error coming from the explicit Φ gradient, as given in equation 3.19. The pressure, using von Neumann boundary conditions, exhibits errors on all boundaries. Both shown errors also feature certain spikes on domain boundaries. Or more precisely, in the corners of whole domain for mono domain calculations and in multi domain cases also on corners of sub domains, placed on domain limits. These come from the fact that in the code, both the mono and multi domain solvers do not solve for the values on the edges of (sub)domains. This is a consequence of using influence matrix as multi domain method and finite differences for discretization. As shown in section on multi domain method 4.2 in chapter 3, the influence matrix defines the interface values on behalf of imposing continuous condition for values and their first derivatives across interfaces. The derivatives are therefore set only in normal directions to interfaces. This, and the need to avoid inclusion of values from other interfaces when a certain one is observed, are the reasons why no edge values are needed on multi domain solver level. The use of finite differences then excludes the need to know the values on edges of sub domains on mono domain solver level. Therefore edge values are at first undefined, leading to appearance of the mentioned spikes in relative errors. However, the spikes in errors should then appear also on other edges, not only those on domain limits. And above all, they should be visible in variable plots. This is not so as the exact values are applied to the edges on domain limits, while inner edges have values defined through interpolation with second order Shepherd's method. This makes for better plots of certain variable, but consequently also causes spikes in error plots. These are much more pronounced on domain limits, since exact values are assigned to edges there, resulting in zero errors. The zero errors are however not seen everywhere, as only every second mesh line is used to make the shown plots. The plots otherwise show that results in these mono and multi domain

calculations are practically equal, as also confirmed by their L_2 norms coinciding.

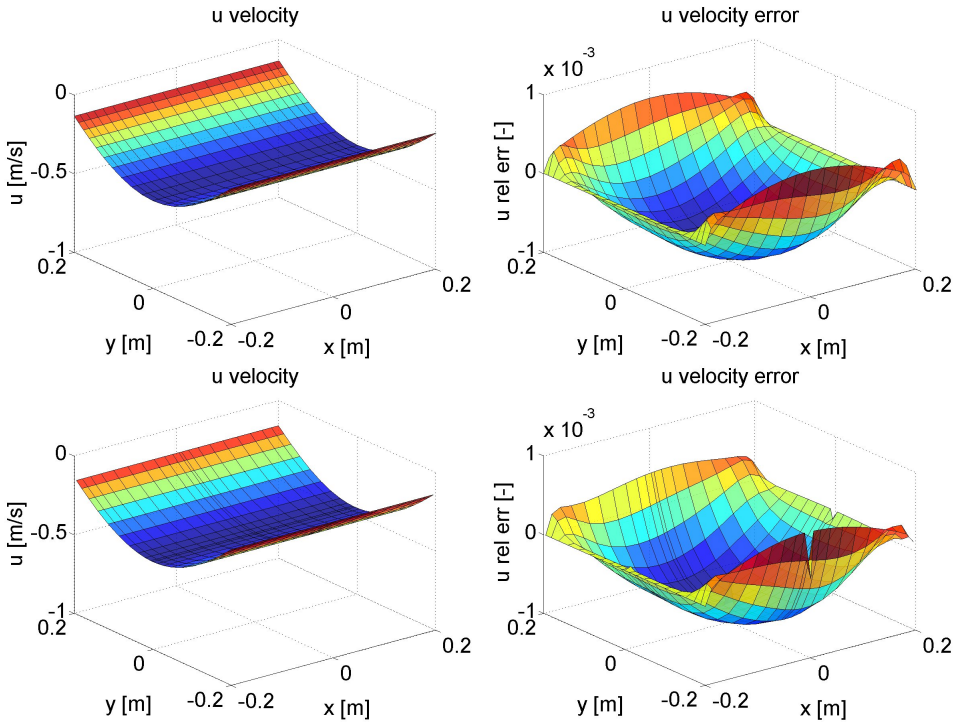


Figure 6.7: Calculated results (left) and relative errors (right) for u velocity in mono (top) and multi (bottom) domain calculation on starting grid at $t = 25,2\ s$.

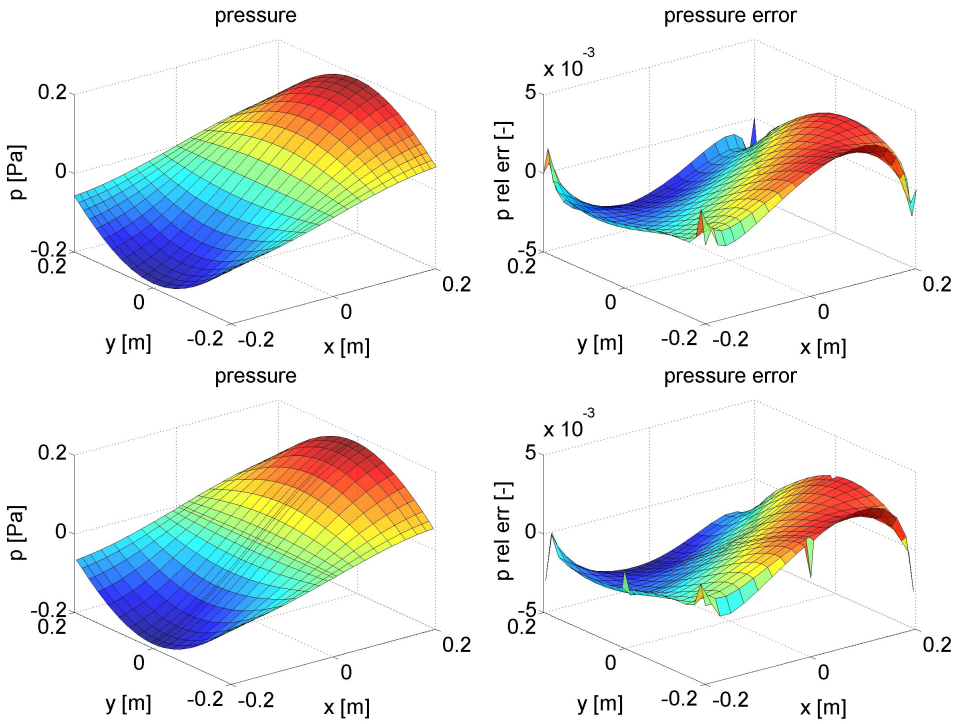


Figure 6.8: Calculated results (left) and relative errors (right) for p in mono (top) and multi (bottom) domain calculation on starting grid at $t = 25,2\ s$.

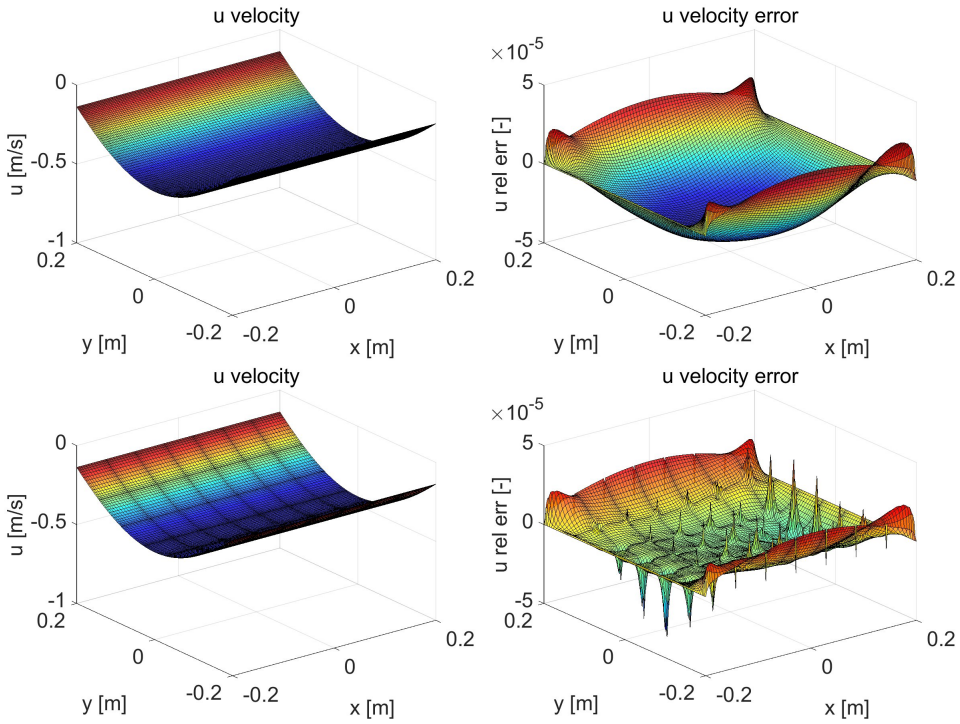


Figure 6.9: Calculated results (left) and relative errors (right) for u velocity in mono (top) and multi (bottom) domain calculation on the most refined grids at $t = 25,2\ s$.

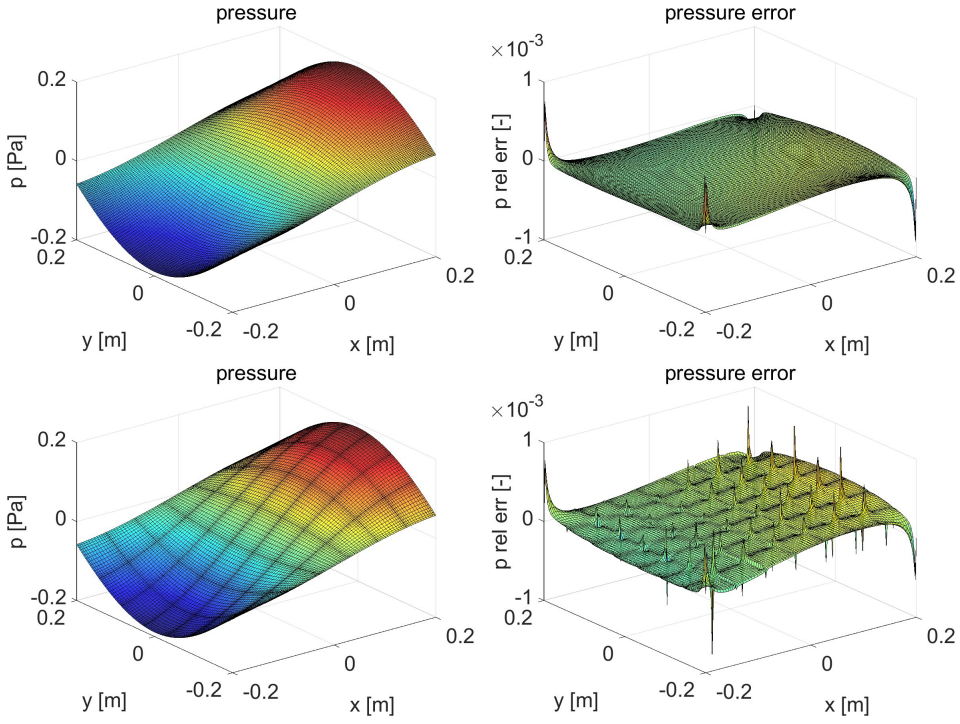


Figure 6.10: Calculated results (left) and relative errors (right) for p in mono (top) and multi (bottom) domain calculation on most refined grid at $t = 25,2\ s$.

The explained reason for the spikes in error plots serves also as an explanation for observed spikes in other error plots given in this work. However, it does not reveal the complete reasons for their appearance. Results on more refined grids have to be consulted to discuss this. Therefore figures 6.9 and 6.10 show same information as the two figures discussed before, but for the case of the most refined grids. It is immediately obvious that although comparison of errors in mono and multi domain computations still shows same order of magnitude and also main shape, there are many spikes included in the multi domain results. These are noticed also on edges of sub domains inside the domain and in fact do not refer only to the edges any more, but to the points in their vicinity as well. The reason for this is not in some error done in the code but in an effect also observed and explained in [80]. There, it is shown that such spikes in pressure, as observed in mono domain solution on figure 6.10, appear with the rotational pressure correction schemes, such as also used in the original algorithm in MFLOPS-3D. These spikes are actually not a drawback but an improvement. As discussed in section 2.2 of chapter 4, where various versions of projection methods are presented, pressure correction schemes which do not use rotational form (do not introduce Φ variable and obtain pressure from it with taking viscous terms into account) suffer from imposition of numerical boundary layer for pressure and thus affect its order of accuracy. As shown in [80], such schemes do not introduce spikes, but errors of higher magnitude on whole domain limits. The rotational form therefore prevents the errors from appearing on whole domain limits, but leads to spikes in and at the corners. These are in [80] referred to also as corner singularities and cause decrease in global convergence rate for the pressure. With them, explanation why goal pressure order of accuracy is set at 3/2 is also given. The article also shows the before mentioned effect that spikes appear only in domains with sharp corners, while in smooth domains, such as domains using periodic boundary conditions or circular domains, the spikes disappear. In them, the second order accuracy for pressure can be expected, while in domains such as ours, the 3/2 is the realistic goal. To conclude with the explanation of spikes, it should be stressed that they appear if boundary conditions for Φ do not impose some numerical boundary layer. This is also ensured on each interface with the multi domain method, therefore the spikes appear also on and at the edges of sub domains, especially since no overlapping of domains is applied. Secondly, this explanation can be combined with the one given before, referring to the spikes which appear strictly on edges. The second order interpolation applied to edges inside the domain cannot get rid of these spikes on refined meshes any more, since the values it uses all come from affected vicinity of an edge. The exact values imposed on the edges on domain limits however can only return zero error values. And finally, the presence of spikes in pressure errors on edges of sub domains causes also appearance of them in velocity results. The velocity and pressure are namely connected through projection.

Going back to the results on fine grid and comparing them with those on the coarse, it is clear that the magnitude of errors on the fine grid drops much more for velocity. For pressure, the highest errors caused by the spikes are only slightly lower, but on the other hand, the complete error field levels out. The spikes and their increased number in multi domain computations could well be the reason why the errors on the two most refined grids increase. However, although straightforward, this is not taken to be the proper explanation. Namely, in each sub domain, the spikes present only a part of the domain. With increased number of sub domains, the ratio between amount of spikes and sub domains remains equal (domains do not overlap, each sub domain edge has its own spike). The possibility of less points per sub domain (as is the case for two most refined grids) making the spike wider also seems not to play an important role. The first four grids all have alternating amount of points (21 or 31 per x, y) and they express monotonic convergence. Moreover, spikes also appear for velocities but reported errors drop monotonically.

2.2 VOA of the new algorithm and code in case of incompressible flow

Since the new algorithm was developed from the same basis as the one for incompressible flow, it is expected that the order of accuracy in incompressible flow case does not differ much from before shown results. Only PIUS and PNCS versions of the new algorithm were tested, using the multi domain computations. There is no need to test all versions of the algorithm, since only one outer iteration is needed in incompressible flow case and the source term is zero. This eliminates the differences between the two versions with updated and constant source term basis. Moreover, σ_p constant is also set as zero, since source term equals zero. CG iterations for Φ are still performed, according to description of how algorithm works. Zero σ_p value is used since in such cases the otherwise applied σ_p value causes a lot of CG iterations as there is no basis for its use. Predictor velocities $\vec{v}^*$ on the other hand effectively do not use CG method, although no change was applied. The reason is that the constant value there comes from the ratio of liquid viscosities and densities, which is the only ratio possible in this case. Therefore the CG method terms on the RHS of equations for $\vec{v}^*$ cancel.

The VOA results revealed interesting characteristics of the new algorithm when it comes to incompressible flow simulations. Same analytical flow expressions were used as for the incompressible flow code. Since the material properties have to be defined, they were chosen to be the same as those used for later presented cavitating flow results. That is $\rho_l = 10kg/m^3$, $\rho_v = 3kg/m^3$, $\mu_l = 0,05Pas$, $\mu_v = 0,01Pas$. Non uniform mesh is used, as the calculations include also the α transport equation, which demands use of upwind discretization scheme. The results for u and v velocities' orders of accuracy for both used algorithm versions are given on figure 6.11, while results for pressure are shown on figure 6.12. The results deserve some discussion from which important conclusions for the following cavitation cases follow.

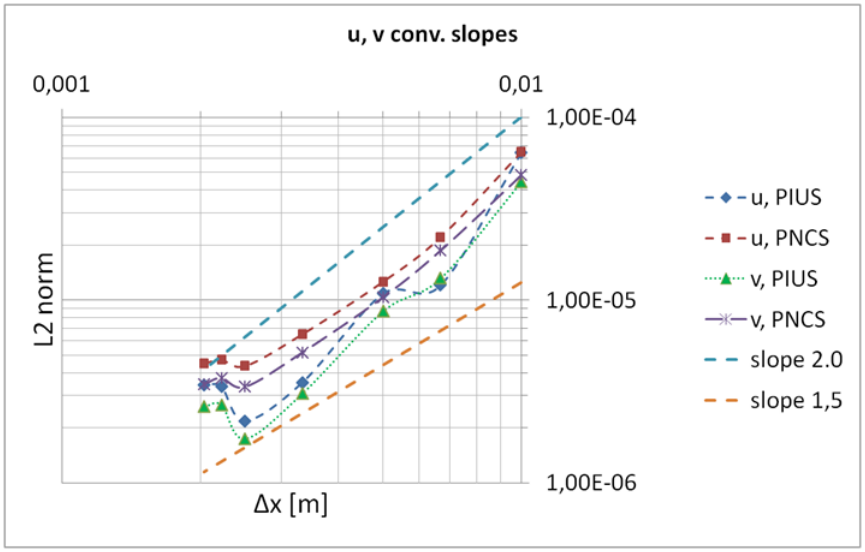


Figure 6.11: Convergence slopes for u and v velocities obtained in multi domain calculations with PIUS and PNCS versions of the new algorithm for the incompressible flow case.

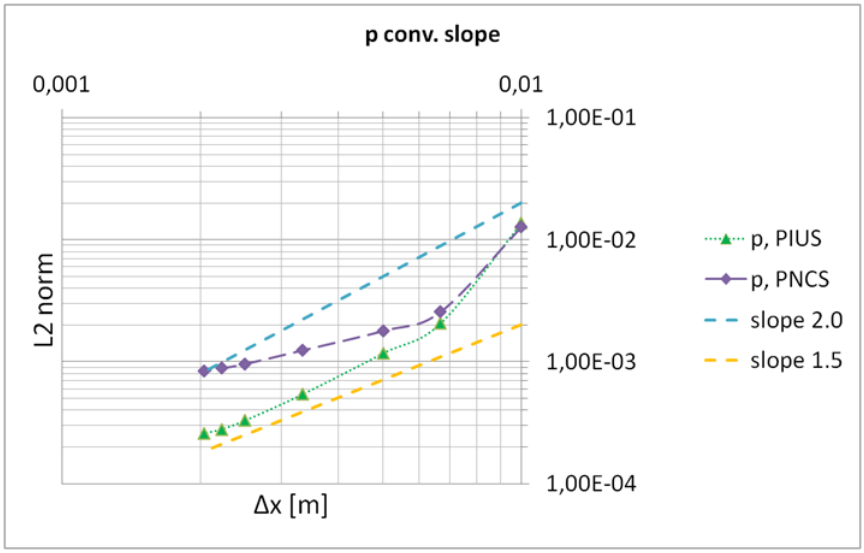


Figure 6.12: Convergence slopes for pressure obtained in multi domain calculations with PIUS and PNCS versions of the new algorithm for the incompressible flow case.

Firstly, both pressure results reveal large drop from starting to second used grid. This is a consequence of results not being in asymptotic range of convergence in the case of starting grid. On it, results also express considerable errors in certain regions, which range even up to 50 %. Since it was also observed that for cavitating cases the starting grid does not return stable simulations in multi domain computations, while in mono domain ones results exhibit similar errors, it was decided that the results from starting grid cannot be used in VOA for the new algorithm and code. The reason for such under performance on this grid is not in an error in the code but in the boundary conditions applied for pressure. These have to be set on one side as Dirichlet boundary conditions, which raises errors in both computation cases presented here and, in fact, all versions of the new algorithm. A consequence of these is also the increase in L_2 norm values for pressure. Contrary to these L_2 norm values, those for velocities become smaller on same grids compared to the original code. Moreover, it can be observed that increase in errors here does not happen on most refined grids for pressure L_2 norms but for velocities. The value of L_2 norm could therefore be the key why the increase in errors happens for most refined grids. In could be possible that the iterative solutions for influence matrix have convergence criteria set too low and at certain level do not permit L_2 norms to drop further with grid refinement. As written, pressure L_2 norms here are higher than those for original incompressible flow algorithm and do not show an increase for most refined grids, while the opposite happens for velocity. The convergence criteria for influence matrix solution was indeed found to be the reason for increased norms or errors on most refined grids, although not for velocities here, but pressure in previously presented results for original code. The explanation for this before shown phenomena is therefore included here, because it was the increase in velocity L_2 norms on most refined grids with new algorithm which prompted some trials with increased convergence criteria for influence matrix solutions. As mentioned in section about the multi domain method 4.2 in chapter 3, the Krylov solver convergence criteria for velocities and Φ influence matrices was set as 10^{-8} and 10^{-5}, respectively. The velocity convergence criteria is not believed to affect much here presented velocity L_2 norms on most refined grids. However, when Φ influence matrix solution convergence criteria was set to equal criteria for velocity, test cases with the original algorithm revealed that L_2 norms on two most refined grids became smaller. Nevertheless, the most refined grid still

returned higher L_2 norm for Φ than penultimate grid. A monotonic drop in errors is therefore still not obtained. Importantly, the L_2 norms for Φ are for that algorithm not as small as those for velocities, and even with the increased convergence criteria they still express issues on most refined grids. Which led to discovery of the actual reason why increase in L_2 norms for Φ happens and why it cannot be removed with current methods in MFLOPS-3D. The solution of Φ influence matrix was namely found to express much worse convergence than the one for velocities. Up to four times more iterations to obtain Φ interface values were demanded when convergence criteria was increased, which is a high price for a partial improvement. The amount of iterations was found unacceptable, explanation why is shown in later part about performance analysis. Here, it should only be stated that in order to keep reasonable performance of the code, the usual convergence criteria for Φ influence matrix was retained. Especially since even increased criteria with much increased amount of iterations did not prevent increase in L_2 norms on the most refined grid. Nevertheless, this explanation still does not give the reason why velocity L_2 norms increase in here presented cases with new algorithm. For this, a discussion about pressure errors is needed first. Furthermore, this increase in errors on two most refined grids, although different from the one with original algorithm, led to the decision to again not take these two grids into definition of order of accuracy. Therefore the results considering VOA with new algorithm come from only four grids in case of multi domain computations and all grids except the starting one in case of mono domain computations. The results for VOA following from four grids in this case are given in table 6.6.

Table 6.6: Orders of accuracy p_s defined from curve fitting to L_2 norms of observed variables for PIUS and PNCS algorithm versions in incompressible flow case.

case\p_s	u	v	p
PIUS	1,9	2,14	1,87
PNCS	1,65	1,75	1,0

Results in table 6.6 show that only pressure incremental algorithm follows the performance of original incompressible flow algorithm. Pressure non incremental algorithm on the other hand returns only first order accuracy for pressure. This is at first a surprise. However, the before mentioned Dirichlet conditions on one side have to be considered. Although the algorithm solves for Φ, which includes in itself also viscous terms, only pressure values can be used for these boundary conditions. We tried to implement boundary conditions for Φ as well, but this made simulations unstable. Therefore pressure values, although necessary, introduce an error on the boundary. This error would not present much concern in real flow cases, except the need to make outflow zone (pressure imposed on the outlet) longer. Here, the effect is more profound. To illustrate it, pressure results and their relative errors are given on figure 6.13 for both used versions of the algorithm. Contrary to figures showing the results before, these feature data from all mesh lines since this is important for the following explanation. The mesh in the example has 121 points in x and y directions. It is clear that errors in both cases are higher than in the case of incompressible flow code. Knowing that the Dirichlet boundary conditions for Φ are imposed on $x = 0,2\ m$ boundary, it is also well shown how much bigger the errors in PNUS version are on that boundary. Consequently, the pressure errors are generally higher overall and pressure non incremental versions suffer from a decrease in order of accuracy. This affects also the velocity order of accuracy, causing it to decrease generally, as can be observed on figure 6.11 and in table 6.6. The results for pressure incremental version also show interesting features. Since this uses connection $\Phi = dp$, there is no error on the boundary with Dirichlet conditions. But

there is a notable jump in error right at it. Moreover, the two algorithms reveal also difference in spikes on domain limits. The spikes on edges inside the domain are not obvious but generally still present in both algorithms, the reason for them has been given before. Where pressure non incremental algorithm has spikes on domain limits, the pressure incremental has a constantly higher error present on these limits in general. This is an expected feature since the pressure incremental algorithm, contrary to the other one, neglects viscous terms in $\Phi - p$ connection and therefore corresponds to standard form of pressure correction schemes. These are in [80] shown to suffer from exactly such pressure errors appearing on the domain limits as a numerical boundary layer is imposed on them. These errors and the error jump at the limit with Dirichlet conditions are also an explanation for oscillations which can be seen in the relative errors plot for pressure incremental algorithm. The increased errors on and close to boundaries namely mean that the pressure gradient in these locations is affected. As pressure gradient is in pressure incremental algorithm used in $\vec{v}^*$ solution, it leads to build up and spread of oscillations. These were found to have greater presence on coarser grids and in multi domain cases. Pressure non incremental algorithm does not express such oscillations.

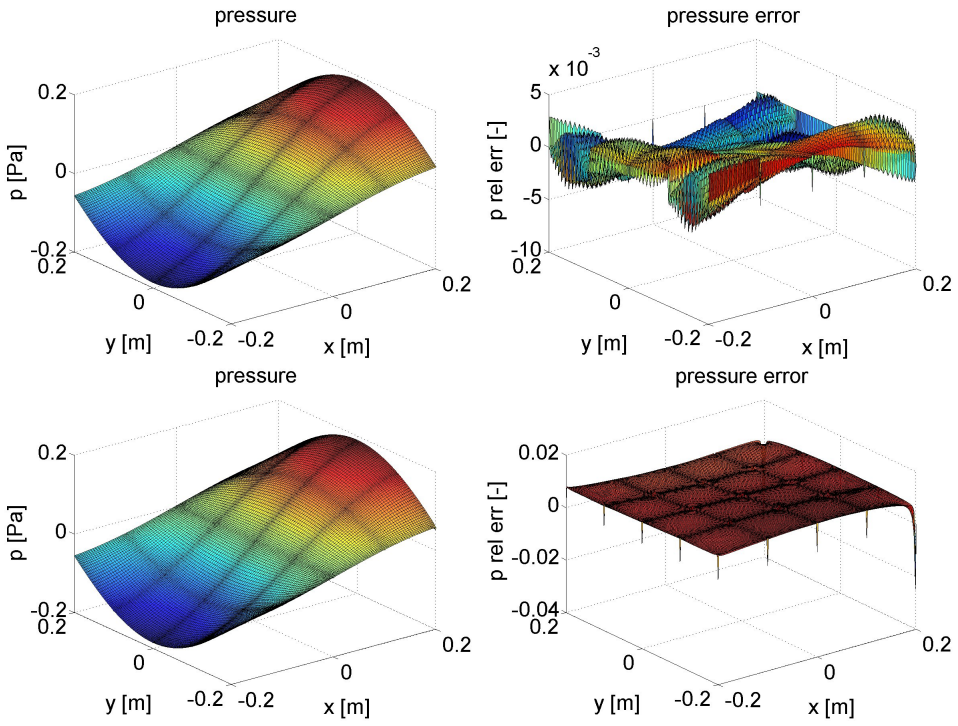


Figure 6.13: Calculated pressure (left) and its relative errors (right) for PIUS (top) and PNCS (bottom) versions of the algorithm on a medium grid at $t = 25,2\ s$.

It can be concluded that although no change in order of accuracy was expected for both algorithms, the pressure non incremental one in MMS tests suffers considerably from the use of pressure values for the needed Dirichlet boundary conditions. Therefore its order of accuracy drops to $3/2 - 2$ for velocities and 1 for pressure. Pressure incremental algorithm on the other hand shows smaller impact and seems to retain 2^{nd} order of accuracy for all variables. Indeed, Dirichlet boundary conditions for Φ are here completely correct and could help at retaining the order of accuracy despite the imposed numerical boundary layer. But all boundary conditions for Φ in this algorithm version introduce some issues with oscillations, which can be seen only in error plots. Finally, it can also be said that the increased errors for pressure in non incremental

algorithm and errors on whole boundaries accompanied by oscillations in pressure incremental one are the reasons for increased L_2 velocity norms on two most refined grids. Although the pressure errors could still be decreasing with grid refinement, as shown on figure 6.12, this might not be sufficient to enable also decrease in velocity errors on two most refined grids.

2.3 Conclusions from VOA with incompressible flow

The incompressible flow tests set the reference orders of accuracy and also guidelines for the following cavitating flow tests. General conclusions obtained from them can be listed as:

- It was confirmed the average grid spacing Δx can be used for definition of overall orders of accuracy also on non uniform grids.
- Mono and multi domain results correspond to each other, except on two most refined grids. There, multi domain calculations show increase in errors, which follows from different causes.
- In case of incompressible flow algorithm, the increase in Φ errors is caused by lower convergence, and with it also lower convergence criteria, of Krylov solver for Φ influence matrix. Increased criteria does not completely remove the observed issues and because of lower convergence also greatly increases the amount of influence matrix solution iterations.
- Increase in velocity L_2 norms on most refined grids is expressed in case of the new algorithm. This is a consequence of Φ solution errors, imposed by the mixed boundary conditions or pressure correction scheme.
- As two most refined grids show certain increase in errors of multi domain calculations in all cases, the results from them are omitted when defining the order of accuracy.

Using above given points, the conclusions for reference orders of accuracy can be summed as:

- The original code is well capable of achieving goal orders of accuracy as all variables return 2^{nd} order of accuracy. Indeed pressure surpasses its set goal order of accuracy (3/2).
- The new algorithm in case of pressure non incremental version suffers from use of mixed boundary conditions for Φ. The needed pressure values on boundary with Dirichlet conditions cause considerably lower orders of accuracy. These are the first order for pressure and orders between 3/2 and 2^{nd} for velocity.
- On the contrary, the pressure incremental version of the new algorithm returns better and expected results. They are on the same level as for original code, that is 2^{nd} order of accuracy for all variables. This is achieved although the use of $\Phi = dp$ equality makes the algorithm correspond to standard form of pressure correction schemes, for which a lower order of accuracy for pressure would be expected. Argument is that exact Dirichlet conditions for Φ on one boundary could help retain the order of accuracy.
- However, the pressure incremental algorithm shows appearance of oscillations because of increased errors on the boundaries with von Neumann conditions (where a numerical boundary layer is imposed) or next to the boundary with Dirichlet conditions.

As main observed issue, the lower accuracy of non incremental new algorithm version, is a consequence of MMS test case alone (variable pressure present on Dirichlet boundary), it was concluded results are acceptable and can be taken as a reference point for the following cavitating flow cases.

3 Verification of the order of accuracy in cavitating flow

In contrast to previous presentation of results with incompressible flow case, the results for VOA of new algorithm with before defined cavitating flow cases are presented all together in one section. The reason for this is that the results do not show large difference from reported results in incompressible flow case. Moreover, eventual explanation of results for a certain variable demands inclusion of effects caused by another variable, therefore it is better to present all results together. Convergence slopes are here given for a certain variable for all four versions of the algorithm at once. Contrary to incompressible flow VOA, the results from mono and multi domain calculations are given in separate graphs, since inclusion of them all together would make the graphs difficult to examine, although the results well correspond to each other.

3.1 Results and discussion

The convergence slopes for u velocity and added curves showing highest and lowest order of accuracy are given on figure 6.14. Graphs for both multi and mono domain results are given on this figure. Following figures 6.15, 6.16 and 6.17 present same graphs and information for v velocity, pressure and α, respectively. As before, the curves added to the graphs give a good illustration of the obtained order of accuracy for a certain variable, while more exactly defined orders of accuracy are given in tables. Table 6.7 reports the values in multi domain cases. These are obtained from four grids only, as explained in previous section. Table 6.8 gives same information for mono domain cases, with the difference that results from all grids are used, except from the coarsest one. The results from it are also excluded here in general, since they do not fall into asymptotic range of convergence in this case.

Table 6.7: Orders of accuracy p_s defined from curve fitting to L_2 norms of observed variables for all four versions of the new algorithm in multi domain calculations.

case \ p_s	u	v	p	α
PICS	1,62	1,47	1,29	1
PIUS	1,46	1,64	1,6	1
PNCS	0,74	0,8	0,7	0,77
PNUS	0,74	0,8	0,7	0,74

Table 6.8: Orders of accuracy p_s defined from curve fitting to L_2 norms of observed variables for all four versions of the new algorithm in mono domain calculations.

case \ p_s	u	v	p	α
PICS	1,6	1,46	1,26	1
PIUS	1,38	1,55	1,43	1
PNCS	0,73	0,82	0,65	0,78
PNUS	0,73	0,82	0,65	0,75

The graphs and tables show that the mono and multi domain results are almost practically identical for all versions of the new algorithm. Only the PIUS version expresses a notable difference. Its pressure order of accuracy is higher in multi domain cases. The reported results for this version are actually best ones, showing that PIUS equals goal 3/2 order of accuracy for pressure

and same is also found for velocities (taken as average). The PICS version is very close, with interestingly inverse highest accuracies in velocities (with u having highest accuracy instead of v). Pressure with it expresses generally lower order of accuracy, close to $1,3$. This is still very close to the goal value of $3/2$, hence the results for both pressure incremental versions of the algorithm are taken as satisfying ones. Considering that they are obtained with the imposed influence of Φ boundary conditions, presence of complete α range in the domain and most importantly the quick changes in pressure sign and amplitude, discussed in section 1.1, these results can be taken as completely acceptable and show the good convergence of the new algorithm. Bearing all the mentioned effects in mind, the drop in accuracy is not big compared to the incompressible flow.

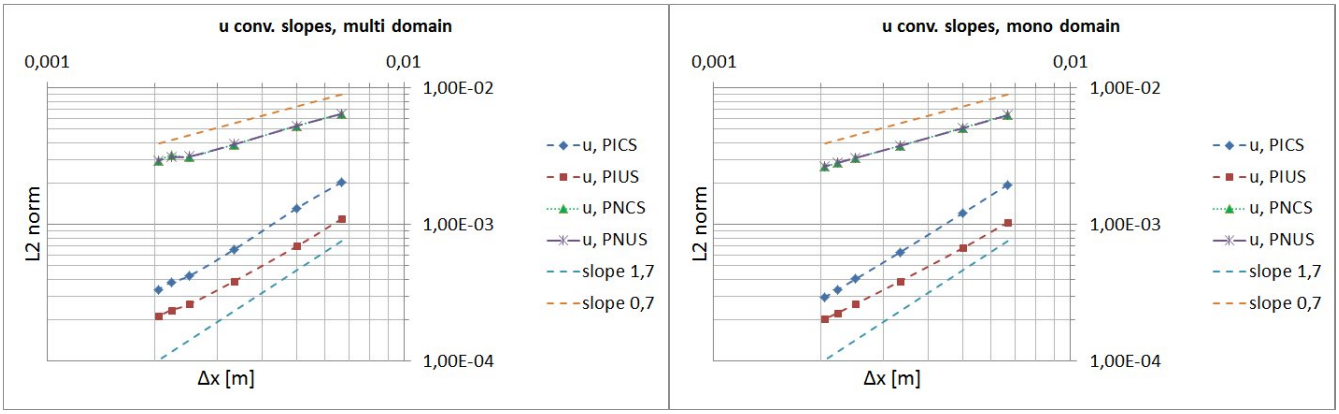


Figure 6.14: Convergence slopes for u velocity obtained in multi (left) and mono (right) domain calculations with all four versions of the new algorithm.

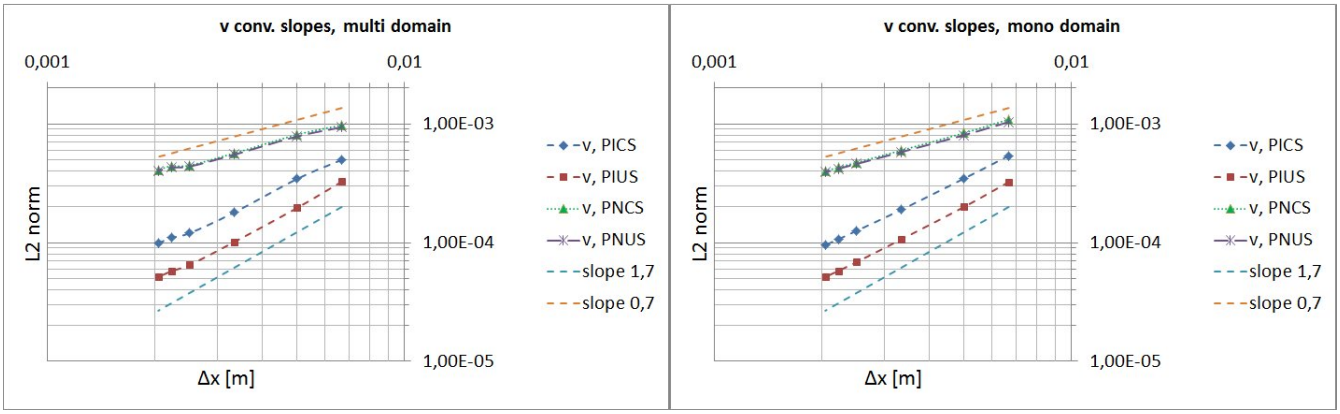


Figure 6.15: Convergence slopes for v velocity obtained in multi (left) and mono (right) domain calculations with all four versions of the new algorithm.

Same cannot be said for the pressure non incremental versions of the algorithm. Following the graphs and tables, both PNCS and PNUS versions interestingly show practically same results in either mono or multi domain cases. They also show the important effect pressure has on the reported orders of accuracy. The accuracy for pressure itself is already lower for cavitating flow cases, which means that the before reported order $p_s = 1$ in incompressible flow case here falls below unity. What is more striking is that same happens with velocities too. In both incremental and non incremental versions, velocities show in general higher order of accuracy than pressure, but this still does not prevent them to drop below unity in the two non incremental versions.

The reason for this, and also for the drop in PICS and PIUS versions, is the high dependence of all variables on accuracy of pressure solution. As shown in the explanation of the new algorithm development, this is caused by the importance the source term S has in such a flow and therefore also its solutions. As the term follows from and affects pressure solution, it defines the projection step and hence sets the accuracy of velocity solutions. Therefore the accuracies for velocities in all versions of the algorithm drop, most notably in pressure non incremental, which is no surprise when the discussed errors imposed on pressure by boundary conditions are combined with the effect the pressure here has on all variables. The situation is therefore for the case of cavitating flow much different from the incompressible one, with accuracy of velocities being set much more strictly by the accuracy of pressure solution. Same effect is seen also for α solution, which reports at most first order accuracy. This is not expected, since the solution features second order accurate temporal and spatial schemes. But the effect that pressure has on it through source term, and also through decreased order of accuracy for velocities, causes α solution to exhibit lower order of accuracy. In case of pressure non incremental versions this order drops below unity, as for velocities. Which only further confirms here discussed relations and effects that pressure exhibits.

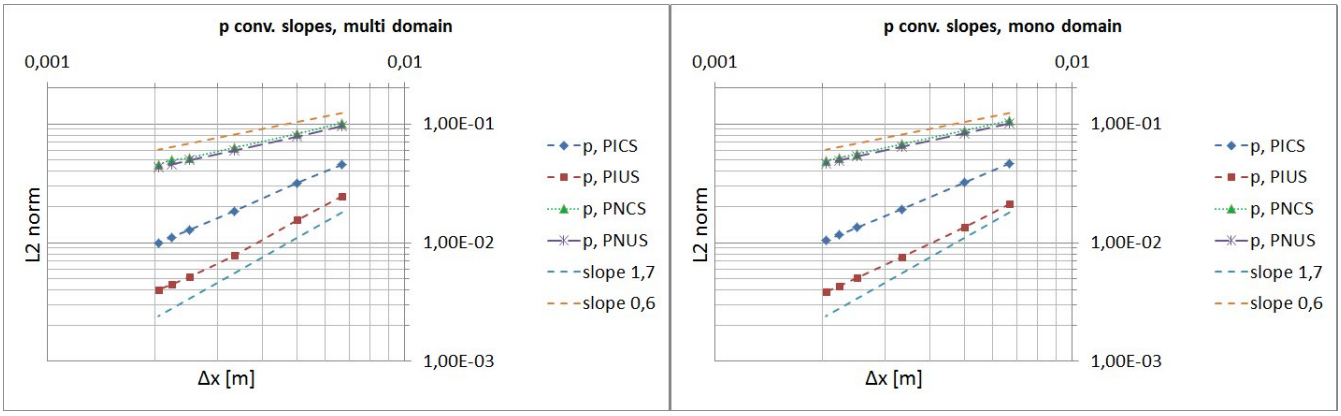


Figure 6.16: Convergence slopes for pressure obtained in multi (left) and mono (right) domain calculations with all four versions of the new algorithm.

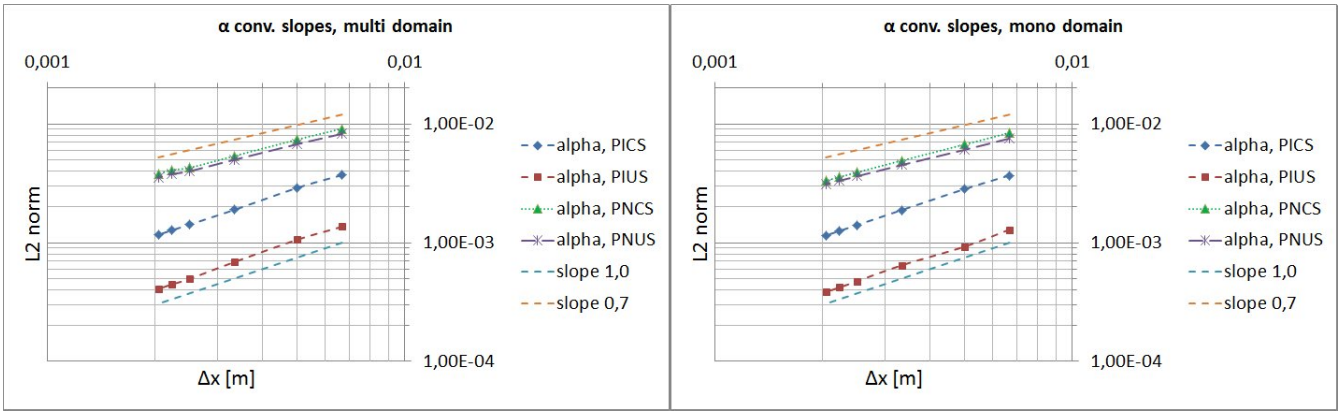


Figure 6.17: Convergence slopes for α obtained in multi (left) and mono (right) domain calculations with all four versions of the new algorithm.

Regarding α solution, the question about its effect on the accuracy of all other variables also exists. As the pressure affects α through S and velocity, α could also affect velocity and pressure

solutions through density and viscosity. This is indeed the case, however the effect of α is rather a small one. This was found with performing calculations with PICS version, where α was not solved but exact values were used instead. This enabled also the use of compact finite differences for all derivatives, that is, including those of variables defined by α. Only v velocity expressed an increase in order of accuracy, from $p_s = 1,47$ to $p_s = 1,69$. Other two variables returned same results. Therefore it can be concluded that α solution, although obtained in a different manner than solutions of other variables, does not considerably affect the overall order of accuracy. It is rather solution of other variables which affect the α solution accuracy.

One other observation can be made from the given graphs with convergence slopes. This is that the increase, although small, in velocity L_2 norms for multi domain calculations is present only for the two pressure non incremental algorithms, while the incremental ones express only slightly decreased convergence slopes. This is not equal to incompressible flow case, but it shows that same behaviour, caused by pressure errors, is still present.

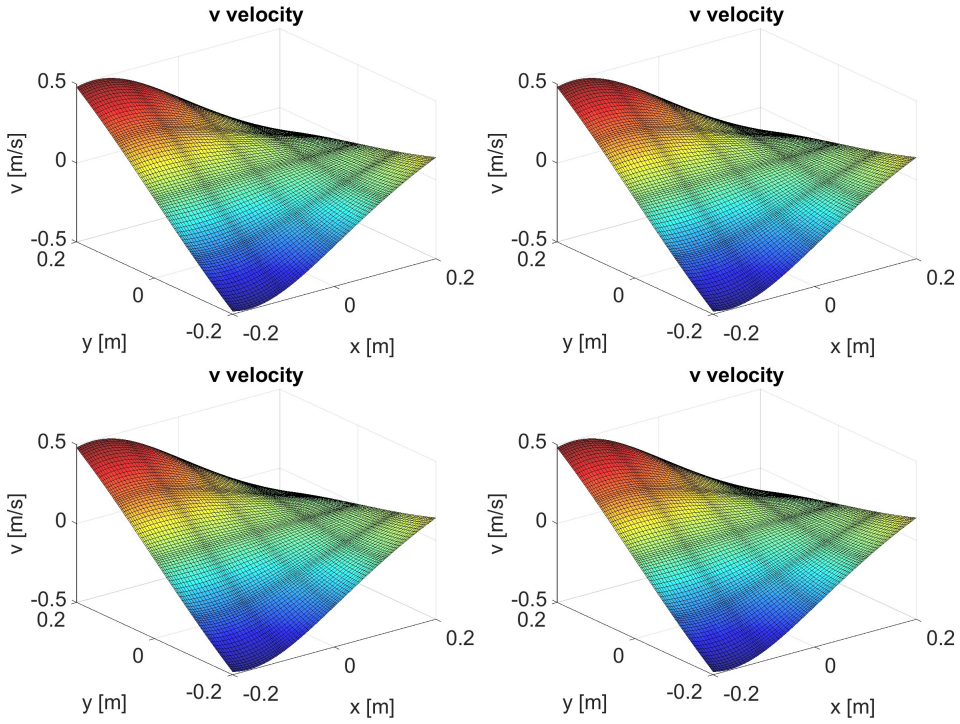


Figure 6.18: Results for v velocity at $t = 25,2\ s$ for PICS (top left), PIUS (top right), PNCS (bottom left) and PNUS (bottom right) version of the algorithm. Results come from multi domain calculations on grid with 81 points in x and y direction. All grid lines are plotted.

Additionally to the shown VOA results, a more graphical presentation of instantaneous errors for certain variables should also be given. In such manner, some already mentioned characteristics of both the test case and the algorithm versions can be better shown also with results from the cavitating flow MMS cases as well. Errors of these results are used since not much can be read directly from results themselves. As in the case of incompressible flow simulations it is namely very hard to differentiate between different calculations just by looking on results or values for a certain variable. An example is given on figures 6.18 and 6.19, which present v and α results from multi domain calculations on the grid with 81 points in x and y directions. Results from all four versions of the algorithm are given at time $t = 25,2\ s$. Both figures show that there are no notable differences between the four versions on such level of observation. Knowing the figures

show multi domain results, which bear more issues than mono domain ones, and observing the monotonic nature of convergence slopes, it can be seen that there is no need to show plots of variables for all versions of the algorithm and mono or multi domain calculations. To present results for certain variable, one plot is sufficient.

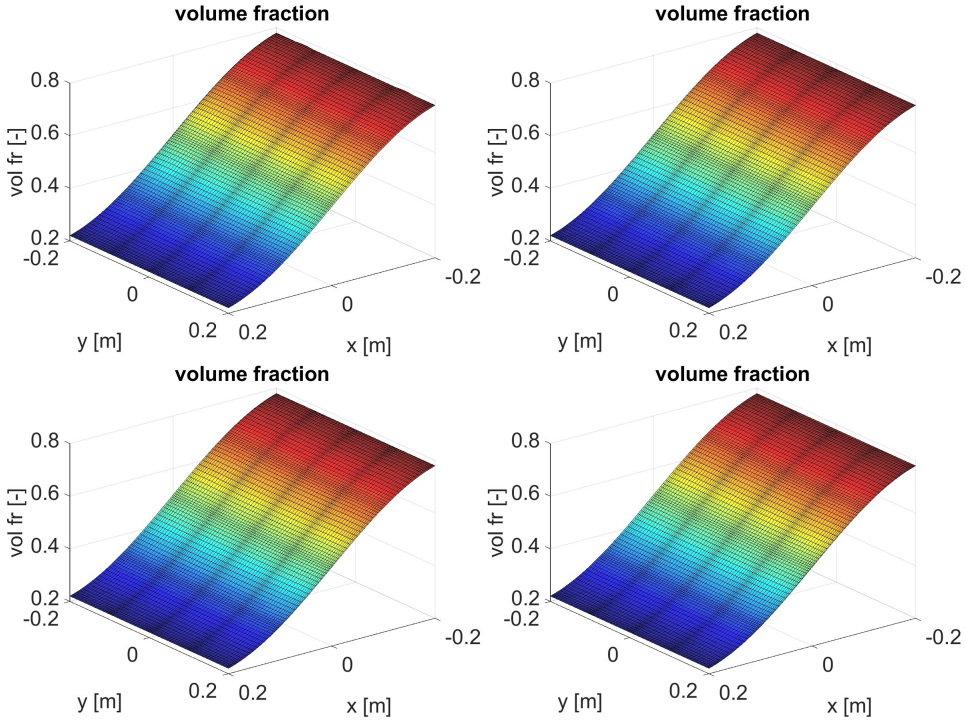


Figure 6.19: Results for α at $t = 25,2\ s$ for PICS (top left), PIUS (top right), PNCS (bottom left) and PNUS (bottom right) version of the algorithm. Results come from multi domain calculations on grid with 81 points in x and y direction. The view is rotated for 180°. All grid lines are plotted.

The error plots on the other hand show more information. These plots are given on figures 6.20–6.25, which show relative errors for u velocity, pressure and α. The plots combine the errors obtained in mono and multi domain calculations, which are presented for cases of grid with 81 points in x and y direction and the most refined grid. The results from coarser grid are given for time $t = 25,2\ s$, while the ones for finest grid are plotted at time $t = 12,6\ s$. These are especially interesting, since they capture errors at sudden pressure change. The pressure results for multi domain calculations with PICS and PNUS versions of the algorithm at this time are given in figure 6.26. Combining results on them with plots on figure 6.1 and graphs on figure 6.2, it is clear that shown errors are plotted for an instant right after the switch in pressure sign. Corresponding to before given explanation, other calculations also return seemingly same results and therefore do not need to be shown. The two given results were chosen to better show that even the pressure non incremental versions, with their higher pressure errors, can well handle sudden pressure changes.

Relative error plots in figures 6.20–6.25, especially those for coarser grid (where no sudden pressure change is present), show that all the characteristics, described for the incompressible flow case already, are here in general only repeated. The pressure incremental versions show presence of numerical boundary layer for pressure, resulting in higher errors on whole boundaries. Same errors as before can be also seen at the boundary with Dirichlet conditions for Φ. A point not stressed

before is that these errors cause velocities to express similar jump in errors at the boundaries as well, since they affect $\nabla\Phi$ term in projection step. Consequently, the oscillations can be seen in them too. From figures 6.20 and 6.22 it can be seen that the multi domain cases express higher oscillations than mono domain, which corresponds to observations in incompressible flow case. Oscillations become smaller also with the refinement of the grid. This is here not proved since the plots of errors on finest grids are given only for each second line. However, since the oscillations become much smaller on the finest grid and they do not affect the results, it was opted to skip their illustration. Regarding their presence and presence of before discussed jumps in pressure and velocity errors in pressure incremental versions, the non incremental results again show a different picture. No error jumps or oscillations are noted, only before explained spikes. This confirms that pressure non incremental versions, with the redefined $\Phi - p$ connection for compressible flow, retain the ability to avoid creation of numerical boundary layer for pressure. Nevertheless, higher overall pressure errors are obvious and still a consequence of the Dirichlet condition for Φ. Interestingly, both non incremental versions show results for pressure and velocity which are very similar or nearly identical regardless of mono or multi domain calculations or the version used. This is additional illustration for observations from graphs with convergence slopes and tables with the defined orders of accuracy.

The description of plotted relative errors in previous paragraph is focused on characteristics which were observed already in incompressible flow case. However, the plots in figures 6.20–6.25 include other features as well. To begin, all α relative errors show presence of two peaks in vicinity of $x = 0,2\ m$ coordinate. These peaks are more pronounced on coarse grid (figure 6.24). The reason for them is unknown, since no other variable expresses similar error while the two peaks can be observed with all versions of the algorithm. As α uses Dirichlet boundary conditions, its errors are otherwise zero on the boundaries, hence the large jumps in their vicinity. Interestingly, although α errors show good correspondence between mono and multi domain results, they can differ significantly between PICS and PIUS or PNCS and PNUS versions. The lowest errors, which are always obtained for the versions with updated source term, show that α errors depend considerably on the treatment of source terms during outer iterations. This is not observed so clearly for any other variable.

The other very distinctive feature of the given plots refers to the results at time $t = 12,6\ s$. As mentioned, these include the effect of sudden pressure change and are only given for case of the most refined grid. The pressure incremental algorithm versions show considerable errors for pressure on the boundary where the sudden change happens. These errors express same magnitude, regardless of the version or type of calculation. Interestingly, pressure non incremental versions show no such errors, meaning that the used $\Phi - p$ connection is effective even in this case. On the other hand, this confirms that the large errors observed in such conditions in incremental versions are a consequence of the numerical boundary layer. Still, the effect of this numerical layer is limited only to the boundary. Figure 6.27 shows that pressure errors, excluding the problematic boundary, are smaller than on the coarse grid (figure 6.22), where no such pressure change is taking place. Furthermore, figure shows the numerical boundary layer on y boundaries is more pronounced as well and increases in magnitude towards the problematic boundary. Finally, these pressure errors also explain higher u velocity errors expressed for PICS and PIUS results on fine grid, shown on figure 6.21. Interestingly, although these errors are expressed as a higher jump at $x = -0,2\ m$ boundary, there are no higher oscillations caused by them. The pressure non incremental versions on the other hand return results which do not express such errors in u, since the mentioned numerical boundary layer effect does not occur. Nevertheless, the impact of higher errors caused by Dirichlet conditions for Φ in them is very clear in all conditions.

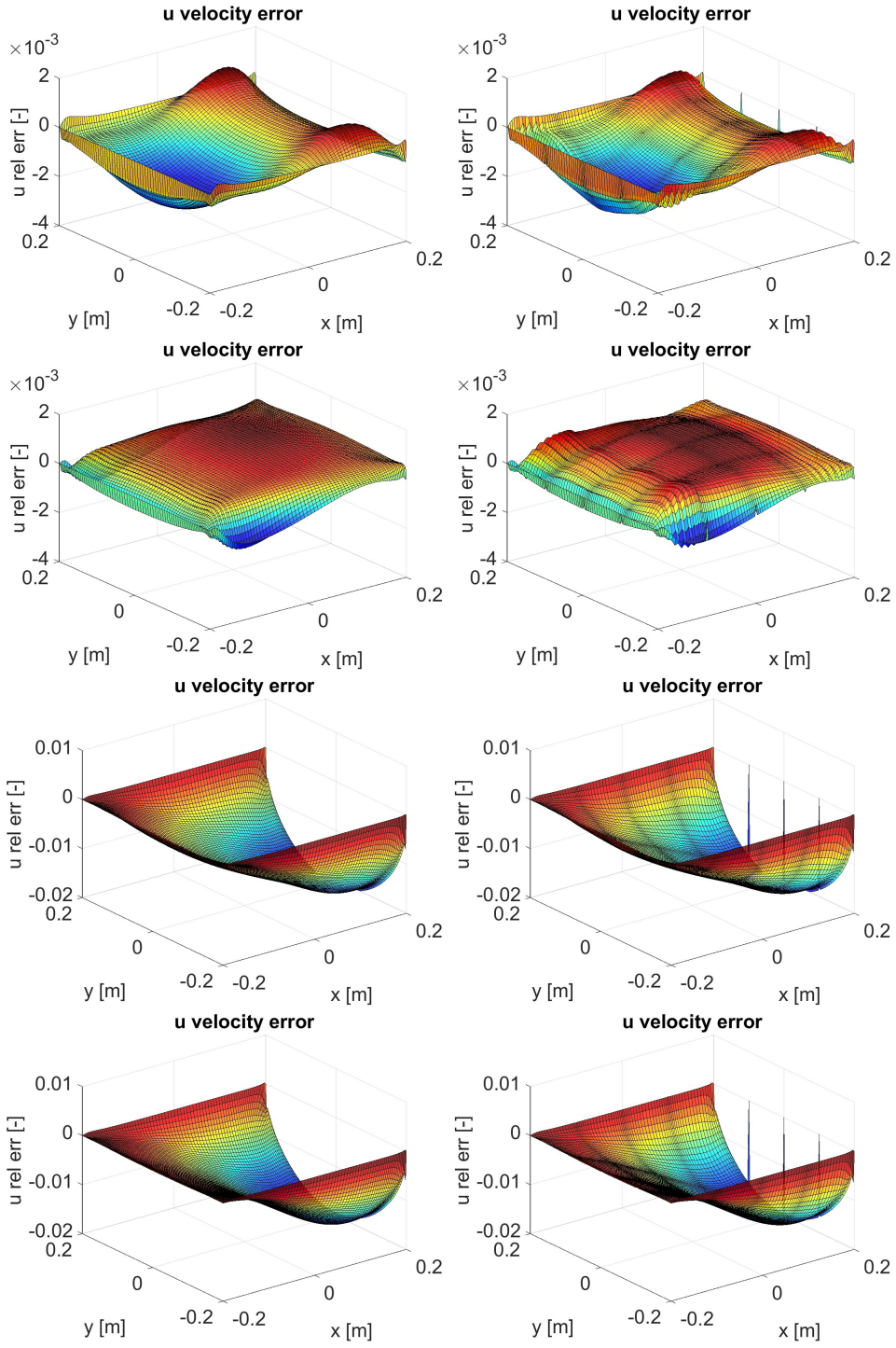


Figure 6.20: u velocity relative errors at $t = 25,2\ s$ on grid with 81 points in x nad y directions. Left column includes mono domain results, while right column shows multi domain ones. Top line gives results with PICS, second with PIUS, third with PNCS and bottom with PNUS version of the algorithm. All grid lines are plotted.

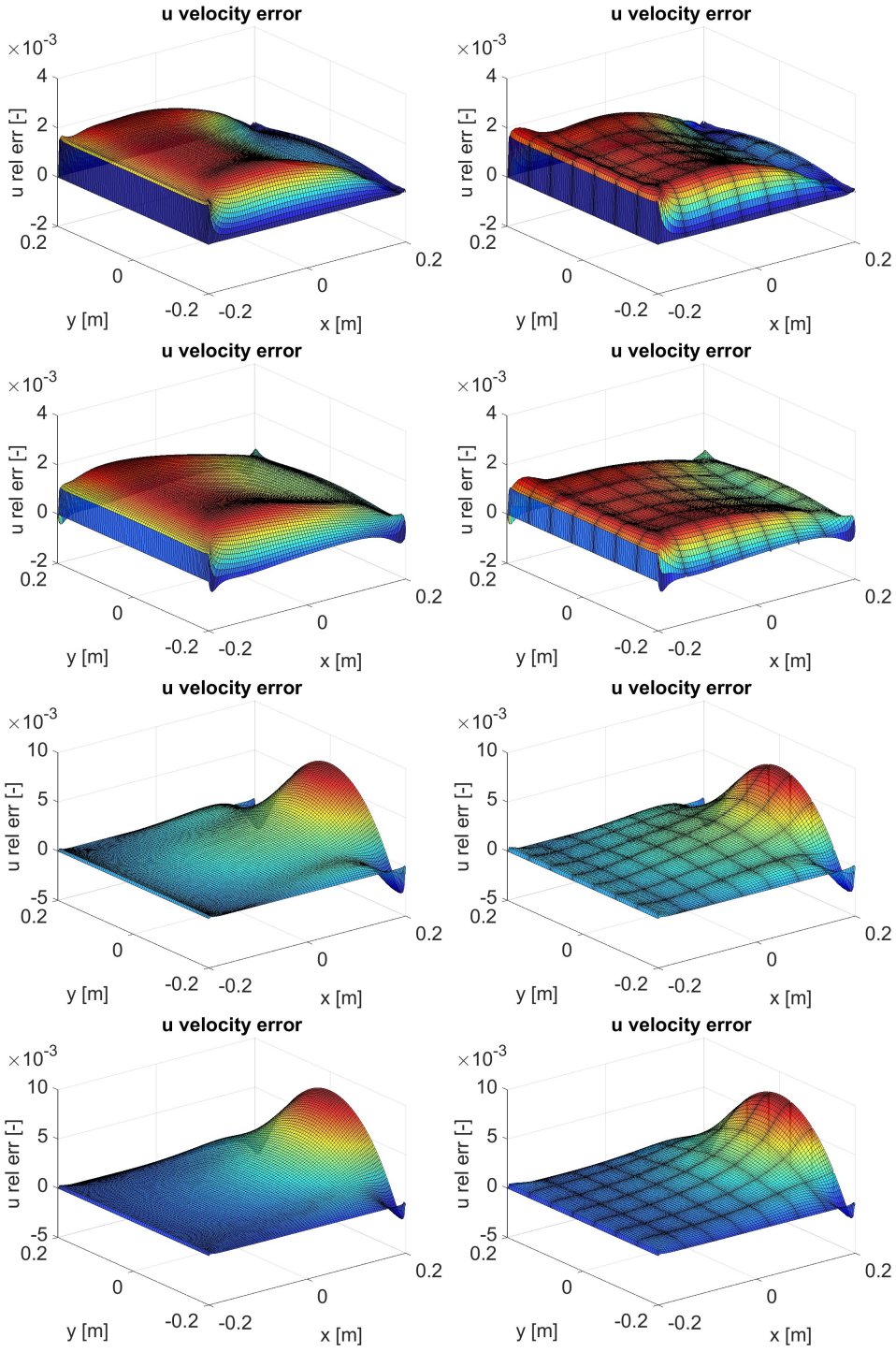


Figure 6.21: u velocity relative errors at $t = 12,6\ s$ on finest grid. Left column includes mono domain results and the right one the multi domain results. Top line gives results with PICS, second line with PIUS, third with PNCS and bottom with PNUS version of the algorithm. Every second grid line is plotted.

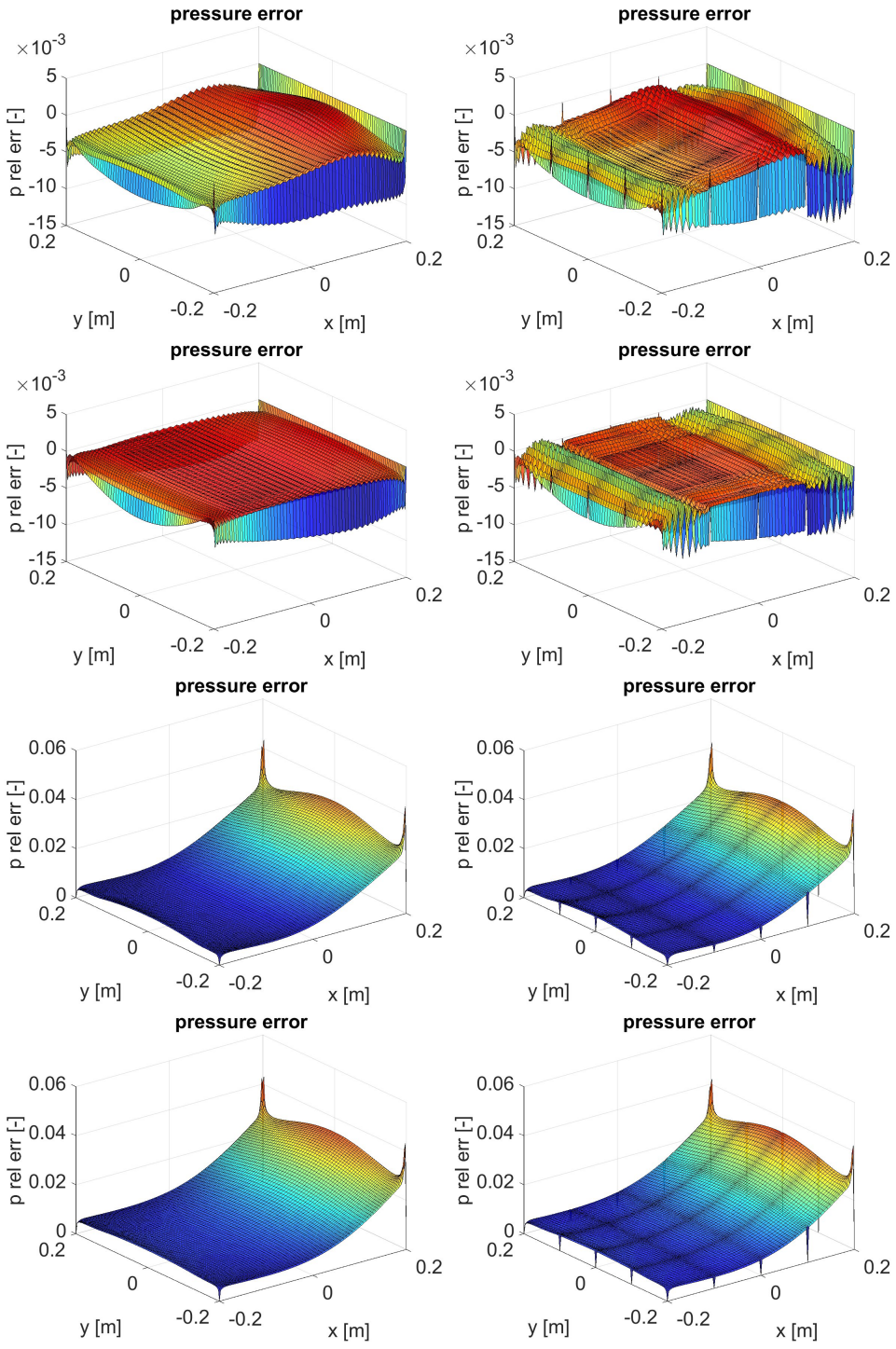


Figure 6.22: Pressure relative errors at $t = 25,2\ s$ on grid with 81 points in x nad y directions. Results are given in same manner as on previous figure 6.20.

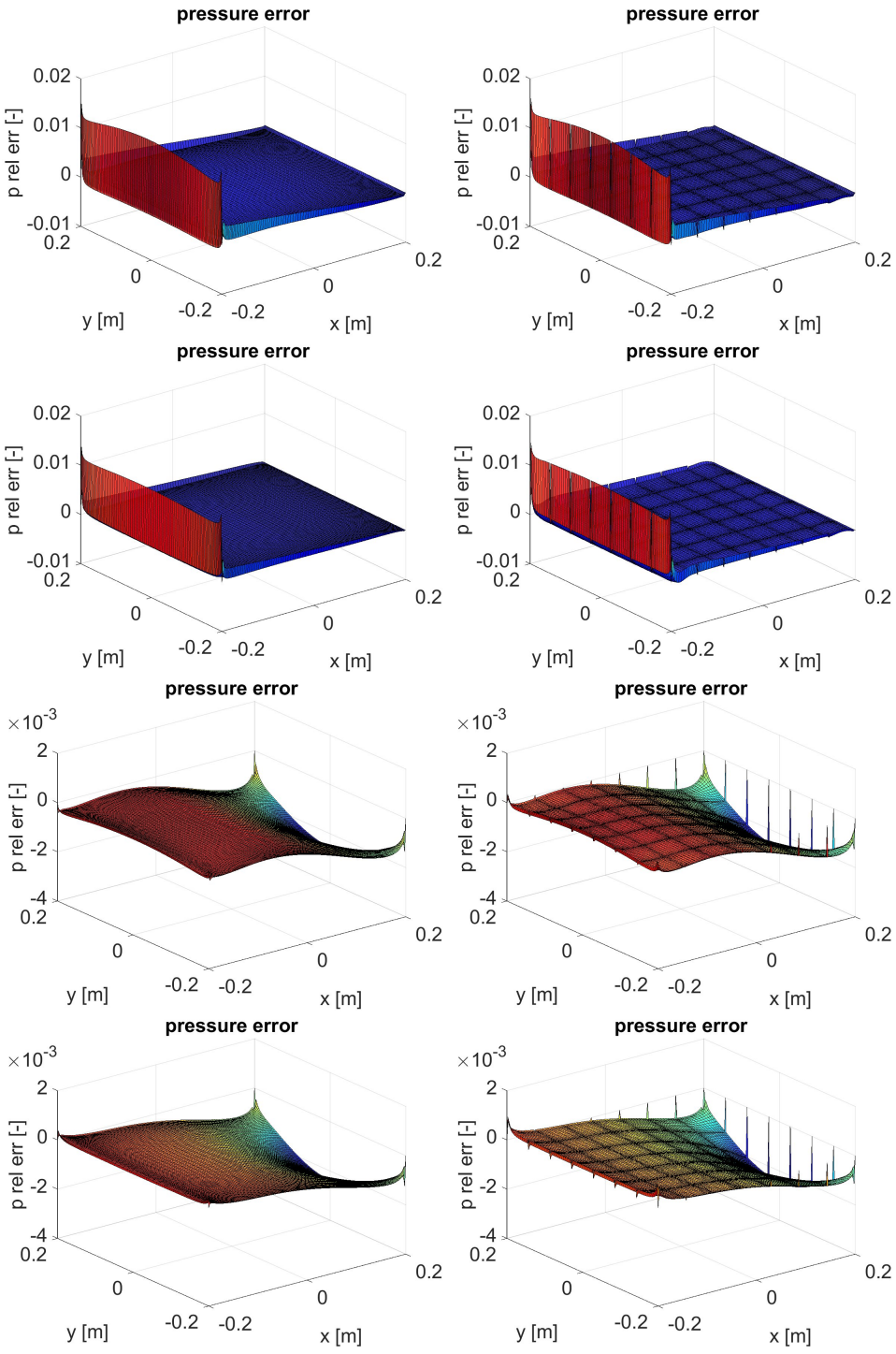


Figure 6.23: Pressure relative errors at $t = 12,6\ s$ on finest grid. Results are given in same manner as on figure 6.20. Every second grid line is plotted.

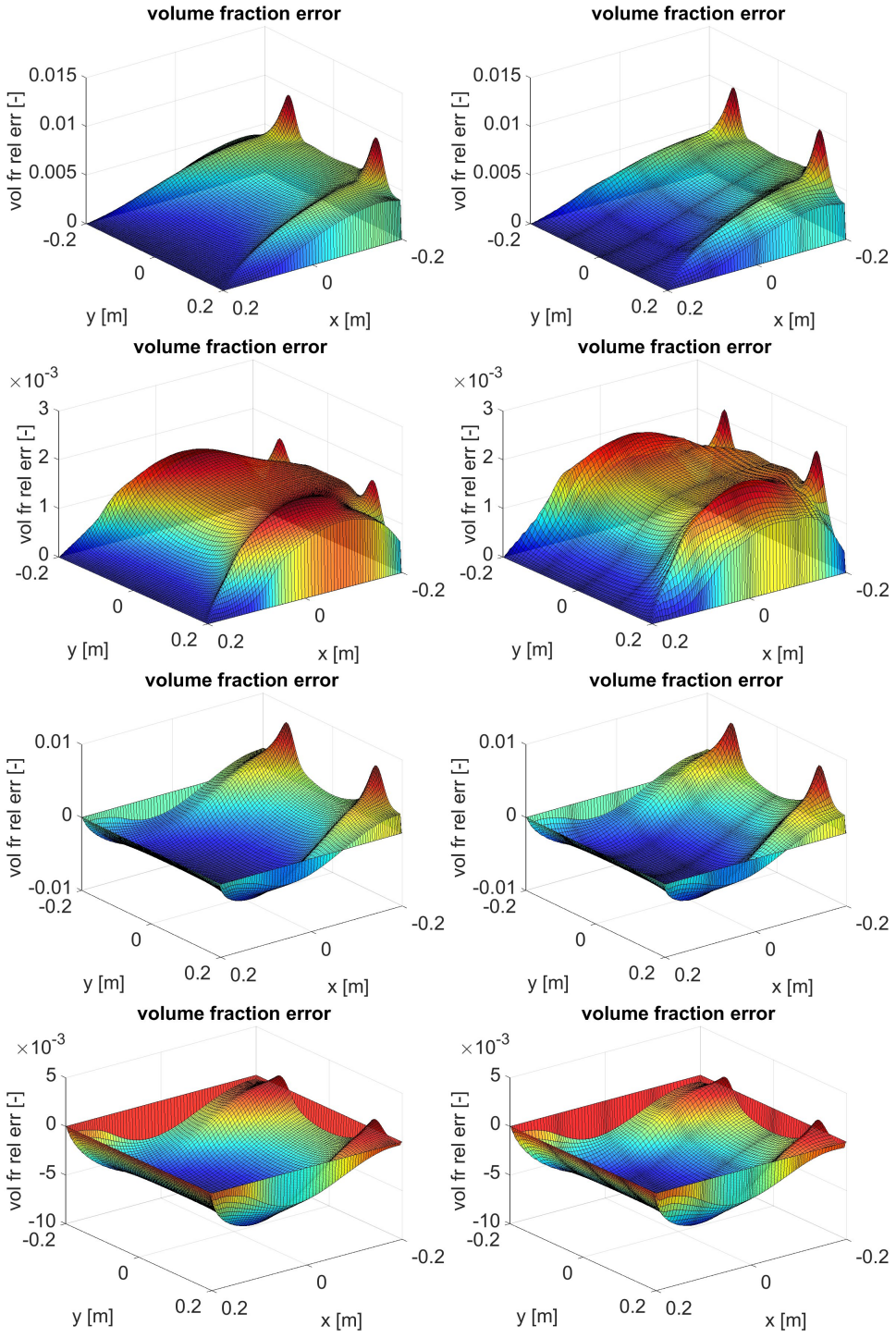


Figure 6.24: α relative errors at $t = 25,2\ s$ on grid with 81 points in x nad y directions. Results are given in same manner as on previous figure 6.20.

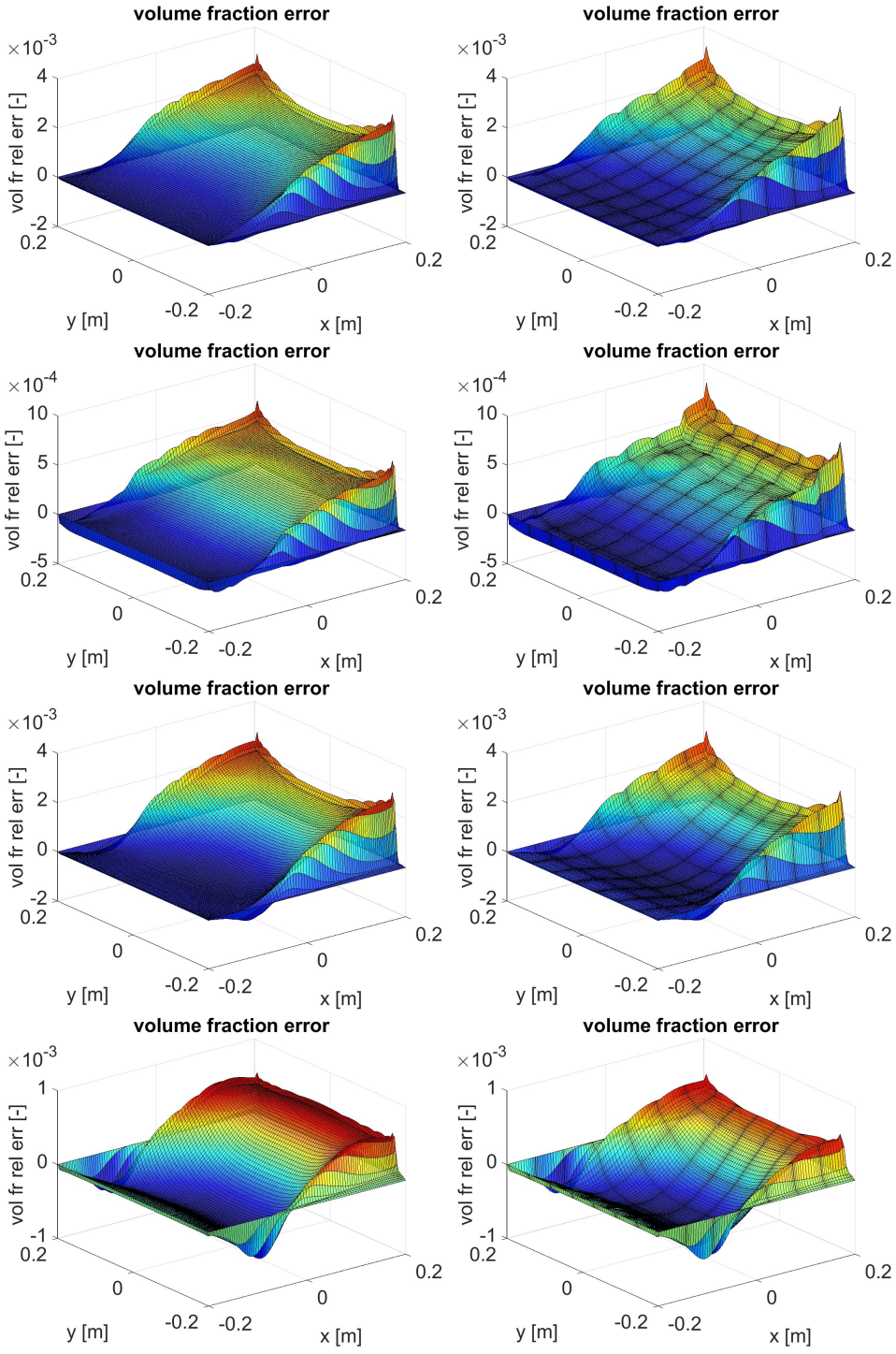


Figure 6.25: α relative errors at $t = 12,6\ s$ on finest grid. Results are given in same manner as on figure 6.20. Every second grid line is plotted.

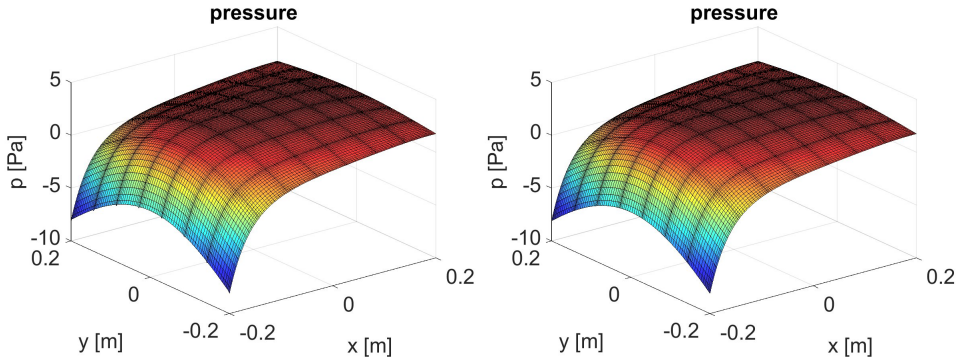


Figure 6.26: Pressure results at $t = 12,6\ s$ on finest grid for PICS (left) and PNUS (right) version of the algorithm. Every second grid line is plotted.

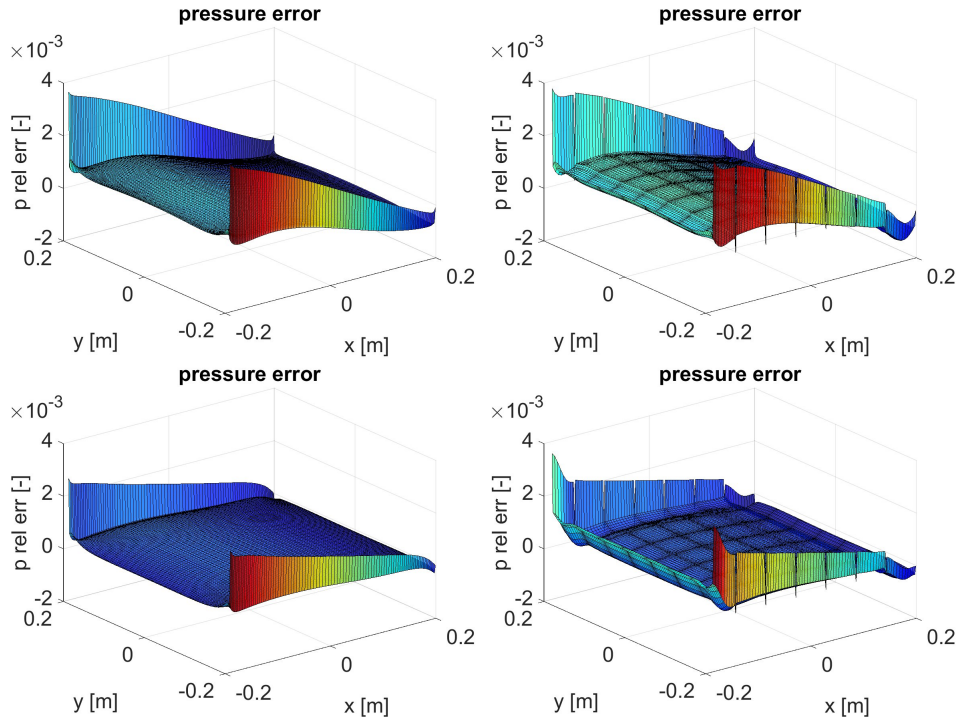


Figure 6.27: Pressure results at $t = 12,6\ s$ on finest grid for mono (left) and multi (right) domain calculations. Upper line gives results with PICS and lower with PIUS version of the algorithm. Every second grid line is plotted, the first ten in x direction are omitted.

3.2 Conclusions from cavitating flow VOA

VOA in cavitating flow case leads to the following conclusions about orders of accuracy of new algorithm and also code for cavitation simulations:

- Generally, the orders of accuracy drop compared to incompressible flow results.
- The most important variable here, defining the orders of accuracy of all variables, is pressure. The pressure order of accuracy drops compared to incompressible flow and all other variables exhibit drops to similar values. This is a consequence of higher impact pressure has on all results through source term S.
- The much greater influence of pressure is best seen in pressure non incremental versions, which exhibit a drop below unity for all variables. The pressure accuracy does not drop much compared to incompressible flow case, yet orders of accuracy for other variables decrease considerably.
- As expected from incompressible flow results, incremental versions exhibit better orders of accuracy with best results obtained with PIUS version. This gives 3/2 order of accuracy for velocities and pressure and first for α. PICS version is close to it, with only pressure exhibiting bit lower accuracy.
- The mentioned orders below unity in pressure non incremental versions are not to be taken as definitive. The reason is that these versions suffer higher effect of mixed boundary conditions for Φ, which combined with the greater role of pressure in cavitating case is the reason for such low results.

It can be argued that results could be better. There are different ways in which better results could be obtained, but there are also arguments against them. These ways and arguments are:

- Periodic boundary conditions could be applied, with which the errors caused at the boundary with Dirichlet conditions for Φ would be avoided, automatically increasing the reported accuracy for all versions of the algorithm. Especially the PNCS and PNUS versions can be expected to return much increased values in both flow cases. This was not done, firstly since the mono domain solver is not applied in a way to enable periodic simulations. Therefore no comparison with multi domain results would follow. Secondly and more importantly, results with such conditions hide important effects, and as it is argued in [80], they cannot be accepted to define actual orders of accuracy.
- Higher orders of accuracy could be obtained by lowering α range, which would also decrease the CG iterations for $\vec{v}^*$ and Φ solutions. This was not applied, since a more thorough test and valid VOA is done with the use of full α range present. Furthermore, it can also be said that the complete span of possible orders of accuracy is obtained in such a manner. The order of accuracy for certain variable was defined before for incompressible flow and here for another limiting case, with complete α range present. The α itself otherwise shows lowest order of accuracy among all variables, which was found to be a consequence of pressure effect and does not play an important role in the final reported orders of accuracy.
- Finally, the orders of accuracy could be increased also with lowering the impact of sudden pressure changes. As this is not done, it follows that in general 3/2 order of accuracy is possible to be achieved even in case of higher pressure and source term changes, which is an important quality for cavitating flow simulations.

For the case of PNCS and PNUS algorithm versions, an obvious option would be also to use a domain which could impose a boundary with a constant and zero pressure value. The devised analytical flow expressions sadly prevent this. But such results for verification, as enabled and shown with the expressions here, are to our knowledge reported for the first time, since no other code or algorithm has been exposed to such VOA procedure. Bearing in mind the limits imposed by the boundary conditions, effect of pressure on all variables and characteristics of the test case, the reported orders of accuracy, especially for pressure incremental versions, show good convergence characteristic of the algorithm.

Given conclusions follow in great part from obtained convergence slopes. The plots of relative errors, discussed after the definition of orders of accuracy, give additional points supporting them. These plots confirm that same characteristics as observed in incompressible flow case are transferred to the cavitating cases. Most importantly, errors clearly show the effect of mixed conditions for Φ on accuracy of pressure results. They also show the direct connection between velocity and pressure errors, as the form of first is set by the latter. Interestingly, the cases with sudden pressure change reveal that numerical boundary layer imposed in PICS and PIUS versions leads to higher pressure and velocity errors in such conditions. Same cannot be seen in PNCS and PNUS algorithm versions, yet these two versions show greater errors overall, originating from the boundary with Dirichlet Φ condition imposed. This is another illustration of the reason for low reported orders of accuracy for pressure non incremental versions. To finish with these conclusions, the plots of α errors are somehow a different story. These do not seem to follow same behaviour as errors of other variables. Nevertheless, convergence curves well show how pressure affects these results too, causing α order of accuracy to equal at most one.

4 Tests concerning increased factors in forcing terms and stability of the new algorithm

In the conclusion of previous chapter describing MMS, it is mentioned that some additional tests were applied in order to define code's and algorithm's response to the changes of factors in devised analytical flow expressions. These tests are presented here. In them, factors a, g, C were increased separately. When one was increased others were kept the same as in the case used for VOA. The density and viscosity ratios ρ_l/ρ_v and μ_l/μ_v were increased in a separate case as well. The purpose was to see how much more demanding it would be to get a stable computation for a certain case and not to perform VOA, neither to get results with small errors or in asymptotic range of convergence. This was done as we did not want to spend too many CPU hours on these tests, therefore also the factors were not generally increased by much. However, applied meshes and time steps are almost the limiting ones, meaning that they still enable stable computations and give a good estimation for the increased demands. Only PICS version of the algorithm was used in each of these test cases. Other three versions were used in two selected cases to make comparison between different versions.

Mentioning that performing of stable computations was observed in these tests, it should be said that regarding estimation of algorithm stability, two approaches were used. First approach was used mainly during development phase, where the algorithm was compared with a version using Dirichlet boundary conditions for Φ on all boundaries. This is possible in tests with MMS as the pressure values are known. The used version for this comparison was PICS, first stable version developed. This is also practical version for implementation of such boundary conditions as Φ in it equals pressure difference in time. The use of pure Dirichlet boundary conditions completely removes discussed issues with compatibility (some are still present even in mixed boundary conditions) and makes for the most stable solution procedure. It was found that no source term linearisation and implementation of CG method is needed to solve for Φ when Dirichlet conditions are used. The solution is namely equally difficult as the solution for $\vec{v}^*$. Therefore the algorithm version using Dirichlet boundary conditions for Φ sets the bar for stability and makes for a very good estimation of the new algorithm. As mentioned, when algorithm was developed, the results were compared with this version. When same results with both were obtained, the development of new algorithm in general was at the end, since highest possible stability, enabled by Dirichlet conditions for Φ, was reached. Additionally, tests with various density and viscosity ratios were performed with both algorithms, and it was surprisingly found that they expressed same demands for calculation settings. If not satisfied, simulations became unstable either at same or almost same time step. This is not to say that such behaviour is generally present, but it backed the conclusion that highest possible stability of the algorithm is achieved. This was a very important point, since such stability was only gradually achieved.

The second approach for stability estimation was done after all four algorithm versions were developed. Since the stability of the developed algorithm was at first confirmed only through the use of PICS version and its counterpart using Dirichlet boundary conditions for Φ, comparison between all four new algorithm versions had to be done too. In such manner, the stability of all versions can be confirmed. This was done in form of searching for settings at which simulation still stably proceeds although it produces notable errors, since this can also reveal if there are huge differences in stability between the four versions. In order not to spend too many CPU hours, two test cases were chosen, one with increased C and one with increased ρ_l/ρ_v and μ_l/μ_v ratios. The versions of the algorithm all proved to be equally stable, with only minor differences between them.

In the following four sections, the main focus is first on presentation of results, obtained to define demands and response of the algorithm and code to cases with increased factors and material property ratios. As mentioned, these results were obtained with the use of PICS version of the algorithm. In cases with increased C and material property ratios, comparison with the other three versions in even more aggressive settings is given to confirm similar stability of all versions of the algorithm.

4.1 Increased C factor

Looking to analytical expressions (5.9) - (5.14), increase in C factor increases amplitudes of all variables except α. Consequently it was found that for successful resolution of cases with increased C, mesh has to be refined, while time step should be lowered. The refinement demand does not have linear behaviour. Namely the highest C we tested was $C = 2$, which demanded use of 36 sub domains (six in x and y directions), each with $25, 25, 15$ points in x, y, z directions. Time step was decreased to $\Delta t = 0,0025\ s$ in order to obtain stable simulations. This means that mesh had to be more than three times more refined than the starting mesh, while corresponding time step was decreased for even more. Results for u and v velocity components, α and p from such calculation with PICS algorithm version are given together with corresponding exact results on figure 6.28. The relative errors of calculated variables are given on the same figure in the last column. The presented results show the state after 6150 time steps.

The results show that errors are kept below one percent. The errors should be taken considering the chosen settings as nearly limiting ones for stable simulations, thus the error magnitude can hardly be discussed. Nevertheless, it is obvious pressure has highest errors and this in position of highest values. Which is somehow expected, as the form of pressure shows the presence of discussed fast pressure switch. α plots have a different plot view, as there is an obvious increase in error from the boundary to the first point inside, not seen from default view position.

This test case was one of the two chosen for stability comparison of the four algorithm versions. This was done as C constant is the easiest one to govern the CFL number with, which was also used as a criteria to find the limiting settings for the four proposed versions of the algorithm. At first, it was found that all versions produce equally good results with same mesh and time step of $\Delta t = 0,0025\ s$. Results obtained with the PIUS version are shown on figure 6.29, to give a comparison with already presented results of PICS version. The results look the same and errors reveal that they are indeed very equal (the PIUS version returns slightly lower errors).

Then, the CFL number was increased with increase in time step. It was found that generally, $\Delta t = 0,0027\ s$ still returns stable results in all algorithm versions. The corresponding highest CFL was found to be $CFL_{max} = 3,41$. The PICS version is able to produce still good results also at slightly longer $\Delta t = 0,00273\ s$, where other three versions of the algorithm quickly diverged. This makes it the most stable algorithm in this regard. Regarding the before mentioned $\Delta t = 0,0027\ s$, results are a good illustration how close to the actual limit of stability the before presented results at $\Delta t = 0,0025\ s$ are. Figures 6.30 - 6.33 give the relative errors for u, v, p and α of the four algorithm versions after 6150 time steps with $\Delta t = 0,0027\ s$. The error amplitudes are higher than in before given examples for $\Delta t = 0,0025\ s$. The errors exhibit strongest similarities between the two pressure non incremental versions, which corresponds well with the behaviour noted in VOA. The highest errors are observed for the PIUS version, where a strong peak at $y = 0\ m$ and $x = 0,2\ m$ is seen. This is a surprise regarding mentioned similarity for this and PICS version at $\Delta t = 0,0025\ s$. The general limit of $\Delta t = 0,0027\ s$ with otherwise many similarities between the plotted errors is a proof that all versions share same stability properties.

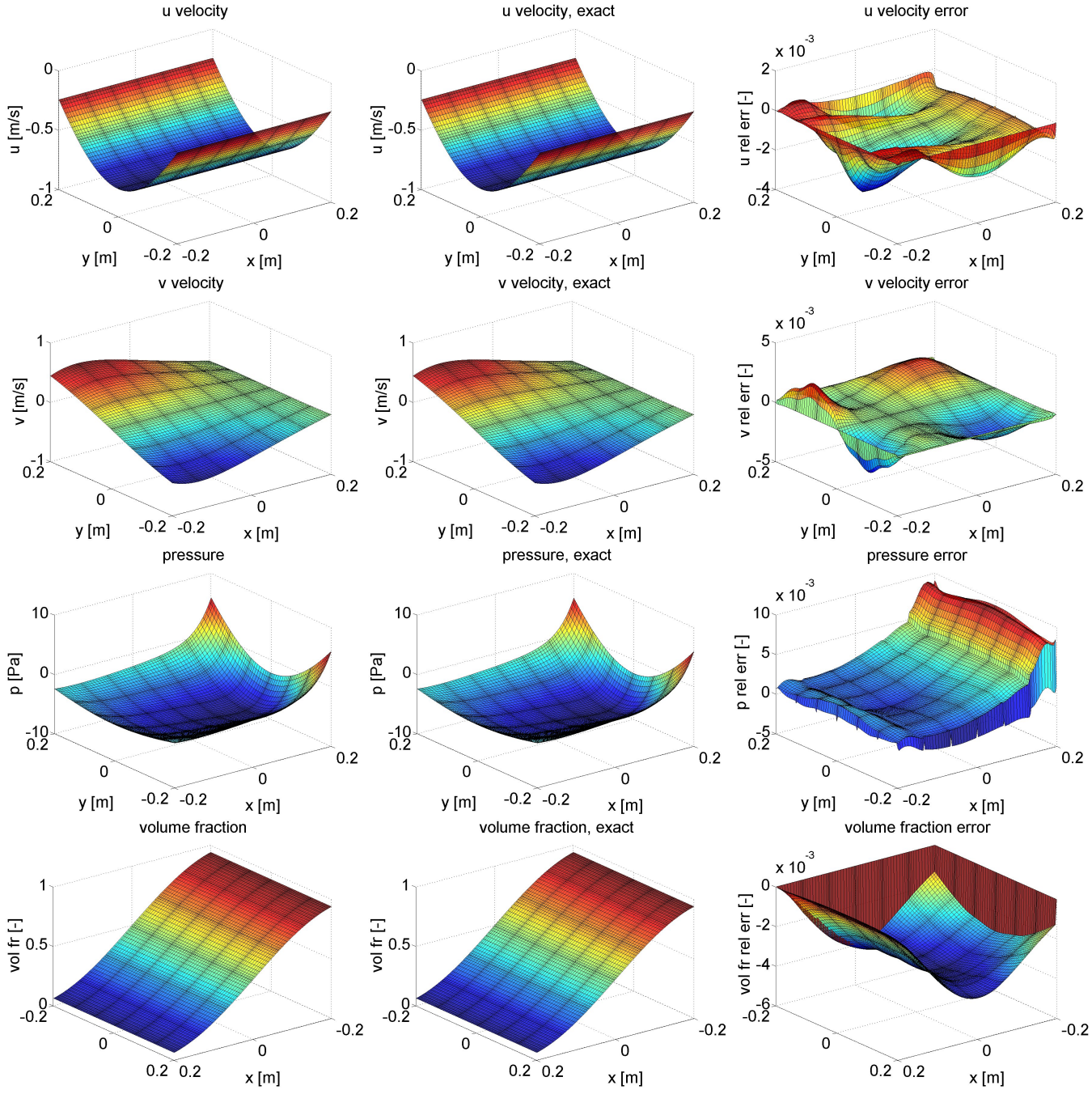


Figure 6.28: The u, v velocity, pressure and α results from top to bottom row for increased C factor after 6150 time steps. Results are obtained with PICS algorithm version. The first column shows calculated values, second shows exact values and the third corresponding relative errors. Plots concerning α are rotated for 180° around z axis.

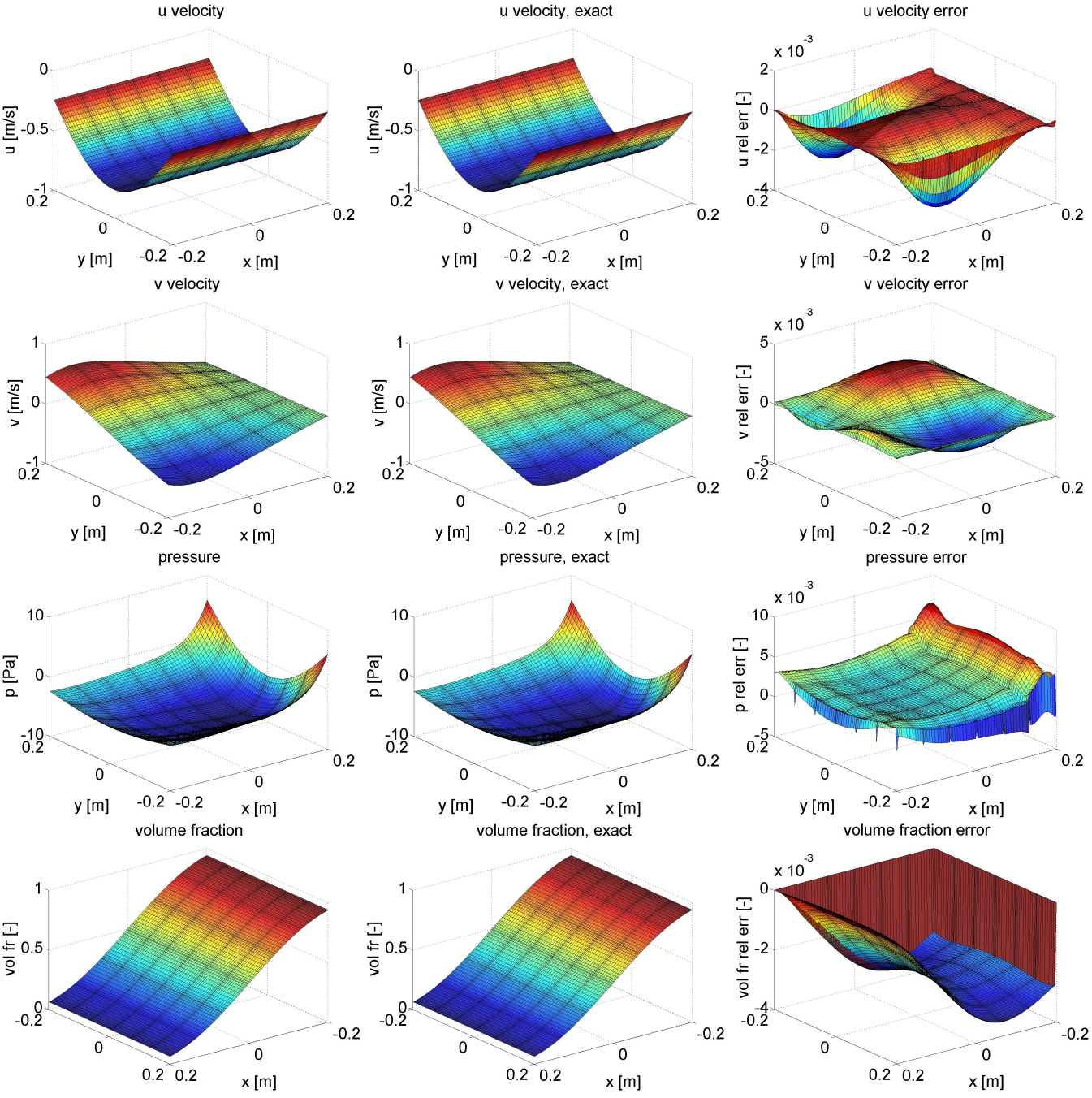


Figure 6.29: The u, v velocity, pressure and α results from top to bottom row for increased C factor after 6150 time steps. Results are obtained with PIUS version. The first column shows calculated values, second shows exact values and the third corresponding relative errors. Plots concerning α are rotated for 180° around z axis.

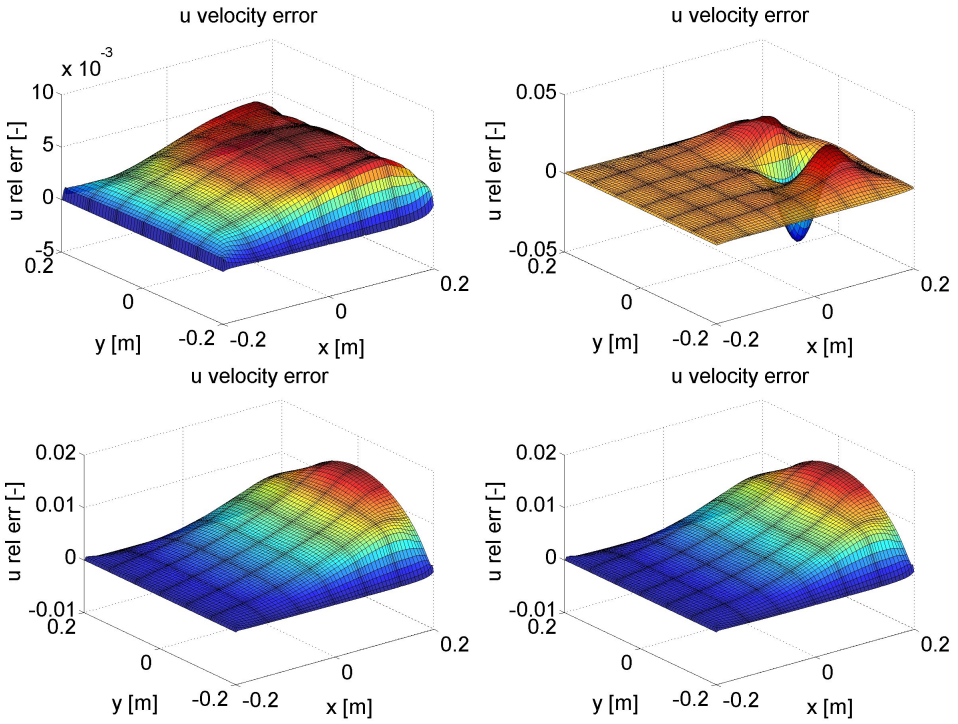


Figure 6.30: u velocity relative errors after 6150 time steps with $\Delta t = 0,0027\ s$. First row gives results for PICS (left) and PIUS (right), the second for PNCS (left) and PNUS (right) versions.

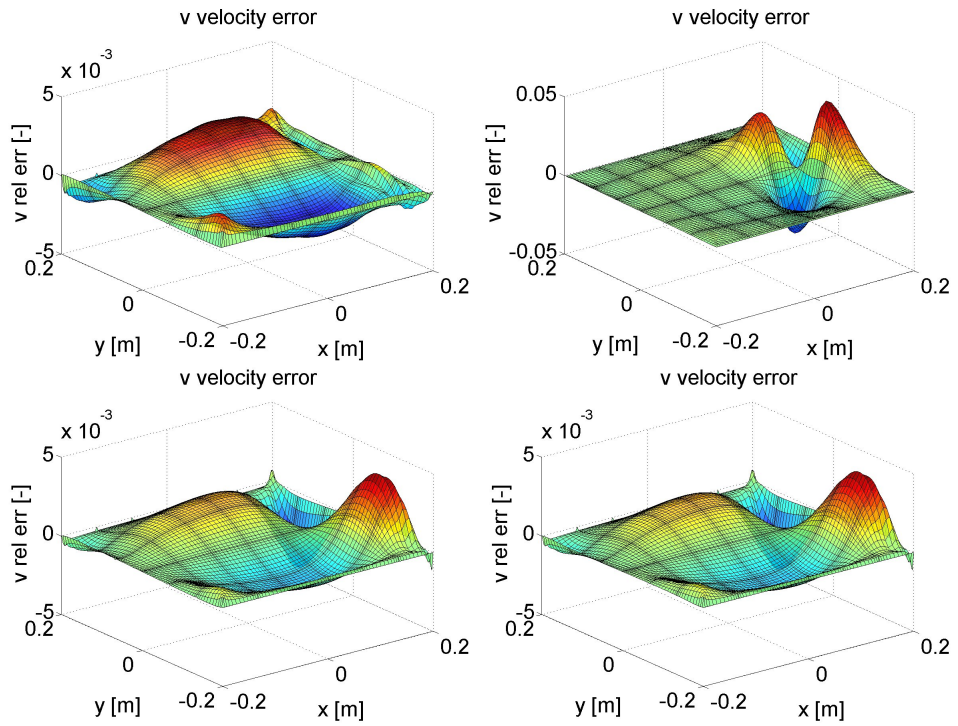


Figure 6.31: The relative errors of v velocity after 6150 time steps with $\Delta t = 0,0027\ s$. Results are given in same manner as on figure 6.30.

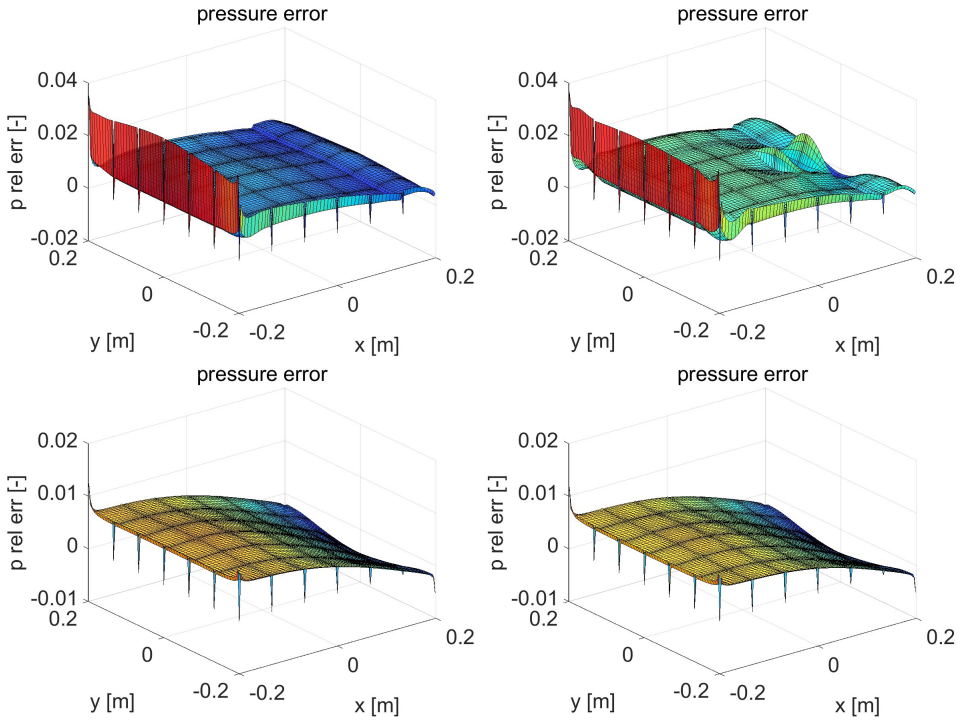


Figure 6.32: Pressure relative errors after 6150 time steps with $\Delta t = 0,0027\ s$. Results are given in same manner as on figure 6.30.

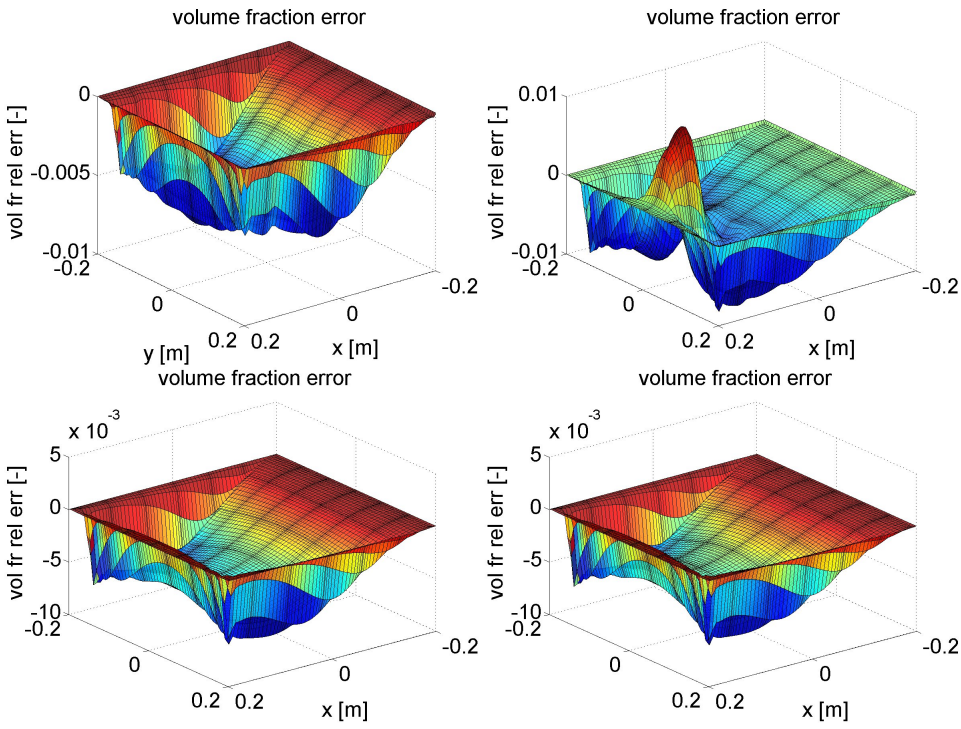


Figure 6.33: The relative errors of α after 6150 time steps with $\Delta t = 0,0027\ s$. Results are given in same manner as on figure 6.30 except that the plots are rotated for 180° around z axis.

4.2 Increased g factor

Increase in g factor is meant to test response of the algorithm to the increased frequency of temporal oscillations. However, in equations (5.9) - (5.14), g also increases amplitudes of v velocity and pressure. Because of this effect and because g is taken as a product of π with an integer, increase in it was found to be very demanding with keeping the domain limits unchanged. Importantly, the stability of calculations was here of secondary concern, as it was revealed that simulations can proceed in a stable manner although quite high errors in computed variables are noted. Therefore the quest here was to find a still stable case with not too high errors. It was found that good results with a value of $g = 4\pi$ can be reached with refinement of the mesh in x direction, while in y direction, the mesh is kept much coarser. The reason for this is simple. As g increases pressure amplitude, it also causes higher sudden pressure change. As it was shown, this occurs in vicinity of domain limits in x direction. Hence the pressure features even higher spatial and also temporal gradients during these sudden pressure changes. Since pressure heavily affects all other variables, it becomes clear why the mesh needs to be much refined in x direction in order to lower the errors. However, the fact that simulations remain stable even with such higher pressure changes gives an additional proof for stability of presented solution procedure for Φ. The domain, in which still stable and roughly acceptable results were obtained in case of $g = 4\pi$, was split into 80 sub domains, 20 in x direction and only 4 in y. Each sub domain had $31, 21, 17$ points in x, y, z direction respectively, while time step was $\Delta t = 0,001\ s$. Figure 6.34 shows results obtained after 3850 time steps were performed. The high errors are clearly seen, causing still obvious deviations between the calculated and exact results. Importantly, pressure results show how much higher the pressure amplitudes during the sudden pressure change are in this case.

4.3 Increased a factor

Increase in a causes increase in spatial oscillations, but as with g, there is a side effect which is an increase in pressure amplitude. Since $K1 = 0$ was used, care must be taken for ax product not to exceed value of $0,5$. Otherwise points with $\alpha = 0$ are reached and calculations become impossible because of discussed pressure singularity issues. Therefore for a case with $a = 8\pi$, domain in x direction was limited to $\{-0,05; 0,05\}\ m$. It is however less demanding to increase a than previous two constants. Only twelve sub domains, three in x and four in y, were needed to successfully solve case with such a. Each sub domain had $25, 25, 13$ points in x, y, z directions and time step with $\Delta t = 0,0025\ s$ was used. Figure 6.35 shows the results obtained after 2550 time steps were performed.

Results show quite high errors in velocities, especially v. Pressure and α errors are of same magnitude and highest at boundaries. Especially α errors show interesting oscillating nature in y direction. This might be in connection with the shape of α which changes only in x and presence of high error in pressure on the boundaries, causing the source term to feed the observed α oscillations. Again, the intention of the calculation was not to get best results but obtain a still stable calculation and define demands for it when a certain factor is increased. In case of a, no special requirements are noted.

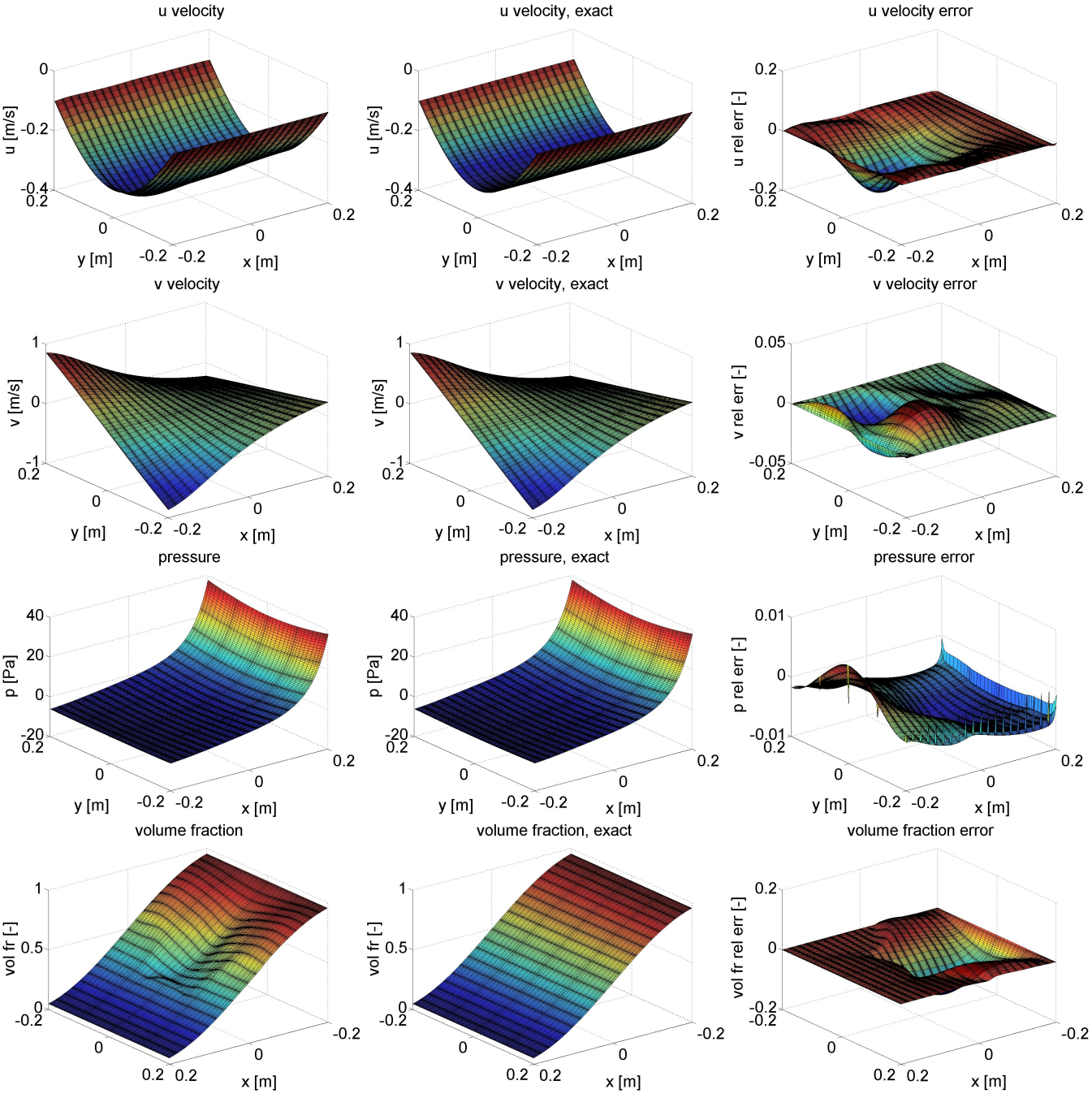


Figure 6.34: The u, v velocity, pressure and α results from top to bottom row for increased g factor after 3850 time steps. Results are obtained with PICS version. The first column shows calculated values, second shows exact values and the third corresponding relative errors. Plots concerning α are rotated for 180° around z axis.

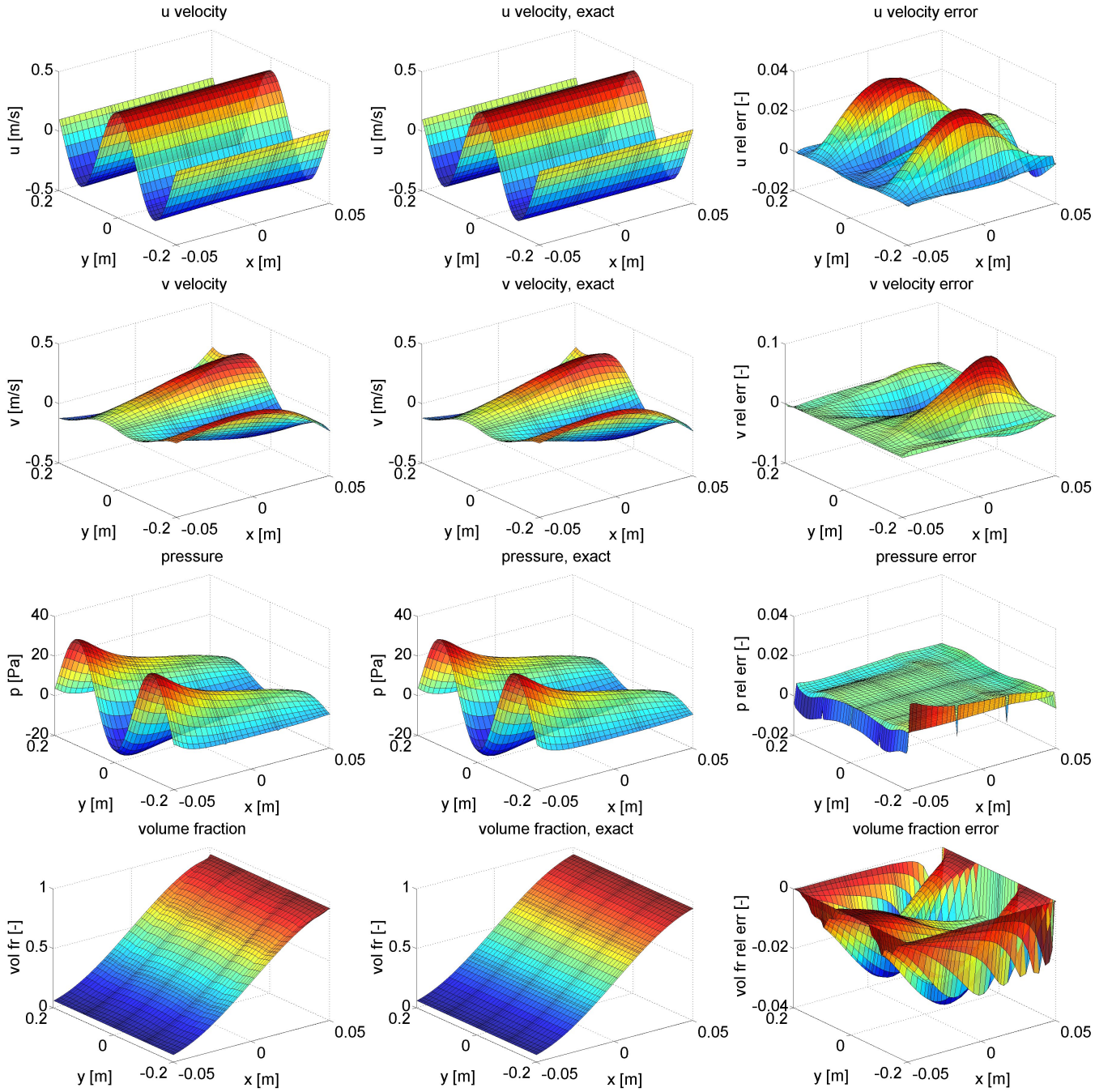


Figure 6.35: The u, v velocity, pressure and α results from top to bottom row for increased a factor after 2550 time steps. Results are obtained with PICS version. The first column shows calculated values, second shows exact values and the third corresponding relative errors.

4.4 Increase in density and viscosity ratios

Increase in these ratios is the most important test from this group. This had to be done in order to predict how the code reacts to realistic phases and ratios of their variables, which can sometimes cause considerable problems for stability. Here, density ratio is more important than viscosity, as it has bigger influence on governing equations and it is also higher than realistic viscosity ratio. The priority was therefore to obtain realistic density ratio and come as close to the realistic viscosity ratio as possible. It was found that the code accepts realistic density ratio of $\rho_l/\rho_v = 1000/0,02$ without problems ($\rho_v = 0,02\ kg/m^3$ being the vapour density at $p = 2\ kPa$ and temperature $T \approx 20\ °C$). In fact, the vapour density was even decreased to $\rho_v = 0,001\ kg/m^3$ in the following tests. Requirements to perform stable simulations were mesh and time step refinement. The size of the domain is the same as for VOA tests. The mesh had to be refined in x direction as α changes according to it. Viscosity ratio, which was of secondary interest here, can also be high, but mixture viscosity had to be kept much higher than in reality. Reason is in increased Re number, which would lead to more CPU power used for stable simulations. For the density ratio of $\rho_l/\rho_v = 1000/0,001$ we then found that stable and not too CPU intensive simulations can be performed with viscosities of $\mu_l = 20\ Pas$ and $\mu_v = 0,1\ Pas$. The ratio between them is twice higher than realistic ratio ($\mu_l = 0,001\ Pas$ and $\mu_v = 10^{-5}\ Pas$ for viscosities at 20 $°C$). Simulations with such ratios were done using 14 sub domains, 7 in x and 2 in y, with $31, 31, 11$ points in x, y, z directions in each. Figure 6.36 shows results with PICS version after 8500 time steps with $\Delta t = 0,0015\ s$ were performed.

This case was the second one used to compare the stability of all algorithm versions. This was done because of the before mentioned possible issues with high density and viscosity ratios. Like in the case of C increase, calculations with same settings were done at first also for other three algorithm versions. Figure 6.37 shows results for PNUS version. It can be seen that results are again very similar, which is confirmed by similar magnitude of errors. This seems to differ by factor of two between the two shown cases. Same behaviour was found for other two versions of the algorithm and the highest similarity was again observed between PNCS and PNUS versions.

The stability of the versions was then further compared with additional decrease in both viscosities. The goal was to find which would be the lowest mixture viscosity each version could accept and still run in a stable manner. For this, vapour viscosity was dropped to $\mu_v = 0,025\ Pas$ and liquid viscosity was gradually decreased. Time step was increased to $\Delta t = 0,0035\ s$ while mesh was unchanged. This was done because definition of stable calculations limit in this case was found to be troublesome, therefore it was decided to use higher CFL to help advance the search. It was found that still stable computations can be performed with $\mu_l \approx 4,0\ Pas$. The limiting μ_l was found to vary slightly between algorithm versions. The lowest viscosity was reached for PICS version with $\mu_l = 3,5\ Pas$. For both pressure non incremental versions, $\mu_l = 4,0\ Pas$ was found to be the limit. PIUS version demanded the highest viscosity to still proceed in a stable manner as $\mu_l = 4,3\ Pas$ had to be used. Figures 6.38 - 6.41 give u, v, p and α relative errors for the four algorithm versions. These were obtained with $\mu_l = 4,3\ Pas$ for PIUS version and $\mu_l = 4,0\ Pas$ for other three. Errors are in general much increased compared to the before presented two cases, for cca factor of ten. u velocity expresses similar error in all versions, while v is highest for PIUS (mind the scale) and lowest for PICS version. Similar behaviour can be observed for pressure, where the errors are especially high (more than 30 %). Interestingly, although PICS version seems to produce lowest errors for pressure and v velocity, it produces highest errors for α. Nevertheless, it seems to be able to accept lowest μ_l and mixture viscosity. Because of this, and lowest errors for p and v, it can be considered as most stable algorithm version in this case too.

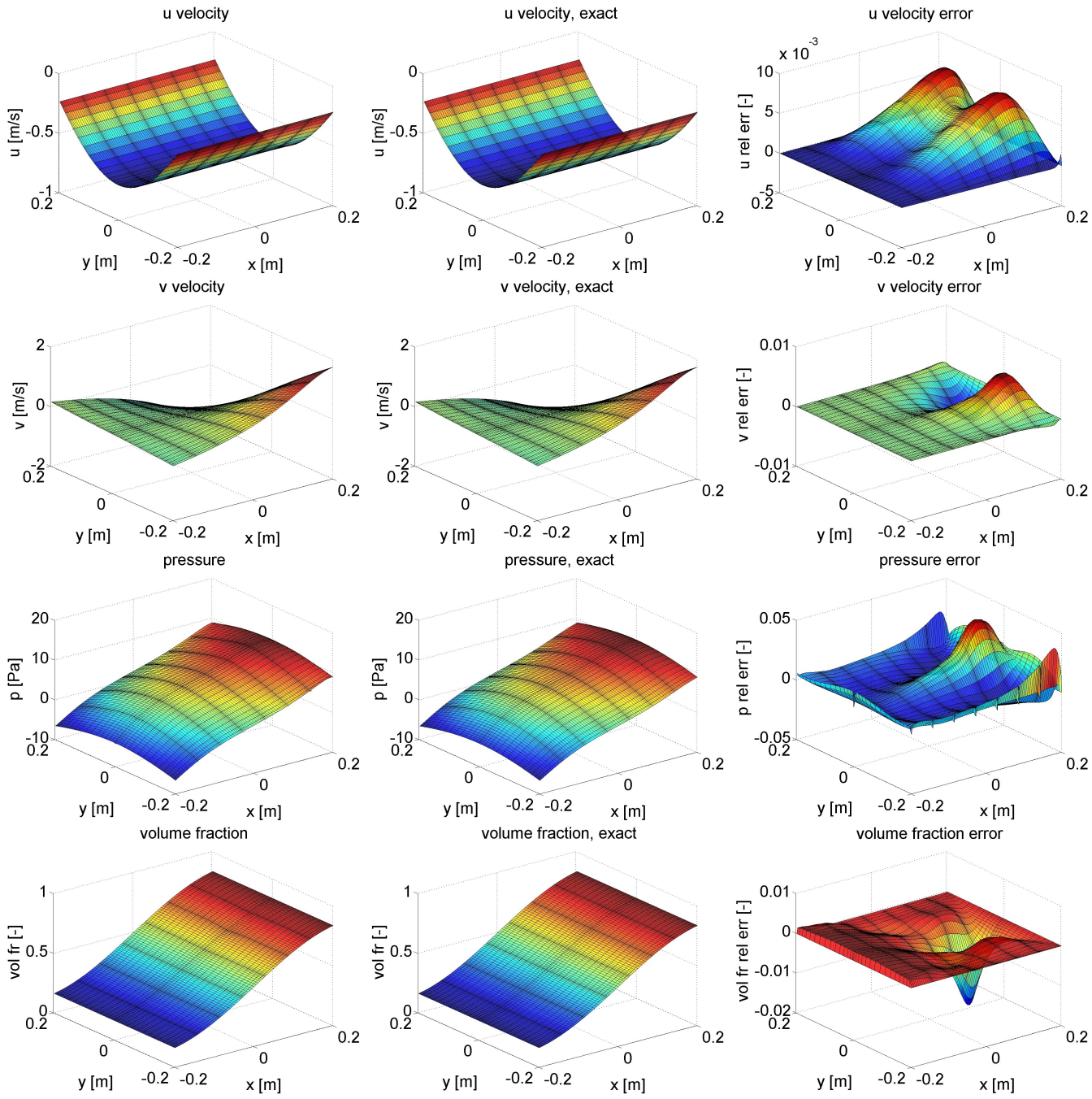


Figure 6.36: The u, v velocity, pressure and α results from top to bottom row for increased ρ_l/ρ_v and μ_l/μ_v ratios after 8500 time steps with $dt = 0,0015\ s$. Results are obtained with PICS version. The first column shows calculated values, second shows exact values and the third corresponding relative errors.

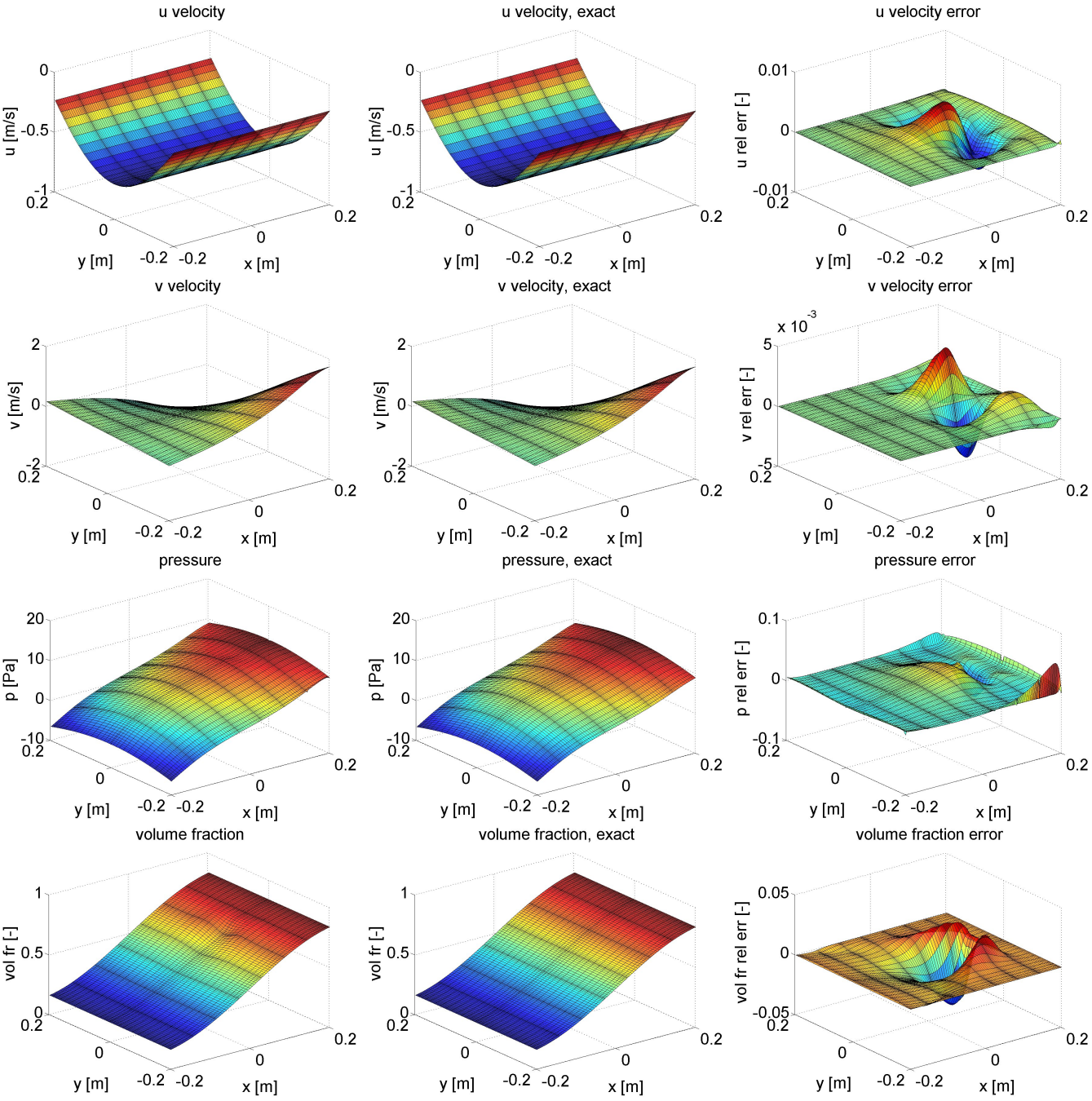


Figure 6.37: The u, v velocity, pressure and α results from top to bottom row for increased ρ_l/ρ_v and μ_l/μ_v ratio after 8500 time steps with $dt = 0,0015\ s$. Results are obtained with PNUS version. The first column shows calculated values, second shows exact values and the third corresponding relative errors.

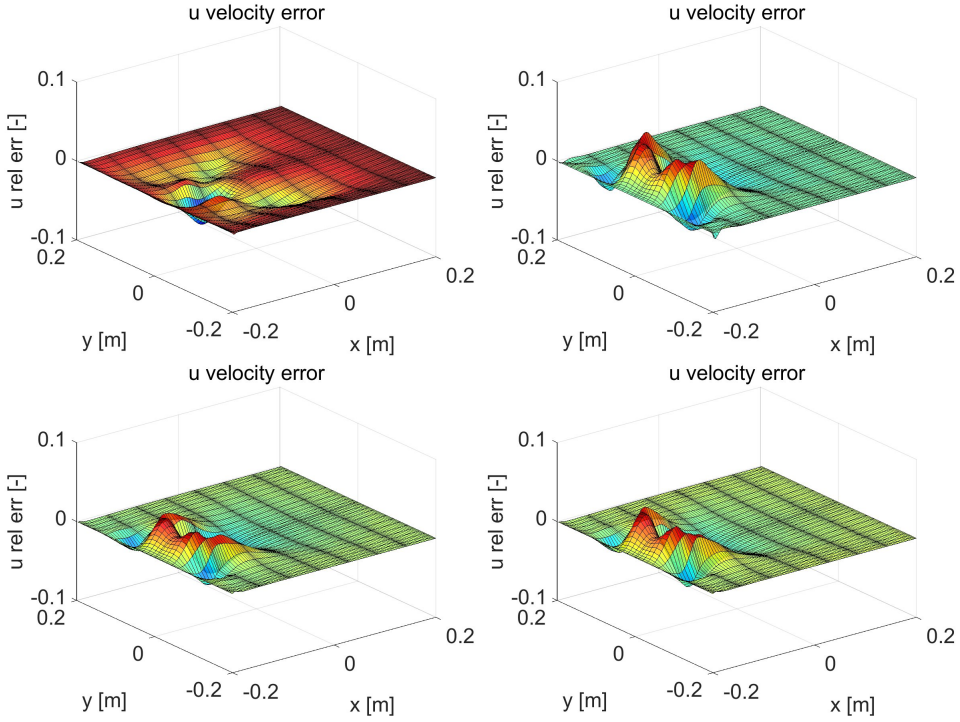


Figure 6.38: The relative errors of u velocity for increased density and viscosity ratios after 8500 time steps with $\Delta t = 0,0035\ s$. First row gives results for PICS (left) and PIUS (right), the second for PNCS (left) and PNUS (right) versions.

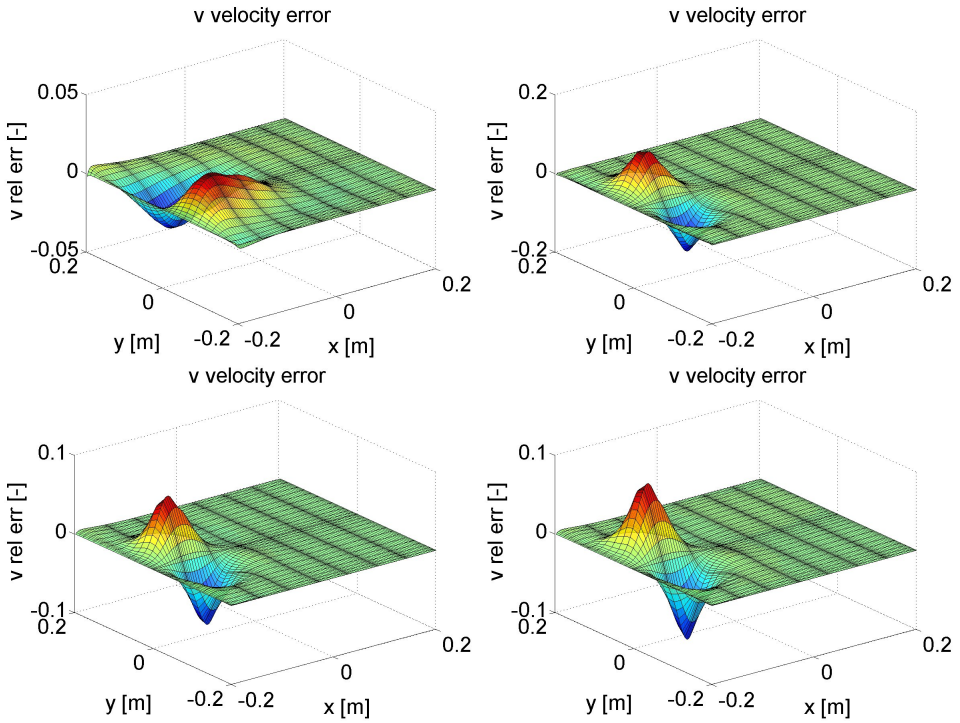


Figure 6.39: The relative errors of v velocity for increased density and viscosity ratios after 8500 time steps with $\Delta t = 0,0035\ s$. Results are given in same manner as on figure 6.38.

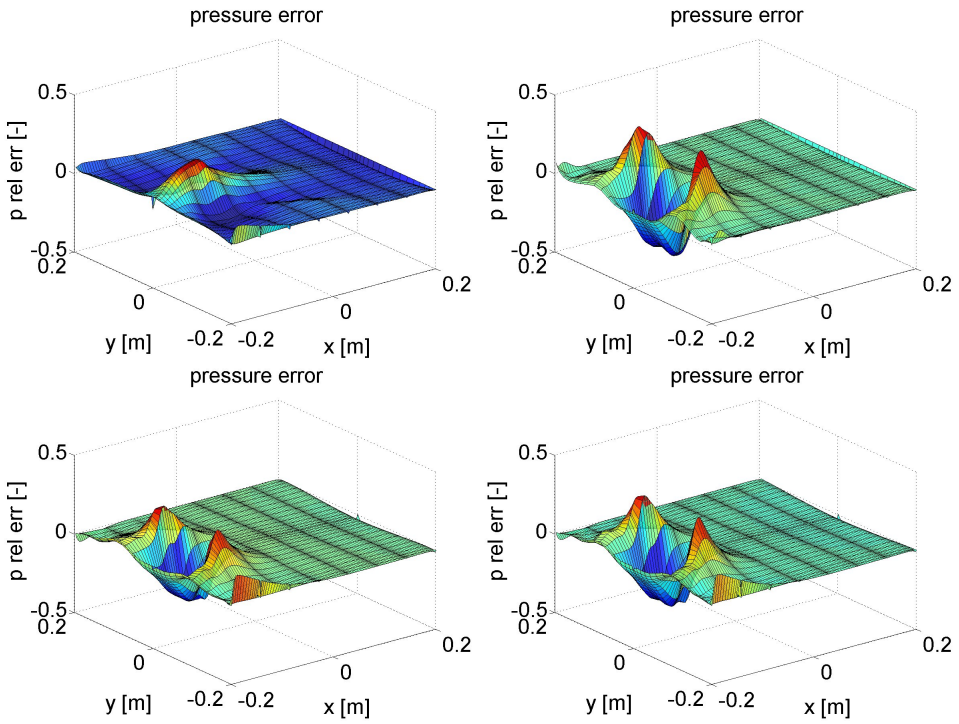


Figure 6.40: The relative errors of pressure for increased density and viscosity ratios after 8500 time steps with $\Delta t = 0,0035\ s$. Results are given in same manner as on figure 6.38.

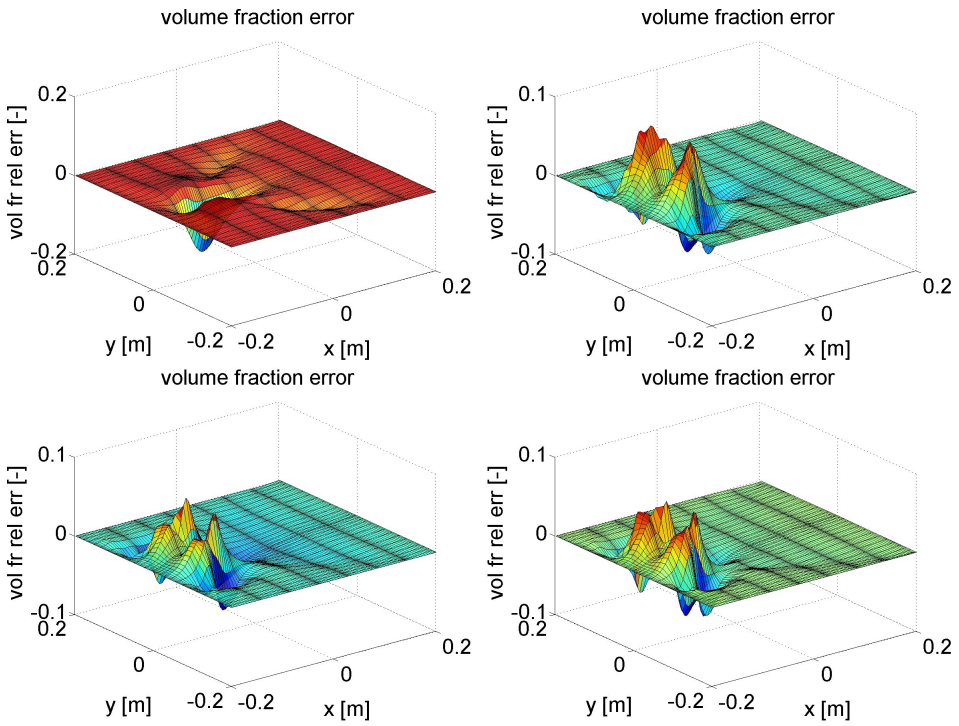


Figure 6.41: The relative errors of α for increased density and viscosity ratios after 8500 time steps with $\Delta t = 0,0035\ s$. Results are given in same manner as on figure 6.38.

4.5 Conclusions

To conclude with this test case and also section dealing with increased factors in analytical flow expressions, it can be said that PICS version of the algorithm proved to have a slightly better performance than other versions. However, it is apparent that the four versions do not differ much when it comes to performing calculations with certain increased effect. Finally, a strong point for the performance of the algorithm is its ability to accept realistic density ratio without any particular demands.

5 Performance analysis of the code and algorithms

This section is devoted to the analysis concerning computational performance of the code and the effect the used algorithms have on it. The section is split into three parts:

- At first, a general analysis is given with strong and weak scaling tests, for which the original code was used. This reveals some concerning points related to before shown increase in Φ errors on most refined grids in incompressible flow multi domain cases.
- Then comparison between performance of the original code to solve same cases using mono or multi domain computations follows. Together with it, comparison of the original algorithm with the new one through the use of incompressible flow case is also presented.
- Finally, performance comparison of all four versions of the new algorithm in cavitating flow case is given.

5.1 Strong and weak scaling tests

If a code is to be used for expensive simulations, such as DNS, it has to be able to perform or scale well with increased amount of CPU. In order to asses the performance of MFLOPS-3D with influence matrix as multi domain method, the strong and weak scaling tests were performed on ADA super computer in IDRIS facilities in Orsay, France [132]. The computer features 332 computing nodes, each with four Intel Sandy Bridge E5-4650 processors with 8 cores. The nodes have in general 4 GB memory per core with some also permitting 8 GB. Combined performance is 233 $Tflops/s$, the nodes are connected with InfiniBand.

The strong and weak scaling tests apply different approach to estimate performance of the code. The strong scaling tests suppose a starting case solved as a mono domain case. The domain is then split into multiple sub domains where the total amount of points remains the same. The weak scaling on the other hand proposes to solve the case by using more and more sub domains in which the amount of points stays the same, therefore increasing the total amount of points in the process. The original version of the code was used for these tests, as the mono and multi domain solvers used in it are also used in the new version for cavitating flow. The described MMS incompressible flow test case was used in cubical computational domain with $0,4$ m edges. Uniform mesh was applied. 200 time steps were used to obtain desired information, where it was at first ensured that the influence matrix solutions settled. As these are obtained iteratively, they exhibit a much increased amount of iterations during first ten time steps. Therefore the recording of iterations and computational time was applied after first 20 time steps were done. It was also preferred to perform strong and weak scaling tests using unsteady flow case, as this gives a better average. The iterative procedure namely cannot perform same amount of iterations for each condition. The following two sections present the results from the two scaling tests done.

5.1.1 Strong scaling

The domain for strong scaling was chosen to have 73 points per direction, which enabled these tests to be performed at most with 729 sub domains. Therefore each sub domain had at least 9 points per direction, which is also close to the limit imposed by stencils of compact finite differences. The used cases are given in table 6.9. The information about the size of influence matrix is also included and applies to both influence matrices used (for pressure and velocity). It is apparent that although more information has to be transferred with increased amount of sub domains, the influence matrices get smaller. Their size therefore strongly depends on the size of an interface.

Table 6.9: Cases used to perform strong scaling tests.

CPU per direction	CPU total	points per CPU	influence matrix size [Mb]
1	1	73	
2	8	37	1100
3	27	25	1300
4	64	19	1200
6	216	13	839
8	512	10	613
9	729	9	528

The performance was estimated for each variable with comparing average times needed to obtain a complete variable solution in a time step, the average amount of iterations done to obtain interface values or influence matrix solution per time step, average time to obtain an influence matrix solution, average time to make influence matrix iteration and from it also the time to perform one influence matrix iteration per grid point. These results are shown on graphs in figures 6.42 – 6.46 for u velocity and Φ. Other two velocities expectedly express same results as u.

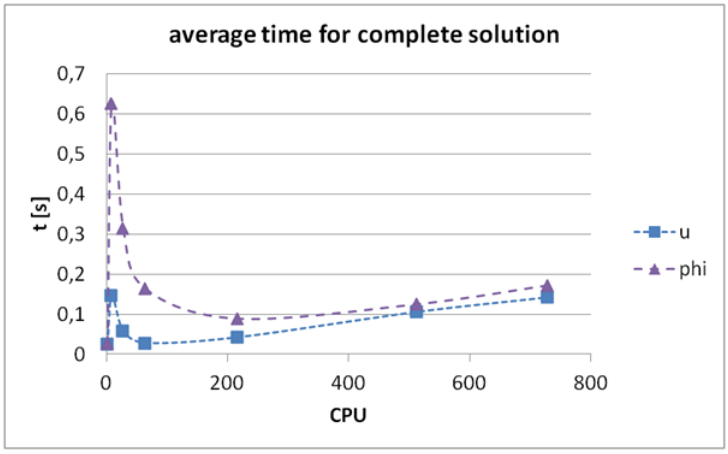


Figure 6.42: Strong scaling results for average time needed to obtain a complete time step solution for u or Φ.

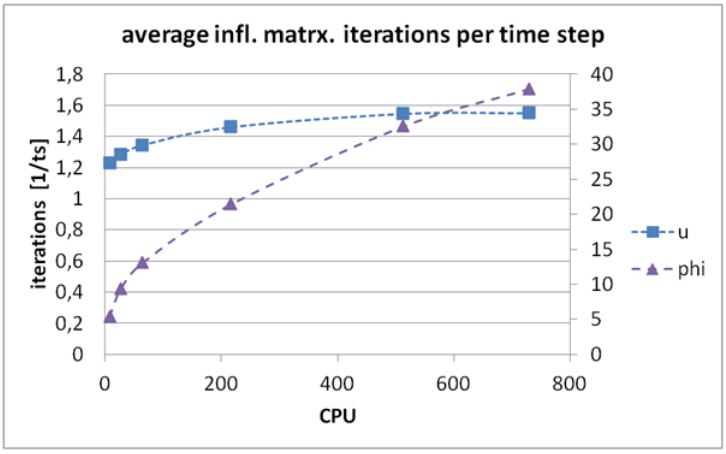


Figure 6.43: Strong scaling results for average amount of iterations performed per time step to obtain influence matrix solution for u or Φ. Secondary scale (right) is for Φ.

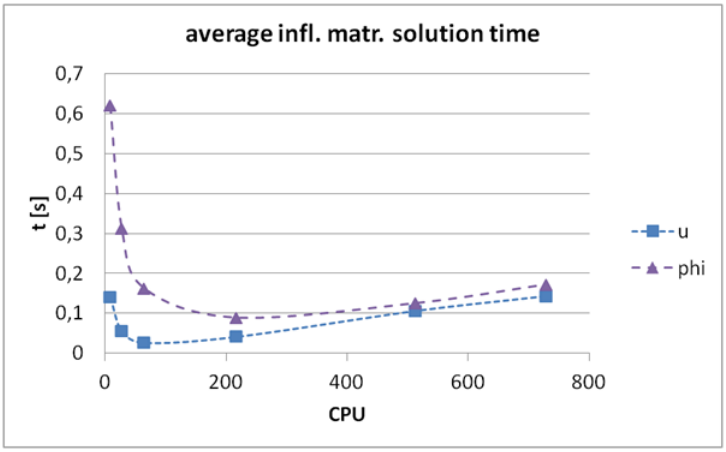


Figure 6.44: Strong scaling results for average time to obtain influence matrix solution for u or Φ.

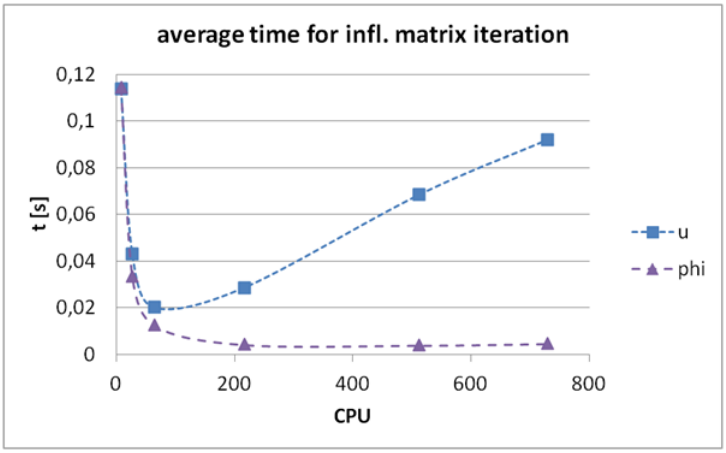


Figure 6.45: Strong scaling results for average time of an iteration in influence matrix solution for u or Φ.

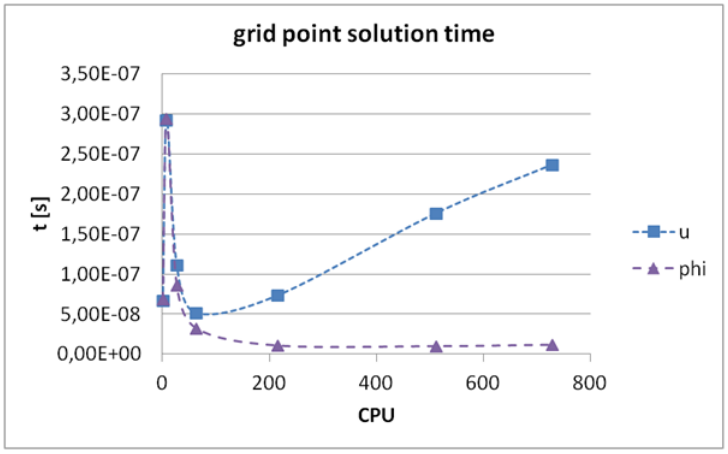


Figure 6.46: Strong scaling results for average time needed per grid point to perform an influence matrix solution iteration for u or Φ. First point shows average time for a point solution in mono domain case.

The average times to obtain one time step solution on figure 6.42 reveal that mono domain case is the fastest. The case with two sub domains per direction returns the worst performance, which is then quickly brought down to the mono domain level again with adding more sub domains. With further increase in sub domains, the performance starts to drop again. Interestingly, the increase in time to obtain a time step solution is quite slow with much increased amount of CPU, showing good scalability in this test case. The reason for this can be further located on graph in figure 6.44. This shows that with increase in amount of sub domains, the time to obtain influence matrix solution of a certain variable increases slowly, despite much increased amount of interfaces and consequently computations and communications which need to be done between sub domains. This is a strong point for influence matrix use. However, the resemblance between this graph and the one in figure 6.42 shows that the time to obtain influence matrix solution practically defines the complete solution time and with it performance of the code. The mono domain solver, although used twice in each time step, in multi domain cases takes practically negligible computational time as it deals with smaller amount of points.

Since it is obvious that on one hand, largest sub domains cause worse performance of influence matrix, while on the other hand smaller sub domains lead to increase in CPU, a compromise has to be done. The most practical cases seem to be those which use 25 and 19 points per direction (27 and 64 CPU cases in the graphs). These are also close to the saddle point in the shown graphs, where the graph on figure 6.43 is excluded. This shows the amount of iterations done on average to obtain complete influence matrix solution. These iterations increase with amount of sub domains. Where the increase is practically not an issue in case of velocities, it shows a problem in case of Φ solution, where iterations increase considerably with increase in sub domain number. This is certainly unwanted, but a good point is that here, the before discussed increase in time to obtain influence matrix solution does not follow increase in iterations directly. The graph in figure 6.45 shows the reason for this and interesting difference between u and Φ solutions. The time it takes to perform one iteration for influence matrix solution in case of Φ decreases towards an asymptote with amount of iterations while same cannot be said for velocity. The graph on figure 6.46 only completes this observation.

5.1.2 Weak scaling

Weak scaling was performed using sub domains with 25 points per direction (15625 points per sub domain), which was found as most practical size in strong scaling. The number of sub domains was then increased to 1000, the chosen cases are shown in table 6.10. Table also includes information about the time step and influence matrix size. The time step was changed to keep similar CFL as the mesh is gradually refined. The influence matrix size in this case grows considerably as a whole, but not per CPU. The size per CPU reaches an asymptote. Time needed to set whole influence matrix shows similar behaviour.

Table 6.10: Cases used to perform weak scaling tests.

CPU per direction	CPU total	points per direction	time step [s]	infl. mat. size [Gb]	inf. mat. size per CPU [Mb]	time to set infl. mat. [s]
1	1	25	0,0225	0,193	24	7,9
2	8	49	0,0115	3,8	59	11,3
4	64	97	0,0058	41	80	12,1
8	512	193	0,0029	85	85	13
10	1000	241	0,0023	151	87	13

The performance in weak scaling was estimated in same manner as before. The graphs which show same information in this case are given on figures 6.47 – 6.51. There are some similarities between the strong and weak scaling results. Average amount of influence matrix iterations done per time step, given on figure 6.48, shows again only slow increase in amount of iterations for velocity, while amount of iterations for Φ grows considerably with increase in sub domain number. The average time to perform an iteration, shown on figure 6.50, is also similar to that in strong scaling. The time for u iteration grows, while for Φ it reaches an asymptote with increase in sub domains. Interestingly, although a sub domain here has more points and larger interfaces than in strong scaling cases, time orders of magnitude are comparable. The graph on last figure 6.51 again confirms this. It also shows that the time it takes to perform an iteration per grid point decreases here for velocity as well. This is a consequence of increase in total amount of points in the domain.

However, the differences between strong and weak scaling show problems exist. The main difference is in the development of the average time to reach a complete solution given on graph in figure 6.47. As in strong scaling, this time is defined by the time it takes to obtain influence matrix solution, confirmed by graph on figure 6.49. The increase in time to obtain velocity solution is not an issue here, its growth is comparable to the one in strong scaling (for cases with more than 60 CPU). The time to obtain Φ solution however increases dramatically and is constantly an order of magnitude higher than time for u solution. Therefore it is the most important factor in defining performance of the code. Here the increase in amount of iterations for Φ influence matrix solution becomes problematic. As the graph on figure 6.50 shows, the average time for one influence matrix iteration does not change much with number of sub domains, therefore high increase in time to obtain Φ solution follows directly from increased amount of iterations. For a better performance of the code, the amount of iterations for Φ influence matrix should be decreased. This is especially important for DNS simulations, where more than 1000 CPU are expected to be used by default in real cavitating flow configuration.

There exists one other option to improve the performance, but seems less practical. As mentioned, solution times for Φ influence matrix are constantly one order of magnitude higher

than for u (graph in figure 6.49). Observing same graph in case of strong scaling on figure 6.44, this can well point out that the ratio between the u and Φ solution time depends on amount of points in a sub domain. Namely, for a similar amount of points in a sub domain, the strong scaling also exhibits a considerable ratio in time between Φ and u solution. If this holds generally and not just for observed strong scaling case with certain amount of sub domains used, it means that the difference in time to obtain influence matrix solution could be brought down by substantially decreasing amount of points in a sub domain. This is eventually not practical, even if larger and larger clusters are built nowadays, as it could demand increase in amount of sub domains to the 3^{rd} power. Not to mention that with increase in sub domain amount there is always a considerable increase in influence matrix iterations for Φ. Even if these in smaller sub domains (judging by strong scaling tests) would generally not cause considerable difference between Φ and u solution times, their amount is still too high. Which leads back to the before given need to decrease it. Especially since the weak increase in iterations for velocity in both scaling tests points towards the problem of weak convergence in Φ influence matrix solution.

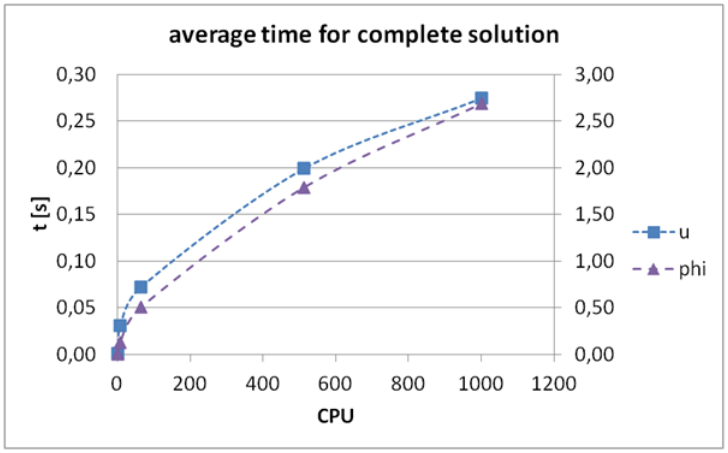


Figure 6.47: Weak scaling results for average time needed to obtain a complete time step solution for u or Φ. Secondary scale (right) refers to Φ results.

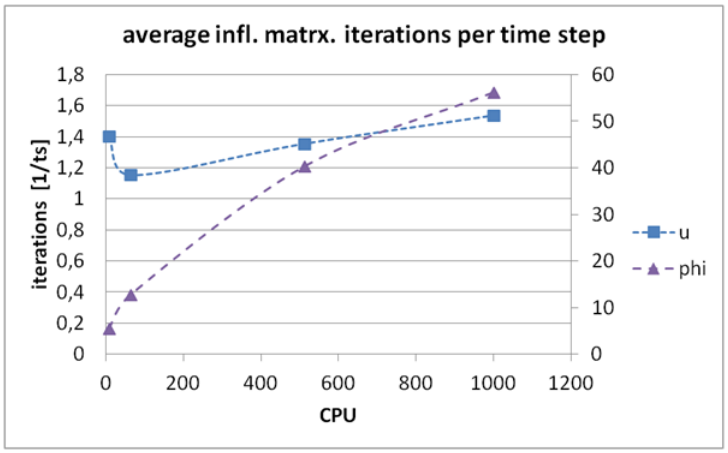


Figure 6.48: Weak scaling results for average amount of iterations performed per time step to obtain influence matrix solution for u or Φ. Secondary scale refers to Φ.

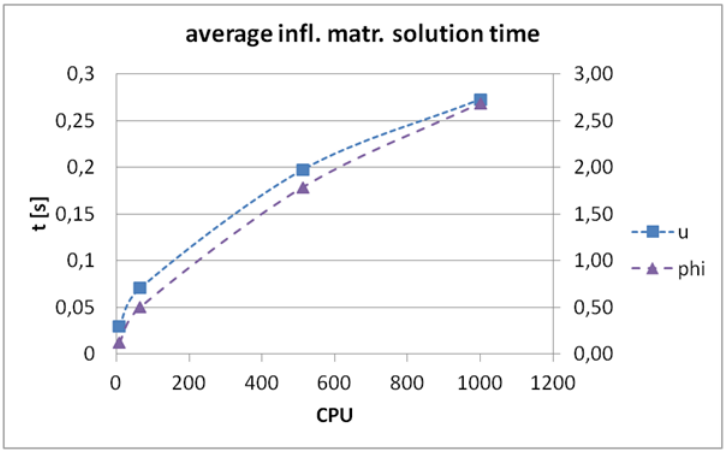


Figure 6.49: Weak scaling results for average time to obtain influence matrix solution for u or Φ. Secondary scale refers to Φ.

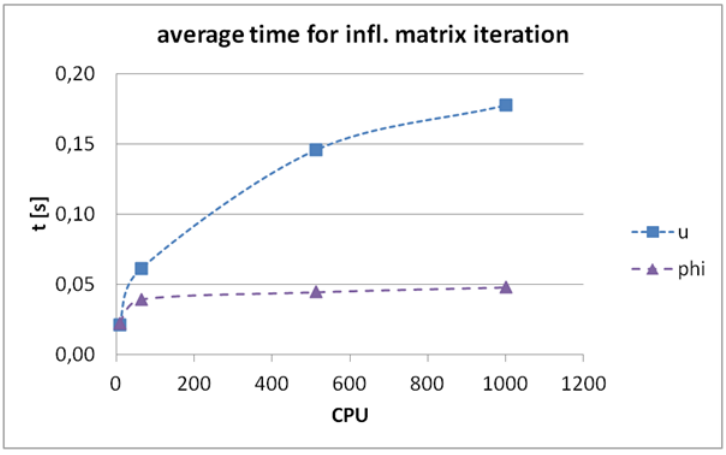


Figure 6.50: Weak scaling results for average time of an iteration in influence matrix solution for u or Φ.

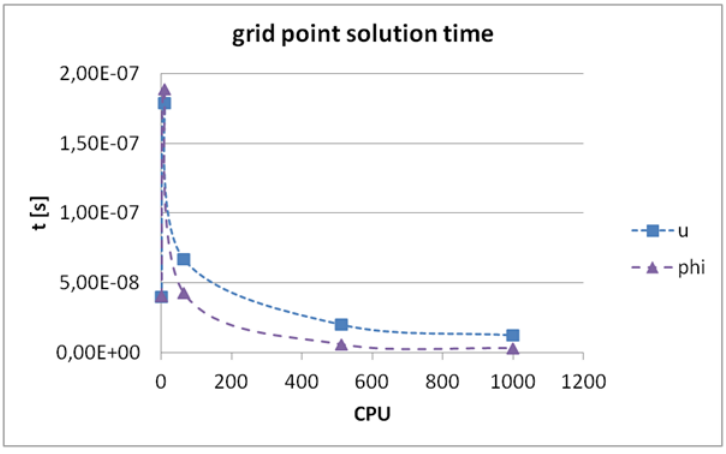


Figure 6.51: Weak scaling results for average time needed per grid point to perform an influence matrix solution iteration for u or Φ. First point shows average time for a point solution in mono domain case.

The statement about influence matrix solutions having poor convergence in case of Φ should be better discussed. In chapter about new algorithm development, more precisely in section on Φ solution issues 2.2, are some paragraphs referred to the compatibility condition problem in Poisson equation. It is said that if compatibility is not ensured, the Poisson-Neumann problem either starts to demand more iterations from an iterative solver or makes it impossible for a direct one to obtain results. Regarding the use of influence matrix, the paragraphs refer to [87, 113], where it was revealed that compatibility issues are shown through the influence matrix. Which gives the reason why iterative solver used for it performs more and more iterations, does not achieve convergent solution and possibly even obtains discontinuous gradients across interfaces. But here, the problems with ensuring compatibility do not seem possible at first. MMS case with original code is used, where known velocities from analytical flow expressions are imposed on boundaries. Hence the compatibility as discussed in mentioned section 2.2 should be satisfied and influence matrix solutions should express good convergence. However, it is also stressed that compatibility issues discussed there concern continuous compatibility condition, which depends on boundary conditions, while compatibility should be also satisfied on the discrete level. And it is this discrete level compatibility which raised concerns and is considered as the cause of poor Φ influence matrix solution convergence. One reason for it is the difference between compact schemes, used for general first and second derivatives, and the schemes, used for second derivatives in Laplace operator. It can be seen in sections dealing with them in chapter 3 that the second derivatives, as used in the Laplace operator for the mono domain solver, do not follow from first derivatives, used to derive $\vec{v}$, $\vec{v}^*$ and Φ. They are defined directly and can have different stencils in same points. According to theory in [123, 124, 107], combination of such schemes does not ensure discrete compatibility and is considered as a reason for increased iterations amount in solving Poisson-Neumann problem. The other reason is in compact schemes themselves, which are not ensured to respect conservative formulation for the first derivative. Looking into works such as [87, 104, 95] and especially to the basic presentation of compact schemes in [94], it can be noted that these schemes do not conserve certain quality globally on their own. However, no treatment was applied to compact schemes in original MFLOPS-3D regarding this, therefore another point for problems caused by discrete compatibility is raised. Combined with the transfer of compatibility issues to the influence matrix solution stated in [87, 113] and inclusion of discrete compatibility treatment in those works, it was assumed the problem of increased amount of iterations for Φ influence matrix solution follows from insufficient ensuring of discrete compatibility. Especially as no considerable increase is observed for velocity in both weak and strong scaling. The problems could well lie also somewhere else, but only this possibility was studied a bit more at the end of the thesis. The reason is in the late discovery of this issue and following lack of time to try different improvements. The results connected to this study will be given in the following chapter.

The poor convergence for Φ is also mentioned as the cause for increased errors or L_2 norms in the most refined multi domain cases in MMS verification tests of original code. As these feature more sub domains, iterations for Φ in them increase in accordance to here observed results. Although the iterations have a finite number, meaning some converged result is achieved, the question is how good is this result. Namely, the increase in convergence criteria of Krylov solver for Φ influence matrix from 10^{-5} to 10^{-8} did show decrease in L_2 norms, though the decrease was still not monotonic. Importantly, amount of iterations was up to four times bigger, but the limit, set at 10000 iterations, was still not reached. This shows that probably the results converge only to a certain level at which Krylov solver or PETSc then stops performing iterations. Where the convergence is satisfactory for less refined grids or in cases with less sub domains, it becomes insufficient for the two most refined grids. This then causes observed increase in L_2 norms of Φ.

5.1.3 Conclusions from scaling results

The conclusions from both strong and weak scaling results, which can be restated at this point, are:

- The influence matrix solution determines the performance of the code as it presents the most time to obtain a certain solution.
- Regarding the performance of mono domain solver, use of large sub domains would be desirable. But for them, the influence matrix tends to perform worse and express higher difference in efficiency to solve for velocities or Φ. Smaller sub domains return similarly fast solution for both variables, but demand too many CPU. Therefore a compromise to use domains with approx. 25 points per direction was applied.
- For such size of sub domains, the computational time grows considerably with increase in their amount. This follows from the increase in amount of Φ influence matrix iterations when more sub domains are used and the solution time demanded for them in comparison to velocity influence matrix iterations.
- Increase in iterations for Φ shows weak convergence of Φ influence matrix solution. For this, lack of well ensured discrete compatibility is assumed to be the reason.
- As continuous compatibility is ensured through boundary conditions in here used tests, ensuring of discrete compatibility could be the key to improving the performance. Following theory, this should decrease amount of iterations for Φ influence matrix solution. Since the time it takes to perform one such iteration does not grow with number of sub domains, better performance should follow.
- Moreover, it is concluded that the increase in L_2 norms for Φ on most refined grids in VOA for incompressible algorithm is caused by the mentioned weak convergence. Hence its improvement is a crucial goal in providing efficient tool for DNS simulations.

5.2 Old and new algorithm performance comparison through incompressible flow case

The strong and weak scaling tests already reveal all the most important characteristics regarding performance of the code and its solvers. However, as the new algorithm applies many changes, especially in form of iterative procedures such as inner iterations, which feature CG method, and outer iterations, which include all projection method steps, its performance is severely affected. It is therefore important to define how it compares to the original algorithm. Since the comparison follows from results obtained in VOA tests, the reference point is set by defining the performance of the code and original algorithm in incompressible MMS flow case used for VOA. For this reason, comparison of the computational times in mono and multi domain simulations is considered at first. Since multi domain computations show great effect of influence matrix solution, the average iterations done to obtain these solutions are also stated. These results are then compared to the ones obtained from VOA of the new algorithm in the same incompressible flow case. Such comparison gives the most direct definition of how the new algorithm performs to the original one. As all mentioned results and comparisons are based on using incompressible flow case, they are all included in this section.

First, computational times for original algorithm are presented. These are given as average time to obtain complete time step solution. In contrast to before presented weak and strong scaling results, which regard only use of mono or multi domain solvers, these times therefore include all steps performed in an algorithm. Regarding them, an important distinction has to be made. Contrary to all other results presented in this work, these computational times do not follow from results obtained on Ada cluster. It was found that the mono domain calculations with the original code on this cluster express extremely long computational times. The reason for them is unknown, especially since the times were even longer than those for mono domain simulations done with new algorithm in cavitating flow case, where both mentioned inner and outer iterations take place. The scaling results presented before are however not affected by this, since the issues do not come from mono or multi domain solver. These solvers actually report computational times in accordance to performed weak and strong scaling tests in all simulations on Ada cluster. Because of these particular issues, the comparison of computational times for incompressible flow algorithm in mono and multi domain cases here rather follows from results obtained on another cluster. This is HPC Prelog [133], which is used by Faculty of Mechanical Engineering in Ljubljana, Slovenia. This cluster features performance wise very similar processors (Xeon X5670), but a lower capabilities for parallel simulations with higher amount of CPU since Ethernet is used for communication between the nodes. These have 12 CPU each. Because it is a much smaller cluster than Ada it was used extensively for algorithm development while all otherwise here reported convergence tests were done on Ada. Even mono domain ones, as they express issues only for original algorithm. Nevertheless, a few tests were performed also on HPC Prelog, including here used ones with original algorithm. The results for mono and multi domain computational times from them are given on figure 6.52.

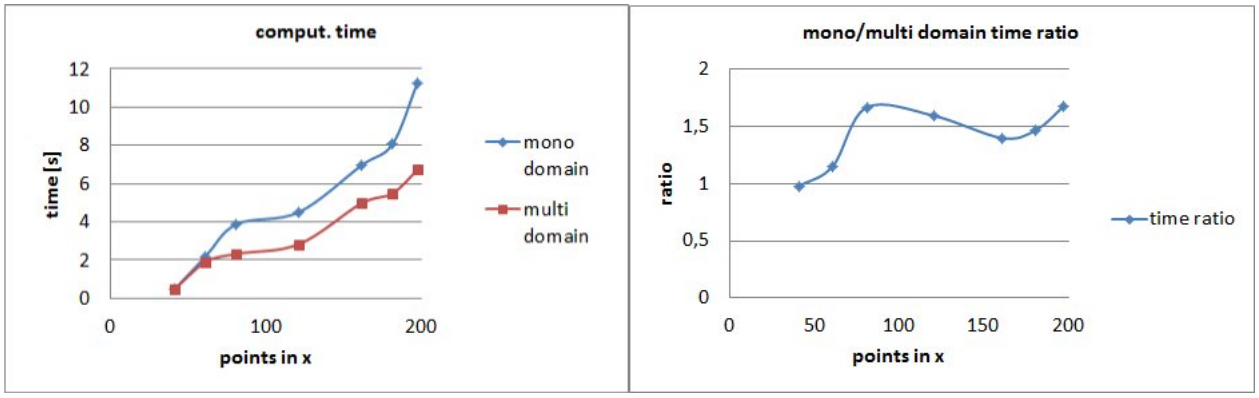


Figure 6.52: Graphs showing average computational times to obtain one complete time step solution in mono and multi domain calculations with original algorithm (left) and the ratios between these times (right).

It can be seen that computational times are equal at first. With increase in amount of total points, where sub domains keep the points in the mentioned most practical range (approximately 25 per direction), multi domain calculations expectedly become faster. The difference is however not a large one, which agrees with the before given results in strong scaling cases. Since all other results follow from calculations on Ada, it seems appropriate to show also problematic results obtained on this cluster. These results are given on figure 6.53 in same form as results on figure 6.52. The difference to mono domain results obtained on HPC Prelog is obvious. The multi domain computational times also show differences. The reported times are not equal but on

same order of magnitude, which is not an issue considering differences in processors and node communication. However, same form of the curve for multi domain computational time would be expected. Like for the issues in mono domain results, it is also unknown what causes this difference.

Although differences in computational times are noted for multi domain cases performed with incompressible flow algorithm on Ada and HPC Prelog, there are no differences between the two clusters in the reported amount of influence matrix iterations. This is another confirmation that the issues do not come from used solvers. The average amount of influence matrix iterations performed per time step is given on figure 6.54. The results correspond with those given in weak and strong scaling tests. The amount of iterations for u and v velocity is equal and nearly constant, while substantial increase in Φ influence matrix iterations with increase in amount of sub domains is present. Amount of iterations for w velocity is higher than for u or v and increases with amount of sub domains. This was found to be a consequence of w being constantly zero.

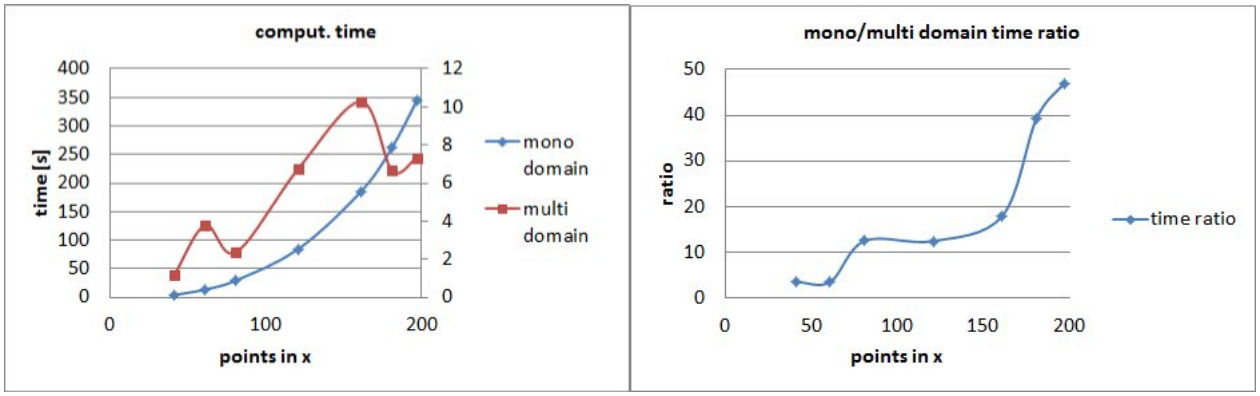


Figure 6.53: Figure presents graphs, equal to those on figure 6.52, but with results obtained on Ada cluster. The values of multi domain average computational times on left graph are given on secondary axis in y.

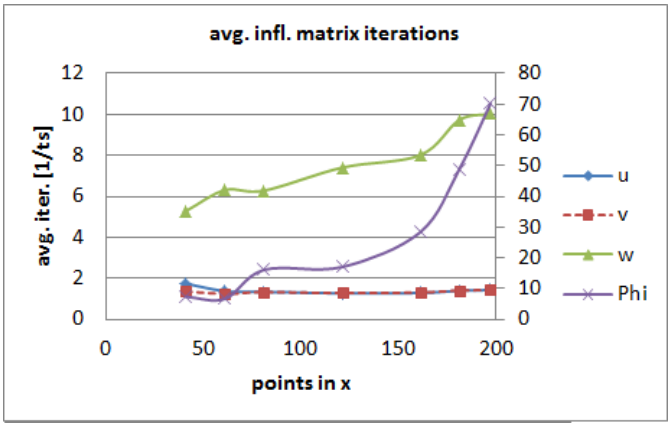


Figure 6.54: Graph shows average influence matrix iterations per time step for all four variables. Φ iterations have their values on the secondary axis.

The presented results, though including the issues for computational time, give the reference point for defining the performance of the new algorithm. For incompressible flow case however, only multi domain calculations with PIUS and PNCS new algorithm versions were done, since

the focus was mostly to confirm the order of accuracy is still on the level of original algorithm. Therefore only multi domain computational times and amount of influence matrix iterations can be compared. Because of the issues with the use of original algorithm on Ada, the incompressible flow calculations with the new algorithm were done on Ada and HPC Prelog. Here, computational times obtained on both clusters are given, but only results from HPC Prelog can be used to directly compare performance of the two algorithms. Results from Ada will be used later, for comparison with cavitating flow case calculations. The average computational times with PIUS and PNCS algorithm versions are given for HPC Prelog on figure 6.55. It can be seen that times in both PIUS and PNCS versions are nearly identical. Same figure also includes a graph showing ratios between these times and those obtained with original algorithm, giving direct comparison of new and old algorithm performance. The results on coarser grids or cases with less sub domains show that new algorithm performs better, while on more refined grids, the new algorithm becomes roughly two times slower than original. Because of this, it can be concluded that the new algorithm is overall up to two times slower than the original in same flow configuration. Results for computational times obtained on Ada are given on figure 6.56. This features also a graph showing the ratio between computational times on Ada and HPC Prelog. Regarding the computational times, it can be seen that no issues, like those observed with the original algorithm, are present. Curve for computational times follows the shape of the one obtained on Prelog, there are no large swings present. It is also obvious that Ada returns faster multi domain calculations. Graph with the ratios shows that Ada is 2 to 2,5 times faster. Such difference was noted only for the case of multi domain calculations, confirming the effect of different node communication speed.

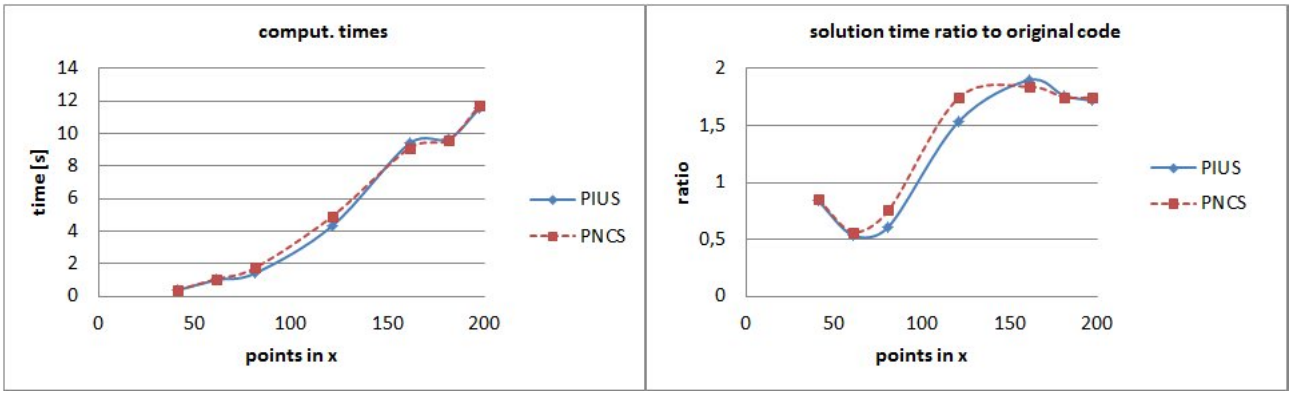


Figure 6.55: Left graph shows average computational times obtained with PIUS and PNCS versions in incompressible flow case on HPC Prelog. Right graph shows ratios between these times and those for same calculations with original algorithm on HPC Prelog.

A reason why the new algorithm is up to two times slower than the original one can be given with discussion on iterations done in it and their comparison with the ones in original algorithm. As described, new algorithm performs inner and outer iterations. Only one outer iteration was performed in this case, since continuity was satisfied in it already. However, inner or CG iterations are performed in comparison to original algorithm. Although they are here not needed, they were not prevented (regarding them, new algorithm was left to run in described manner). Because of them, it is also better to give average amount of influence matrix iterations as average per one CG iteration. Therefore the average number of CG iterations per time step should also be shown. This average amount of CG iterations for velocities and Φ is given with graphs on figure 6.57, while corresponding average amount of influence matrix iterations per one CG iteration is

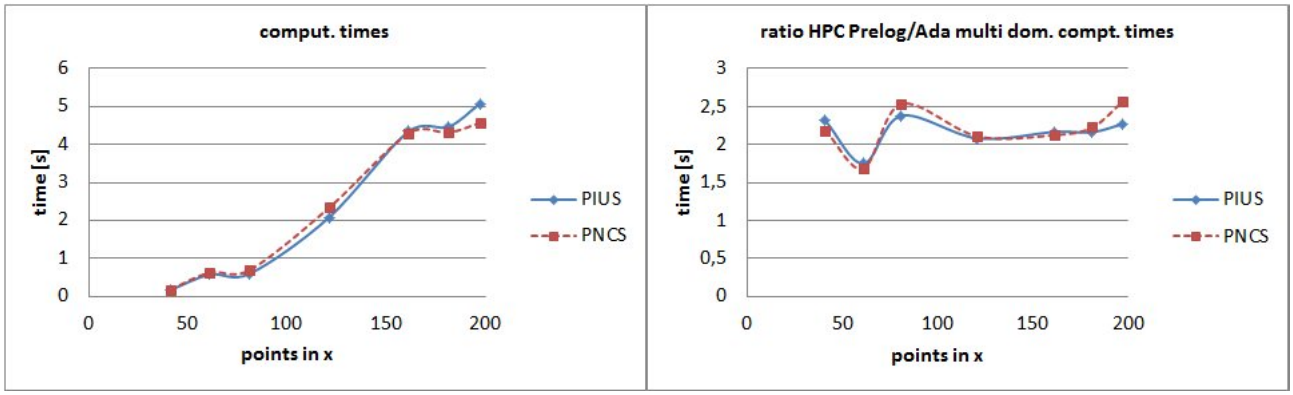


Figure 6.56: Left graph shows average computational times obtained with PIUS and PNCS versions in incompressible flow case on Ada. Right graph shows ratios between these times and those for same calculations on HPC Prelog.

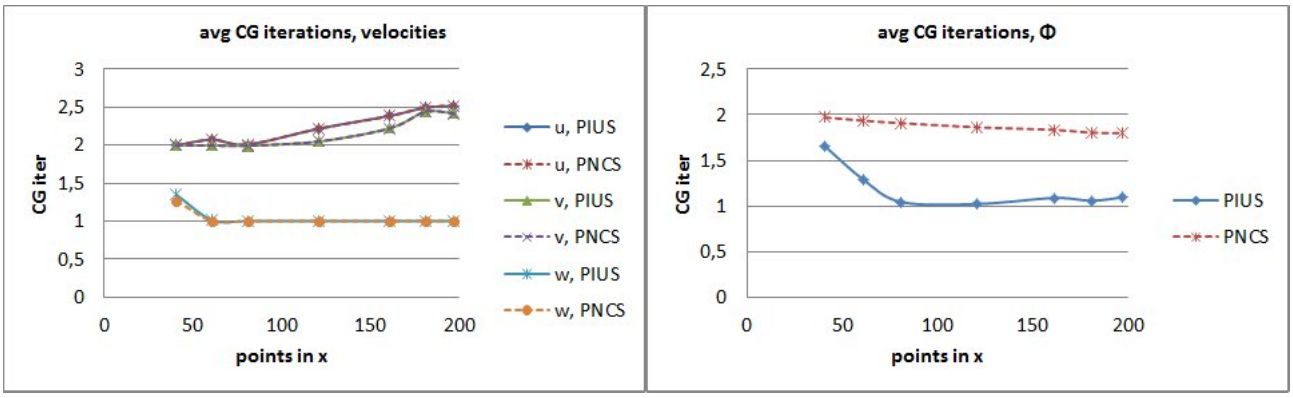


Figure 6.57: Left graph shows average amount of CG iterations for velocities per time step with PIUS and PNCS algorithm versions in incompressible flow case. Graph on the right shows same information for Φ solution.

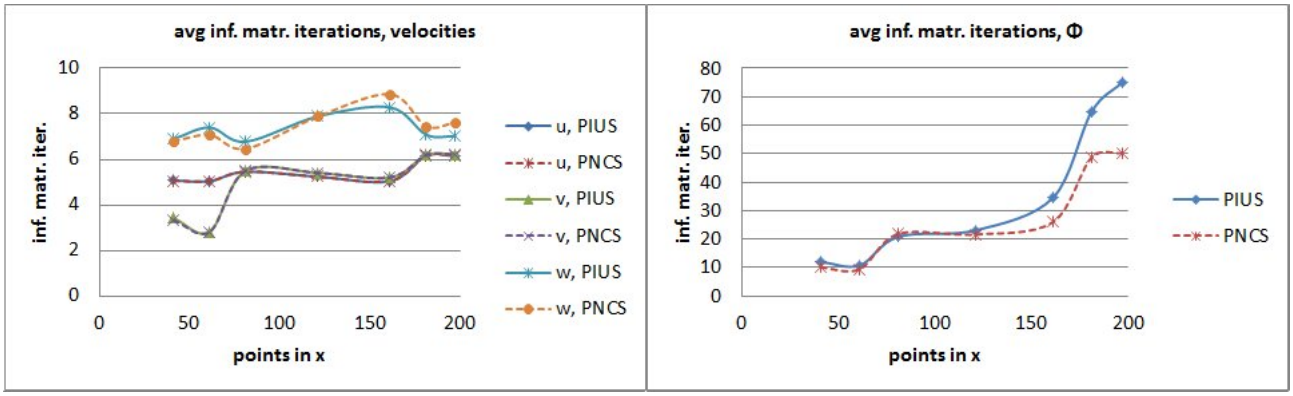


Figure 6.58: Left graph shows average influence matrix solution iterations for velocities performed per one CG iteration with PIUS and PNCS algorithm versions in incompressible flow case. Graph on the right shows same information for Φ solution.

given on figure 6.58. Importantly, calculations on Ada and HPC Prelog returned same results for these iterations. Graphs on figure 6.57 show that more than one CG iteration is done in general. This is not surprising as even in incompressible flow case, the imposed convergence criteria for CG iterations needs to be met and the first CG iteration compares its results with the last ones from previous time step. However, the convergence criteria is quickly met and only around two CG iterations are done at most. On the other hand, this means that the mono and multi domain solvers are applied up to two times more often than in the original algorithm, wich well agrees with up to two times longer simulations reported on figure 6.55. The actual ratio is however not defined only through the performed CG iterations, but also with influence matrix solution iterations. Results for these on figure 6.58 show that despite using mixed boundary conditions, the amount of iterations for Φ per one CG iteration does not drop much. Which shows that relaxation of compatibility condition does not help much with the amount of iterations done for Φ influence matrix solution. PIUS version actually returns same amount as the original algorithm while PNCS version reports less iterations on most refined grids. Considering higher amount of CG iterations done for Φ solution in this version, this decrease is in the end not making a difference. On the other hand, u and v velocities return same amount of iterations for both PIUS and PNCS versions, which is up to 6 times higher than in the case of original algorithm. Iterations for w velocity are otherwise in same range for both algorithms.

5.2.1 Conclusions from direct comparison of old and new algorithm computational performance

To conclude the presentation of direct performance comparison of new and original algorithm, it can be stated that:

- Regardless of the version, the new algorithm performs up to two times slower than original one in incompressible flow simulations.
- The main reason is in increased use of mono and multi domain solver, demanded by the performed CG iterations. Around two iterations are done at most, which corresponds well with the computational time ratio.
- The influence matrix iterations show considerable increase for the two non zero velocities.
- High amount of influence matrix iterations is still done for Φ solution despite the use of mixed boundary conditions and relaxed compatibility constraint. It follows that issues with this solution persist also in the case of new code.

5.3 Performance of new algorithm in MMS cavitating flow case

As for the incompressible flow cases, the discussion of new algorithm computational performance follows from calculations done in VOA. These were performed using all four versions of the algorithm in mono and multi domain configurations. Therefore performance comparison between different versions and their effect on differences between mono and multi domain calculations can be presented.

Firstly, average times to obtain a time step solution using all four versions of the algorithm in mono and multi domain calculations are shown on figure 6.59. As in the VOA, the two pressure non incremental versions return practically identical results. Interestingly, in mono domain cases, these two versions show worst performance, while situation is inverse in multi

domain cases. There, PIUS version shows worst performance with PICS being in between it and the two non incremental versions, which are fastest. Since multi domain simulations are the ones to be performed practically, the fastest simulations are therefore obtained with either of the two pressure non incremental versions. Regarding these results, some additional remarks can be also given. It was stated before that original algorithm returned problematic results regarding computational times on Ada. All results given here were obtained on Ada, but they show no issues, as was also the case for incompressible flow calculations with new algorithm on Ada. All computational time curves in multi domain cases again correspond well with the shape of the curve obtained in that incompressible flow case, there are no huge swings as observed for the original algorithm. On the other hand, the mono domain cases return curves with shape corresponding to the problematic one for use of original algorithm on Ada. However, as it was stated before, these times are still lower than the problematic ones, despite the cavitating case returning in general slower simulations.

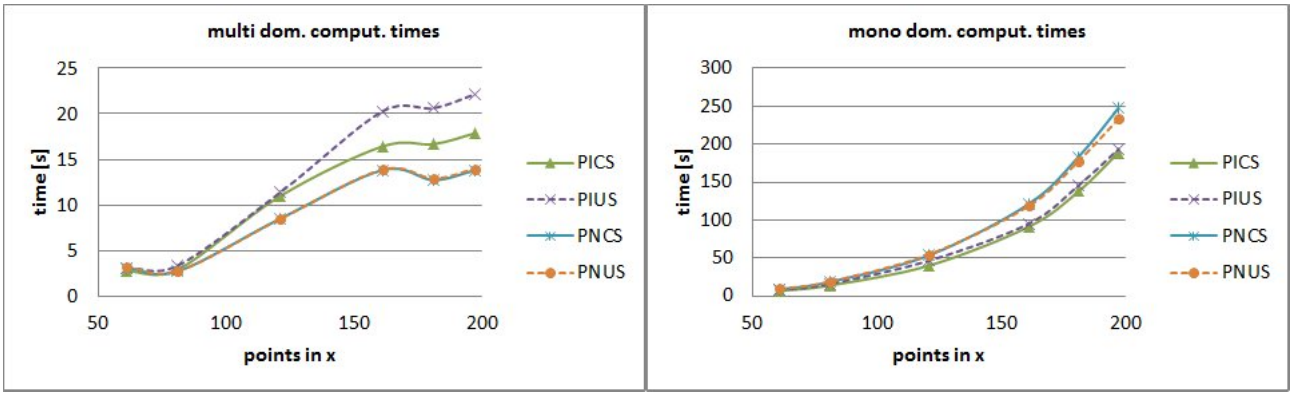


Figure 6.59: Average computational times for a time step solution for all four versions of the algorithm in multi (left) and mono (right) domain computations.

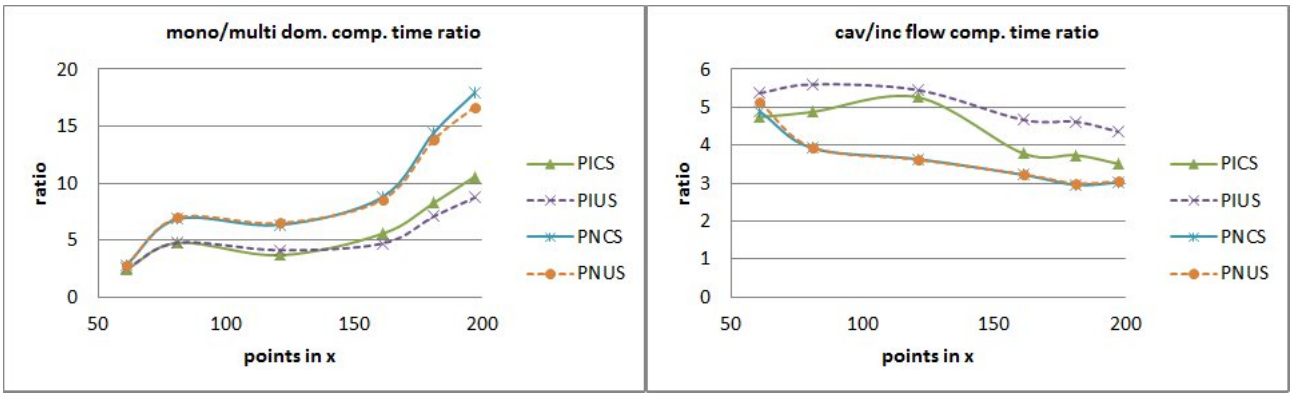


Figure 6.60: Left graphs shows ratio between mono and multi domain cavitating flow computational times for all four new algorithm versions. Right graph gives a ratio between cavitating and incompressible flow computational times in multi domain cases.

Shown computational times can be used to give some performance comparisons. At first, mono and multi domain computational times can be compared to establish the ratio between the two and compare it with the ratio for original algorithm. On the other hand, times for multi domain computations can be compared with those obtained with the PIUS and PNCS versions in

incompressible flow case computations done on Ada. The two comparisons are given with graphs on figure 6.60. The ratio between mono and multi domain computational times is obtained by dividing the times from corresponding versions of the algorithm. The ratio between cavitating and incompressible flow multi domain computational times is obtained by dividing results of non incremental versions with results for PNCS version and incremental versions' results with results for PIUS version in incompressible flow.

Time ratio between mono and multi domain computations is for new algorithm in cavitating flow $5-10$ times higher than the one observed for original algorithm on figure 6.52. This is not an issue and the reasons for it will be given later with discussion on performed CG iterations. The ratio between multi domain computational times in cavitating and incompressible flow simulations is more important, and it can be seen that cavitating flow simulations are less than 6 times slower than incompressible flow ones. The difference also becomes smaller with increase in sub domains number and is generally the worst for PIUS version. The best or lowest difference is obtained for pressure non incremental versions, in accordance to their shortest computational times shown on figure 6.59. Regarding these, only 3 times longer simulations are done for cavitating flow case on most refined grids. A comparison with original algorithm can be made too, although only indirectly. It was shown in previous section that the new algorithm is up to two times slower than original one in same incompressible flow simulations. Therefore the new algorithm in cavitating flow can be from 6 to 12 times slower than original algorithm. Which is a large drop in performance. But it is also the worst drop possible. This follows from the fact that complete α range is present in the performed cavitating flow simulations, with source term being non zero globally. This results in increased CG iterations for both velocities and Φ solutions. On the other hand, two to three outer iterations are also performed opposing to one in incompressible flow simulations. It therefore follows that the shown ratios between performance of the new algorithm in cavitating flow case and either new or original algorithm in incompressible flow case are the worst possible ones.

The average amount of CG and influence matrix iterations is presented in same manner as before. That is, average CG iterations for all four considered variables are given as average amount done per one time step. For multi domain cases the average influence matrix iterations are given as average amount per one CG iteration. Here, the fact that two to three outer iterations are usually done per one time step is not considered. The reason is that this would make the analysis very complex. On one hand, CG iterations for certain variable normally drop with outer iterations. At the same time, influence matrix iterations express drop with performed CG iterations. On the other hand, this is not a strict rule, since the sudden pressure changes or α having an instantaneous range of $\alpha = \{0; 1\}$ can cause CG iterations not to drop with outer iterations. And same can apply for influence matrix iterations. For the two most refined grids, these can also on general change less between CG iterations. Therefore it is impractical to present CG and influence matrix iterations in connection with outer iterations. However, it should be stated that most of the computational time usually falls to the first outer iteration (more than half as a rule). The average amounts of CG iterations for velocities and Φ in both multi and mono domain cavitating case calculations are given on figures 6.61–6.64. Corresponding average amounts of influence matrix iterations in multi domain calculations are given all on figure 6.65.

It can be seen all variables demand nearly same amount of CG iterations in mono and multi domain calculations. The only exception is that all variables bar w velocity show some increase of CG iterations on two most refined grids in multi domain calculations. u velocity is the extreme case, with iterations growing for more than two times. The actual cause of this was not defined, but it seems it is somehow connected with Φ influence matrix solution, which expresses same

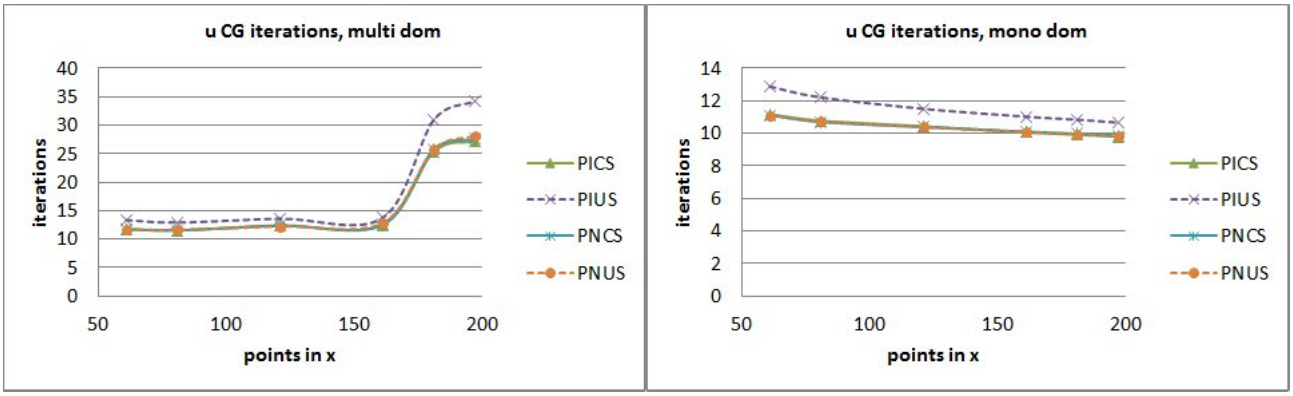


Figure 6.61: Graphs showing average amounts of CG iterations for u velocity done in multi (left) and mono (right) domain cavitating case calculations with all four new algorithm versions.

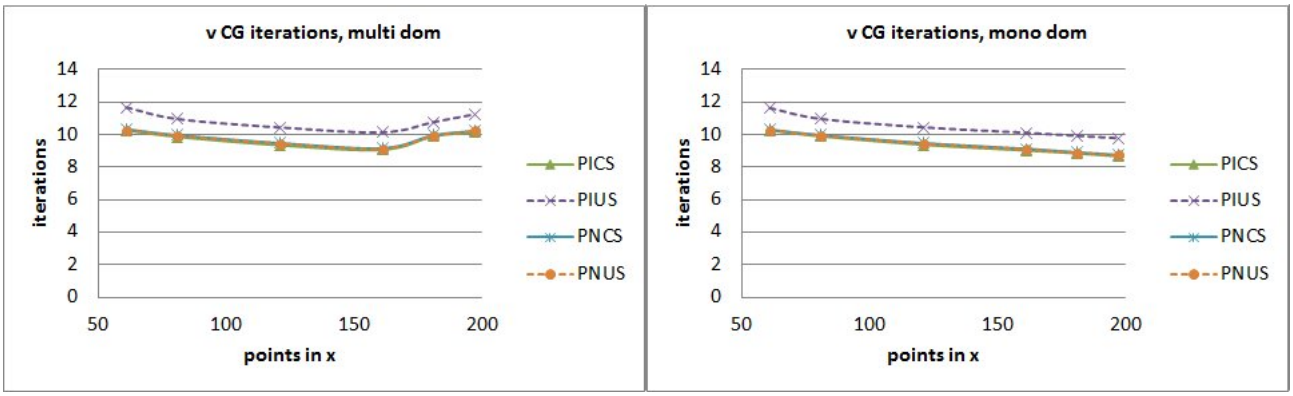


Figure 6.62: Graphs showing average amounts of CG iterations for v velocity done in multi (left) and mono (right) domain cavitating case calculations with all four new algorithm versions.

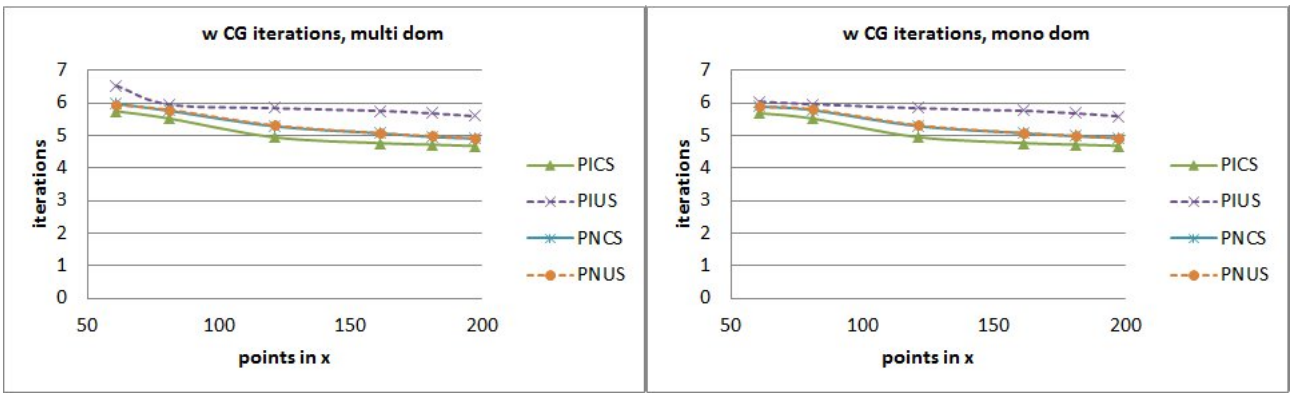


Figure 6.63: Graphs showing average amounts of CG iterations for w velocity done in multi (left) and mono (right) domain cavitating case calculations with all four new algorithm versions.

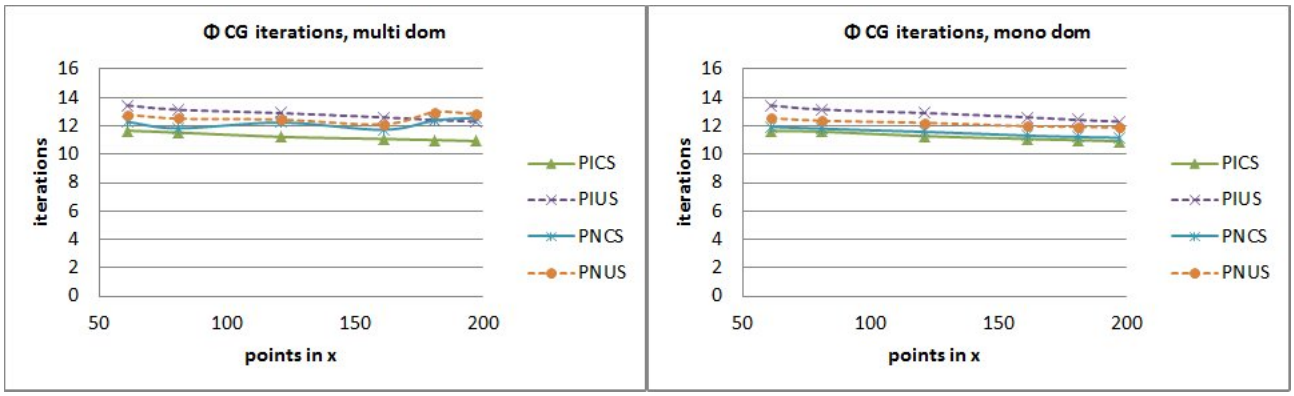


Figure 6.64: Graphs showing average amounts of CG iterations for Φ done in multi (left) and mono (right) domain cavitating case calculations with all four new algorithm versions.

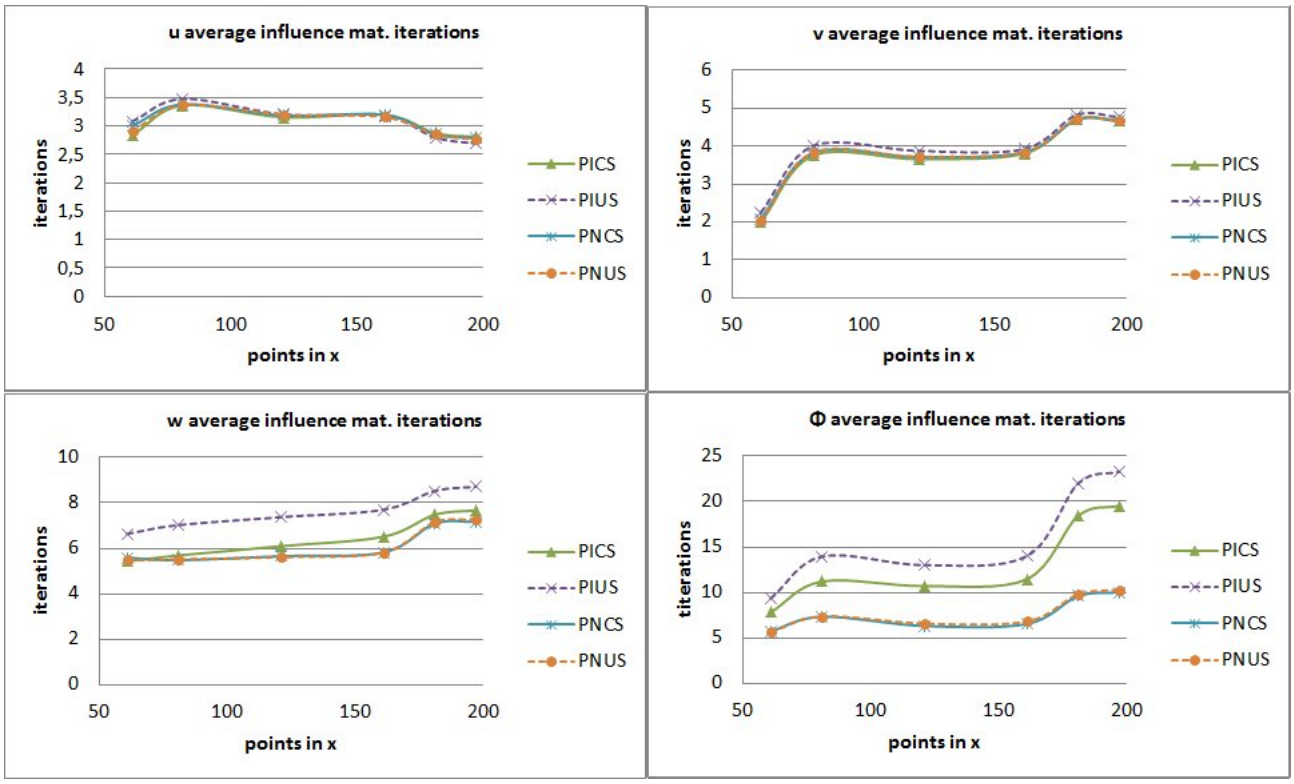


Figure 6.65: Graphs showing average amounts of influence matrix iterations for all four considered variables in multi domain calculations of cavitating case with the four new algorithm versions.

increase in iterations on most refined grids no matter the considered case or algorithm. Nevertheless, good correspondence of average amount of CG iterations in mono and multi domain cases is obvious and shows no issues with the algorithm itself. PIUS version makes most iterations for all four variables while PICS demands least iterations for w velocity and Φ. Otherwise PICS, PNCS and PNUS versions return same amount of CG iterations. This shows that algorithm is capable to converge in equally good manner in all versions. Considering the amounts of CG iterations, it follows that excluding observed increase on most refined grids, iterations slowly decrease with increase in amount of sub domains or mesh refinement. Approximately $10 - 13$ CG iterations are done for u velocity, which is 5 to 6 times more compared to the CG iterations done for incompressible flow case. Situation is very similar for v velocity, which shows $8 - 12$ CG iterations are done for cavitating flow case, returning roughly same ratio compared to incompressible flow. w velocity returns lowest amount of iterations, 5 to 6 are done on average. Since only one CG iteration for w is done in incompressible flow, the ratio between the two is again the same as before. Which leads to the conclusion that for the new algorithm, it takes up to 6 times longer to obtain velocity solutions in the case of cavitating flow than in incompressible flow. Regarding CG iterations for Φ, there are $11 - 14$ iterations done, meaning from 7 to nearly 14 times more CG iterations are done than for the incompressible flow case. Finally, compared to incompressible flow computations with original algorithm, where equivalent of one CG iteration is done for each variable, this means even higher difference. Again, this might seem as a huge increase, but as it was said before, α varies between zero and unity, while S is also constantly present. This makes considered cavitating flow case to be highly demanding on CG iterations, presenting the worst possible performance. Therefore here reported increase is not considered as a problematic one. It also explains the mentioned differences in computational times given on figure 6.60. There shown ratio between computational times in mono and multi domain simulations is $5 - 10$ times higher than for the original algorithm. Since many CG iterations are done for each variable, it is normal that the difference between mono and multi domain computational times should be increased, as both mono and multi domain solvers are applied correspondingly more often. Same applies also for time ratio between multi domain calculations of cavitating and incompressible flow case with the new algorithm. This shows less than 6 times increased computational time in cavitating flow case, which again agrees well with the stated increase in CG iterations for velocities and Φ. The increase is however not completely direct with CG iterations. This follows from two reasons. Firstly, the average amount of influence matrix iterations is for certain variables much different from that in incompressible flow. And secondly, these iterations can vary (decrease) with CG iterations. Average influence matrix iterations per one CG iteration are presented on figure 6.65.

Comparing shown average influence matrix solution iterations, it follows that for w and Φ, two pressure incremental algorithm versions demand higher amount of iterations. Otherwise the average amount of iterations agrees well between different versions. This, combined with the highest amount of CG iterations noted for PIUS version, also explains why this version reports longest computational times in multi domain cases. On the other hand, PICS version returns second longest times, despite making the least CG iterations for w or Φ. Reason is in effect shown in weak scaling, that amount of influence matrix iterations defines computational time. The amount of these iterations is second highest for PICS version, hence the second longest computational times in multi domain cases. This observations however do not explain why the two pressure incremental versions return shortest computational times in mono domain cases. The reason for these remains unknown. Regarding the before mentioned different amount of influence matrix iterations compared to incompressible flow case, it can be seen that approximately 3 iterations are done for u velocity. This is almost half less than in incompressible flow calculations

with new algorithm. But three times more than with original algorithm. For v velocity, difference between new algorithm in cavitating and incompressible flow is smaller, as only one less influence matrix iteration is done in cavitating flow cases. That is, 2 to 5 iterations are done for cavitating flow. Which is also the increase compared to original algorithm. For w velocity, all versions except PIUS report slightly lower amount of iterations in cavitating flow than in incompressible. However, as the original algorithm reports amount of iterations in same range, it follows that in all cases, algorithms report similar amount of iterations for w. This is expected and correct result, since w velocity is always zero. The only difference regarding w velocity is therefore higher amount of CG iterations done in cavitating flow. Finally and most interestingly, Φ influence matrix iterations show a considerable drop compared to incompressible flow calculations with both new and original algorithm. From 5 at least to 25 iterations at most are done, compared to 10 to 75 in incompressible flow case. As stated, most iterations are done with two pressure incremental versions. Considerable increase can be noted for two most refined grids for all versions, showing presence of discussed issues with Φ influence matrix solution also in cavitating flow case. The lower overall amount of iterations in cavitating flow case was verified to not be a consequence of much increased amount of CG iterations, which could lead to the presented results with drop in influence matrix iterations. Instead, it was for instance noticed that PIUS version on the most refined grid does approximately 30 influence matrix iterations per CG iteration at most. Same was observed for all four versions of the algorithm. Therefore the reason for the decrease is somewhere else. It was found that the reason is in non zero σ_p constant, used for CG iterations. This changes Φ solution from Poisson to Helmholtz problem, which increases convergence of the multi domain solver. Hence lower amount of Φ influence matrix iterations is reported.

5.3.1 Conclusions

The whole chapter about verification and performance tests is with these last presented results at the end. Conclusions from them, combined with some before stated ones, present here also the closure for this part of the thesis. It can be first said that regarding the differences between them, the four versions of the new algorithm show quite similar computational performance results. Other conclusions are:

- PIUS version seems to be the slowest in practical, multi domain computations. PNCS and PNUS versions are fastest in them, while PICS version is between other three.

- Despite being slower, PIUS and PICS versions are currently the most appropriate choice for actual cavitating flow simulations. The reason is in their higher orders of accuracy.

- Among these two versions, PICS is the more practical choice as it has better computational performance, slightly higher stability than other versions in tests with increased factors and orders of accuracy close to PIUS version.

- Non incremental versions show potential for real flow simulations in having better computational performance and not imposing numerical boundary layer. But they should be first proved able to return higher orders of accuracy. A test case, avoiding issues with Dirichlet boundary condition for Φ, has to be devised for this.

- Pressure non incremental versions in performance tests also retain a characteristic strongly expressed in VOA. Both versions return practically same results in these tests too.

- Considering all new algorithm versions in cavitating flow case, the amounts of CG iterations done in comparison to incompressible flow case increase by roughly $5-6$ times for velocities and $7-14$ times for Φ. Influence matrix iterations drop slightly for velocities, but considerably for Φ. This returns less than 6 times slower simulations for cavitating than incompressible flow case.
- Considering original algorithm, the increase in computational time is even bigger. This is however not taken as an issue because the cavitating flow test case is defined in a manner to invoke a lot of iterations to be performed.
- The difference in real flow cases should therefore be smaller, but remain in the range of ratios which are on one side set between new and original algorithm in incompressible flow case and on the other between new algorithm in cavitating flow case and original algorithm.
- Importantly, the shown computational times can be made shorter with improvement of Φ influence matrix solution. Although this reports less iterations, for which the change of Φ equation from Poisson to Helmholtz problem is responsible, there still exists a considerable increase in them for two most refined grids. Which shows issues with Φ influence matrix solution remain present also in this case.

Chapter 7

Towards DNS simulations

Although new algorithm was developed and verified, actual DNS simulations of turbulent cavitating flow were with it in the end not performed. The reason for this is in the obstacles that use of MFLOPS-3D imposed. The goal of this chapter is to present them briefly and offer either developed or proposed solutions for them. It should be stressed however, that the issues do not come from the new algorithm, but are noticed in the original version of the code. Not the very first one, using spectral methods, but the one which served as a basis for development of the new algorithm. The issues were consequently transferred to the code with new algorithm. One of the issues, the increased amount of iterations for Φ influence matrix solution with increased amount of sub domains, was mentioned in previous chapter. The reason for it was only roughly discussed. There is another issue, in our opinion connected with this increase in iterations, which is the appearance of oscillations leading to unstable results. Finally, when simulations in the chosen venturi geometry were conducted, serious issues with mapping were revealed. These and their solution are also the first to be presented.

1 Instabilities caused by mapping and their removal

As chapters 1 and 2 show, the cavitating flow is intended to be simulated in one of the most widely used venturi test sections, often applied for validation and also verification of developed codes for cavitating flow simulations. This is the venturi with $18° - 8°$ converging-diverging part. The height and the width of its inlet channel is 5 mm, which is conditioned by the mentioned experiments with X-rays. The venturi test section demands use of mapping, which was explained in chapter 3, section 5. An example of mesh applied to it is given in figure 7.1 for an illustration. The shown and in simulations used domain has periodic boundary conditions in z direction, where the chosen width was 5 mm, same as the inlet height. It was found that mapping, applied to this domain as presented in section 5, leads to completely unstable simulations. This was a surprise, since mapping was successfully used in MFLOPS-3D with spectral methods to simulate turbulent flow with adverse pressure gradient. Results can be seen in [89, 90, 78]. Indeed the geometry in this case is more difficult for stable simulations as it has sharper edges than the smooth bump used in those calculations. However, the problem was not caused by presence of sharp edges, as calculations immediately diverged, even if almost stationary flow was applied with extremely low CFL number. Smooth edges were also tried and caused no change. Example of results, obtained after only two time steps for the laminar flow case with maximum flow velocity of $u_{max} = 0,1\ m/s$, is given on figure 7.2.

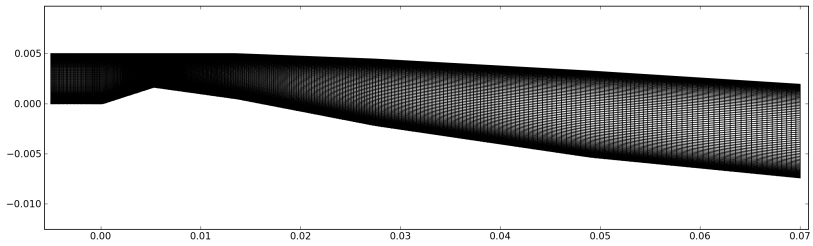


Figure 7.1: Example of studied venturi test section and mesh applied to it. The inlet is 5 mm high.

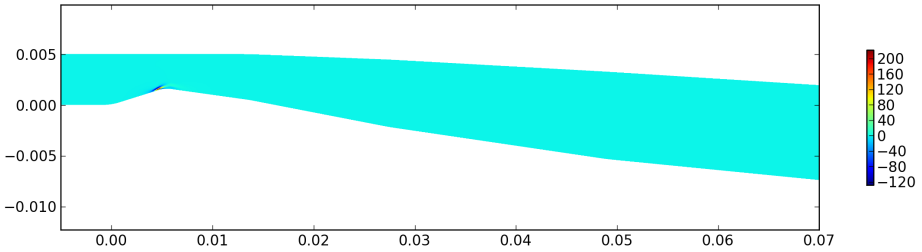


Figure 7.2: Example of problematic results obtained for the venturi test section after only two time steps. Velocity is given in [m/s].

Looking back to section 5 of chapter 3, it can be restated that the only physical coordinate changing with mapping is $\bar{y}$. The physical $\bar{x}$ and Cartesian x coordinates show same horizontal position of points as the mesh lines in y or $\bar{y}$ are aligned. The coordinate $\bar{y}$ depends on the lower and upper domain limits $\bar{\eta_1}$ and $\bar{\eta_2}$. These depend on their position in $\bar{x}$ or equally x. Therefore mapping effects depend on distance between $\bar{\eta_1}$, $\bar{\eta_2}$ limits and gradients of their change with x. The Venturi test section limits have same ratio between throat and inlet height as those used in before mentioned successful simulations with bump geometry (2/3). Even their change gradients are in the same range. Therefore the issues with mapping were even bigger surprise and the only explanation is they are caused by different discretization methods and following mono and multi domain solver changes. However, it was noted that if the venturi has a larger throat section and thus larger minimal distance between $\bar{\eta_1}$, $\bar{\eta_2}$, stable simulations are possible. This led to the discovery that the L_η operator is the root cause.

The L_η operator describes the terms which appear in Laplacian operator with mapping and are different from pure Cartesian derivatives. The operator is here restated for with equations (7.1)−(7.4). Importantly, and as explained in section 5, none of its terms can be treated implicitly. Equations for $\vec{v}^*$ and Φ are therefore adapted to include this operator on RHS, as shown with equations (3.104) and (3.105).

$$L_\eta = L_{\eta,x} + L_{\eta,y} + L_{\eta,z} \tag{7.1}$$

$$L_{\eta,x} = \frac{\partial^2}{\partial y^2}\left(\frac{\partial y}{\partial \bar{x}}\right)^2 + 2\frac{\partial x}{\partial \bar{x}}\frac{\partial y}{\partial \bar{x}}\frac{\partial^2}{\partial x \partial y} \tag{7.2}$$

$$L_{\eta,y} = \frac{\partial^2}{\partial y^2}\left(\left(\frac{\partial y}{\partial \bar{y}}\right)^2 - 1\right) \tag{7.3}$$

$$L_{\eta,z} = 0 \tag{7.4}$$

Since stable simulations are possible in venturi configurations with a larger throat and mapping effects depend exactly on the distance between upper and lower domain limit, it was assumed that at least one of the terms in L_η becomes too high, causing either $\vec{v}^*$ or Φ solution to become too explicit and unstable. This term was found to be $L_{\eta,y}$ and the troubled equation was found to be equation for Φ. To explain why this is so, the connection between physical and Cartesian coordinates $\bar{y}$, y is given at first with equation (7.5). The y_b and y_a values in it are the upper and bottom limit of mapped, rectangular geometry.

$$y = \frac{\bar{y}\,(y_b - y_a) + \bar{\eta}_2 y_a - \bar{\eta}_1 y_b}{\bar{\eta}_2 - \bar{\eta}_1} \tag{7.5}$$

If equation (7.5) is put into (7.3), equation (7.6) follows and reveals that $L_{\eta,y}$ term magnitude depends on the local ratio between mapped domain limits y_b, y_a and actual domain limits $\bar{\eta}_1$, $\bar{\eta}_2$. The factor following from this ratio is denoted as NC. This then multiplies the second derivative in Cartesian coordinates, meaning that the whole $L_{\eta,y}$ term can become higher than the implicitly treated Cartesian second derivative in y. Here it can be added that the whole $L_{\eta,y}$ expression ensures that that same implicitly treated Cartesian second derivative is multiplied only by unity. Therefore Laplace operator in Cartesian coordinates is included on the LHS of equations with terms multiplied only by unit factors, which enables its eigendecomposition. And thus the use of described direct mono domain solver. Considering a simple venturi geometry shown on figure 7.3, with same ratio between throat and inlet heights of 2/3, NC term amplitude variation can be easily shown. This is plotted on same figure to show the term becomes the highest in the region where the flow is the most demanding to solve. This was also found to be the main reason why mapping returned highly unstable simulations in case of considered venturi test section. The amplitude of NC and therefore explicitly treated second derivative in y becomes higher than amplitude of same, implicitly treated derivative. Equation for $\vec{v}^*$ accepts this without any issues but Φ solution returns completely wrong results and causes unstable simulations. The negative values shown on figure 7.3 follow from L_η inclusion on the RHS of equations for $\vec{v}$ and Φ.

$$L_{\eta,y} = \frac{\partial^2}{\partial y^2}\left(\left(\frac{y_b - y_a}{\bar{\eta}_2 - \bar{\eta}_1}\right)^2 - 1\right) = NC\frac{\partial^2}{\partial y^2} \tag{7.6}$$

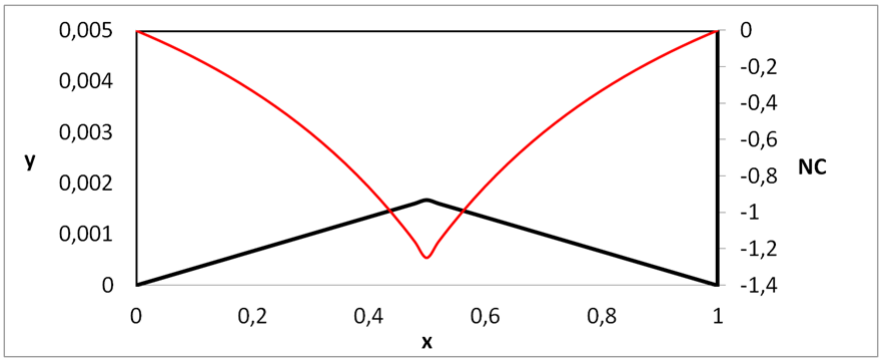


Figure 7.3: Example of a simple venturi geometry to show NC factor amplitude changes. The venturi geometry is given with black lines while the NC factor amplitude is given with the red line and measured on the secondary (right) scale.

The problem was solved in a quite simple manner. Since highest amplitude of NC is achieved at the throat or narrowest part, the most important thing is to try and make it smaller there. The equation for Φ, here repeated with equation (7.7), has to be changed in a manner to keep its LHS constant but diminish the explicitly treated $L_{\eta,y}$. Equation comes from the original algorithm since this development was done and tested with it.

$$\Delta\Phi = \frac{3}{2\Delta t}\left(\nabla\cdot\vec{v}^* + \vec{G}_\eta\cdot\vec{v}^*\right) - L_\eta\Phi_e \tag{7.7}$$

The goal of diminishing explicit $L_{\eta,y}$ was achieved by multiplying equation (7.7) with factor 2, thus introducing a solution for double Φ or Φ_D as shown with equation (7.8). Multiplying whole equation by 2, the double divergence of $\vec{v}^*$ on left hand side can be split into complete Laplacian in physical coordinates and single divergence of $\vec{v}^*$, as shown with equation (7.9). Assuming equality of Φ and Φ_e (in converged solution), the two terms with L_η operators can be joined, leaving equation again with $-L_\eta$ term. However, the added $\Delta\Phi$ term (practically $\Delta\Phi_e$) now hides one second derivative in y, which can be added to $L_{\eta,y}$ and shift the amplitude of NC by one. This effect on NC is shown on figure 7.4, where new amplitude of NC can be compared to the previous one. Final equation for double Φ is given in (7.10).

$$2\Delta\Phi = \Delta\Phi_D = 2\left(\frac{3}{2\Delta t}\left(\nabla\cdot\vec{v}^* + \vec{G}_\eta\cdot\vec{v}^*\right) - L_\eta\Phi_e\right) \tag{7.8}$$

$$\Delta\Phi_D = \Delta\Phi + L_\eta\Phi + \frac{3}{2\Delta t}\left(\nabla\cdot\vec{v}^* + \vec{G}_\eta\cdot\vec{v}^*\right) - 2L_\eta\Phi_e \tag{7.9}$$

$$\Delta\Phi_D = \Delta\Phi_e + \frac{3}{2\Delta t}\left(\nabla\cdot\vec{v}^* + \vec{G}_\eta\cdot\vec{v}^*\right) - L_\eta\Phi_e \tag{7.10}$$

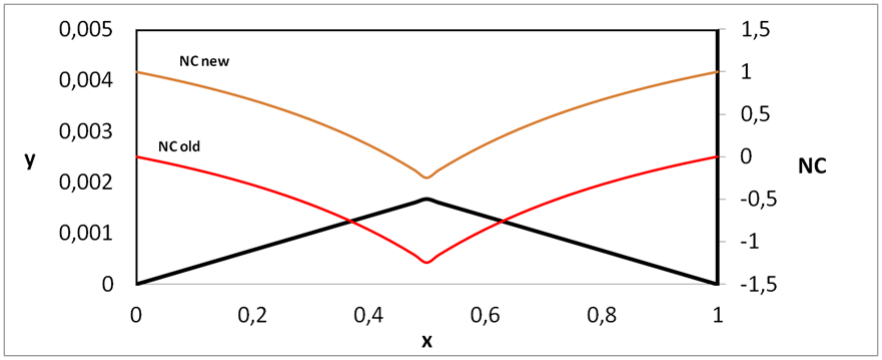


Figure 7.4: Example of a simple venturi geometry showing changed NC factor amplitude. The red line shows NC term before doubling Φ equation, the orange one gives NC amplitude after.

Figure 7.4 clearly shows that with this treatment, the NC amplitude and with it problematic explicitly treated second derivative is smallest at the throat. NC does grow towards the inlet and outlet or generally regions where mapped and real geometry have same height. However, these regions represent easier flow conditions. With this treatment, stable simulations in Venturi test section or any other kind of Venturi geometry with as narrow throat were obtained. For example, figure 7.5 shows the simulations of laminar flow in chosen test section and following results obtained with increasing velocity and turbulence on the inlet. This example is taken from simulations which were used to initialize the flow in the computational domain with accelerating

the velocity at the inflow. At the same time, results from later considered LES simulations in periodic channel were imposed on the inlet in order to gradually replace laminar flow with realistic turbulent flow. This is the reason the velocity profile on the bottom image follows turbulent one and some minor turbulent forms can be observed at the inlet.

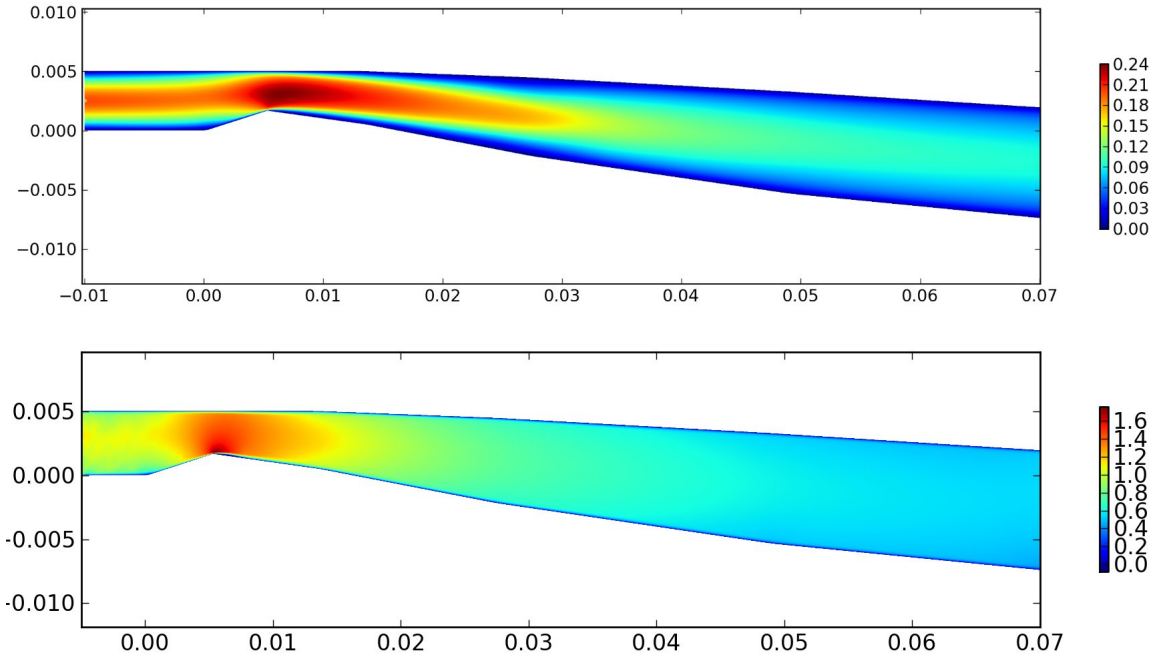


Figure 7.5: Example of stable flow simulations in venturi geometry obtained after solution for Φ_D was introduced. First image shows laminar flow results, while second shows the results obtained with increasing velocity and turbulence on the inlet. The domain in second case is slightly shorter than in the first. Velocity is given in $[m/s]$.

It should be mentioned that solution for Φ_D did not impose additional issues. The boundary conditions were adapted with simply multiplying the before defined ones by 2 (in mapping, von Neumann conditions are not homogeneous). After a solution for Φ_D was found, it was divided by 2 and obtained Φ was used in projection equation to gain $\vec{v}^{n+1}$. Same procedure is also applicable for the case of new algorithm, where mixed boundary conditions are used for Φ. Therefore Φ value imposed on the outlet has to be multiplied by 2. Furthermore, it can be argued that same approach could be applied to $\vec{v}^*$ solution and that the applied multiplication by 2 is suitable only for the here used Venturi test section. The procedure was indeed also applied to $\vec{v}^*$ solution but no differences were observed, hence there is no need for it in this case. And it is true that the multiplication by 2 well fits the NC in this geometry, since its amplitude follows from local ratio of mapped and physical channel heights. However, the procedure is universal and in case of different ratio of heights, different value can be used in multiplication to well diminish NC at the throat. Some other factors were actually tried in this case too, and the equation for Φ was after multiplication split in different manners, but here presented multiplication by 2 proved to return best results.

Finally, an important fact not stressed before should be also brought to attention. Application of mapping introduces iterative steps for solutions of any variable already to the original code. As these iterations are in essence done in same manner as CG iterations introduced in the new algorithm, they are in the new code included in CG iterations. Which consequently means that

for flows in venturi geometry, original and new code are expected to exhibit lower differences in computational performance than observed in MMS tests discussed before. As only original code was actually used for simulations of flow in venturi, a good comparison between the two cannot be given. But it was observed that the new algorithm performs roughly twice the amount of CG iterations (all outer iterations combined) for Φ in shown MMS cavitating flow case than original algorithm does iterations for Φ_D in simulating venturi flow. Combined with less influence matrix iterations observed with the new algorithm because of the Poisson equation for Φ being changed into Helmholtz, this could well make the differences between the computational performance of the two codes much smaller than reported in previous chapter. There exist some questions nevertheless, such as which convergence criteria for iterations should be used. Mapping and CG iterations namely use criteria which can be considerably different, especially for $\vec{v}^*$ solution. Or if there are any effects between the use of non zero σ_p and Φ_D. These questions remain open for future work.

2 Issues with channel flow simulations

This section briefly presents the activities regarding real flow simulations and issues met in them. It also serves for illustration why it was assumed that discrete compatibility, mentioned in previous chapter, is the cause of poor influence matrix performance in case of Φ solution. Therefore it also gives an introduction for the next section, where an approach towards reaching discrete compatibility is presented.

The real flow simulations considered in this chapter are all simulations done without MMS. These are periodic or non periodic channel flow and Venturi channel flow simulations. There was a considerable amount of such simulations done. The plan was to simulate periodic turbulent channel flow with LES models to obtain realistic turbulent flow data, which can then be imposed onto the inlet of Venturi geometry. Some results were obtained, however many issues were noted. Because of them, simple laminar flow in non periodic channel was considered and interesting results were found. In the following chapters, the periodic turbulent flow simulations are therefore first presented and followed by the discoveries in the non periodic laminar channel flow. The Venturi results are not presented, since the focus is on issues appearing already in simple channel flow configurations. But same issues appear also in Venturi geometry.

2.1 Turbulent periodic flow simulations

Turbulent periodic flow simulations were done in a channel with periodicity imposed also in z direction. Meaning that also later intended DNS simulations of cavitating flow in Venturi are meant to be performed with periodicity imposed in z direction. The reasons for this are first in same conditions being generally applied to simulations of cavitating flow in a venturi and second in avoiding additional issues with the walls, like boundary layers on them and also secondary flows (Prandtl's secondary flows of second kind). The effects of walls are also questionable, as the boundary layers are thin. Moreover, the experiments as done in [64, 65], with which the results are to be validated, use images from one side of Venturi test section. Therefore wall effects cannot be defined.

As mentioned in chapter 2, the goal Reynolds number based on channel half height is $Re = 16700$. This corresponds to roughly friction Reynolds number of $Re_\tau \approx 850$. MFLOPS-3D code has not yet been run at such high Reynolds numbers, neither were some turbulent channel flows

at such numbers available to us. Since it is much better for DNS simulations to have realistic turbulent flow imposed on the inlet, periodic turbulent flow simulations were performed. Which was also a nice opportunity to try MFLOPS-3D with compact finite differences in all three directions. Original, incompressible flow algorithm was used. LES models, already implemented in the code and used in calculations described in [93], were applied in these simulations to decrease computational costs. Dynamic LES model known as WALE (Wall Adapting Local Eddy-Viscosity) [134] was found to perform best and was chosen to perform all simulations. These were set following instructions described in [77] to perform validation of LES results in turbulent channel flows. The length of the domain was $l = 6h$, height $H = 2h = 0,005\ m$ and width $w = 3h$. The mesh had local refinement in each sub domain applied in x and z directions, while y direction had local and global refinement applied. Local refinement was advised from other users of the code, as it improves the impact of using forward or backward compact schemes at sub domain limits on the accuracy. Global refinement in y is used to ensure appropriate y^+ values at the wall. Same meshing laws ($tanh$) were used for both refinements as shown for local refinement in MMS cases, but δ coefficient was changed. The periodic flow simulations demanded the use of a forcing term to keep the mass flow in the domain constant.

Velocity in LES simulations was gradually increased towards the goal mean velocity of $\bar{u} = 6,71\ m/s$, which corresponds to before mentioned Re number. At first, $Re_\tau = 400$ simulations were conducted, since these results can easily be validated with an existing database following work in [100] and are also used in the mentioned instructions to validate LES results in [77]. This lower friction Reynolds number corresponds to mean velocity of $\bar{u} = 2,84\ m/s$. Turbulent flow in them was initialized by using results from a turbulent channel flow with $Re_\tau = 550$ performed and described in [101]. These results were interpolated to the here used grids with their mean velocity adjusted. After good simulations settings were defined, including enough refined mesh and time step, velocity was increased to achieve $Re_\tau = 550$ again. Same domain was used for this, with even more refined mesh and time step. General simulation settings, including mesh quality, time step length (and CFL number) and amount of sub domains, which returned best results can be seen in table 7.1. The subgrid-scale model constant c of WALE model was chosen as $c = 0,5$. Simulations were run for more than five flow through times.

Table 7.1: Settings for LES periodic turbulent channel flow simulations (WALE model) at $Re_\tau = 400$ and $Re_\tau = 550$.

Re_τ	Δy^+ (max,min)		Δx^+ (max,min)		Δz^+ (max,min)		$\bar{u}$ [m/s]	Δt [μs]	CPU (x,y,z)	points per sub domain
400	14	0,2	37	26	16	10	2,84	1,6	9/6/3	21/27/33
550	21	0,18	31	12	20	8	4,1	0,1	10/7/3	21/27/35

Simulations produced some valid results. For example, obtained mean velocity profiles are given in figure 7.6, where the mean profiles from DNS simulations at same Re_τ are also given. However, the simulations expressed also some issues, mostly in form of oscillations near interfaces, which could even lead to divergent results at some instances. In the presented simulations, these problems were treated with slight variation of Δt and c between consequent simulations. Therefore it is also said that the simulation settings in table 7.1 are only general. But issues became more pronounced with further increase in velocity, leading to useless simulations at $Re_\tau = 850$. Example of good instantaneous results and results where oscillations cause simulation to diverge in case of $Re_\tau = 400$ are given on figure 7.7 with xy plots of u velocity fields.

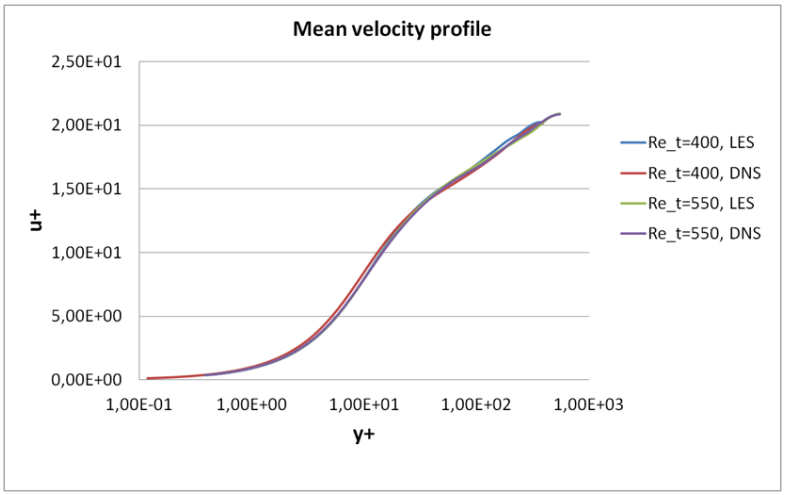


Figure 7.6: Mean velocity profile results obtained in LES simulations of periodic turbulent channel flows with $Re_\tau = 400$ and $Re_\tau = 550$. The DNS results come from [100] for $Re_\tau = 400$ and [101] for $Re_\tau = 550$.

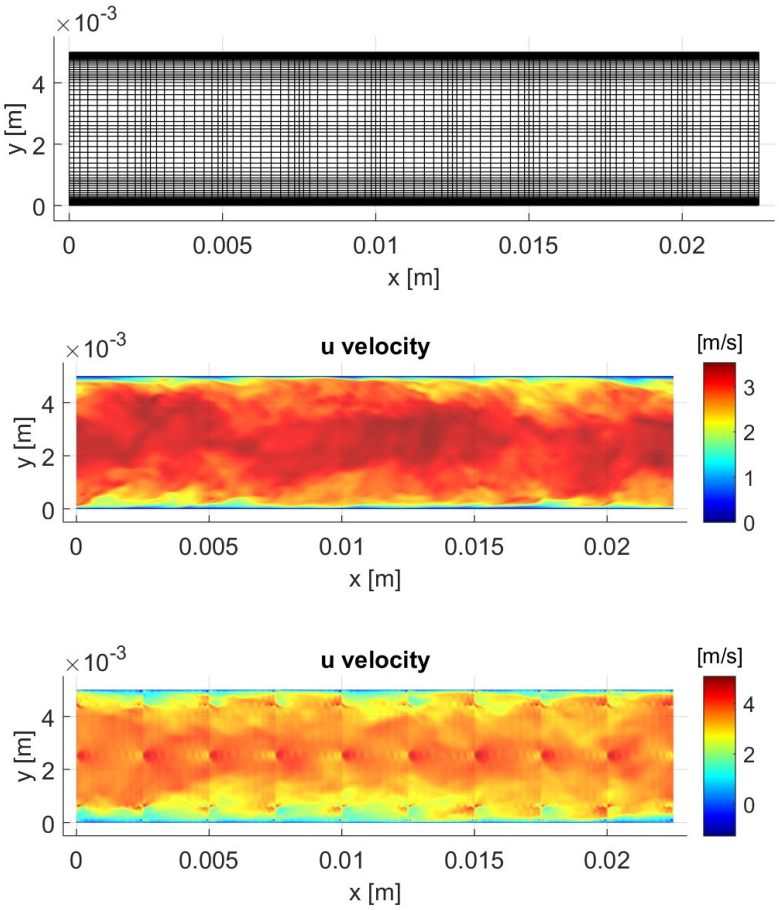


Figure 7.7: Examples of instantaneous u velocity for $Re_\tau = 400$ flow. Top image presents the used mesh with every second line drawn. Middle image presents good results while the bottom one gives an example of developed oscillations for almost same simulation settings as for the good results.

Different causes of these problems were considered. For instance, too short domain in x was ruled out with tests on twice longer grids which returned same issues. Spanwise dimension was also varied and same results were obtained. The mesh was, as it can be seen in table 7.1, enough refined. In fact, following [77], the mesh fits into the most refined grids tried in there performed validation. The problem of using collocated grid was also considered. The odd-even coupling is indeed possible as nearly all points use central schemes. However, from discussions with other users of MFLOPS-3D, this possibility did not receive much support. Therefore staggered grid was not applied in such simulations. Since same pattern of oscillations occurring in vicinity of interfaces, consequent increase in influence matrix iterations for Φ at same time (from average 40 in the two presented cases to twice more) and finally unstable simulations was observed at practically each Re_τ, it was decided to try and run some very basic laminar flow simulations to see if same effect can be captured already in them. In such manner, the issue of kinetic energy not being conserved can also be ruled out. As mentioned in sections about skew symmetric form of non linear terms in NS momentum equations 3.3 and 2.1.1, inability to conserve kinetic energy also leads to unstable simulations. However, as issues, if they appear, in laminar flow can hardly be caused by inability to conserve kinetic energy, the reason must be somewhere else.

2.2 Laminar channel flow simulations

Laminar flow simulations were contrary to turbulent flow performed without periodicity in x. The chosen channel has a square cross section with height of $H = 0,002\ m$. Length varied, depending on the test case performed, between $l = 0,006\ m$ and $l = 0,009\ m$. Incompressible flow algorithm was used, as in cases of turbulent periodic channel flow. Ratio between density and viscosity, otherwise used as Re constant in equations of this code, was set to realistic ratio $\rho/\mu = 995015$ (same was applied in before presented cases). Mean flow velocity used was either $\bar{u} = 0.5\ m/s$ or $\bar{u} = 0.25\ m/s$, resulting in Reynolds number for half channel height of $Re = 500$ or $Re = 250$. Non periodic conditions in x demand use of convective boundary condition from equation (4.45). Convection velocity of $U_c = 0,5$ was used in it (other values were also tried but lead to same conclusions). The flow in whole domain was initialized with parabolic, exact profile for developed laminar flow.

It was observed that oscillations develop even in these basic simulations, pointing towards serious issue in the code. Interestingly, they were found not to appear in directions where only one sub domain was used. Example is given on figure 7.8, where a case with mean flow velocity of $\bar{u} = 0,5\ m/s$ in a channel with length $l = 0,006\ m$ is considered. In the case with one sub domain in stream wise direction, no oscillations are revealed and simulation seems to proceed in a stable manner. In case where two sub domains in same direction are used, oscillations appear and lead to unstable simulations. Both cases used in total 41 points in x, 3 sub domains with 61 points total in y and 2 subdomains with 41 points total in z. Same time step $\Delta t = 2,16\ \mu s$ was used in both and corresponding CFL is $CFL = 0,04$. Additional prove that oscillations appear with multiple sub domains in certain direction is the pressure plot for seemingly stable simulations on figure 7.9. It can be seen that oscillations still appear in pressure, but only in y direction where 3 sub domains are used.

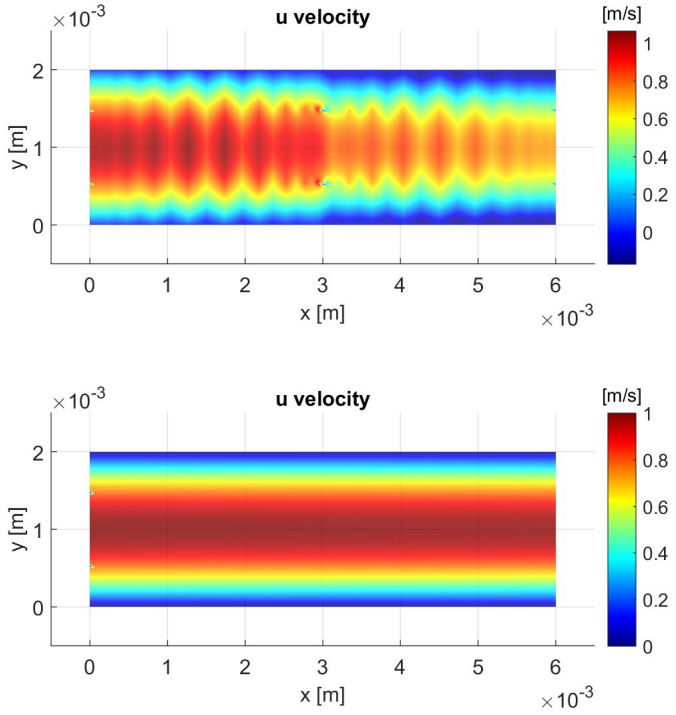


Figure 7.8: Example of oscillations appearing in for the case with two sub domains in x direction and no oscillations appearing for the case of only one sub domain in same direction.

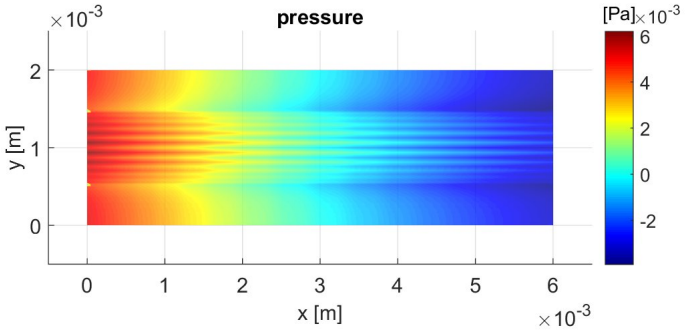


Figure 7.9: Example of oscillations appearing in p plot for the case with one sub domain in x and 3 in y direction. It is obvious oscillations appear only in y.

With this behaviour discovered, it was tried to better locate what could lead to diminishing of oscillations in results. Here considered examples will be given for a domain with $l = 0,009\ m$ length and mean velocity $\bar{u} = 0,25\ m/s$. Domain was split into 3 sub domains in x and y and 2 in z. 5000 time steps with $\Delta = 2,16\mu s$ were done in each, corresponding to $CFL = 0,0165$. Different settings were tried. At first, a reference case was set with the use of unchanged simulations, that is, incompressible flow algorithm as implemented in MFLOPS-3D with homogeneous von Neumann boundary conditions for Φ and convection boundary condition on the outlet as described before. Then, cases with mixed boundary conditions for Φ (in original algorithm), no projection or 2^{nd} order accurate compact schemes were also performed. Since some oscillations were found in all

simulations, two additional cases with staggered grid were performed, one with homogeneous von Neumann and one with mixed boundary conditions for Φ. All cases, differences between them and their notation are given in table 7.2. Staggered grid configuration was implemented in a very simple manner, on the sub domain level only. Two grids were applied in each sub domain. Pressure grid was added an additional point, its points were put in between the usual grid, on which velocity was solved. Only one point needed to be added as interfaces of sub domains on both grids coincide. Therefore the coordinates of first and last point on a grid line in certain direction are the same. Bilinear interpolation was used to translate results from one mesh to another. This was done as otherwise complete class of derivative schemes would have to be defined. Instead, interpolated values can be directly put into compact schemes (explicit stencils) without changing them. Such approach also demands only minor changes to mono and multi domain solver routines, in order to make them operate on both grids.

Table 7.2: Different performed laminar flow cases in order to study appearance of oscillations.

name	mesh	Φ boundary condition	compact schemes order of accuracy	projection performed
REF	collocated	homogeneous von Neumann	4^{th}	yes
NoPr	collocated	homogeneous von Neumann	4^{th}	no
REF2nd	collocated	homogeneous von Neumann	2^{nd}	yes
REFMix	collocated	mixed	4^{th}	yes
STAGh	staggered	homogeneous von Neumann	4^{th}	yes
STAGm	staggered	mixed	4^{th}	yes

Appearance of oscillations in velocity, pressure and velocity divergence field was observed. Instantaneous plots for these variables after 5000 time steps are given on figures 7.10–7.12. On figures 7.10 and 7.11, u velocity and pressure are given with xy and xz plots. Both come from either middle position in z or y. On figure 7.12, velocity divergence on xy plot is given in two ways, including and excluding its values near interfaces. Divergence is namely highest at the interfaces, which hides oscillations that appear inside sub domains.

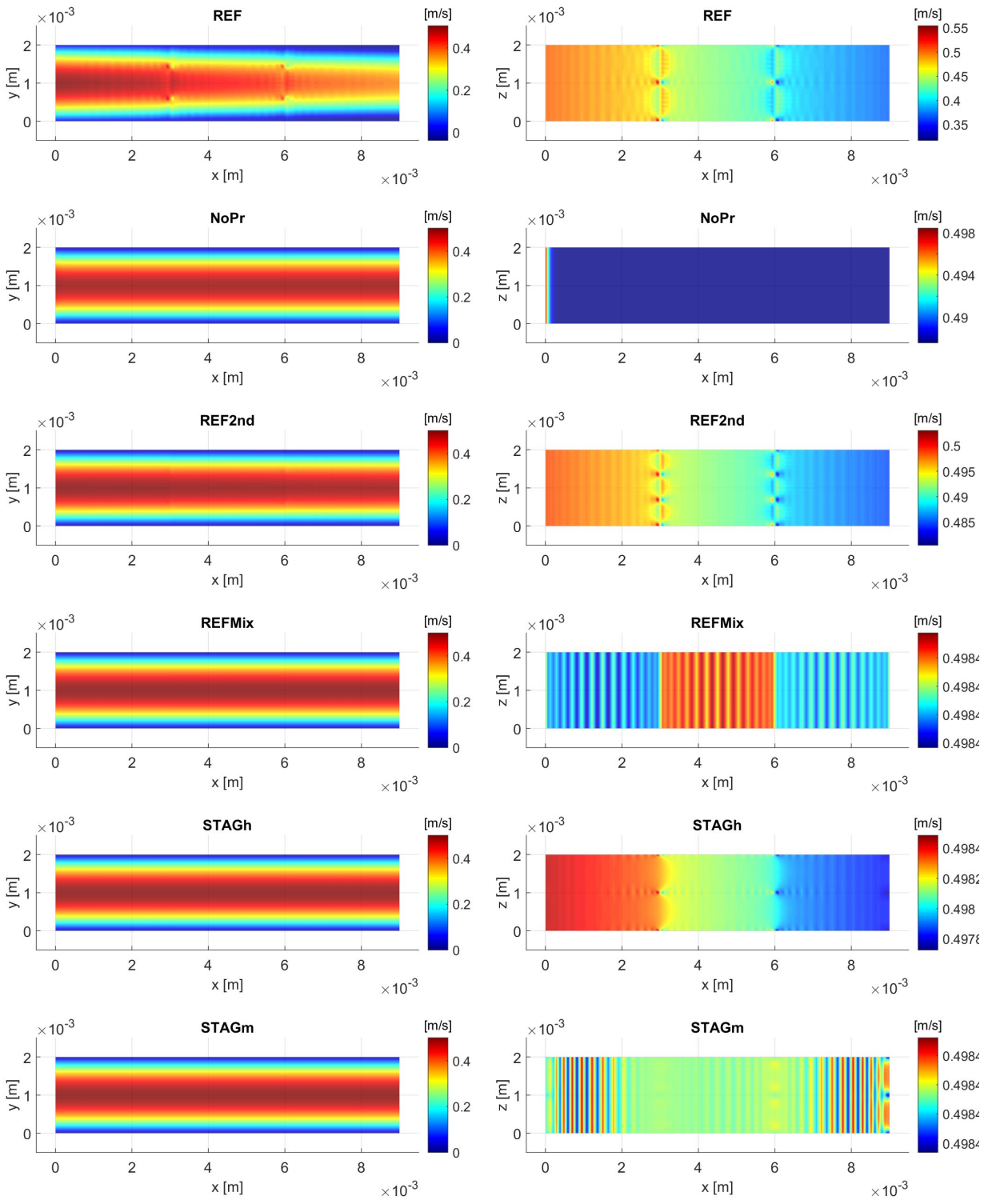


Figure 7.10: Plots of u velocity fields on middle xy (left) and xz (right) planes in cases of different simulation settings. The titles reveal from which case the results came.

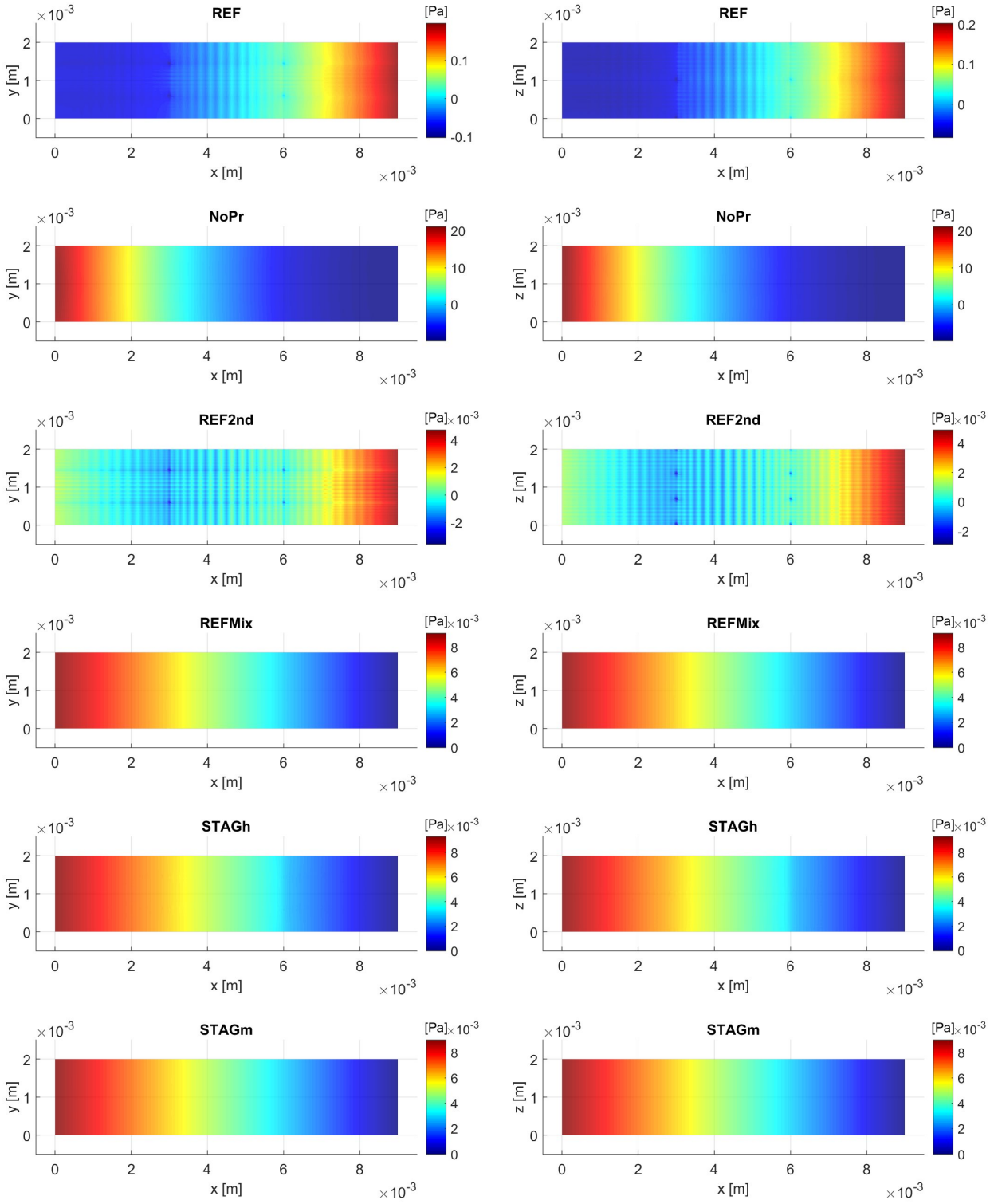


Figure 7.11: Plots of pressure fields on middle xy (left) and xz (right) planes in cases of different simulation settings. The titles reveal from which case the results came.

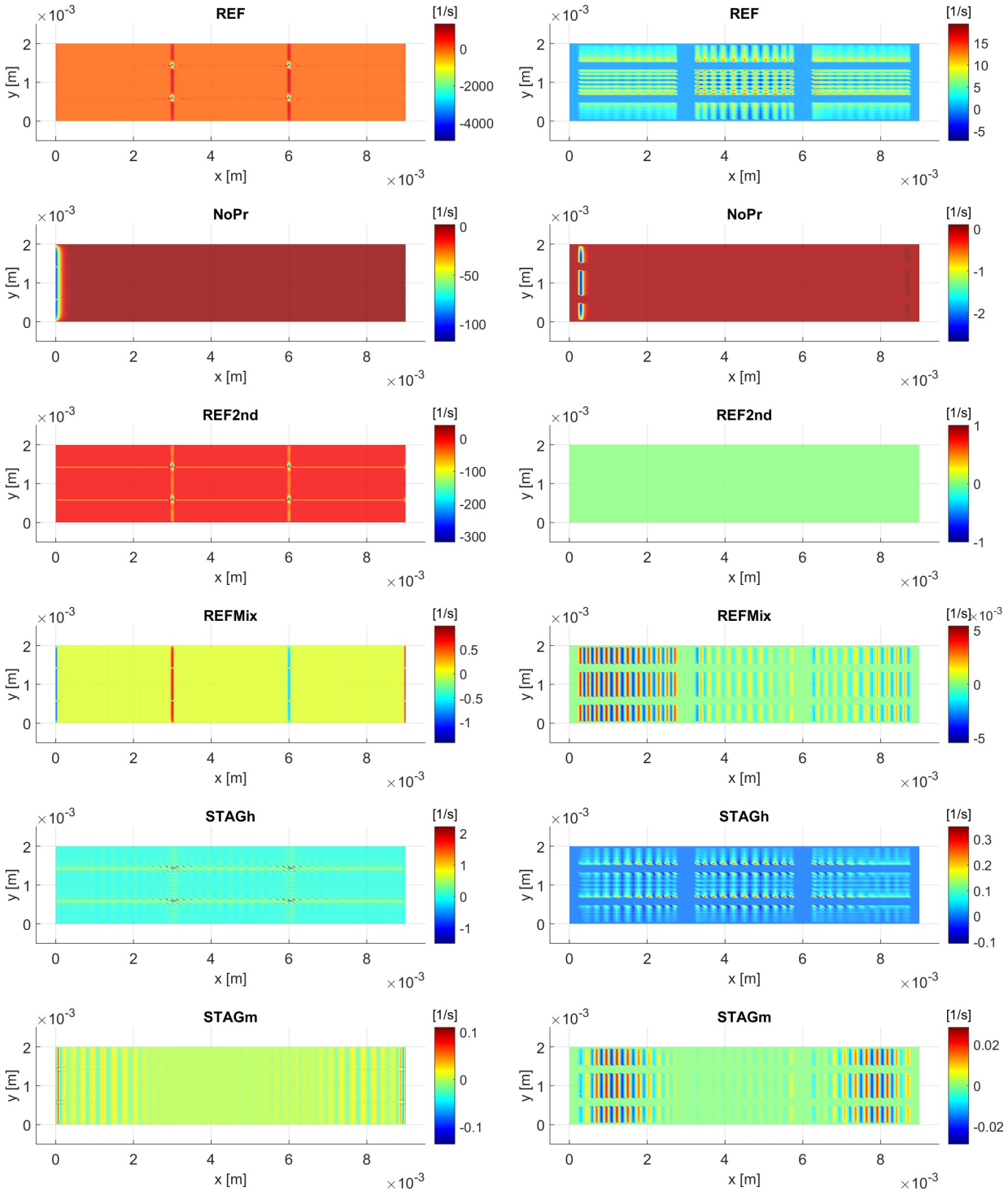


Figure 7.12: Plots of $\nabla \cdot \vec{v}$ fields on middle xy planes in cases of different simulation settings. The left column gives unclipped results while the right one shows results without values in three point on both sides of an interface. The titles reveal from which case the results came.

Figures show that all cases, except the $NoPr$, show oscillations either in one or all three observed variables. Therefore the oscillations follow from the solution for Φ and its subsequent use in projection step. The predictor velocity $\vec{v}^*$ solution shows no issues on its own, showing correct performance of mono and multi domain solvers in this case. The caused oscillations are clearly most pronounced or have certain jumps at interfaces. This, combined with their general presence in all cases, suggests that the multi domain solver or influence matrix solution for Φ is the cause of issues. The worst case is REF, with huge oscillations appearing in all variables. The results are also completely false, with pressure being higher on the outlet than inlet and velocity therefore decelerating towards the outlet. The $REF2nd$ case shows oscillations can be made smaller with the use of lower order schemes. Higher pressure on the outlet however still shows that results are not appropriate. Therefore the cause of issues has to be somewhere else. The leap towards better results is clearly made with the use of either staggered grid or mixed boundary conditions. In all these cases pressure results show correct values and velocity oscillations clearly drop in amplitude, as they can be only seen on xz plots. They are still present in velocity divergence plots, but their amplitude is smaller than in other cases. What is interesting in these results is a clear jump in the base velocity (onto which oscillations are superimposed) across interfaces on xz plots for $REFMix$ and $STAGh$ cases. This coincides with the before given statement from [87, 113], saying that inability to satisfy compatibility condition leads to the Poisson-Neumann singularity problems be expressed through influence matrix solution, which can even lead to discontinuous gradients over the interfaces. This is exactly what happens in these cases, as velocity is obtained through using Φ gradient in the projection step. Proof of this is given with plots of u velocity and Φ gradient in x direction at middle y coordinate. Plots are given on figure 7.13 for cases REF, $REFMix$ and $STAGh$. It can be seen that Φ gradient has a peak at each interface. Since data was saved from only one interface although the two overlap, the actual discontinuous results are not seen in Φ gradient plots. However, their presence is clear with plots of u velocity, which for the two non staggered cases exhibits a considerable jump observed already from xz plots. In case of staggered grid, the jumps happen over a group of points at the interfaces.

These results and experience from developing the new algorithm (large discontinuities observed if homogeneous von Neumann conditions were used) led to conclusion that inability to satisfy compatibility condition is the main cause of oscillations and subsequent instabilities. However, as mentioned in chapter 3 and section about strong and weak scaling tests 5.1 of previous chapter, two kinds of compatibility can be considered. Continuous, defined by boundary conditions, and discrete. Continuous is ruled out in cases with MMS because of exact velocities being imposed on the boundaries. In periodic flows, such as presented before, this is also not an issue. Here however, continuous compatibility can play an important role as it demands equality between inflow and outflow. This was also the reason why mixed boundary conditions for Φ were tried. They do show an improvement as they relax this condition, but not remove it completely. This agrees with discussion given in section 2.2.3.1 of chapter 4, where same effect is argued. To rule out continuous compatibility as a cause completely, another two tests were done, using either homogeneous von Neumann ($REF\ ext$) or mixed boundary conditions ($REFMix\ ext$), together with an exact velocity imposed on the outlet. In such manner, the boundary conditions are ensured to satisfy continuous compatibility condition. Results showed no improvement compared to the results obtained in already shown $REFMix$ case, which is confirmed with xz plots of u velocity on figure 7.14 and graphs of Φ gradient on figure 7.15. Additionally, the graphs on figure 7.16 show the relative differences between inflow and outflow for REF, $REFMix$, $STAGh$ cases and the additional case with homogeneous von Neumann conditions with exact velocity on the outlet ($REF\ ext$). Relative differences are obtained by dividing difference between inflow

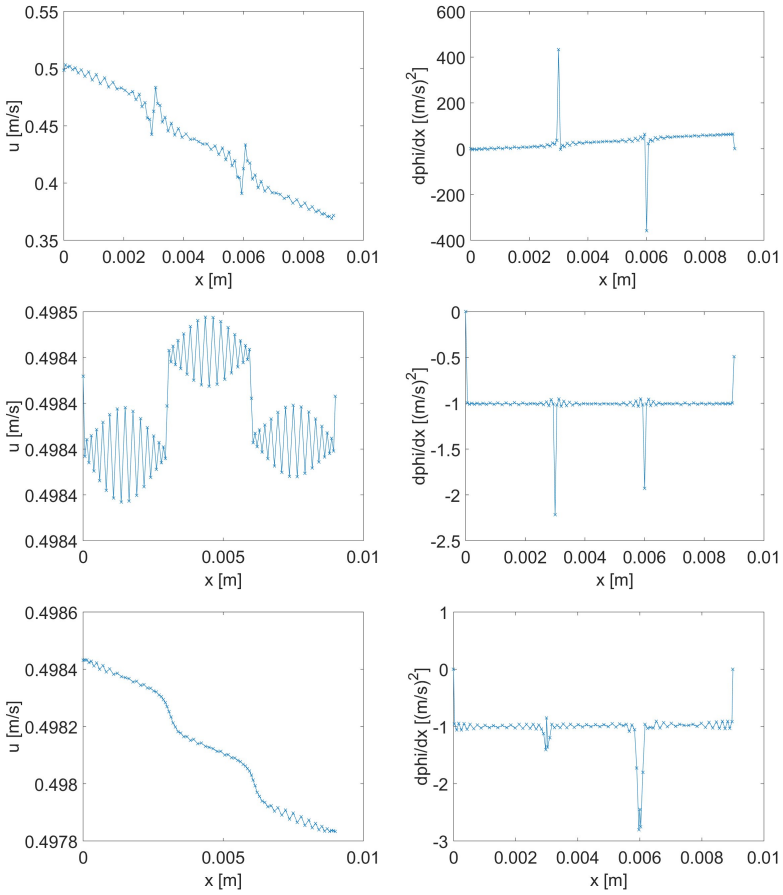


Figure 7.13: Graphs showing u velocity (left) and Φ gradient (right) in x direction at middle y position. Top line shows results for REF, middle for $REFMix$ and bottom for $STAGh$ case.

and outflow surface integrals with channel cross section. It can be seen that REF case violates equality of inflow and outflow greatly, fails to ensure continuous compatibility and hence returns greater instabilities. $STAGh$ ensures continuous compatibility in a better manner without special treatment at the outlet, although the difference grows in time and would probably lead to divergent results as in REF case. Dirichlet conditions for Φ at the outlet cause only small violation of considered compatibility. Exact outflow completely ensures it, yet the results are not better than for $REFMix$ case. It follows that although continuous compatibility is ensured perfectly, results still exhibit issues, most notably discontinuous Φ gradient. Hence the only other option considered is that discrete compatibility is not ensured. Regarding this, velocity divergence in time was also observed and compared to inflow-outflow difference. These two are related through equation (4.44), which is based on Gauss theorem. Results for same four cases are given on figure 7.17, where relative value of velocity divergence volume integral is plotted. This was obtained by dividing the result of divergence volume integral with domain volume. It can be seen that divergence results generally correspond with the difference between inflow and outflow, which is expected. However, $REFMix$ and $STAGh$ cases reveal there are differences between the two quantities. The $STAGh$ case expresses a notable divergence when plotted against the worst case REF, something which cannot be noticed in observing only inflow-outflow difference. And $REFMix$ has a different shape of the curves for two observed quantities. This was taken as a confirmation that there are differences between the two observed quantities, confirming the

possibility of discrete compatibility not being satisfied and causing issues.

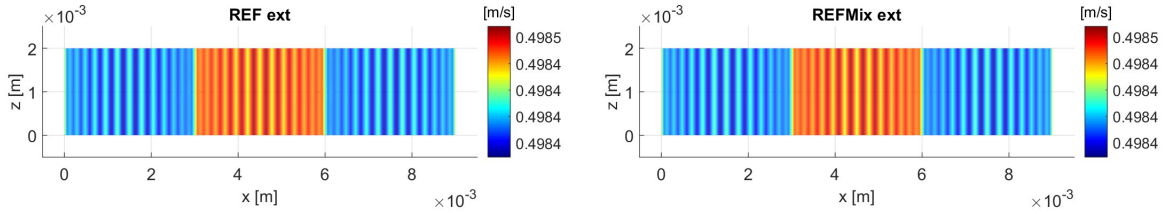


Figure 7.14: u velocity fields on middle xz plane in cases of using exact outlet velocity. The left plot gives results with homogeneous von Neumann conditions for Φ, right for the mixture ones.

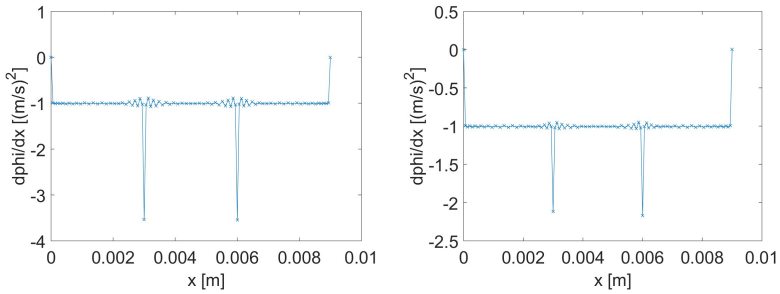


Figure 7.15: Φ gradient in x direction at middle y position for exact outlet velocity. The left graph gives results with homogeneous von Neumann conditions for Φ, right for the mixture ones.

Another criteria was also observed in order to define if compatibility is an issue or not. As it was shown in strong and weak scaling tests, amount of Φ influence matrix iterations increased with increase in sub domain amount. One of the goals to improve performance of the code is to decrease amount of these iterations, for which satisfaction of compatibility was mentioned as a crucial factor. The average amount of Φ influence matrix iterations was observed in here performed test cases and surprisingly, the lowest amount was found for the worst case, REF. 29 iterations are performed for Φ influence matrix in it on average. The cases with completely ensured continuous compatibility $REF\ ext$ and $REFMix\ ext$ returned 38 and 72 iterations on average, respectively. This shows that continuous compatibility satisfaction does not help improve performance of the Φ influence matrix solution. Interestingly, use of mixed boundary conditions causes a considerable increase in amount of iterations, as $REFMix$ case also returned 72 iterations on average. The worst results were noted for staggered grid case, where $STAGh$ produced 76 iterations on average while $STAGm$ case made 120. These findings can therefore only show that use of mixed boundary conditions can, contrary to expectations, cause even an increase in Φ influence matrix iterations. It can also be restated that compatibility constraint is with them only relaxed but not also removed. On the other hand, they do leave the question of discrete compatibility unanswered. It can be only speculated that high amount of iterations in cases with better ensured continuous compatibility can be caused by discrete compatibility not being ensured. The presence of highest amount of iterations in case of staggered grid further supports this, as the derivatives used for Φ and $\vec{v}$ do not correspond to each other since interpolation from one grid to another is involved. On the other hand the lowest amount of iterations in REF case gives an idea to be enabled by both discrete and continuous compatibility not being ensured.

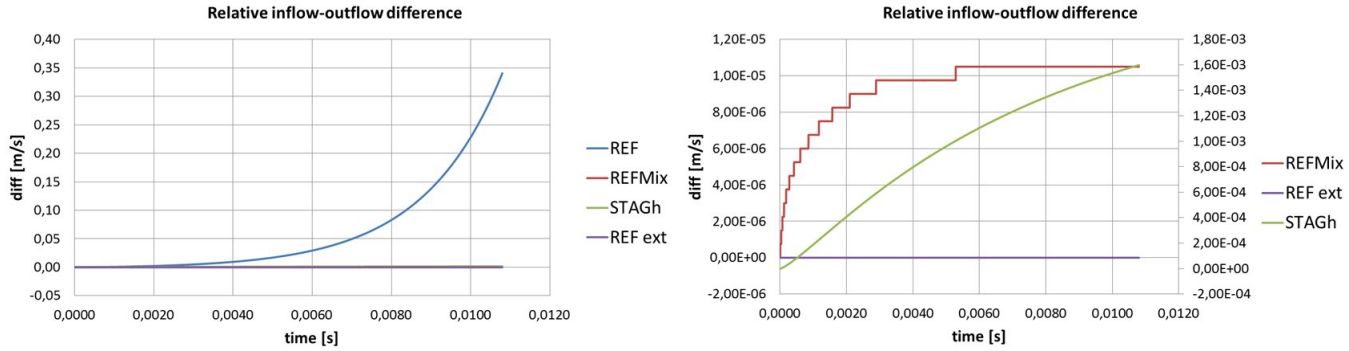


Figure 7.16: The relative difference between inflow and outflow REF, $REFMix$, $STAGh$ cases and $REF\ ext$. The graph on the right shows the difference of better cases more clearly. $STAGh$ line on it corresponds to the secondary scale (right).

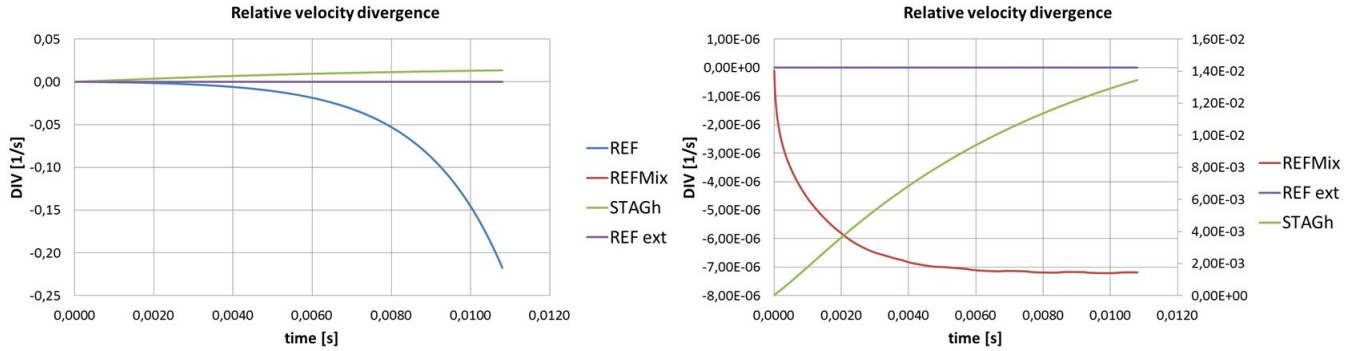


Figure 7.17: Relative velocity divergence plots for REF, $REFMix$, $STAGh$ cases and $REF\ ext$. The graph on the right shows results for better cases more clearly. $STAGh$ line on it corresponds to the secondary scale (right).

2.3 Conclusions from turbulent and laminar channel flow simulations

Turbulent periodic channel flow simulations revealed to be more and more problematic to perform when Re number was increased. Since oscillations in them were noted and led to instabilities for all Re numbers considered, laminar flow cases were studied to find their cause. These test cases confirmed that issues appear because of a problem with influence matrix solution for Φ. Oscillations namely appeared only in directions with more than one sub domain and if solution for Φ was applied. Highest values of velocity divergence being located on the interfaces are another point supporting that issues originate from the mentioned solution. More so are further tests with laminar flow, which revealed presence of discontinuous Φ gradients over interfaces. In accordance with other cases where influence matrix was used as multi domain method, these are a sign of issues with compatibility conditions. This furthermore corresponds with oscillations being highest in cases where continuous compatibility is not ensured. But even if this is ensured, oscillations still persist, although with smaller magnitude. For the case of laminar or low Re turbulent flow, this can be handled to make no notable impact on results. The claim follows from results with the laminar flow cases here, MMS test cases (where continuous compatibility is completely ensured) and smaller problems with oscillations encountered at $Re_\tau = 400$ periodic channel flow than in other two flows with higher Re_τ. However, overall presence of oscillations and even instabilities caused in case of turbulent channel flow, where continuous compatibility is

ensured by periodic conditions, point towards the issue of other, discrete compatibility not being satisfied. This is supported with discontinuous gradient of Φ appearing even in laminar flow cases where continuous compatibility is completely ensured and the fact that divergence volume integral does not show overall same behaviour as the inflow-outflow difference. The amount of iterations performed for Φ influence matrix solution on the other hand returns surprising result of lowest amount of iterations for the worst case observed (REF). Still, this does not dismiss the conclusion that most probably discrete compatibility problems cause the observed issues with oscillations, which can then lead to unstable simulations and also increase the amount of iterations for Φ influence matrix solutions. Especially as the staggered grid cases, where derivatives of Φ and $\vec{v}$ are not from same basis, reveal highest amount of iterations done for Φ influence matrix.

3 Attempt to address discrete compatibility problem

The discrete compatibility, although referred to as probable cause of issues with instabilities, was not thoroughly explained in the previous text. In previous chapter, section about weak and strong scaling tests 5.1, it was for instance only mentioned that the schemes defining second derivatives in Laplace operator do not follow from schemes defining first derivatives and that the first derivative schemes are not verified to ensure conservative formulation. Accoring to theory in [123, 124, 107, 87, 104, 94] discrete compatibility is therefore not ensured. Why this is so, follows directly from explanation of compatibility condition constraint in section 2.2.2.1 of chapter 4. In it, the Gauss theorem is used to connect the surface integral of boundary conditions in equation (4.41) with the volume integral of Poisson equation for Φ, as equations (4.42)−(4.44) show. This is here repeated with equations (7.11) and (7.12) with an important difference-Laplace is written as divergence of first derivative and not directly as second derivative operator. Discrete compatibility demand follows from the presented volume integrals and simply means that discretized equations must satisfy them. It is therefore a counterpart of continuous compatibility demand, only on the level of discretized equations. From this, the two stressed reasons why MFLOPS-3D does not satisfy it follow and will be better referred to in the following text. Possible solutions for the two reasons are also introduced and discussed.

$$\oint \nabla\Phi \cdot d\vec{S} = \int \nabla \cdot \nabla\Phi dV = 0 \tag{7.11}$$

$$\int \nabla \cdot \nabla\Phi dV = \int -\frac{3}{2\Delta t}(\nabla \cdot \vec{v}^{n+1} - \nabla \cdot \vec{v}^{*})dV = \int \frac{3}{2\Delta t}\nabla \cdot \vec{v}^{*}dV = 0 \tag{7.12}$$

3.1 Conservative formulation of first derivative

Firstly, the conservative formulation of the first derivative should be referred to as it gives the basis for the Laplace operator definition importance. In equation (7.12), the last integral demands that the discretized velocity divergence, multiplied with differential volumes dV, sums to zero. This is only possible if the first derivatives are conservative globally, that is, they are defined in a manner to enable zero result of summation. Which is difficult when compact finite differences are used. In the most often used configuration, where a uniform mesh is used and any non unifom mesh is mapped into uniform, the problem affects the derivatives on and at the boundaries, which use either forward or backward schemes [94]. Namely, on uniform grids, where the differential volumes are equal, it is only these schemes which cause the summation of discrete derivatives to not equal zero. Therefore the discrete compatibility condition is not ensured. In [94], a remedy

is proposed through the definition and use of weighting coefficients. For a uniform mesh as used in that work, a general compact scheme for central points on a grid is given with equation (3.26), while the boundary points use schemes defined with (3.32). When schemes for all points in a grid line are combined, they form a system of equations which can be simply written with equation (7.13). This is practically a repeat of equation (3.44) given in section 3.2.4 of chapter 3, where compact schemes in MFLOPS-3D are discussed. For N points in a line, $\mathbf{A}$ and $\mathbf{b}$ are $N \times N$ matrices representing implicit and explicit coefficients.

$$\mathbf{A}[f]' = \mathbf{b}[f] \tag{7.13}$$

In [94], the mentioned remedy to ensure conservative formulation of first derivatives is based on enabling $\mathbf{b}$ columns 2 to $N-1$ to sum to zero. From this, the weighting coefficients follow and affect $\mathbf{b}$ coefficients of schemes at the boundaries. What this does is to enable the Gauss theorem to hold on discrete level since it lets only the values on boundary nodes to contribute to fluxes across boundaries. The values of derivatives, obtained on the other side of equation (7.13), have their integral value therefore set by boundary variable values. Meaning that if the continuous compatibility condition imposing equality of inflow and outflow is satisfied, the discrete one follows. Mathematically, this is shown with equations (7.14) and (7.15). Equation (7.14) follows from multiplication of system of equations (7.13) with inverse of $\mathbf{A}$, resulting in the actual expression for discretized first derivatives. Equation (7.15) then gives the expression for discretized line integral, which, if first derivative is conservative and the mentioned columns in $\mathbf{b}$ sum to zero, has to equal the difference set by boundary values f_1 and f_N. Since the derivatives in $[f]'$ are written for a grid line, the integral in this case can only be line integral and dV is changed with dx, but the logic applies also on whole domain and therefore volume integral.

$$[f]' = \mathbf{A}^{-1}\mathbf{b}[f] \tag{7.14}$$

$$\int f'dx = \sum_{i=1}^{N} f_i' dx_i = \sum_{i=1}^{N} \left(\mathbf{A}^{-1}\mathbf{b}[f]\right)_i = f_N - f_1 \tag{7.15}$$

The solution to ensure discrete compatibility with weighting coefficients is used for example in [87, 104]. The two works are related, with [87] using also influence matrix as multi domain method, which makes the approach interesting for application in case of MFLOPS-3D code. The use of weighting coefficients is well presented in [104], where it is shown that the treatment is applied to predictor velocity $\vec{v}^*$ divergence in Poisson equation in order to ensure its volume integral to equal zero and hence conserve mass. The discretized second derivatives forming Laplace operator are then adjusted to account for the same weighting coefficients in order to keep the equality between LHS and RHS of Poisson equation. Here, a question of universal applicability of this treatment remains opened, since the work uses second order central differences to discretize terms in Laplace operator. Such scheme is known to return zero value of the volume integral or satisfy discrete compatibility with simple treatments applied to it, as follows from [123, 124]. The algorithm also uses a specific derivation of projection method, enabling no decrease in spatial order of accuracy despite second order central differences used in Poisson equation. In [87] only compact schemes are used for discretization and the use of weighting coefficients is apparently not equal. As it is mentioned there, a renormalization technique including same weighting coefficients is used on predictor velocity field in order to ensure mass flow balance and compatibility with it, while the derivative operators do not have weighting coefficients applied.

Although the overall use of compact schemes and influence matrix as multi domain method in [87] makes the renormalization approach more interesting to be applied in MFLOPS-3D than weighting used in [104], there is a consideration which raises doubts about implementation of either of the two approaches. As written, both of them utilise weighting coefficients which come from ensuring that $\mathbf{b}$ columns 2 to $N-1$ sum to zero. However, the weighting is applied to coefficients obtained from uniform grid, where differential volumes are equal and central schemes do not contribute to the problems of discrete compatibility violation. In MFLOPS-3D, the non uniform grids are not mapped into uniform, therefore the weighting procedure as shown in [94] has to be applied differently, by taking the possible contribution to compatibility violation from all schemes, not only boundary ones. Moreover, the weighting has to be done for each specific case of non uniform grids. An attempt to implement weighting was done by one other user of the code in our laboratory and at the moment found unsuccessful. This however does not mean weighting is impossible, only more complex in such a case. It therefore still remains a valid and promising option for resolution of discrete compatibility issues and the work on it is ongoing.

3.2 Laplace operator definition

Previous section shows how important it is for the first derivative to be conservative. This makes the basis for ensuring discrete compatibility. However, this compatibility also demands that volume integral of whole Poisson equation equals zero. It is not enough that only first derivative is conservative, Laplace operator needs to reflect this too. As written, this is done in [104] by applying weighting coefficients also to the discretized Laplace terms, to ensure equality of LHS and RHS of Poisson equation. However, since the weighting is difficult to be applied for MFLOPS-3D, this was in this thesis not considered. Furthermore, the Laplace operator in that work has terms defined with second order central schemes. On the other hand, the first derivative importance offers additional view. Namely, Laplace operator is defined as $\Delta = \nabla \cdot \nabla$. Poisson equation is obtained after divergence is applied to projection equation. Meaning simply that Laplace of Φ follows from derivation of projection equation and if the LHS and RHS of Poisson equation are then to be equal, the Laplace of Φ should strictly be defined with terms that are first derivatives of $\nabla\Phi$. In such manner, the procedure of obtaining Poisson equation from projection, as shown again with equations (7.16) and (7.17), is obeyed also on discrete level. Furthermore, if first derivative has conservative formulation, Laplace obtained by applying it to the gradient operator can also be expected to express same characteristics. As mentioned, such derivation of Laplace operator is not done in MFLOPS-3D, where terms in it follow directly from compact finite differences.

$$\nabla\Phi = -\frac{3}{2\Delta t}(\vec{v}^{n+1} - \vec{v}^{*}) \tag{7.16}$$

$$\nabla \cdot \nabla\Phi = -\frac{3}{2\Delta t}(\nabla \cdot \vec{v}^{n+1} - \nabla \cdot \vec{v}^{*}) \tag{7.17}$$

The importance of ensuring $\Delta = \nabla \cdot \nabla$ equality in regards to discrete compatibility is stressed in [123, 124, 107]. In [107], an interesting point is raised, saying that if the discussed equality is not ensured, the projection method is only approximate and not exact. It is also stated that with compact finite differences, it is impossible to devise a compact Laplacian equal to compact $\nabla \cdot \nabla$. Therefore, if a compact Laplacian is applied directly instead of compact $\nabla \cdot \nabla$, the projection method can only be approximate and the zero divergence condition is satisfied only to the order of the method. Which therefore applies also to the discrete compatibility condition.

On the other hand, simple second order central scheme is also known to not satisfy discrete compatibility automatically on collocated grids [123, 124]. As it is shown in [123], this again follows from inequality between Δ and $\nabla \cdot \nabla$ if the scheme is directly used to obtain first and second derivative. It is worth to notice that in this case, the second derivatives forming Laplace return zero value of discrete volume integral, which is not the case with the first derivatives of velocities, therefore the discrete compatibility is not ensured. All three works propose a similar remedy to ensure desired equality. At first, the discretized projection equation is proposed with $\nabla\Phi$ and predictor velocities $\vec{v}^*$ defined at middle positions between the points or, as used in [107], on cell edges. There, staggered grid is used, therefore no interpolation is needed for velocities, as in the case of collocated grids in [123, 124]. The schemes in [107] are also directly written for the use of staggered grids and demand no specific adaptation to obtain $\nabla\Phi$. In [123, 124], second order central scheme simply uses values from two points next to the middle point to obtain this gradient. Applying divergence to such discrete projection equation enables one to obtain Φ in computational nodes with ensured equality between $\Delta = \nabla \cdot \nabla$ and both sides of Poisson equation. In [123], a proof of discrete compatibility is given with the discrete volume integral of both sides of Poisson equation equaling zero. In [124], same approach is used and adapted to general curvilinear coordinates. In [107], such an approach towards forming Poisson equation is said to enable exact projection, thus it is also a key to ensuring discrete compatibility (velocity divergence condition is ensured).

These observations about Laplace operator definition led to a proposal for MFLOPS-3D. As Laplace operator in the code follows directly from compact schemes for second derivative, only approximate projection is done. This leaves an option for improvement of discrete compatibility if $\Delta = \nabla \cdot \nabla$ equality can be respected somehow. Moreover, as no weighting coefficients are used in [107], such an improvement was an even more attractive option. However, the obvious issue is that in the three works, where equality between discretized LHS and RHS of Poisson equation is ensured, some mid point values are always used. An approach as the one applied in [123] and [124] was considered, but it was decided to first try to obtain the desired equality of Laplace operator directly on collocated grid. Because of this, the projection equation is not adapted at first, as in the presented three works. Instead, divergence is applied directly to its both sides. This presents no issues for the RHS of projection equation. However, the LHS with following Laplace operator demands special attention. The compact schemes used to directly describe Laplace operator terms have to be changed to enable implementation of $\Delta = \nabla \cdot \nabla$. The equation (3.84), which describes derivation of terms forming Laplace operator was addressed in order to do this. This equation is described in section 4.3 of chapter 3 and is here repeated for convenience with (7.18). It presents a second derivative discretization on one grid line, that is, for one direction, but complete Laplacian is then built from it.

$$\mathbf{A}[f]'' = \mathbf{b}[f] \tag{7.18}$$

As it is obvious, second derivatives are obtained straight from variable values. In order to ensure $\Delta = \nabla \cdot \nabla$ equality, the second derivatives should be obtained from the first derivatives, with schemes which are equal to those used to obtain $\nabla \cdot \vec{v}^*$ on the RHS of Poisson equation. This was done purely in a numerical way, as building schemes to represent $\nabla \cdot \nabla$ is complex and results in stencils extending throughout the domain [107]. Hence the schemes are not compact any more. The possibility to inverse implicit coefficients matrix $\mathbf{A}$ was used, which enabled derivation of second derivatives from definition for the first as shown with equation (7.19). The discretized second derivatives and hence also terms in Laplace operator, applicable for eigen decomposition, therefore follow from equation (7.20).

$$\mathbf{A}[f]'' = \mathbf{b}[f]' = \mathbf{b}\mathbf{A}^{-1}\mathbf{b}[f] \tag{7.19}$$

$$[f]'' = \mathbf{A}^{-1}\mathbf{b}[f]' = \mathbf{A}^{-1}\mathbf{b}\mathbf{A}^{-1}\mathbf{b}[f] \tag{7.20}$$

The actual implementation of such second derivatives demanded much more attention than simple use of inverse implicit coefficients matrix. The schemes used to obtain first derivatives and hence $\nabla \cdot \vec{v}^*$ have to be the same as those used to obtain second derivatives f'' from the first f' in equation (7.20). If not, the desired equality is not imposed. Here, two issues arise. Firstly, the schemes to be used in Laplace operator or final discretized second derivative must be able to return vector $[f]''$ which does not include points from boundaries or interfaces. As shown in section 4.3 of chapter 3, the mono domain solver treats values from these points as boundary conditions. Hence for a sub domain with N points in a certain direction, vector $[f]''$ in equation (7.20) has $N-2$ values. Otherwise the final system to be solved has more equations than unknowns. On the other hand, compact schemes used to obtain other first or second derivatives do not have this limitation. As same group of schemes is needed to obtain first and from it second derivatives in order to impose desired equality, adaptation of stencils at the boundaries is required. The goal is to obtain stencils, which for schemes in points $2 : N-1$ do not require f'' values in first or last points. Equally said, these are schemes which include values from first or last point in implicit stencils only if written for these two points. Such schemes are in MFLOPS-3D used to obtain Laplace operator directly, therefore they were here used to define schemes for both first and from them following second derivatives. The stencils used in these schemes are given in table 3.2 in mentioned section 4.3. However, the schemes are only used for points $2 : N-1$. The first derivative schemes in first and last point use same stencils as before. They can be seen in table 3.1. These schemes are important as influence matrix method uses derivatives defined from them. It then follows that schemes for first derivative are purely forward or backward on the first and last two points on a grid line. It is also important to note that here described schemes for second derivative affect only the second derivative as used for Laplace operator. If a second derivative is needed otherwise, as for viscous terms on the RHS of equation for $\vec{v}^*$ in the new algorithm, it is obtained directly as before, without using the procedure described here. This and schemes applied in it are strictly used for the general first derivatives and for second derivatives in Laplace operator of the mono domain solver.

The second issue, which follows with implementation of Laplace from (7.20) and using same schemes to obtain second derivative from the first one, is connected with boundary conditions and their application. In case of compact Laplace operator used otherwise, section 4.3 in chapter 3 shows either Dirichlet or von Neumann conditions are easily applied. Compact schemes defining Laplace operator in that case namely enable explicit stencils where variable or first derivative values in boundary points can be used. Here however, situation is more complex. In order to present it and give the solution, equation (7.20) has to be rewritten with a more thorough form and some matrices have to be written with terms. Matrices are also renamed, since $\mathbf{A}$ and $\mathbf{b}$ applied with vector $[f]''$ do not have same size as those originating from $[f]'$. This vector has N terms, hence its matrices $\mathbf{A}$ and $\mathbf{b}$ are both of size $N \times N$. They and their terms are denoted with subscript d. Matrices $\mathbf{A}$ and $\mathbf{b}$ connected with vector $[f]''$ have $(N-2)\times(N-2)$ and $(N-2)\times N$ terms, respectively. They have subscript s assigned. Equation (7.20) can therefore be expressed as (7.21).

$$[f]'' = \mathbf{A}_s^{-1}\mathbf{b}_s[f]' = \mathbf{A}_s^{-1}\mathbf{b}_s\mathbf{A}_d^{-1}\mathbf{b}_d[f] \tag{7.21}$$

The $\mathbf{A}_d^{-1}\mathbf{b}_d[f]$ part is the cause of boundary conditions application complexity. If Dirichlet conditions are given, this presents no issues for implementation. All four matrices in equation above can be multiplied to form an equivalent of matrix $\mathbf{B}$ from section 4.3. Then, first and last columns of $\mathbf{B}$ can be multiplied with first and last values in $[f]$, products are then put on the RHS of a system of equations to be solved and $\mathbf{B}$ without the mentioned two columns is used in eigen decomposition. This is all very much the same as before. However, when von Neumann conditions are specified, same approach cannot be used as vector $[f]$ does not include first derivatives. Indeed the von Neumann conditions can be implemented on the level of $[f]'' = \mathbf{A}_s^{-1}\mathbf{b}_s[f]'$ expression. Meaning that first or last columns from $\mathbf{A}_s^{-1}\mathbf{b}_s$ product can be multiplied with the corresponding von Neumann conditions and put on the RHS of the system to be solved. However, the otherwise used replacement of $[f]'$ vector with of $\mathbf{A}_d^{-1}\mathbf{b}_d[f]$ means that in this case unknown values f_1 or f_N can now be included in all rows of (7.21). Which has to be prevented. A solution for the problem exists on behalf that $\mathbf{A}_d^{-1}\mathbf{b}_d[f]$ replaces $[f]'$ while f_1' or f_N' are known. If $\mathbf{A}_d^{-1}\mathbf{b}_d$ product is written as $N \times N$ matrix $\mathbf{C}$ with terms c_{ij}, the equality $[f]' = \mathbf{A}_d^{-1}\mathbf{b}_d[f]$ follows as written below.

$$\mathbf{C}[f] = \begin{bmatrix} c_{11} & c_{12} & . & . & . & c_{1(N-1)} & c_{1N} \\ c_{21} & c_{22} & . & . & . & c_{2(N-1)} & c_{2N} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ c_{(N-1)1} & c_{(N-1)2} & . & . & . & c_{(N-1)(N-1)} & c_{(N-1)N} \\ c_{N1} & c_{N2} & . & . & . & c_{N(N-1)} & c_{NN} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ . \\ . \\ . \\ . \\ f_{N-1} \\ f_N \end{bmatrix} = \begin{bmatrix} f_1' \\ f_2' \\ . \\ . \\ . \\ . \\ f_{N-1}' \\ f_N' \end{bmatrix}$$

What can be obtained from this is the direct expression for either f_1' or f_N', whichever is needed. If f_1' is taken as an example, its equation can be written as equation (7.22). Given the known value of f_1', the variable f_1 value follows the expression in equation (7.23). Since the unknown value can be now in each row of $\mathbf{C}[f]$ expressed through other values, the above given matrix system can be rewritten to exclude unknown value. Or equally said, to include unknown value through other values. For instance, expression for f_2' can be written as given in equation (7.24). Matrix $\mathbf{C}$ shown above is adjusted with the terms shown in this equation. Same procedure is applied if f_N' is known and doubled if both f_1' and f_N' are specified. No issues appear in this case as well, since the c_{1N} and c_{N1} terms are zero (grid should have enough points that certain stencils cannot include all grid points).

$$f_1' = c_{11}f_1 + c_{12}f_2 + c_{13}f_3 + ...c_{1N}f_N \tag{7.22}$$

$$f_1 = \frac{f_1' - (c_{12}f_2 + c_{13}f_3 + ...c_{1N}f_N)}{c_{11}} \tag{7.23}$$

$$f_2' = c_{21}\frac{f_1'}{c_{11}} + f_2\left(c_{22} - \frac{c_{21}c_{12}}{c_{11}}\right) + f_3\left(c_{23} - \frac{c_{21}c_{13}}{c_{11}}\right) + ... + f_N\left(c_{2N} - \frac{c_{21}c_{1N}}{c_{11}}\right) \tag{7.24}$$

With such adaptation of matrix $\mathbf{C}$, there are no excessive unknowns. Furthermore, since f_1' or f_N' are in this work always equal to zero, first or last columns to be multiplied with replaced f_1 or f_N can have zero terms. Similar applies to rows representing given f_1' or f_N' in $\mathbf{C}[f]$ as inclusion of f_1 or f_N, written through equations as (7.23), cancels all the terms in corresponding rows, bar

the known derivative value. In such manner, the von Neumann conditions are in mono domain solver applied on the level of $[f]'' = \mathbf{A}_s^{-1}\mathbf{b}_s[f]'$ expression directly. If f_1' or f_N' were not zero, this would have to be respected in the revised matrix $\mathbf{C}[f]$ as the mentioned columns cannot be taken as zero any more. Which would also affect application of such boundary conditions through $[f]'' = \mathbf{A}_s^{-1}\mathbf{b}_s[f]'$, but no considerable changes are needed. In all cases, the final matrix $\mathbf{B}$ to be eigen decomposed follows from columns $2 : N-1$ in matrix $\mathbf{C}$ multiplied with $\mathbf{A}_s^{-1}\mathbf{b}_s$. The two remaining columns from matrix $\mathbf{C}$ are used only for application of Dirichlet boundary conditions (or non zero von Neumann conditions). In such case, they are multiplied with $\mathbf{A}_s^{-1}\mathbf{b}_s$ and give terms to be included on the RHS of system of equations to be solved, as described before.

The presented approach to ensure $\Delta = \nabla \cdot \nabla$ equality was in MFLOPS-3D tried only with original algorithm and with 4^{th} order compact schemes for first derivatives. The reason is the overall 4^{th} order accuracy of these schemes and presence of compatibility issues already in the original code. From them, schemes with same order of accuracy follow also for composed Laplace operator, for each possible combination of boundary conditions. Convergence tests results showing this are given in the Appendix. In simulations however, it was found that some important improvements are enabled with it, but a lot of issues still need to be addressed. On the plus side, redefined Laplace operator proved to produce better results for mono domain flow simulations than the otherwise applied one. In these simulations, the usual convective boundary condition for outflow velocity from equation (4.45) was used with homogeneous von Neumann condition for Φ. Simulations with redefined Laplace operator proved to be stable, while the compact Laplace operator led to divergence. Comparison of pressure and u velocity results for case with mean velocity $\bar{u} = 0,7\ m/s$ can be seen on figure 7.18. Mesh (non uniform) with 35 points in each direction was used with time step resulting in $CFL = 0,08$.

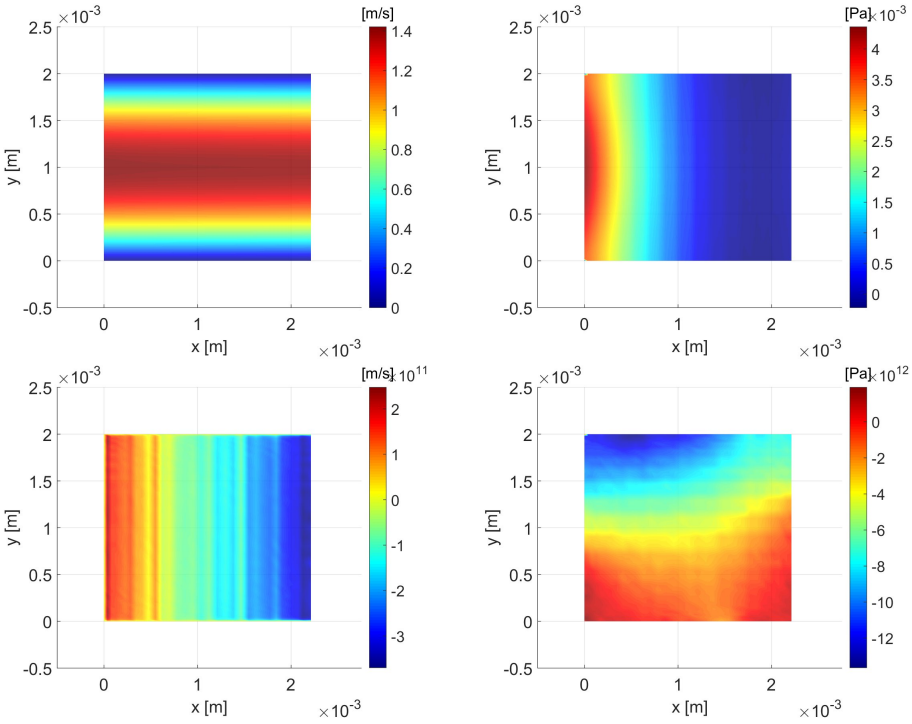


Figure 7.18: Graphs showing u velocity (left) and p (right) results on xy plane at middle z position. First line gives results with redefined Laplace operator while second gives results obtained with usual compact Δ. Both results show state after 1500 time steps were performed.

It can be seen that results with the new operator return expected values and no issues with oscillations. Since only mono domain solver is used, meaning that singular Poisson-Neumann problem is solved in a direct manner, the opinion was that the discrete compatibility affecting influence matrix solution is resolved. If not, the null eigen value issues and outflow conditions which do not ensure continuous compatibility should prevent direct solver to produce stable simulations. The null eigen values were indeed produced, one per direction, and were even smaller than for the case of compact Laplace operator (they always have some value, although close to zero). Other eigen values are always real and negative. This is same behaviour as observed with the compact Laplace operator, where simulations were unsuccessful. It is worth to also write that simulations with redefined Laplace did not demand any special treatment because of null eigen values. The compact Laplace on the other hand produced unstable simulations if the division with null eigen values was treated or not (results in case of such division have zero values applied). Hence, it was expected that multi domain simulations would return better results as well. These simulations however showed a completely different picture and revealed that issues with unstable simulations and appearing oscillations are, if anything, only worse. The results for u velocity and pressure, from same simulations as before but with three sub domains in x used, are shown on figure 7.19. It can be seen that both velocity and pressure express much worse results, where interestingly, pressure follows same form as in the case with usual Laplacian operator. Only its amplitudes are much higher. Oscillations are also obvious in both results, thus it can be considered that the discrete compatibility issues still persist.

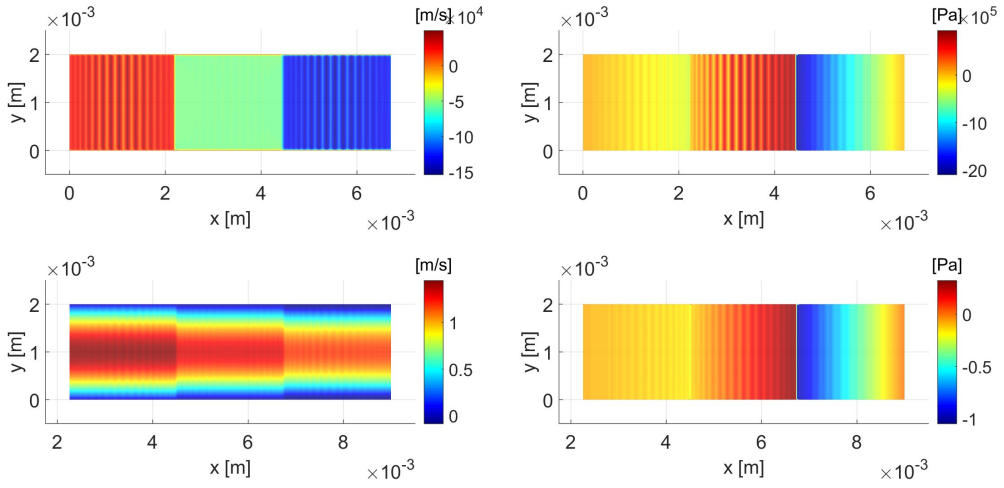


Figure 7.19: Graphs showing u velocity (left) and p (right) results on xy plane at middle z position. First line gives results with redefined Laplace operator while second gives results obtained with usual compact Δ. Both results show state after 1500 time steps were performed and three sub domains in x direction were used.

Since good results were obtained in shown mono domain simulations, the new Laplace operator was also tested with MMS tests. These do not show so many issues with compatibility, for which the exact velocity boundary conditions are responsible. Mono domain cases were run, since the multi domain ones return worse results. It was found that the redefined Laplace operator actually does not return consistent results, with L_2 norms not decreasing with mesh refinement. If the mesh was very refined, simulations could even diverge. The reason for such behaviour was not defined. Results are especially interesting since no issues regarding consistency of the new schemes for Laplace operator were revealed in convergence tests for it.

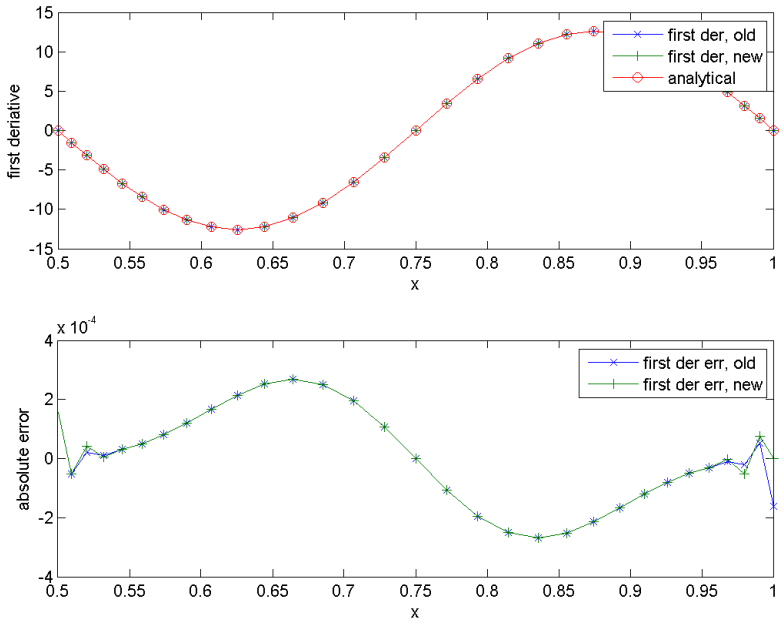


Figure 7.20: First derivative results and absolute error for a case of cosinus function derivation on non uniform grid with 31 points. Results for both usual and new schemes are given. The results with the new schemes have Dirichlet boundary conditions on the starting point and von Neumann on the end.

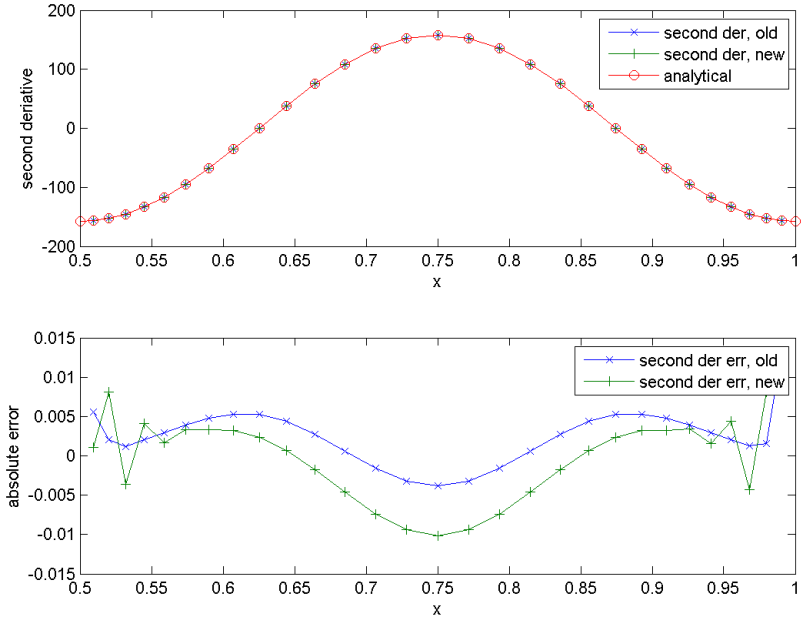


Figure 7.21: Second derivative results and absolute error for a case of cosinus function derivation on non uniform grid with 31 points. Results for both usual and new schemes are given and both schemes have Dirichlet conditions applied at first point and von Neumann at the last.

Although the reasons for inconsistent simulations with redefined Laplace operator were not defined, some conclusions can still be given. The improved performance in the case of non periodic flow in mono domain simulations shows that the new approach to implement Laplace operator which satisfies $\Delta = \nabla \cdot \nabla$ equality with compact schemes is possible. No staggered grids are needed for it as well, as in [107]. However, the approach needs to be further improved since multi domain simulations show that discrete compatibility issues persist while MMS tests revealed inconsistent behaviour. The two could be also well connected. Namely, use of forward or backward schemes is increased with the redefined Laplace operator, making way to increased Runge phenomena effect at the boundaries and worse first derivative conservation properties. The increased Runge phenomena was confirmed in performing convergence tests to determine order of accuracy of second or first derivatives obtained from usual and new compact schemes. Example is given with figures 7.20 and 7.21, where both first and second derivatives of a simple cosinus function on non uniform grid (same as the one used in laminar flow simulations) are given. The example is a general one from the point of mixed boundary conditions used in it to show the effect of presented von Neumann conditions implementation. Boundary conditions are not necessary, but convergence tests of derivative schemes were performed with them implemented in order to verify their correctness. The only results where they are not used are for usual schemes for first derivative. In can be seen that new schemes return first derivative with only slightly increased oscillations caused by Runge phenomena at the boundaries. The second derivative on the other hand shows much increased oscillations, no matter which boundary conditions are used. The error for middle points is increased too, although this is not assumed to be the cause of issues. Especially since errors otherwise can be even smaller than those from original schemes. Furthermore, as the increased use of forward and backward schemes increases Runge phenomena effects, it can be also taken to worsen the conservative characteristics of first derivative, especially as no weighting is used. Since influence matrix uses the problematic (forward or backward) schemes, this can indeed cause worse results in multi domain cases. Thus two future research directions can be set. One is to diminish the Runge phenomena with change in the used schemes. The other is to try and implement weighting procedure. With this however, the need for such Laplace operator as presented here could well even disappear.

3.3 Conclusions about attempts to improve discrete compatibility

Although this section presented the attempts to improve discrete compatibility, which is believed to be the reason for increased amount of influence matrix iterations for Φ and instabilities in real flow simulations, it rather opened new possibilities on how MFLOPS-3D code can be improved.

The section first deals with the known problems of compact schemes not ensuring conservative formulation of first derivative by themselves. This can well be the reason for discrete compatibility violation in MFLOPS-3D and ensuring it was also shown to be the solution used in many cases where compact schemes are applied. Ensuring of first derivative conservative formulation is with MFLOPS-3D more difficult than in works where it was applied because of compact schemes being here used directly on non uniform grids. As such it was not tried in the scope of this thesis, but an attempt at it was nevertheless done in our laboratory, although for the moment unsuccessful.

Another approach was tried instead of ensuring conservative first derivative formulation. It was namely shown in different works that the way in which Laplace operator is defined is important regarding discrete compatibility and also exactness of projection methods. Therefore new Laplace operator definition was applied, introducing a composition of discretized second derivatives through first derivative compact schemes. This required resolution of boundary condition

issues, which resulted in demands to use specific schemes and also specific inclusion of von Neumann boundary conditions. The new schemes for Laplace operator showed to retain the order of accuracy from the first derivative compact schemes used for their derivation. They also importantly showed capability to return stable mono domain simulations of incompressible channel flow, which was not possible with the original compact Laplace operator. However, they did not turn out to be the remedy as multi domain simulations with them were unstable while MMS mono domain test case showed that they actually suffer from inconsistency.

Although unsuccessful, the new Laplace operator still shows interesting results and also points to directions in which MFLOPS-3D can be improved. These are the ensuring of conservative first derivative formulation of the chosen compact schemes and improvements regarding presence of Runge phenomena. Both of these issues are considered to be affecting simulations in a more profound way when the new Laplace operator is used and work on them should be done in the future regardless of which Laplace operator is chosen for use. It was not stressed, but this work should be first done for the case of original code, and then also for the new one, with the new algorithm. The reason is in the differences between the codes and the fact that the issues appear already in the original MFLOPS-3D, which is simpler than the one for cavitating flow simulations.

Conclusions and future work

The work in this thesis presents development of a new algorithm for cavitating flow simulations, suitable for use in codes which enable efficient or fast DNS simulations and offer more flexibility regarding geometry of the domain. The need for DNS simulations comes on one hand from the desire to have a better description of turbulence-cavitation interactions, which open a lot of unanswered questions and also make development of better turbulence and cavitation models difficult. On the other hand, it also comes from the availability of better experimental results for cavitating flows, introduced by development and use of novel measuring techniques. Such simulations can therefore now be better validated and offer more insight into the problem of turbulence-cavitation interactions. However, the need for a new algorithm for them originates from the demands imposed on the numerical methods to be used. These base on desired ability for fast DNS since cavitating flows are more cumbersome to be simulated than incompressible ones and there are also many cavitation models which can be used. The request to make simulations faster promotes use of direct, fast solvers, which can be further combined with specific multi domain methods to enable good code scalability. Influence matrix technique is such a method. Importantly, spectral methods are the usual choice for discretization when fast DNS is requested. But they are here discouraged because of geometries in which cavitating flow is to be simulated. Instead, compact finite differences are used. Influence matrix as multi domain method and direct mono domain solver promise good computational performance with them too. But consequently, an algorithm, capable to return stable and accurate simulations of cavitating flows while keeping the LHS of solved equations constant, is requested.

Such an algorithm was developed in the scope of this work. Although not stressed, the algorithm can be used also in LES simulations. This was not put to discussion since this possibility is quite straightforward. MFLOPS-3D code, developed and used in our laboratory before, was taken as a basis for its development. The code was before this work changed from using spectral methods in two directions to discretization with compact finite differences in all three directions. It also features the mentioned influence matrix technique as multi domain method. Use of compact finite differences in three directions actually makes MFLOPS-3D a unique code. The reason is in the consequent combination of direct mono domain solver with iterative one for influence matrix, as this is here used as 3D multi domain method. Application of mapping to enable simulations in various geometries further opens the possibilities for applications of MFLOPS-3D. This therefore seemed to be a suitable tool for implementation of abilities to simulate cavitating flow. Indeed the goal was also to finally perform the desired DNS simulations with it and propose some improvements for cavitation and turbulence models. But the development of the needed algorithm was more time consuming than planned while MFLOPS-3D was found to express considerable issues in pressure solution on multi domain level. The work was therefore focused on the algorithm development and, at the end, also on some attempts to improve pressure multi domain solution.

The new algorithm is created for simulations of cavitating flows using homogeneous mixture

approach with single phase modelling. This was chosen since it is the most practical and widely used approach, able to produce good results and also presents a subject of on going research to propose better turbulence and cavitation models. Furthermore, it was decided to use cavitation models with additional transport equation for gaseous phase as these offer better cavitation description. It was also chosen to neglect thermal and phase compressibility effects as considered cavitating flow to be simulated includes water at room temperature without appearance of shocks. Importantly, the governing equations for such kind of cavitatition modelling were in the past shown to be successfully adapted to algorithms based on projection methods. Such an algorithm is also implemented in the original MFLOPS-3D. More precisely, the projection method is a version of the Kim and Moin scheme or rotational non incremental pressure correction scheme. This was then used as a basis to develop the new algorithm.

Although it is generally well documented how to solve chosen governing equations with projection methods, the known procedures could not be directly applied. Especially the pressure solution was found to be very troublesome. The issues imposed by the need to keep LHS of equations constant led to the introduction of different solution procedures, from which the completely linearised source term and introduction of CG method were found to offer the desired solutions. The latter also turned out to give the needed tool for implementation of the new algorithm. Though a stable solution for pressure is possible with completely linearised source term, the implementation of CG method led to the desired stability with also much improved computational performance. The new algorithm was then proposed in four versions, where the CG method is used in both $\vec{v}^*$ and Φ solutions. Versions differ between each other in using pressure incremental or non incremental scheme and in the way in which the source term is treated during outer iterations. The algorithm namely also proposes outer and inner iterations. The first are a consequence of increased amount of explicit terms in all steps of the algorithm, while the latter come from the inclusion of CG method. The algorithm therefore offers different manners on how to proceed in simulations. Importantly, its pressure incremental versions are similar to algorithms otherwise used for cavitating flow simulations. But the non incremental ones might offer an improvement in DNS simulations, as pressure non incremental schemes have in the past shown to include less issues with stability in multi domain simulations.

All versions of the new algorithm were verified to produce correct results in the scope of this thesis with MMS method. This is another important contribution of this work, as a valuable MMS test case for cavitating flow was devised in order to develop and verify the new algorithm. Such a test case is to our knowledge a first of its kind and could be an interesting option for future verification and development of numerical tools for cavitating flow simulations. This test case and its counterpart for incompressible flow were used to perform verification of the order of accuracy for new and original algorithm, as well as analysis of computational performance of the MFLOPS-3D code with both algorithms. Furthermore, the cavitating flow test case was also used to assess response of the new algorithm in cases where certain flow effects are more pronounced. Results show that the new algorithm can in case of incompressible flow retain the order of accuracy observed with original one. This is 2^{nd} order for both velocity and pressure. The goal order for latter is actually considered as 3/2, according to the literature. The algorithm did however reach such accuracy only with pressure incremental versions (PIUS or PICS). The non incremental ones (PNUS and PNCS) were found to suffer from the use of inaccurate, but still required, pressure values imposed on the boundary with Dirichlet conditions for Φ. It follows that this effect is more a consequence of the chosen MMS test case and that results with pressure non incremental algorithm versions are still acceptable. Nevertheless, a more suitable MMS test case should be devised to prove that the two non incremental versions can return better accuracy.

Especially since the difference is even bigger in verification with cavitating flow case. There, pressure was found to define orders of accuracy of other variables. These all change to values, similar to the one for pressure. PIUS version returned best results and showed that the goal order of accuracy for pressure (3/2) is possible to be achieved even in a very demanding cavitating flow case, which includes sudden pressure changes and complete range of α values. PICS version returned similar results, with a bit lower pressure order of accuracy, but the problematic ones are the results with the two non incremental versions. In cavitating flow case their pressure order of accuracy drops below unity, and with it also accuracy of all other variables. On behalf of these results, it also follows that PIUS and PICS versions are currently the most suitable for real flow simulations. PICS version is the most practical one, which follows from three observations. First, PICS version was able to return slightly better stability in cases where increased flow effects were observed through increase of factors in devised MMS analytical flow solutions. Second, the computational performance tests showed PICS version returned faster multi domain computations than most accurate version, PIUS. Finally, only drawback to PIUS version is its slightly lower pressure order of accuracy. However, the computational performance tests also showed that two non incremental versions return fastest multi domain computations. Therefore a proof that these two versions are able to reach higher orders of accuracy is even more important. If this is indeed so, the PNCS or PNUS versions should be taken as most appropriate for real flow simulations. The computational performance tests otherwise showed that the new algorithm is in incompressible flow simulations up to two times slower than the original one. Inner and outer iterations are the reason for this. Regarding cavitating flow cases, the new algorithm shows to be up to six times slower than in the cases of incompressible flow. Which means that there is a considerable difference between performance of new algorithm in cavitating flow case and original algorithm for incompressible flow. This is however not taken as problematic, since the cavitating flow case was designed to invoke a lot of inner and also outer iterations with the inclusion of complete α range and sudden pressure changes. The presented ratio between the two algorithms is therefore considered as worst possible. Furthermore, the mentioned ratios are not average but the highest ones noted in results, which only confirms this.

There were however some issues observed in the performed multi domain verification tests. The observed error norms in them were noted to express an increase for the case of two most refined grids. The issues with this increase do not come from the same cause in cavitating and incompressible flow case. In cavitating flow case, the increase was not found to be a problematic one. But in the case of incompressible flow, the increase cannot be neglected. Furthermore, it can also severely affect future real incompressible or cavitating flow simulations with MFLOPS-3D. It was namely discovered that this increase follows from worse convergence of influence matrix solution for Φ. Which effectively means issues with multi domain pressure solution. This worse convergence was to no surprise also found to be very problematic in computational performance analysis. The influence matrix solutions, or solutions obtained from the iterative multi domain solver, define computational performance of MFLOPS-3D code. Amongst these, solution for Φ is expectedly the most important one. As the solver expresses a considerable drop in convergence for this variable when amount of sub domains is increased, computational time becomes increasingly dependent on it. All versions of MFLOPS-3D code were found to be affected by this, hence an improvement in this regard is essential.

The attempts at performing real flow simulations in form of LES simulations of turbulent channel flow and initialization of flow in venturi geometry only confirmed presence of issues with multi domain solution for Φ. Although some valid periodic turbulent channel flow simulations were performed, the mentioned issues were present in all of them. They were connected to

occurrence of oscillations, which led to further increase in iterations to obtain Φ influence matrix solutions and unstable simulations. Hence an analysis with simple laminar flow cases was done and it was concluded that the issues are most probably caused by discrete compatibility condition not being ensured with compact schemes in MFLOPS-3D. An attempt at treating this was done in the scope of this work with introduction of a new Laplace operator which ensures use of exact projection method. Although interesting, the operator and schemes implemented with it did not show an improvement regarding Φ influence matrix solutions. But they are able to solve Poisson-Neumann problem in mono domain calculations better than other used schemes. Which makes them interesting for further development and use.

Apart from the issues with multi domain solution for Φ, there was another problem encountered in attempts to perform venturi flow simulations. This was caused by the non Cartesian terms of Laplace operator introduced with mapping. These are treated explicitly and demand essentially same iterations as those applied by CG method. It was found that the y direction non Cartesian term causes the issues as it highly increases the presence of explicit terms in the most difficult part of venturi geometry (throat). The issue was successfully treated by doubling the Poisson equation for Φ solution, which brought the problematic term at venturi throat close to zero. The treatment of mapping issues in presented manner is simple and can be applied also to other codes where such approach towards simulations of flows in various geometries is used. Furthermore, the iterations demanded by mapping also mean that the computational performance difference between original and new code should be in venturi flow simulations much smaller than reported in verification tests.

From the presented work it therefore follows that despite a novel algorithm for cavitating flow simulations was developed and verified, and many additional methods were introduced as well, real flow simulations with it were in the end not possible because of the issues caused by the used code. The gained experience from algorithm's development and especially from the work on the problems imposed in real flow simulations raised proposals for future work which can be put into three groups. These are resolution of Φ solution issues, improvement of iterative multi domain solver for Φ and performing real cavitating flow simulations.

For the resolution of Φ multi domain solution issues, the observations from this work make case for two future activities. At first, discrete compatibility issues should be treated. One remedy for this is the application of conservative first derivative formulation. Since MFLOPS-3D uses compact schemes directly on non uniform grids, ensuring of this formulation would make a very interesting development and option for future use of compact schemes. The other possibility, which is also worth to be considered generally, is the use of staggered grid. This is often reported to not just treat the odd-even decoupling but also to be a better way to ensure conservation of properties. Use of staggered grid would considerably change MFLOPS-3D, therefore the application of conservative formulation is to be considered before. Staggered grid could well also be the solution for the second needed work regarding improvement of Φ solution. This is the work on Runge phenomena, which is strongly present because of the completely forward and backward schemes on the interfaces of sub domains. Besides staggered grid, use of filtering could also be a solution for this issue. Finally, ghost points could be introduced for each sub domain in order to ensure use of central schemes on interfaces. These might also improve the issues with conservative formulation. However, the question remains how such points could be applied.

Regarding the improvement of multi domain solver or iterative solver for influence matrix, it should be first mentioned that such work was in the scope of this thesis not considered or applied. The used Krylov solver was just slightly changed when mixed boundary conditions for Φ were introduced. But regarding the high amount of iterations for Φ in all cases and relatively small

applicable sub domain sizes, the improvement of this solver should also be one of the future work points. Especially the improvement of this iterative solver in case of Φ solution. Options are either to use different preconditioners for the applied solver or even change the type of the solver itself. The work on this subject is actually already considered in the scope of one other thesis currently in progress, where MFLOPS-3D is used. The aim of that work is also to enable use of larger sub domains and hence make the use of influence matrix as 3D multi domain method and MFLOPS-3D very competitive against other codes.

The previous two points for future work primarily consider improvements of original MFLOPS-3D code. These should then be implemented also into the new code with the new algorithm. With them, real incompressible or cavitating flow simulations should become realizable. Performing cavitating flow simulations nevertheless still demands a considerable amount of work to be done. At first, periodic turbulent channel flows are to be simulated with LES models in order to obtain desirable data for inflow velocities. At the same time, the flow in venturi test section should be initialized with simulations where its velocity is gradually brought to the goal velocity at which cavitation was observed. Secondly, there are some questions regarding the new algorithm, which should be addressed although the algorithm was successfully verified. These are the questions regarding α transport equation solution and choice of suitable σ_p for certain cavitation model. The latter can be quite straightforward, especially for some empirical cavitation models, while additional care should be given to models with highly different source term in case of vapour creation and destruction and to pressure difference term in models based on bubble dynamics. The solution for α on the other hand could be reconsidered as a whole after improvements for Φ solution will be implemented. At the moment, second order upwind scheme is used for discretization and solutions are obtained separately in each point. A future improvement could therefore be to try and use compact schemes for discretization of this variable (and all dependent on it). At the moment this is not feasible because of Runge phenomena which leads to high discontinuities of α solution on interfaces. Another improvement would also be implementation of an advection equation solver. Although this does not seem to be necessary, it could lead to improvements through the use of a more implicit solution procedure for α. The question how such a solver could be implemented in MFLOPS-3D remains open, as its integration with the multi domain method might be impossible.

As already mentioned, with all of the improvements applied, where the final ones for α do not seem obligatory, real cavitating flow simulations with MFLOPS-3D should become feasible. It is difficult to say how long the implementation of the discussed improvements will take, especially as the use of influence matrix technique as 3D multi domain method is quite a novel approach. The use of compact schemes directly on non uniform grids only adds to the complexity. But successful resolution of the shown issues should in return give a very interesting numerical tool, capable of efficient simulations of highly resolved flows in various geometries. With the new algorithm, also in the case of cavitating flow. However, the new algorithm was already verified to be suitable for such simulations. It can be directly applied and used in other codes which enable fast DNS or LES simulations and offer higher flexibility regarding domain geometry and size. Therefore an interesting and novel approach for performing highly resolved cavitating flow simulations is already available. It is expected that this, with the new and more detailed experimental results, will enable further insight into cavitating flow and its interactions with turbulence. Thus better cavitation and turbulence models should be finally obtained.

Chapter 8

Appendix

1 Compact finite difference schemes, convergence tests

This part of the appendix gives results of convergence tests with which the order of accuracy of compact finite difference schemes is defined. Convergence tests were done with calculating first or second derivative of the chosen function (8.1) on interval $x \in \{0,5; 1,0\}$. Difference between these results and analytical values was used to obtain L_2 norms of errors. L_2 norms are defined with equation (8.2). Equation is given for the case of first derivative results f', its use for second derivatives is analogous. Since L_2 norms were defined for single or multiple points, equation features division with the amount of points on which errors are observed. This is done in order to have estimation done using average error in case of multiple points and to preserve similarities between single and multiple point results. Convergence tests were performed by obtaining results on gradually refined grids. These included from 21 to 281 points and were either uniform or non uniform. Non uniform grids were defined using tanh law given with equation (6.1). Obtained L_2 norms were used to obtain convergence curves, their slopes then defined the order of accuracy. The procedure of defining order of accuracy is therefore analogous to the one done for VOA with MMS.

$$f = \cos\left(\frac{2\pi}{0,5}(x - 0,5)\right) \tag{8.1}$$

$$L_2 = \sqrt{\frac{\sum_i^{N_{pts}} \left(f_i' - f'^{ext}_i\right)^2}{N_{pts}}} \tag{8.2}$$

Following sections present at first the convergence tests' results for the mentioned 4^{th} and 8^{th} order schemes in MFLOPS-3D, discussed in chapter 3. Results for first derivative are followed by those for second. Afterwards, results for newly implemented schemes, aimed at discretely satisfying compatibility condition, are given in same manner. Only 4^{th} order accurate schemes are shown, as these were the only ones used in the new schemes.

1.1 First derivative orders of accuracy

Convergence curves and with them orders of accuracy are given for three different cases in order to show the provided overall and local order of accuracy with used compact schemes. Schemes namely differ depending on their position on the grid, therefore their errors also differ. Errors from all grid points are taken into account to provide global order of accuracy. Middle five points provide the ideal order of accuracy as schemes are strictly central. First and last point schemes are purely forward or backward, hence their order of accuracy is decisive for the overall accuracy.

It can be seen that schemes referred to as 4^{th} order accurate provide this or better order of accuracy in all points. Schemes referred to as 8^{th} order accurate provide this and even higher order of accuracy only in points where central schemes are used. Overall they provide only 4^{th} to 5^{th} order accuracy because of lower accuracy of the boundary schemes. The waves observed for central schemes are a consequence of machine round off error being reached.

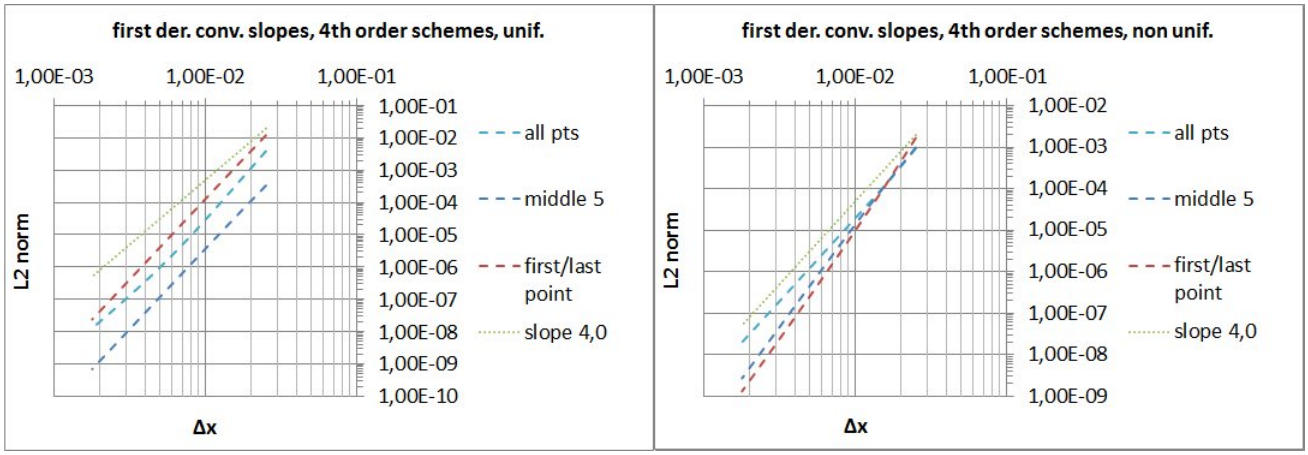


Figure 8.1: Convergence slopes for first derivatives obtained with 4^{th} order schemes in MFLOPS-3D on uniform (left) and non uniform (right) grids.

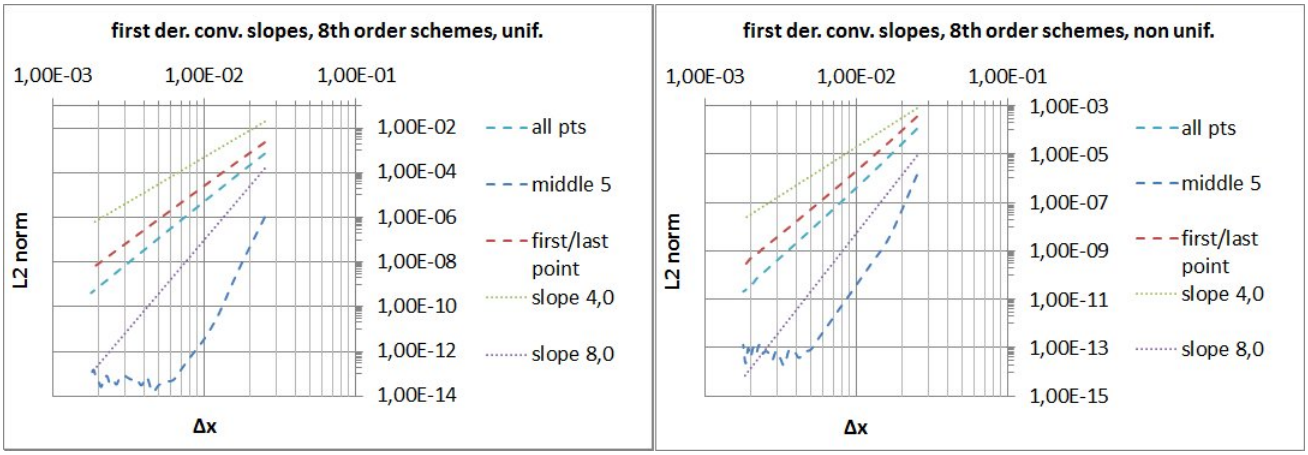


Figure 8.2: Convergence slopes for first derivatives obtained with 8^{th} order schemes in MFLOPS-3D on uniform (left) and non uniform (right) grids.

1.2 Second derivative orders of accuracy

Convergence results are, as for first derivatives, given for different cases. Importantly, here presented second derivatives are obtained with schemes used to discretize Laplace operator terms. These differ from other compact schemes in used stencils near boundaries. Since the schemes are in those points also specifically derived to enable inclusion of von Neumann boundary conditions, additional results for convergence in first or last points with such schemes are given as well. These results are marked with *vn*, while other results in first or last points are marked as *dir*.

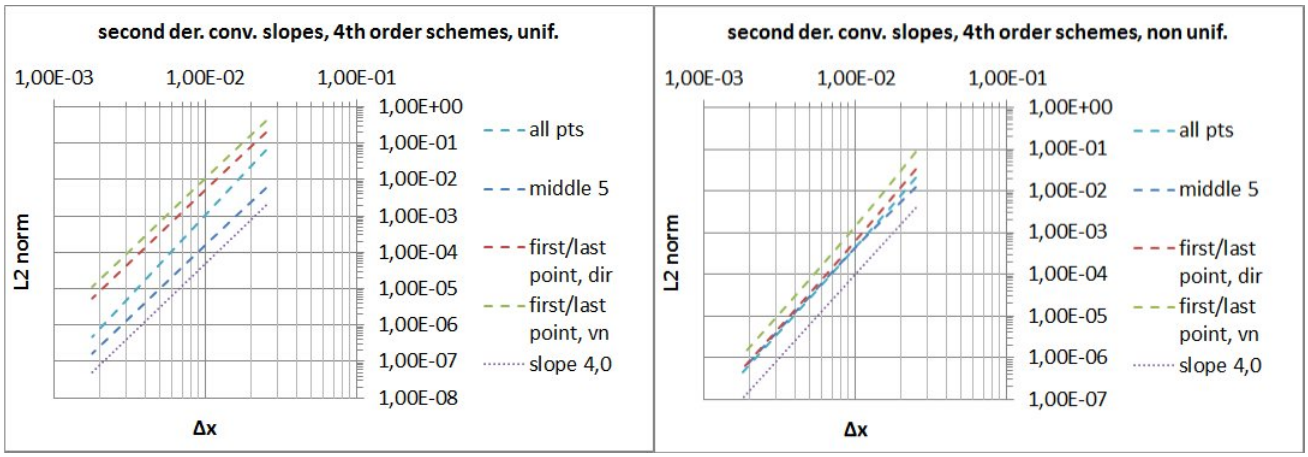


Figure 8.3: Convergence slopes for second derivatives obtained with 4^{th} order schemes in MFLOPS-3D on uniform (left) and non uniform (right) grids.

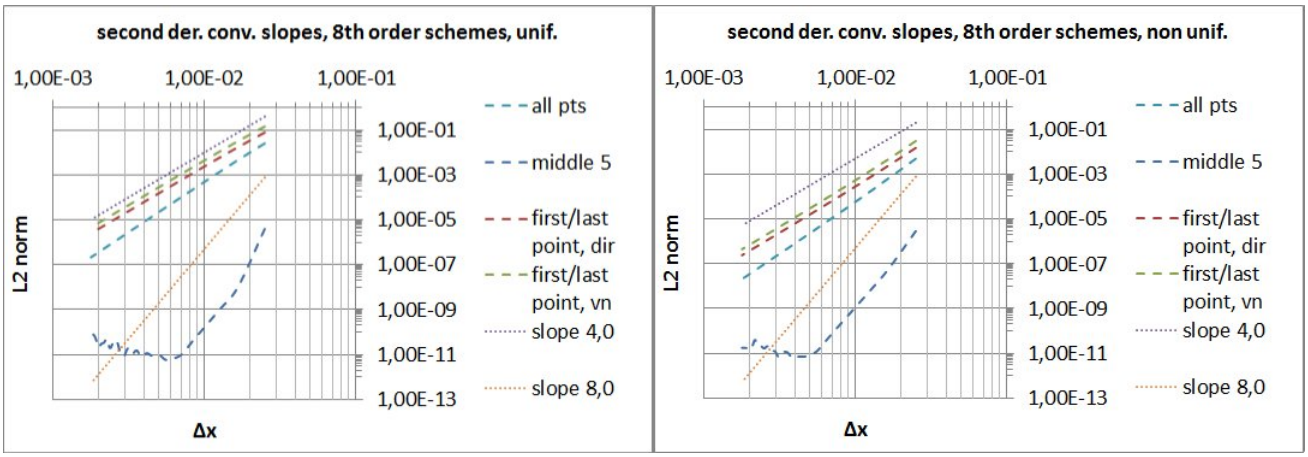


Figure 8.4: Convergence slopes for second derivatives obtained with 8^{th} order schemes in MFLOPS-3D on uniform (left) and non uniform (right) grids.

As it can be seen, schemes referred as 4^{th} order accurate provide overall 4^{th} order accuracy also for these derivatives. And those referred as 8^{th} order accurate again provide this accuracy only in middle points or for central schemes. Use of schemes which enable inclusion of von Neumann conditions shows no effect on convergence and order of accuracy. This is also the reason why only results for first and last points are presented. Change of boundary schemes to include von Neumann conditions namely causes changes for all results, since the complete system of equations composed from the schemes is affected. But since von Neumann results do not affect convergence

in boundary points, they also do not affect it globally in the domain. To conclude, schemes for general second derivative were found to provide equal results, therefore they do not need to be shown.

1.3 First derivative order of accuracy with new schemes for discrete compatibility

Convergence results for this derivative are given in a slightly different manner than before. New schemes have to account for the von Neumann boundary conditions already in the definition of first derivative. Two main groups of results are given on this basis. One shows convergence in case of only Dirichlet conditions present on both sides of the grid line. The other shows results for the case of using von Neumann conditions in same manner. Results are otherwise given as before, for uniform and non uniform grid and with inclusion of different groups of points. Results are given only for 4^{th} order accurate schemes.

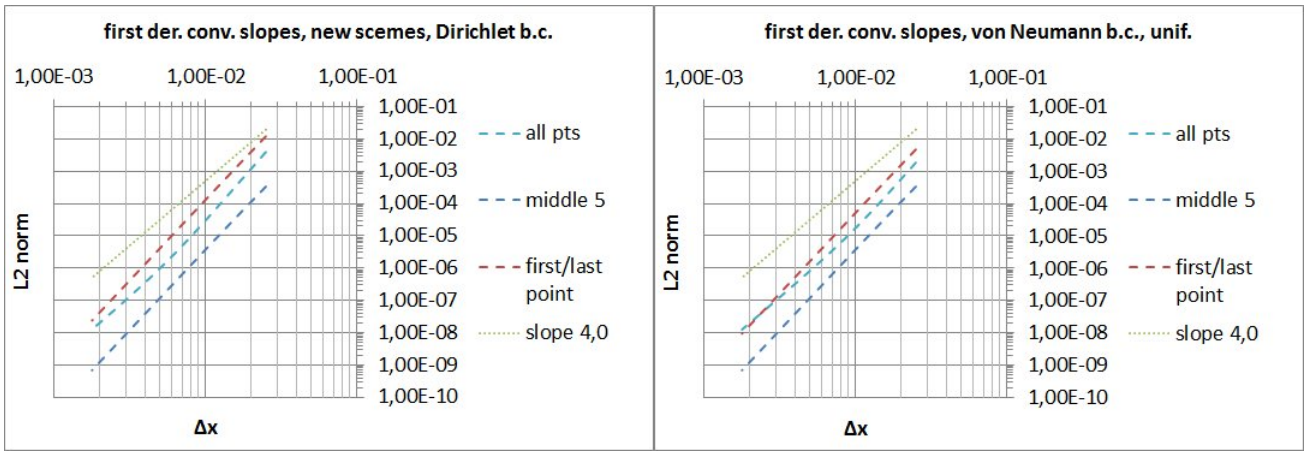


Figure 8.5: Convergence slopes for first derivatives on uniform grid obtained with schemes proposed to ensure discrete compatibility. Left graph shows results with Dirichlet, right with von Neumann conditions.

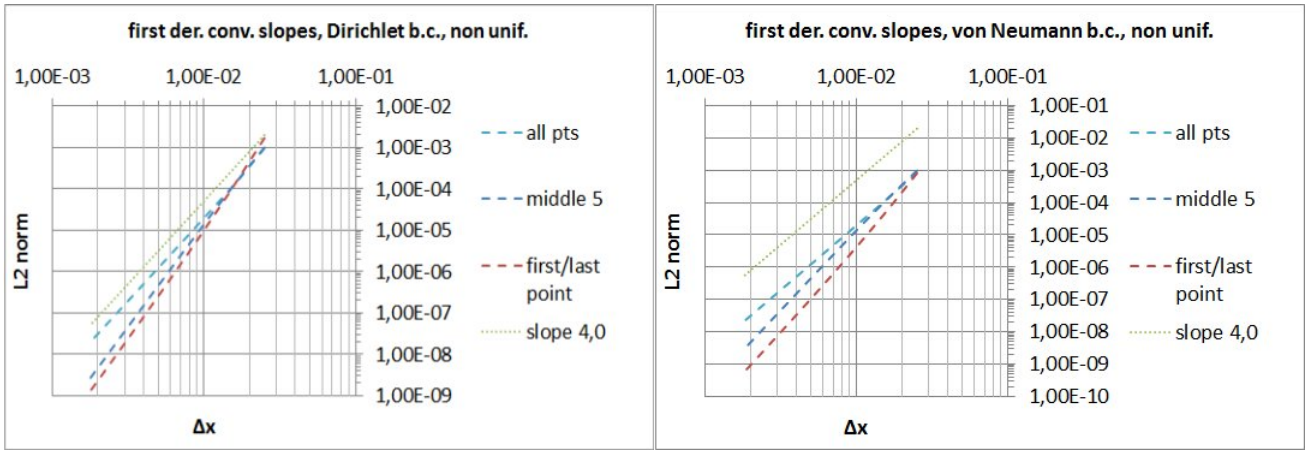


Figure 8.6: Convergence slopes for first derivatives on non uniform grid obtained with schemes proposed to ensure discrete compatibility. Left graph shows results with Dirichlet, right with von Neumann conditions.

Results show that 4^{th} order accuracy is ensured in all points regardless of mesh and boundary conditions used.

1.4 Second derivative order of accuracy with new schemes for discrete compatibility

These results are given in same manner as before presented ones for first derivatives with new schemes, as the effect of boundary conditions was already included in them. Results essentially present convergence of the new Laplace operator terms. As before, 4^{th} order accuracy is ensured in all points regardless of mesh and boundary conditions used.

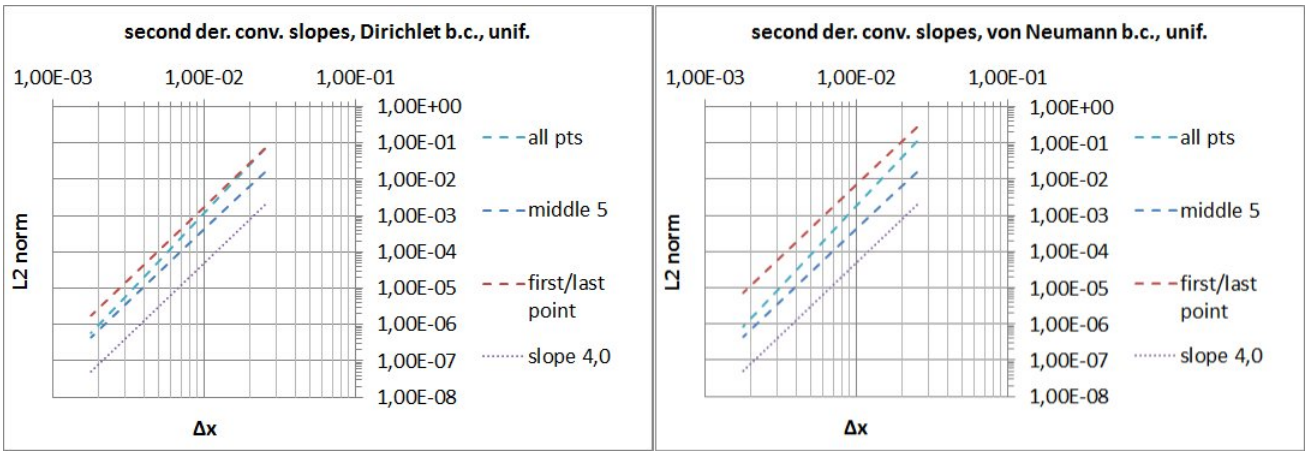


Figure 8.7: Convergence slopes for second derivatives on uniform grid obtained with schemes proposed to ensure discrete compatibility. Left graph shows results with Dirichlet, right with von Neumann conditions.

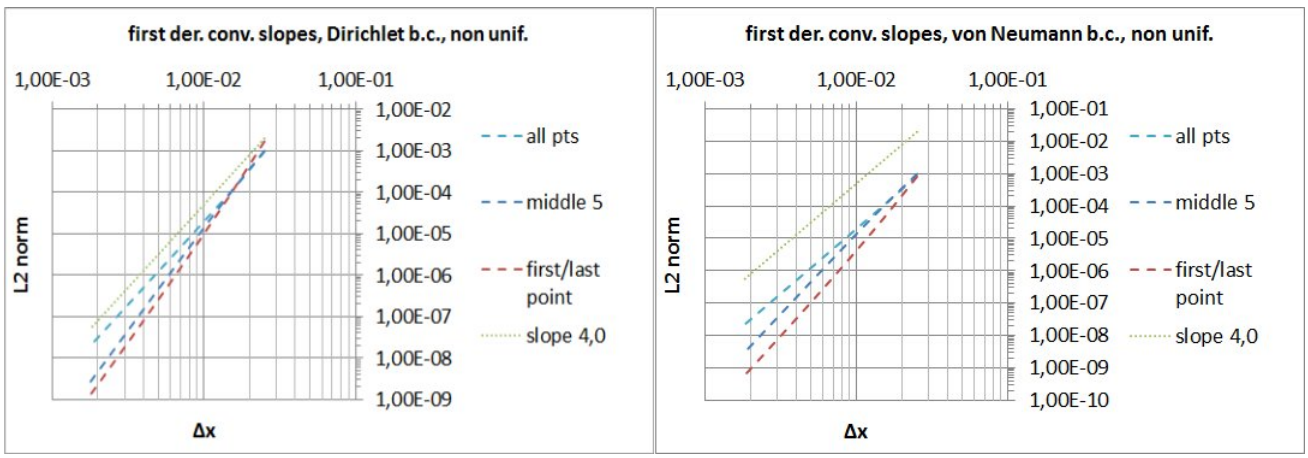


Figure 8.8: Convergence slopes for second derivatives on non uniform grid obtained with schemes proposed to ensure discrete compatibility. Left graph shows results with Dirichlet, right with von Neumann conditions.

2 3D eigendecomposition in MFLOPS-3D

The 3D eigendecomposition in MFLOPS-3D is the key to the direct mono domain solver. Procedure on which it is based is here still much simplified. Some important mathematical operations, which make final diagonalisation possible, are only briefly described. The author is also very grateful to (at the moment still) PhD student in LML laboratory, Ilkay Solak, who contributed a lot of insight into how the procedure of 3D eigendecomposition is applied. The equations which are written also follow from the work of this student and are here written in a more simplified manner than applied by him.

To begin with, a general Helmholtz equation solved with the mono domain solver is repeated with equation (8.3). As written in chapter 3, section 4.1, variables $D_x^{(2)}$, $D_y^{(2)}$ and $D_z^{(2)}$ each represent a 2D matrix composed of coefficients for discretized 2^{nd} derivatives in Laplacian operator. σ is the constant value, and U is the rank-3 tensor of variables we look for. F on RHS is rank-3 tensor following from known values in the domain.

$$D_x^{(2)}U + D_y^{(2)}U + D_z^{(2)}U - \sigma U = F \tag{8.3}$$

Eigendecomposition procedure is applied starting with the equation above. Before going to it, a point given in section about boundary conditions 4.3 of chapter 3 should be repeated. Matrices $D_x^{(2)}$, $D_y^{(2)}$ and $D_z^{(2)}$ in equation (8.3) are not complete. Originally they have as many columns as there are points per certain direction. But since boundary values are known for each sub domain, the products of these columns with corresponding values are included in tensor F. There are of course also no second derivatives written for boundary points. Therefore it follows that for M points in x direction, N in y and P in z, matrix $D_x^{(2)}$ has $(M-2)\times(M-2)$ elements, $D_y^{(2)}$ has $(N-2)\times(N-2)$ and $D_z^{(2)}$ has $(P-2)\times(P-2)$ elements. Tensors U and F have $(M-2)\times(N-2)\times(P-2)$ elements. It is obvious that multiplication of 2D matrices with rank-3 tensor U has to be done carefully for each Laplace operator term since derivatives in different directions are in question. The notations for elements locations are not used here, but the matrices need to multiply tensor U each in its specific direction, where certain term from a matrix can be applied to whole slice or $2D$ matrix inside tensor U. As a consequence, it is possible to write equality in equation (8.4) which importantly affects application of eigendecomposition. $D^{(2)}$ in it represents matrix with coefficients for discretized second derivative in y or z directions while $D^{(2)T}$ is its transpose. This equality is here not explained particularly but rather a simple point for its understanding is given. This is that the rows in tensor U give the position of an element in x direction. Hence multiplications in D_y^2U or D_z^2U need to be specified as multiplications where U elements come from horizontal directions. It then easily follows that $UD_y^{(2)T}$ or $UD_z^{(2)T}$ give same results. Importantly, this is simplified also by applying same matrix $D^{(2)}$ to any line in certain direction of U tensor to obtain desired derivatives, which is a consequence of solver being used strictly on rectangular grids (mapping is used to ensure this).

$$D^{(2)}U = UD^{(2)T} \tag{8.4}$$

As mentioned, the above given equality for y or z direction derivatives affects also application of eigendecomposition in 3D which finally leads to diagonalised system of equations. It was otherwise used in same manner also for eigendecomposition and diagonalisation in 2D cases, which can be seen in [104, 135]. Besides it, three other equalities also need to be used as well and are given in equations (8.5), (8.6) and (8.7). $\mathbf{\Lambda}$ in them presents general diagonal matrix of eigenvalues while S is matrix composed of corresponding eigenvectors.

$$D^{(2)} = S\mathbf{\Lambda}S^{-1} \tag{8.5}$$

$$S^{-1}D^{(2)}S = \mathbf{\Lambda} \tag{8.6}$$

$$D^{(2)T} = (S^{-1})^T\mathbf{\Lambda}S^T \tag{8.7}$$

In order to obtain final diagonalised system of equations, equation (8.3) is at first written as (8.8).

$$D_x^{(2)}U + UD_y^{(2)T} + UD_z^{(2),T} - \sigma U = F \tag{8.8}$$

Then, eigendecomposed $D_x^{(2)}$ is used do apply diagonalisation in x direction by multiplication with S_x^{-1} from the right side. Equation (8.9) follows and if $S_x^{-1}U$ and $S_x^{-1}F$ are written as $\widehat{U}$ and $\widehat{F}$ respectively, equation can be written as (8.10).

$$\mathbf{\Lambda}_x S_x^{-1}U + S_x^{-1}UD_y^{(2)T} + S_x^{-1}UD_z^{(2)T} - S_x^{-1}U\sigma = S_x^{-1}F \tag{8.9}$$

$$\mathbf{\Lambda}_x\widehat{U} + \widehat{U}D_y^{(2)T} + \widehat{U}D_z^{(2)T} - \widehat{U}\sigma = \widehat{F} \tag{8.10}$$

Before the next step, the equality (8.4) is applied to the third term in equation above, to obtain (8.11). This is done as diagonalisation in y direction is done next, by applying multiplication with $(S_y^{-1})^T$ from the left. Equation (8.12) is obtained, where each term on the LHS includes $\widehat{U}(S_y^{-1})^T$ product. This can be written as $\overline{U}$. Equally the product of $\widehat{F}(S_y^{-1})^T$ is written as $\overline{F}$, giving equation (8.13).

$$\mathbf{\Lambda}_x\widehat{U} + \widehat{U}D_y^{(2)T} + D_z^{(2)}\widehat{U} - \widehat{U}\sigma = \widehat{F} \tag{8.11}$$

$$\mathbf{\Lambda}_x\widehat{U}(S_y^{-1})^T + \widehat{U}(S_y^{-1})^T\mathbf{\Lambda}_y + D_z^{(2)}\widehat{U}(S_y^{-1})^T - \widehat{U}(S_y^{-1})^T\sigma = \widehat{F}(S_y^{-1})^T \tag{8.12}$$

$$\mathbf{\Lambda}_x\overline{U} + \overline{U}\mathbf{\Lambda}_y + D_z^{(2)}\overline{U} - \overline{U}\sigma = \overline{F} \tag{8.13}$$

Finally, using equality (8.4) for the third term again, equation (8.14) follows, to which diagonalisation in z direction can be applied by multiplication with $(S_z^{-1})^T$. Equation (8.15) is obtained.

$$\mathbf{\Lambda}_x\overline{U} + \overline{U}\mathbf{\Lambda}_y + \overline{U}D_z^{(2)T} - \overline{U}\sigma = \overline{F} \tag{8.14}$$

$$\mathbf{\Lambda}_x\overline{U}(S_z^{-1})^T + \overline{U}(S_z^{-1})^T\mathbf{\Lambda}_y + \overline{U}(S_z^{-1})^T\mathbf{\Lambda}_z - \overline{U}(S_z^{-1})^T\sigma = \overline{F}(S_z^{-1})^T \tag{8.15}$$

The starting system of equations is with the equation above finally diagonalised. Left side includes terms with $\overline{U}(S_z^{-1})^T$, which equals $S_x^{-1}U(S_y^{-1})^T(S_z^{-1})^T$, while on the right we have $S_x^{-1}F(S_y^{-1})^T(S_z^{-1})^T$. Rearranging terms on the left we obtain equation (8.16), which is the final diagonalised system of equations. Solution in certain point m, n, p can with it be directly obtained following equation (8.17), where $\bar{u}_{m,n,p}$ and $\bar{f}_{m,n,p}$ are point values from U and F tensors multiplied with inverse eigenvector matrices while $\Lambda_{x,m}$, $\Lambda_{y,n}$ and $\Lambda_{z,p}$ are eigenvalues in the corresponding point m, n, p.

$$(\mathbf{\Lambda}_x + \mathbf{\Lambda}_y + \mathbf{\Lambda}_z - \sigma)\, S_x^{-1} U (S_y^{-1})^T (S_z^{-1})^T = S_x^{-1} F (S_y^{-1})^T (S_z^{-1})^T \tag{8.16}$$

$$(\Lambda_{x,m} + \Lambda_{y,n} + \Lambda_{z,p} - \sigma)\, \bar{u}_{m,n,p} = \bar{f}_{m,n,p} \tag{8.17}$$

Since the equation above provides values forming $S_x^{-1} U (S_y^{-1})^T (S_z^{-1})^T$ tensor, the final solution for searched values in tensor U follows from multiplication with appropriate eigenvectors.

As mentioned, matrix tensor multiplications, such as this last one including $S_x^{-1} U (S_y^{-1})^T (S_z^{-1})^T$ tensor, need to be done respecting directions of derivatives from which 2D matrices with their eigenvectors and eigenvalues follow. No notation was used in here given explanation to denote this. MFLOPS-3D uses the mentioned LAPACK procedures (*dgemm* routine) which enable easier handling of such matrix-tensor operations.

3 Proof for used connections between p and Φ

This part of the appendix deals with the proof of the extensively used connection between pressure p and intermediate variable Φ. This connection is an important one as it was the first key to stable calculation of pressure for both versions of the projection method used. Indeed it is in the end used only for the pressure non incremental version, but its existence is nevertheless important also for the pressure incremental version. The reason is that the connection includes viscous terms effects, which are in the shown pressure incremental versions of the algorithm neglected. This, correspondingly to description and results in [80], causes a numerical boundary layer in pressure solution and hence lowers the accuracy of pressure near boundaries. Furthermore it also causes occurrence of oscillations. Existence and proof for the complete Φ - p connection therefore means that this boundary layer could be eliminated also in pressure incremental versions of the algorithm, if the viscous terms could be somehow applied without affecting the stability of calculations. This was in this work not done and remains a potential improvement.

The proof for the used Φ - p connection is built on the one for incompressible flow in [80] and is given for the general case of projection method. That is, for pressure incremental version. The non incremental demands only the pressure from previous level to be omitted. This previous level can be time or iteration. For explanation here, time levels are used. Furthermore, since there are two intermediate variables developed, the usual Φ and potentially better Φ_2, the explanation is first given for the connection Φ - p and then for Φ_2 - p.

3.1 Φ - p connection proof

Three equations need to be restated first to begin the proof of this connection. These are the momentum equations for $\vec{v}$, $\vec{v}^*$ and projection equation. Momentum equations are given in the form of Helmholtz equation with non-constant coefficients and discretized time derivatives. No CG method is used in equations for $\vec{v}^*$ since converged result is assumed, meaning there is no difference between $\vec{v}^*$ and $\vec{v}_e^*$. The NL terms in both equations are not written with skew-symmetric form for simplicity. Inclusion of this form on the other hand does not change the end result of this proof. Projection equation, which follows as the difference between momentum equations, is given including connections between Φ and velocities or pressure.

$$\left(\Delta - \frac{3\rho}{2\Delta t\mu}\right)\vec{v}^{n+1} = \rho\frac{-4\vec{v}^{n} + \vec{v}^{n-1}}{2\Delta t\mu} + \frac{\rho}{\mu}((\vec{v}\cdot\nabla)\vec{v})^{n,n-1} + \frac{\nabla p^{n+1}}{\mu} - \frac{1}{\mu}(\nabla\mu)\cdot(\nabla\vec{v}^{n,n-1}) - \frac{1}{\mu}(\nabla\mu)\cdot(\nabla\vec{v}^{n,n-1})^{T} - \frac{1}{3}\nabla(\nabla\cdot\vec{v}^{n,n-1}) + \frac{2}{3\mu}\nabla\mu(\nabla\cdot\vec{v}^{n,n-1}) \tag{8.18}$$

$$\left(\Delta - \frac{3\rho}{2\Delta t\mu}\right)\vec{v}^{*} = \rho\frac{-4\vec{v}^{n} + \vec{v}^{n-1}}{2\Delta t\mu} + \frac{\rho}{\mu}((\vec{v}\cdot\nabla)\vec{v})^{n,n-1} + \frac{\nabla p^{n}}{\mu} - \frac{1}{\mu}(\nabla\mu)\cdot(\nabla\vec{v}^{n,n-1}) - \frac{1}{\mu}(\nabla\mu)\cdot(\nabla\vec{v}^{n,n-1})^{T} - \frac{1}{3}\nabla(\nabla\cdot\vec{v}^{n,n-1}) + \frac{2}{3\mu}\nabla\mu(\nabla\cdot\vec{v}^{n,n-1}) \tag{8.19}$$

$$-\nabla\Phi = \frac{3\rho}{2\Delta t}(\vec{v}^{n+1} - \vec{v}^{*}) = -\nabla p^{n+1} + \nabla p^{n} + \mu\Delta(\vec{v}^{n+1} - \vec{v}^{*}) \tag{8.20}$$

As written in chapter 4, section 2.2.1, the needed connection between Φ and p without gradient operators is at first simply proposed to be given with equation (8.21).

$$p^{n+1} = \Phi + p^{n} + \mu\nabla\cdot(\vec{v}^{n+1} - \vec{v}^{*}) \tag{8.21}$$

In order to prove it, gradient operator is at first applied to this connection again. If connection holds, the ultimate results should be the connection in equation (8.20). At first, expression (8.22) is obtained.

$$\nabla p^{n+1} = \nabla\Phi + \nabla p^{n} + (\nabla\mu)\nabla\cdot(\vec{v}^{n+1} - \vec{v}^{*}) + \mu\nabla\nabla\cdot(\vec{v}^{n+1} - \vec{v}^{*}) \tag{8.22}$$

This expression is different from the one in (8.20). By applying the known connection $\Delta a = \nabla\nabla\cdot a - \nabla\times\nabla\times a$, as it is also done in [80] for incompressible flow case, it changes to (8.23).

$$\nabla p^{n+1} = \nabla\Phi + \nabla p^{n} + (\nabla\mu)\nabla\cdot(\vec{v}^{n+1} - \vec{v}^{*}) + \mu\Delta(\vec{v}^{n+1} - \vec{v}^{*}) + \mu\nabla\times\nabla\times(\vec{v}^{n+1} - \vec{v}^{*}) \tag{8.23}$$

It follows that we need to show $(\nabla\mu)\nabla\cdot(\vec{v}^{n+1} - \vec{v}^{*}) + \mu\nabla\times\nabla\times(\vec{v}^{n+1} - \vec{v}^{*}) = 0$. To do this, we need to sum (8.19) and connection between Φ and velocities in (8.20). Equation (8.24) is obtained, where term $f(t^{n,n-1})$ stands for explicitly treated terms on the RHS of equation (8.19). As it can be seen, these are in (8.19) and (8.18) equal with the exception of pressure gradient term.

$$\frac{\rho}{2\Delta t}(3\vec{v}^{n+1} - 4\vec{v}^{n} + \vec{v}^{n-1}) - \mu\Delta\vec{v}^{*} + f(t^{n,n-1}) + \nabla\Phi + \nabla p^{n} = 0 \tag{8.24}$$

Term $\nabla\Phi$ in the equation above can be then replaced with the one expressed from (8.23) for which we want to show that the two additional terms add to nothing. Equation (8.25) is obtained.

$$\frac{\rho}{2\Delta t}(3\vec{v}^{n+1} - 4\vec{v}^{n} + \vec{v}^{n-1}) + \nabla p^{n+1} + f(t^{n,n-1}) - \mu\Delta v^{n+1} = (\nabla\mu)\nabla\cdot(\vec{v}^{n+1} - \vec{v}^{*}) + \mu\nabla\times\nabla\times(\vec{v}^{n+1} - \vec{v}^{*}) \tag{8.25}$$

The LHS of this last equation is nothing else than complete equation (8.18). Therefore the complete RHS, which equals the two additional and unwanted terms from (8.23), equals zero. Which gives the proof that connection between Φ and p, given with equation (8.21), is correct.

3.2 Φ_2 - p connection proof

The proof of this connection follows same procedure as given in previous case. The second possible projection equation, because of which this proof is needed, is here written with equation (8.26). In it, all connections Φ_2 exhibits (with Φ, velocities and pressure) in differential form are included.

$$-\nabla\Phi_2 = -\frac{\nabla\Phi}{\rho} = \frac{3}{2\Delta t}(\vec{v}^{n+1} - \vec{v}^*) = \frac{-\nabla p^{n+1} + \nabla p^n}{\rho} + \frac{\mu}{\rho}\Delta(\vec{v}^{n+1} - \vec{v}^*) \tag{8.26}$$

The first step is again to propose connection between Φ_2 and pressure directly. Removal of gradient operator as done before leads to equation (8.27).

$$\Phi_2 = \frac{\Phi}{\rho} = \frac{p^{n+1} - p^n}{\rho} - \frac{\mu}{\rho}\nabla\cdot(\vec{v}^{n+1} - \vec{v}^*) \tag{8.27}$$

To prove this connection is correct, the connection itself is first multiplied with ρ. This makes it easier applicable to same procedure as done for the first proven connection. Equation (8.28) follows this.

$$\rho\Phi_2 = \Phi = p^{n+1} - p^n - \mu\nabla\cdot(\vec{v}^{n+1} - \vec{v}^*) \tag{8.28}$$

If expression (8.28) has gradient operator applied, one obtain equation (8.29). This is not equal to the expression one would obtain if equation (8.26) was multiplied by ρ. The connection $\Delta a = \nabla\nabla\cdot a - \nabla\times\nabla\times a$ can be used in it again to obtain equation (8.30).

$$\Phi_2\nabla\rho + \rho\nabla\Phi_2 = \nabla p^{n+1} - \nabla p^n - (\nabla\mu)\nabla\cdot(\vec{v}^{n+1} - \vec{v}^*) - \mu\nabla\nabla\cdot(\vec{v}^{n+1} - \vec{v}^*) \tag{8.29}$$

$$\Phi_2\nabla\rho + \rho\nabla\Phi_2 = \nabla p^{n+1} - \nabla p^n - (\nabla\mu)\nabla\cdot(\vec{v}^{n+1} - \vec{v}^*) - \mu\Delta(\vec{v}^{n+1} - \vec{v}^*) - \mu\nabla\times\nabla\times(\vec{v}^{n+1} - \vec{v}^*) \tag{8.30}$$

From equation (8.30) it follows that one needs to show that $\Phi_2\nabla\rho + (\nabla\mu)\nabla\cdot(\vec{v}^{n+1} - \vec{v}^*) + \mu\nabla\times\nabla\times(\vec{v}^{n+1} - \vec{v}^*) = 0$. This is done like before by first summing the projection equation with momentum equation for $\vec{v}^*$. However, the projection equation (8.26) has to be first multiplied by ρ in order to make summation of velocity time derivatives terms possible. Result of this procedure is equation (8.31), where $f(t^{n,n-1})$ again represents explicitly treated terms on the RHS of equation (8.19).

$$\mu\Delta\vec{v}^* - \rho\nabla\Phi_2 = \frac{\rho}{2\Delta t}(3\vec{v}^{n+1} - 4\vec{v}^n + \vec{v}^{n-1}) + \nabla p^n + f(t^{n,n-1}) \tag{8.31}$$

If $\rho\nabla\Phi_2$ expressed from (8.30) is used in equation above, the end result again leads to the equation where the LHS is the NS momentum equation for velocity $\vec{v}$ while RHS includes the terms whose sum we want to be equal to zero. This is here given with equation (8.32), which corresponds to given description. Hence the proof for the used Φ_2 - p connection is given and also confirms that the chosen second projection equation is suitable for use.

$$\begin{aligned}&\frac{\rho}{2\Delta t}(3\vec{v}^{n+1} - 4\vec{v}^n + \vec{v}^{n-1}) + \nabla p^{n+1} + f(t^{n,n-1}) - \mu\Delta v^{n+1} = \\ &\Phi_2\nabla\rho + (\nabla\mu)\nabla\cdot(\vec{v}^{n+1} - \vec{v}^*) + \mu\nabla\times\nabla\times(\vec{v}^{n+1} - \vec{v}^*)\end{aligned} \tag{8.32}$$

4 Used forcing terms from analytical solutions for incompressible flow

This part of the appendix gives an example for forcing terms which follow from the presented analytical incompressible flow expressions in chapter 5, section 3.1. The forcing terms are obtained by inclusion of those analytical expressions into equation (5.2) given in section 2 of same chapter. They are then used in MFLOPS-3D as depicted with equation (5.3). Expressions (8.33), (8.34) and (8.35) define the forcing term for NS momentum equations for u^*, v^* and w^* velocities, respectively.

$$\begin{aligned} F_{t,u} = &- aC^2\cos^2(gt)\sin(ax)\sin(ay) + aC^2\cos^2(gt)\cos(ax)\cos(ay) \\ &- Cg\sin(gt)\cos(ay) + \frac{C}{Re}a^2\cos(gt)\cos(ay) \end{aligned} \tag{8.33}$$

$$\begin{aligned} F_{t,v} = &- C^2a\cos^2(gt)\sin(ax)\sin(ay) + C^2a\cos^2(gt)\cos(ax)\cos(ay) \\ &- Cg\sin(gt)\sin(ax) + \frac{C}{Re}a^2\cos(gt)\sin(ax) \end{aligned} \tag{8.34}$$

$$F_{t,w} = 0 \tag{8.35}$$

Bibliography

[1] F.R. Young. *Cavitation*. McGraw-Hill Book Company Ltd, 1989.

[2] B. Širok, M. Dular, and B. Stoffel. *Kavitacija*. i2, Ljubljana, 2006.

[3] D. Liuzzi. *Two-Phase Cavitation Modelling*. PhD thesis, Faculty of Civil and Industrial Engineering, University of Rome, June 2012.

[4] M. Frobenius. *Numerische Simulation kavitierenden Stromungen in hydraulischen Stromungsmachinen*. PhD thesis, Fakultat fur Mashinenwesen der Technischen Universitat Munchen, December 2003.

[5] I. Senoçak. *Computational Methodology for the Simulation of Turbulent Cavitating Flows*. PhD thesis, University of Florida, 2002.

[6] G. D. Ciocan and M. S. Iliescu. Vortex rope investigation by 3d-piv method. In *Proc. 2nd Int. Meeting of Workgroup on Cavitation and Dynamic Problems in Hydraulic Machinery and Systems, Timisoara*, 2007.

[7] C. E. Brennen. *Cavitation and Bubble Dynamics*. Oxford University Press, 1995.

[8] J. Sauer. *Instationar kavitierende Stromungen–Ein neues Modell, basierend auf Front Capturing (VoF) und Blasendynamik*. PhD thesis, Fakultat fur Mashinenbau der Universitat Karlsruhe (TH), July 2000.

[9] Y. Chen and S. D. Heister. A numerical treatment for attached cavitation. *Journal of Fluids Engineering*, 116:613–618, 1994.

[10] P. Dupont and F. Avellan. Numerical computation of a leading edge cavity. In *Proceedings of the Cavitation 91 Symposium*. ASME-JSME, June 1991.

[11] V. Schütte. Modellierung und berechnung von blasenströmungen und kavitation in laufradkanälen von kreiselpumpen. In *Pumpentagung Karlsruhe '92, Karlsruhe*, 1992.

[12] R. Samulyak, Y. Prykarpatskyy, T. Lu, J. Glimm, Z. Xu, and M. Kim. Comparison of heterogeneous and homogenized numerical models of cavitation. *International Journal for Multiscale Computational Engineering*, 4(3):377–389, 2006.

[13] T. Lu, R. Samulyak, and J. Glimm. Direct numerical simulation of bubbly flows and application to cavitation mitigation. *Journal of Fluids Engineering*, 129:595–604, 2006.

[14] A. Alajbegovic, G. Meister, D. Greif, and B. Basara. Three phase cavitating flows in high-pressure swirl injectors. *Experimental Thermal and Fluid Science*, 26:677–681, 2002.

[15] A. Kubota, H. Kato, and H. Yamaguchi. A new modelling of cavitating flows: a numerical study of unsteady cavitation on a hydrofoil section. *Journal of Fluid Mechanics*, 240:59–96, 1992.

[16] Y. Chen and S. D. Heister. Two-phase modeling of cavitated flows. *Computers and Fluids*, 24(7):799–809, 1995.

[17] Y. Delannoy and J. L. Kueny. Cavity flow predictions based on the euler equation. In *ASME Cavitation and Multiphase Flow Forum*, 1990.

[18] O. Coutier-Delgosha, S. Frikha, and J.A. Astolfi. Influence of the cavitation model on the simulation of cloud cavitation on 2d foil section. *International Journal of Rotating Machinery*, 2008, 2009.

[19] J.L. Reboud, B. Stutz, and O. Coutier-Delgosha. Two-phase flow structure of cavitation: Experiment and modeling of unsteady effects. In *Third International Symposium on Cavitation, Grenoble*, April 1998.

[20] O. Coutier-Delgosha, R. Fortes-Patella, J.L. Reboud, N. Hakimi, and C. Hirsch. Numerical simulation of cavitating row in 2d and 3d inducer geometries. *International Journal for Numerical Methods in Fluids*, 48:135–167, 2005.

[21] D.P. Schmidt, C. J. Rutland, and M. L. Corradini. A numerical study of cavitating flow through various nozzle shapes. In *SAE Technical Paper*. SAE International, May 1997.

[22] D.R. van der Heul, C. Vuik, and P. Wesseling. Efficient computation of flow with cavitation by compressible pressure correction. In *European Congress on Computational Methods in Applied Sciences and Engineering, ECCOMAS 2000, Barcelona*. ECCOMAS, September 2000.

[23] Y. Ventikos and G. Tzabiras. A numerical method for the simulation of steady and unsteady cavitating flows. *Computers and Fluids*, 29:63–88, 2000.

[24] E. Rapposelli and L. d'Agostino. A barotropic cavitation model with thermodynamic effects. In *Fifth International Symposium on Cavitation (cav2003), Osaka*, November 2003.

[25] E. Goncalvès and R. F. Patella. Numerical simulation of cavitating flows with homogeneous models. *Computers and Fluids*, 38:1682–1696, 2009.

[26] B. Charrière, J. Decaix, and E. Goncalvès. A comparative study of cavitation models in a venturi flow. *European Journal of Mechanics B/Fluids*, 49:287–297, 2015.

[27] C.L. Merkle, J. Feng, and P.E.O. Buelow. Computational modeling of dynamics of sheet cavitation. In *Third International Symposium on Cavitation, Grenoble*, April 1998.

[28] N. Singhal, A.K. Vaidya and A.D. Leonard. Multi-dimensional simulation of cavitating flows using a pdf model for phase change. In *ASME FED Meeting, Vancouver*, 1997.

[29] R. F. Kunz, D. A. Boger, D. R. Stinebring, T. S. Chyczewskia, J. W. Lindaua, H. J. Gibelinga, S. Venkateswaranb, and T. R. Govindan. A preconditioned navier-stokes method for two-phase flows with application to cavitation prediction. *Computers and Fluids*, 29:849–875, 2000.

[30] Y. Utturkar. *Computational Modeling of Thermodynamic Effects in Cryogenic Cavitation*. PhD thesis, University of Florida, 2005.

[31] A. Hosangadi, V. Ahuja, and R. Ungewitter. Simulations of cavitating cryogenic inducers. In *40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, Fort Lauderdale, Florida*, July 2004.

[32] D. Dauby, P. Queutey, A. Leroyer, and M. Visonneau. Computation of 2d cavitating flows and tip vortex flows with an unstructured rans solver. In *11èmes journé es de l'hydrodynamique Brest*, April 2007.

[33] A.K. Singhal, L. Huiying, M.A. Mahesh, and Y. Jiang. Mathematical basis and validation of the full cavitation mode. In *Proceedings of ASME FEDSM'01*. ASME, May-June 2001.

[34] M. Athavale and A. Singhal. Numerical analysis of cavitating flows in rocket turbopump elements. In *37th Joint Propulsion Conference and Exhibit. Salt Lake City*, 2001.

[35] P.J. Zwart, A.G. Gerber, and T. Belamri. A two-phase flow model for predicting cavitation dynamics. In *International Conference on Multiphase Flow, Yokohama*, May 2004.

[36] C. F. Delale, G. H. Schnerr, and J. Sauer. Quasi-one-dimensional steady-state cavitating nozzle flows. *Journal of Fluid Mechanics*, 427:167–204, 2001.

[37] T. Xing, Z. Li, and S. H. Frankel. Numerical simulation of vortex cavitation in a three-dimensional submerged transitional jet. *Journal of Fluids Engineering*, 127:714–725, 2005.

[38] *Ansys Fluent 12.0/12.1 Documentation*, 12.1 edition.

[39] J. Zhu, Y. Chen, D. Zhao, and X. Zhang. Extension of the schnerr-sauer model for cryogenic cavitation. *European Journal of Mechanics B/Fluids*, 52:1–10, 2015.

[40] A. Žnidarčič, R. Mettin, and M. Dular. Modeling cavitation in a rapidly changing pressure field - application to a small ultrasonic horn. *Ultrasonics Sonochemistry*, 22:482–492, 2015.

[41] *ANSYS CFX-Solver Theory Guide*, 15.0 edition.

[42] B. Stutz and J.L. Reboud. Two-phase flow structure of sheet cavitation. *Physics of Fluids*, 9:3678–3686, 1997.

[43] B. Stutz and J.L. Reboud. Experiments on unsteady cavitation. *Experiments in Fluids*, 22:191–198, 1997.

[44] B. Stutz and J.L. Reboud. Measurements within unsteady cavitation. *Experiments in Fluids*, 29:545–552, 2000.

[45] Y. Kuhn de Chizelle, S.L. Ceccio, and C.E. Brennen. Observations and scaling of traveling bubble cavitation. *Journal of Fluid Mechanics*, 293:96–126, 1995.

[46] C. Y. Li and S. L. Ceccio. Interaction of single travelling bubbles with the boundary layer and attached cavitation. *Journal of Fluid Mechanics*, 322:329–353, 1996.

[47] S. K. Wang, S. J. Lee, O. C. Jones, and R. T. Lahey. 3-d turbulence structure and phase distribution measurements in bubbly two-phase flow. *International Journal of Multiphase Flow*, 13:327–343, May 1987.

[48] M. Lance and J. Bataille. Turbulence in the liquid phase of a uniform bubbly air-water flow. *Journal of Fluid Mechanics*, 222:95–118, 1991.

[49] J.-Y. Billard, P. Galivel, , and D. H. Fruman. Effect of preturbulence on the cavitating bubble inception. In *Proceedings of ASME FED*, 1993.

[50] T. Baur and J. K. Ngeter. Measurements in the shear-layer behind a surface-mounted obstacle for the determination of coherent structures. In *Ninth Int. Symp. on Applications of Laser Techniques to Fluid Mechanics*, 1998.

[51] S. Gopalan, J. Katz, and O. Knio. The flow structure in the near field of jets and its effects on cavitation inception. *Journal of Fluid Mechanics*, 398:1–43, 1999.

[52] Roger E.A. Arndt. Cavitation in vortical flows. *Annual Review of Fluid Mechanics*, 34(1):143–175, 2002.

[53] V. Aeschlimann, S. Barre, and H. Djeridi. Unsteady cavitation analysis using phase averaging and conditional approaches in a 2d venturi flow. *Open Journal of Fluid Dynamics*, 3:171–183, 2013.

[54] R. T. Knapp. Recent investigations of the mechanics of cavitation and cavitation damage. *Transactions ASME*, 77:1045–1054, 1955.

[55] F. Joussellin, Y. Delannoy, E. Sauvage-Boutar, and B. Goirand. Experimental investigations on unsteady attached cavities. In *Proceedings of ASME FED*, 1991.

[56] Q. Le, J.P. Franc, and J.M. Michel. Partial cavities: global behaviour and mean pressure distribution. *Journal of Fluids Engineering*, 115:243–248, 1993.

[57] Y. Kawanami, H. Kato, M. Tanimura, and Y. Tagaya. Mechanism and control of cloud cavitation. *Journal of Fluids Engineering*, 119:788–794, 1997.

[58] M. Hofmann, H. Lohrberg, G. Ludwig, B. Stoffel, J.L. Reboud, and R. Fortes-Patella. Numerical and experimental investigations on the self-oscillating behaviour of cloud cavitation - part i: Visualisation. In *Proceedings of ASME FEDSM, San Francisco*, 1999.

[59] A. Kubota, H. Kato, H. Yamaguchi, and M. Maeda. Unsteady structure measurement of cloud cavitation on a foil section using conditional sampling technique. *Journal of Fluids Engineering*, 111:204–210, 1989.

[60] S. Gopalan and J. Katz. Flow structure and modeling issues in the closure region of attached cavitation. *Physics of Fluids*, 12, 2000.

[61] M. Callenaere, J.P. Franc, J.M. Michel, and M. Riondet. The cavitation instability induced by the development of a re-entrant jet. *Journal of Fluid Mechanics*, 444:223–256, 2001.

[62] V. Aeschlimann, S. Barre, and H. Djeridi. Velocity field analysis in an experimental cavitating mixing layer. *Physics of Fluids*, 23, 2011.

[63] V. Aeschlimann, S. Barre, and S. Legoupil. X-ray attenuation measurements in a cavitating mixing layer for instantaneous twodimensional void ratio determination. *Physics of Fluids*, 23, 2011.

[64] O. Coutier-Delgosha, M. Hocevar, I. Khlifa, and S. Fuzier. Velocity measurements in cavitating flows by fast x-ray. In *ISROMAC-14*, February 2012.

[65] I. Khlifa, S. Fuzier, O. Roussette, A. Vabre, M. Hočevar, K. Fezzaa, and O. Coutier-Delgosha. Fast x-ray imaging of cavitating flows. *Journal of Fluid Mechanics*, in press 2017.

[66] O. Coutier-Delgosha, R. Fortes-Patella, and J.L. Reboud. Evaluation of the turbulence model influence on the numerical simulations of unsteady cavitation. *Journal of Fluids Engineering*, 125:38–45, 2003.

[67] M. Morgut, E. Nobile, and I. Biluš. Comparison of mass transfer models for the numerical prediction of sheet cavitation around a hydrofoil. *International Journal of Multiphase Flow*, 37:620–626, 2011.

[68] O. Coutier-Delgosha. *Modélisation des écoulements cavitants : étude des comportements instationnaires et application tridimensionnelle aux turbomachines.* PhD thesis, Grenoble Institute of Technology, INPG, 2001.

[69] R.E.A. Arndt, C.C.S. Song, M. Kjeldsen, J. He, and A. Keller. Instability of partial cavitation: A numerical/experimental approach. In *Twenty-Third Symposium on Naval Hydrodynamics. Washington, DC.* The National Academies Press, 2001.

[70] N. Dittakavi, A. Chunekar, and S. Frankel. Large eddy simulation of turbulent-cavitation interactions in a venturi nozzle. *Journal of Fluids Engineering*, 132, December 2010.

[71] B. Ji and X.W. Luo. Three-dimensional large eddy simulation and vorticity analysis of unsteady cavitating flow around a twisted hydrofoil. *Journal of Hydrodynamics, Ser. B*, 25:510–519, September 2013.

[72] K. Okabayashi, T. Kajishima, and T. Ohta. Dns and les of cavitating turbulent flow. In *Sixth International Symposium on Turbulence and Shear Flow Phenomena, Seoul*, June 2009.

[73] M. Mattson and K. Mahesh. Simulation of bubble migration in a turbulent boundary layer. *Physics of Fluids*, 23, April 2011.

[74] G.H. Oweis, I.E. van der Hout, C. Iyer, G. Tryggvason, and S.L. Ceccio. Capture and inception of bubbles near line vortices. *Physics of Fluids*, 17, 2005.

[75] J.H. Seo, Y.J. Moon, and B.R. Shin. Prediction of cavitating flow noise by direct numerical simulation. *Journal of Computational Physics*, 227:6511–6531, 2008.

[76] Y. Bac, J.H. Sco, Y.J. Moon, and B.R. Shin. *Direct Numerical Simulation of Cavitation Noise for a 3D Circular Cylinder Cross-Flow*, pages 605–610. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[77] U. Piomelli. Large-eddy and direct simulation of turbulent fows. Technical report, Department of Mechanical Engineering, University of Maryland, 2001.

[78] M. Marquillie, U. Ehrenstein, and J.P. Laval. Instability of streaks in wall turbulence with adverse pressure gradient. *Journal of Fluid Mechanics*, 681:205–240, 2011.

[79] R.S. Brodkey. *The Phenomena of Fluid Motions*. General Publishing Company, Ltd., 2005.

[80] J.L. Guermond, P. Minev, and J. Shen. An overview of projection methods for incompressible flows. *Comput. Methods Appl. Mech. Engrg.*, 195:6011–6045, 2006.

[81] J.H. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics*. Springer, 3 edition, November 2001.

[82] D. L. Brown. Accurate projection methods for the incompressible navier-stokes equations. *Journal of Computational Physics*, 168:464–499, 2001.

[83] J. Kim and P. Moin. Application of a fractional-step method to incompressible navier-stokes equations. *Journal of Computational Physics*, 59:308–323, 1985.

[84] A.J. Chorin. Numerical solution of the navier-stokes equations. *Math. Comput.*, 22:745–762, 1968.

[85] R. Temam. Sur l'approximation de la solution des equations de navier-stokes par la methode des pas fractionnaires ii. *Arch. Ration. Mech. Anal.*, 33:377–385, 1969.

[86] P. Moin, , and K. Mahesh. Direct numerical simulation: A tool in turbulence research. *Annual Review of Fluid Mechanics*, 30(1):539–578, 1998.

[87] S. Abide and S. Viazzo. A 2d compact fourth-order projection decomposition method. *Journal of Computational Physics*, 206:252–276, 2005.

[88] G. Dahlquist and A. Bjorck. *Numerical Methods*. Prentice-Hall, dover edition, 1974.

[89] M. Marquillie and U. Ehrenstein. Numerical simulation of separating boundary-layer flow. *Computers and Fluids*, 31:683–693, 2002.

[90] M. Marquillie, J.P. Laval, and R. Dolganov. Direct numerical simulation of a separated channel flow with a smooth profile. *Journal of Turbulence*, 9:1–23, 2008.

[91] M. Marquillie and U. Ehrenstein. On the onset of nonlinear oscillations in a separating boundary-layer flow. *Journal of Fluid Mechanics*, 490:169–188, September 2003.

[92] F. Gallaire, M. Marquillie, and U. Ehrenstein. Three-dimensional transverse instabilities in detached boundary layers. *Journal of Fluid Mechanics*, 571:221–233, 2007.

[93] L. Kuban, J.P. Laval, W. Elsner, A. Tyliszczak, and M. Marquillie. Les modeling of converging-diverging turbulent channel flow. *Journal of Turbulence*, 13(11):1–19, 2012.

[94] S.K. Lele. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103:16–42, 1992.

[95] S. Nagarajan, S.K Lele, and J.H. Ferziger. A robust high-order compact method for large eddy simulation. *Journal of Computational Physics*, 191:392–419, 2003.

[96] R.K. Shukla and X. Zhong. Derivation of high-order compact finite difference schemes for non-uniform grid using polynomial interpolation. *Journal of Computational Physics*, 204:404–429, 2005.

[97] L. Gamet, F. Ducros, F. Nicoud, and T. Poinsot. Compact finite difference schemes on non-uniform meshes. application to direct numerical simulations of compressible flows. *International Journal for Numerical Methods in Fluids*, 29:159–191, 1999.

[98] J. Jiménez and P. Moin. The minimal flow unit in near-wall turbulence. *Journal of Fluid Mechanics*, 225:213–240, 1991.

[99] J.C. del Álamo, J. Jiménez, P. Zandonade, and R.D. Moser. Scaling of the energy spectra of turbulent channels. *Journal of Fluid Mechanics*, 500:135–144, 2004.

[100] R.D. Moser, J. Kim, and N.N. Mansour. Direct numerical simulation of turbulent channel flow up to $Re_\tau = 590$. *Physics of Fluids*, 11(4):943–945, 1999.

[101] S. Hoyas and J. Jiménez. Scaling of the velocity fluctuations in turbulent channels up to $Re_\tau = 2003$. *Physics of Fluids*, 18, 2006.

[102] M. Uhlmann and A. Sekimoto. Marginally turbulent flow in a square duct. *Journal of Fluid Mechanics*, oct 2007.

[103] A. Huser, S. Biringen, and F.F. Hatay. Direct simulation of turbulent flow in a square duct: Reynolds-stress budgets. *Physics of Fluids*, 6(9):3144–3152, 1994.

[104] E. Vedy, S. Viazzo, and R. Schiestel. A high-order finite difference method for incompressible fluid turbulence simulations. *International Journal for Numerical Methods in Fluids*, 42:1155–1188, 2003.

[105] M.P. Simens, J. Jiménez, S. Hoyas, and Y. Mizuno. A high-resolution code for turbulent boundary layers. *Journal of Computational Physics*, 228:4218–4231, 2009.

[106] U. Ehrenstein, M. Marquillie, and C. Eloy. Skin friction on a flapping plate in uniform flow. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 372(2020), 2014.

[107] G.A. Reis, I.V.M. Tasso, L.F. Souza, and J.A. Cuminato. A compact finite differences exact projection method for the navier-stokes equations on a staggered grid with fourth-order spatial precision. *Computers and Fluids*, 118:19–31, 2015.

[108] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[109] C.A.J. Fletcher. *Computational Techniques for Fluid Dynamics, Volumes I and II*. Springer-Verlag, Berlin, 1991.

[110] Y. Morinishi, T.S. Lund, O.V. Vasilyev, and P. Moin. Fully conservative higher order finite difference schemes for incompressible flow. *Journal of Computational Physics*, 143:90–124, 1998.

[111] Y. Morinishi. Skew-symmetric form of convective terms and fully conservative finite difference schemes for variable density low-mach number flows. *Journal of Computational Physics*, 229:276–300, 2010.

[112] P. Haldenwang, G. Labrosse, S. Abboudi, and M. Deville. Chebyshev 3-d spectral and 2-d pseudospectral solvers for the helmholtz equation. *Journal of Computational Physics*, 55:115–128, 1984.

[113] G. Fontaine, S. Poncet, and E. Serre. Multidomain extension of a pseudospectral algorithm for the direct simulation of wall-confined rotating flows. *Lecture Notes in Computational Science and Engineering*, 95:261–271, 2014.

[114] G. Danabasoglu, S. Biringen, and C.L. Streett. Application of the spectral multidomain method to the navier-stokes equations. *Journal of Computational Physics*, 113:155–164, 1994.

[115] Satish Balay, Jed Brown, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, 2013. http://www.mcs.anl.gov/petsc.

[116] H. Ganesh. *Bubbly Shock Propagation as a Cause of Sheet to Cloud Transition of Partial Cavitation and Stationary Cavitation Bubbles Forming on a Delta Wing Vortex*. PhD thesis, University of Michigan, 2015.

[117] A. Gnanaskandan and K. Mahesh. A numerical method to simulate turbulent cavitating flows. *International Journal of Multiphase Flow*, 70:22–34, 2015.

[118] S.V. Patankar. *Numerical heat transfer and fluid flow*. CRC Press, 1980.

[119] C. Rezki. *Simulation 2D et 3D des écoulements cavitants : d éveloppement d'un algorithme original dans Code Saturne et étude de l'influence de la mod élisation de la turbulence*. PhD thesis, ENSAM ParisTech, 2014.

[120] *Code Saturne documentation: Code Saturne 4.2.0 Theory Guide*, 4.2.0 edition.

[121] C.D. Dimitropoulos and A.N. Beris. An efficient and robust spectral solver for nonseparable elliptic equations. *Journal of Computational Physics*, 133:186–191, 1997.

[122] R.J. LeVeque. Finite difference methods for differential equations. University Lecture, AMath 585-586 course, 2005.

[123] S. Abdallah. Numerical solutions for the pressure poisson equation with neumann boundary conditions using a non-staggered grid, i. *Journal of Computational Physics*, 70:182–192, 1987.

[124] M.L. Mansour and A. Hamed. Implicit solution of the lncompressible navier-stokes equations on a non-staggered grid. *Journal of Computational Physics*, 86:147–167, 1990.

[125] Y. Li and L. Baldacchino. Accuracy and efficiency of some higher-order schemes on non-uniform grids for fluid flow problems. In *Twelfth Australian Fluid Mechanics Conference, Sydney*, 1995.

[126] L. Eca and M. Hoekstra. Code verificaton of unsteady flow solvers with the method of the manufactures solutions. In *Proceedings of the Seventeenth International Offshore and Polar Engineering Conference*. ISOPE, July 2007.

[127] K. Salari and P. Knupp. Code verification by the method of manufactured solutions. Technical report, Sandia National Laboratories, Albuquerque, New Mexico and Livermore, California, USA, 2000.

[128] L. Eça, G. Vaz, and M. Hoekstra. Code verification of refresco with a statistically periodic manufactured solution. In *Proceedings of ASME 33rd International Conference on Ocean, Offshore and Arctic Engineering, OMAE2014, San Francisco*, June 2014.

[129] D. Pelletier, É. Turgeon, and D. Tremblay. Verification and validation of impinging round jet simulations using an adaptive fem. *International Journal for Numerical Methods in Fluids*, 44:737–763, 2004.

[130] P.J. Roache. Code verification by the method of manufactured solutions. *Journal of Fluids Engineering*, 124:4–10, 2001.

[131] *Maxima Manual*, 5.30.0 edition, 2013. available from http://maxima.sourceforge.net/docs/manual/en/maxima.pdf.

[132] IDRIS-Institut du développement et des ressources en informatique scientifique. Available online at http://www.idris.fr/.

[133] HPC Prelog. Available online at http://hpc.fs.uni-lj.si/.

[134] F. Nicoud and F. Ducros. Subgrid-scale stress modelling based on the square of the velocity gradient tensor. *Flow, Turbulence and Combustion*, 62(3):183–200, 1999.

[135] U. Ehrenstein and R. Peyret. A chebyshev collocation method for the navier-stokes equations with application to double-diffusive convection. *International Journal for Numerical Methods in Fluids*, 9:427–452, 1989.

UN NOUVEL ALGORITHME POUR LA SIMULATION DNS ET LES DES ÉCOULEMENTS CAVITANTS

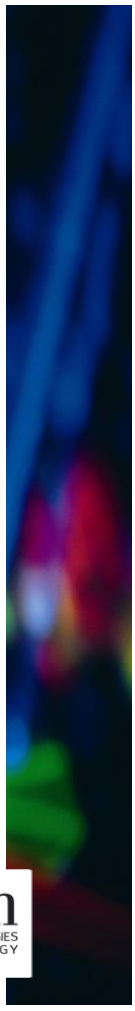
RESUME : Le couplage diphasique-turbulence est une propriété clé des écoulements cavitants, qui est un frein important à l'amélioration des modèles de cavitation et de turbulence. Réaliser des simulations directes (DNS) est le moyen proposé ici pour s'affranchir du modèle de turbulence et obtenir des informations nouvelles sur les phénomènes mis en jeu. Ce type de simulation est exigeant sur le plan numérique, et requiert le développement d'un solveur spécifique intégrant les spécificités des modèles de cavitation. Cela inclue notamment des schémas de discrétisation d'ordre élevé, un solveur direct, et une résolution multi-domaines associée à une parrallélisation efficace. Une discrétisation par différences compactes finies s'avère être le meilleur choix. La contrainte de rapidité et de parrallélisation impose un algorithme où les systèmes résoudre n'impliquent des multiplications des variables implicites que par des coefficients invariants au cours du calcul. Un nouvel algorithme réunissant ces critères a été développé durant cette thèse, à partir de la combinaison de la méthode de Concus & Golub et d'une méthode de projection, qui permet de résoudre les équations associées à la modélisation homogène de la cavitation. Une nouvelle approche de vérification de ce nouvel algorithme est également proposée et mise en œuvre sur la base de la méthode des solutions manufacturées (MMS).

Mots clés : Cavitation, modèle homogène, DNS, différences finies compactes, matrice de l'influence, méthode Concus et Golub

A NOVEL ALGORITHM FOR DNS AND LES SIMULATIONS OF CAVITATING FLOWS

ABSTRACT : Cavitation-turbulence interactions are problematic aspect of cavitating flows which imposes limitations in development of better cavitation and turbulence models. DNS simulations with homogeneous mixture approach are proposed to overcome this and offer more insight into the phenomena. As DNS simulations are highly demanding and a variety of cavitation models exists, a tool devoted specifically to them is needed. Such tools usually demand application of highly accurate discretization schemes, direct solvers and multi domain methods enabling good scaling of the codes. As typical cavitating flow geometries impose limits on suitable discretization methods, compact finite differences offer the most appropriate discretization tool. The need for fast solvers and good code scalability leads to request for an algorithm, capable of stable and accurate cavitating flow simulations where solved systems feature multiplication of implicitly treated variables only by constant coefficients. A novel algorithm with such ability was developed in the scope of this work using Concus and Golub method introduced into projection methods, through which the governing equations for homogeneous mixture modeling of cavitating flows can be resolved. Work also proposes an effective and new approach for verification of the new and existing algorithms on the basis of Method of Manufactured Solutions.

Keywords : Cavitation, homogeneous mixture approach, DNS, compact finite differences, influence matrix technique, Concus and Golub method



ARTS ET MÉTIERS ParisTech

ParisTech INSTITUT DES SCIENCES ET TECHNOLOGIES PARIS INSTITUTE OF TECHNOLOGY