Numerical optimization using stochastic evolutionary algorithms : application to seismic tomography inverse problems

Keurfon Luu

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT

DE L'UNIVERSITÉ PSL

Préparée à MINES ParisTech

Numerical optimization using stochastic evolutionary algorithms: application to seismic tomography inverse problems

Optimisation numérique stochastique évolutionniste : application aux problèmes inverses de tomographie sismique

Soutenue par

Keurfon LUU

Le 28 septembre 2018

École doctorale n°398

Géosciences, Ressources Naturelles et Environnement

Spécialité

Géosciences et Géoingénierie

Composition du jury :

M. CAUMON Guillaume Professeur, Université de Lorraine	*Rapporteur*
M. TARITS Pascal Professeur, Université de Bretagne Occidentale	*Rapporteur*
M. BODET Ludovic Maître de conférences, Sorbonne Université	*Examinateur*
Mme GÉLIS Céline Ingénieur de recherche, IRSN	*Examinatrice*
Mme GESRET Alexandrine Chargée de recherche, MINES ParisTech	*Examinatrice*
M. NOBLE Mark Maître de recherche, MINES ParisTech	*Examinateur*
M. ROUX Pierre-François Ingénieur, BHGE	*Invité*

MINES ParisTech | PSL

Acknowledgements

Tout d'abord, je tiens à remercier mes jambes pour m'avoir supporté tout au long de mon parcours, mes bras pour toujours avoir été à mes côtés, sans oublier mes doigts sur lesquels j'ai toujours pu compter.

Plus sérieusement, je dois mes premiers remerciements à Mark Noble, mon directeur de thèse, qui m'a permis de travailler sur ce sujet tout en me laissant suffisamment de liberté pour me l'approprier (il me semble que le sujet initial portait sur la localisation. . .). Je remercie également ma co-encadrante Alexandrine Gesret pour tout le temps qu'elle m'a accordé et ses précieux conseils durant la rédaction de ce manuscrit.

Je souhaiterais exprimer ma gratitude envers Guillaume Caumon et Pascal Tarits qui ont accepté d'être rapporteurs de ce travail (et de lire ce manuscrit pendant leurs vacances d'été), ainsi que Céline Gélis et Ludovic Bodet pour avoir accepté de faire partie de mon jury de thèse.

Mes sincères remerciements vont aussi à Philippe Thierry qui, en dépit de son agenda de ministre, a trouvé du temps pour m'aider et me conseiller sur le calcul haute performance, et qui m'a notamment permis de faire la rencontre de *Skylake* quand *Katrina* m'a laissé tomber (je parle d'ordinateurs hein, pas taper).

Mention spéciale à tous les autres membres du *Labo de Géophy* : Hervé, Pierre, Véronique, les *deux gars de Magnitude* Nidhal et Pierre-François, Maxime, Charles-Antoine, Yves-Marie, Jihane, Sven, Emmanuel, Yubing, Alexandre, Hao, Julien, Michelle, Tianyou, Ahmed et en passant par le Brésil, Tiago. Merci pour votre bonne humeur et toutes les discussions que l'on a pu avoir pendant les (trop) nombreuses pauses café.

Que seraient ces remerciements si j'oubliais tous les membres de ma famille (de près ou de loin généalogiquement ou géographiquement) qui m'ont connu et soutenu depuis mes premières lignes de commande (sous DOS pour lancer des jeux, bien entendu). En particulier, je dois beaucoup à *baba mama* qui ont fait beaucoup de sacrifices pour ma soeur, mon frère et moi, et qui, malgré leurs modestes moyens, nous ont toujours encouragés et poussés à donner le meilleur de nous.

Enfin, je voudrais dédier ce mémoire à ma femme An qui, plus que toutes les parties de mon corps susmentionnées, n'a jamais cessé de me supporter (dans les deux sens du terme), d'être à mes côtés et de m'encourager durant ces trois longues années. *Xiexie ni wo de baobei*.

Résumé

La tomographie sismique des temps de trajet est un problème d'optimisation mal-posé du fait de la non-linéarité entre les temps et le modèle de vitesse. Par ailleurs, l'unicité de la solution n'est pas garantie car les données peuvent être expliquées par de nombreux modèles. Les méthodes de Monte-Carlo par Chaînes de Markov qui échantillonnent l'espace des paramètres sont généralement appréciées pour répondre à cette problématique. Cependant, ces approches ne peuvent pleinement tirer parti des ressources computationnelles fournies par les super-calculateurs modernes. Dans cette thèse, je me propose de résoudre le problème de tomographie sismique à l'aide d'algorithmes évolutionnistes. Ce sont des méthodes d'optimisation stochastiques inspirées de l'évolution naturelle des espèces. Elles opèrent sur une population de modèles représentés par un ensemble d'individus qui évoluent suivant des processus stochastiques caractéristiques de l'évolution naturelle. Dès lors, la population de modèles peut être intrinsèquement évaluée en parallèle ce qui rend ces algorithmes particulièrement adaptés aux architectures des super-calculateurs. Je m'intéresse plus précisément aux trois algorithmes évolutionnistes les plus populaires, à savoir l'évolution différentielle, l'optimisation par essaim particulaire, et la stratégie d'évolution par adaptation de la matrice de covariance. Leur faisabilité est étudiée sur deux jeux de données différents: un jeu réel acquis dans le contexte de la fracturation hydraulique et un jeu synthétique de réfraction généré à partir du modèle de vitesse Marmousi réputé pour sa géologie structurale complexe.

Mots Clés: algorithme évolutionniste, tomographie sismique, problème inverse, calcul haute performance, intelligence artificielle

Abstract

Seismic traveltime tomography is an ill-posed optimization problem due to the non-linear relationship between traveltime and velocity model. Besides, the solution is not unique as many models are able to explain the observed data. The non-linearity and non-uniqueness issues are typically addressed by using methods relying on Monte Carlo Markov Chain that thoroughly sample the model parameter space. However, these approaches cannot fully handle the computer resources provided by modern supercomputers. In this thesis, I propose to solve seismic traveltime tomography problems using evolutionary algorithms which are population-based stochastic optimization methods inspired by the natural evolution of species. They operate on concurrent individuals within a population that represent independent models, and evolve through stochastic processes characterizing the different mechanisms involved in natural evolution. Therefore, the models within a population can be intrinsically evaluated in parallel which makes evolutionary algorithms particularly adapted to the parallel architecture of supercomputers. More specifically, the works presented in this manuscript emphasize on the three most popular evolutionary algorithms, namely Differential Evolution, Particle Swarm Optimization and Covariance Matrix Adaptation - Evolution Strategy. The feasibility of evolutionary algorithms to solve seismic tomography problems is assessed using two different data sets: a real data set acquired in the context of hydraulic fracturing and a synthetic refraction data set generated using the Marmousi velocity model that presents a complex geology structure.

Keywords: evolutionary algorithm, seismic tomography, inverse problem, high performance computing, artificial intelligence

Contents

Chapter 1

Introduction

Contents

1.1 General context

This thesis is supported by the GEOTREF project (www.geotref.com). This project is funded by ADEME in the frame of les Investissements d'Avenir program. Partners of the GEOTREF project are Kidova, Teranov, MINES ParisTech, ENS Paris, GeoAzur, GeoRessources, IMFT, IPGS, LHyGes, UA, and UCP-GEC. GEOTREF is a multi-disciplinary platform for innovation and demonstration activities for the exploration and development of high geothermal energy in fractured reservoirs.

The initial topic of this work aimed at characterizing the seismicity induced by the hydraulic fracturing process in the context of geothermal energy. In non-conventional hydrocarbon reservoirs or geothermal systems, fluid is usually injected into reservoirs to improve their permeability. This process can lead to the failure of the medium due to the changes of its physical properties (e.g. stress, pore pressure). Populated areas must be monitored in case of increasing seismic rate and/or magnitude. For instance, an earthquake of magnitude 3.4 has been caused by a geothermal project in Basel (Switzerland) in 2006 (Deichmann and Giardini (2009)).

Microseismic monitoring is currently the only method available to follow the state of the fracking and is thus required for security reasons. In the last decades, it has been a key tool to understand and delineate hydraulic fracture geometry (Sasaki (1998), Maxwell *et al.* (2002), Rothert and Shapiro (2003), Warpinski, Wolhart and Wright (2004), Calvez *et al.* (2007), Daniels *et al.*

(2007), Warpinski (2009), Maxwell *et al.* (2010)). It consists in locating microseismic events to constrain the geometry of the fractures. Event locations are strongly affected by the quality of the velocity model used and is usually disregarded in the interpretation (Cipolla *et al.* (2011)). Maxwell (2009) has shown on a simple synthetic model that an error on the velocity of only a few percent can produce a shift in an event location that can reach several hundreds of meters. In routine microseismic monitoring, a layered velocity model is constructed from acoustic logs that measure both compressional (P-) and shear (S-) wave velocities. The velocity model is then calibrated using perforation shots from known locations (Eisner *et al.* (2009)). Finally, the quality of the calibrated velocity model can be assessed by evaluating the errors in perforation shot locations.

In the frame of geothermal energy, microseismic events are typically recorded by a limited number of receivers deployed on the near-surface and can be used to image the reservoir. Such an acquisition geometry is not ideal as the ray paths from the perforation shots to the receivers do not correctly cover the medium traversed. Consequently, a given velocity model that gives small perforation shot location errors is only one of the many velocity models that can accurately relocate the shots. In addition, Gesret *et al.* (2015) showed that more reliable locations of hypocenters with their associated uncertainties can be obtained by propagating the velocity model uncertainties to the event locations. Therefore, an accurate velocity model must be derived and its uncertainties acknowledged to enable reliable interpretation of microseismic data.

Broadly speaking, velocity models are reconstructed through a tomographic procedure that iteratively tries to fit a model to match the observed data. This procedure is usually called an inversion as one wants to find the model that could have generated a given data set. In this manuscript, I particularly emphasize on first arrival traveltime tomography that aims at building the velocity model using the traveltimes observed from the seismic sources to all the receivers. The methodology initially developed to solve the tomographic problem in the context of microseismicity is applied to other tomographic problems that could potentially find an interest in geothermal reservoir characterization. First arrival traveltime tomography is an optimization problem and thus requires an accurate forward problem and an optimization algorithm for its resolution.

In this introduction, I first describe the principles of inverse problems. The limitations of the conventional methods used to solve an inverse problem in geophysics motivate the need to develop more computer efficient algorithms.

1.2 Inverse problems in geophysics

Solving an inverse problem consists in finding the physical model that explains the data by minimizing the misfit between the observed data and the data predicted by the model. Therefore, it is necessary to define the physical law that accurately describes the relation between the data and a physical model. In inversion, this operation is known as the foward modeling. Let $\mathbf{d} = \{d_1, ..., d_N\}$ be a discrete vector of N data points, $\mathbf{m} = \{m_1, ..., m_d\}$ a physical model described by d parameters, and g the forward modeling operator such that

$$\mathbf{d} = g(\mathbf{m}). \tag{1.1}$$

The quality of the data is strongly affected by the level of noise recorded during the measurements. In addition, mathematical models are oftenly subject to hypothesis and approximations and thus do not exactly reproduce the true physics. Therefore, an additional term ε that accounts for both

the errors in the data and the forward modeling must be considered and Equation (1.1) can be rewritten as

$$\mathbf{d} = g(\mathbf{m}) + \varepsilon. \tag{1.2}$$

In first arrival traveltime tomography, $\mathbf{d}$ and $\mathbf{m}$ are the traveltime data and the velocity model, respectively. First arrival traveltimes can be accurately computed by solving the Eikonal equation that links the velocities of the propagation medium to the traveltimes under the high frequency approximation. This approximation is only valid when the signal wavelength is negligible with respect to the characteristic wavelength of the medium. A full and detailed presentation of the dynamic of acoustic waves can be found in Chapman (1985), Sheriff and Geldart (1995) and Červený (2001) and is beyond the scope of this thesis. However, a short description of the underlying Eikonal equation is reviewed in Appendix A. The Eikonal equation is written

$$|\nabla T|^2 = \frac{1}{c^2} \tag{1.3}$$

where $|\cdot|$ denotes the absolute value, ∇ the gradient operator, T the traveltime and c the velocity of the medium. It is a partial differential equation that can be solved either analytically for simple velocity models, or numerically by using ray-tracing (Julian and Gubbins (1977), Červený (1987), Um and Thurber (1987)) or a finite-difference Eikonal solver (Vidale (1988), Vidale (1990), Podvin and Lecomte (1991), Trier and Symes (1991)). In this thesis, the finite-difference Eikonal solver proposed by Noble, Gesret and Belayouni (2014) is used to compute accurate traveltimes.

Inverse problems rely on a scalar objective function that measures the *misfit* between the observed and the theoretical data to assess the quality of the model parameters, and is usually called misfit function. The optimal model corresponds to the model that yields the global minimum misfit function value. In other words, solving an inverse problem requires the minimization of the misfit function E under the constraint $\mathbf{m} \in \Omega$ (feasible space) such that

$$\forall \mathbf{m} \in \Omega, \mathbf{m}_{\text{opt}} = \operatorname{argmin}\left(E(\mathbf{m})\right). \tag{1.4}$$

Although other norms can be found in the literature, the misfit function is usually defined in the least-squares sense in geophysics, following

$$E(\mathbf{m}) = \left[\left(\mathbf{d}^{obs} - g(\mathbf{m})\right)^{\top}\left(\mathbf{d}^{obs} - g(\mathbf{m})\right)\right]^{\frac{1}{2}} \tag{1.5}$$

where $\mathbf{d}^{obs}$ stands for the observed data.

Misfit functions encountered in geophysical inverse problems are ill-posed and highly multi-modal due to the non-linearity between the models and the data (Delprat-Jannaud and Lailly (1993), Zhang and Toksöz (1998)). The non-linearity of first arrival traveltime tomography problems are most oftenly addressed by using derivative-based (i.e. gradient) optimization methods (White (1989), Zelt and Barton (1998), Taillandier *et al.* (2009), Rawlinson, Pozgay and Fishwick (2010), Noble *et al.* (2010)) combined with a regularization procedure to make the problem better posed (Hansen and O'Leary (1993), Menke (2012), Tikhonov *et al.* (2013)). Derivative-based optimization methods are local optimization algorithms that iteratively update a model such that the misfit function value decreases over time, the direction of updates being given by the gradient of the misfit function with respect to the model parameters. Derivative-based algorithms include the steepest descent methods (Nemeth, Normark and Qin (1997), Taillandier *et al.* (2009)), the conjugate gradient methods (Luo and Schuster (1991), Zhou *et al.* (1995)), Gauss-Newton methods (Delbos *et al.* (2006)), Quasi-Newton methods (Liu and Nocedal (1989), Nash and Nocedal (1991)) and Newton methods (Gerhard Pratt, Shin and Hicks (1998)). Nowadays methods can efficiently compute the gradient of the misfit function by using the adjoint state method (Chavent (1974), Plessix (2006)).

As previously mentioned, one of the main challenges dealt when reconstructing a velocity model is the non-uniqueness of the solution. Besides the unconstraining geometry inherent to typical microseismic acquisition, additional sources of approximations and errors can contribute to the non-uniqueness of the velocity model such as the parametrization chosen and the theory behind the forward modeling (Tarantola (2005)). The adjoint state method applied to compute the gradient when using derivative-based algorithms does not provide the sensitivity of the solution to the errors and additional expensive calculations of the Fréchet derivatives are required (Plessix (2006)). Moreover, derivative-based algorithms are local optimization methods that make the assumption of a unique solution that presents a good trade-off between data fit and regularization and strongly depend on the initial solution (Menke (2012)). For highly non-linear problems such as traveltime tomography, these approaches are not suitable when it comes to uncertainty quantification.

From a theoretical point of view, global optimization methods that sample the model parameter space are required to appraise the uncertainties. Geophysicists usually resort to probabilistic approaches that rely on the Bayes theorem combined with Markov Chain Monte Carlo algorithms (MCMC, Tarantola and Valette (1982), Scales and Snieder (1997), Ulrych, Sacchi and Woodbury (2001), Scales and Tenorio (2001)). In the Bayesian framework, the model parameters are random variables distributed accordingly to a probability distribution. Interest of this approach is two-fold: one can use the a priori knowledge about the parameter values to better constrain the estimation of the model parameters (an uninformative prior distribution can be defined otherwise); it thoroughly samples the model space and provides reliable parameter uncertainty estimates which offers a way to tackle the non-linear and non-uniqueness issues. MCMC algorithms have been first introduced in geophysics by Keilis-Borok and Yanovskaya (1967) and have been widespreadly used for optimization and sampling purpose to solve various types of geophysical inverse problem since then. Press (1968), Wiggins (1969) and Press (1970) were the first to apply MCMC algorithms to fit Earth models to seismological data. Cary and Chapman (1988) and Koren *et al.* (1991) addressed the problem of seismic waveform fitting using MCMC methods. Zhang and Toksöz (1998) used an MCMC approach to estimate the uncertainties of a velocity model obtained by a derivative-based optimization method. Other than in seismic studies, they also found applications in magnetotelluric (Jones and Hutton (1979), Jones, Olafsdottir and Tiikkainen (1983), Grandis, Menvielle and Roussignol (1999)) and electrical (Schott *et al.* (1999), Malinverno and Torres-Verdín (2000), Ramirez *et al.* (2005)) studies. A comprehensive but non-exhaustive review of their applications in geophysics can be found in Mosegaard and Tarantola (1995) and Sambridge and Mosegaard (2002). Nonetheless, classical MCMC algorithms can become inefficient with increasing number of parameters and multi-modal distributions as they can be easily trapped in a local mode. Many algorithms based on MCMC have been proposed to address its shortcomings such as the well-known Simulated Annealing (SA, Kirkpatrick, Gelatt and Vecchi (1983)), reversible-jump Markov Chain Monte Carlo (rj-MCMC, Green (1995)) or by using simultaneous interactive Markov Chains (Romary (2010), Sambridge (2014), Bottero *et al.* (2016)). On the one hand, rj-MCMC provides an interesting way to solve inverse problems without the requirement to fix the number of unknowns to optimize *a priori* and has been first applied in geophysics by Malinverno and Leaney (2000). Such inversion algorithms are usually qualified as *transdimensional* since they jump between parameter subspaces of different dimensionality. Bodin and Sambridge (2009) applied this algorithm to the seismic tomography problem. More recently, Piana Agostinetti, Giacomuzzi and Malinverno (2015) solved a three-dimensional local earthquake tomography problem using a transdimensional approach. On the other hand, interactive Markov Chains methods tackle the sequential nature of MCMC algorithms by allowing the sampling to be done in parallel. However, these methods are not straightforward to implement and remain sensitive to their control parameter values which are not easy to tune.

Microseismic data are required to be processed in real-time which motivates the need to develop and implement computationally efficient algorithms. In this thesis, I essentially work with another class of global optimization algorithms that has been growing in popularity in the last decades and are known as evolutionary algorithms. These methods are stochastic optimization methods inspired by the natural evolution of species and simultaneously operate on a set of candidate solutions called a population. Evolutionary algorithms have been quite overlooked by the geophysical community mostly due to the computational cost as the misfit function requires to be evaluated numerous times. Only few applications are available in the geophysical literature. More specifically, they have been applied to solve magnetotelluric inverse problems (Shaw and Srivastava (2007), Grayver and Kuvshinov (2016), Xiong *et al.* (2018)), gravity inversion (Zhang *et al.* (2004), Toushmalani (2013), Pallero *et al.* (2015), Ekinci *et al.* (2016)), history matching for reservoir characterization (Schulze-Riegert *et al.* (2001), Hajizadeh, Christie and Demyanov (2010), Mohamed *et al.* (2010), Fernández Martínez *et al.* (2012)), earthquake location (Sambridge and Gallagher (1993), Růžek and Kvasnička (2001), Han and Wang (2009)), surface-wave inversion (Wilken and Rabbel (2012), Song *et al.* (2012), Poormirzaee (2016)), and traveltime tomography (Tronicke, Paasche and Böniger (2012), Rumpf and Tronicke (2015), Poormirzaee, Moghadam and Zarean (2015)). Nonetheless, with the rise in computational power mainly brought by the parallel architectures of supercomputers, evolutionary algorithms are becoming an interesting alternative to conventionally used optimization methods as they are intrinsically parallel. Originally, evolutionary algorithms have not been designed for uncertainty quantification. However, as stochastic algorithms, they can be run several times to explore and sample different subspaces of the model parameter space (Sen and Stoffa (1996)).

1.3 Overview

The manuscript is organized as followed:

- In Chapter 2, I give an introduction to evolutionary algorithms in the framework of *black-box* optimization. First, I introduce the concepts of *black-box* optimization and evolutionary algorithms. Many evolutionary algorithms have been proposed in the literature. However, this thesis mainly emphasizes on the three most popular and efficient evolutionary algorithms, namely the Differential Evolution (DE), the Particle Swarm Optimization (PSO) and the Covariance Matrix Adaptation - Evolution Strategy (CMA-ES). These algorithms are thoroughly described in Chapter 2. For each method, I detail the algorithm and the underlying equations, and try to give a more intuitive explanation about how they work. A sample working code (written in Python) is also provided for the three methods. Finally, I give a brief description of the hybrid parallel programming paradigm used throughout this thesis to solve seismic tomography problems using evolutionary algorithms in reasonable computation time;

- In Chapter 3, I principally focus on PSO for its efficiency and ease of implementation. I present a new optimization algorithm based on PSO that tackles its main shortcoming. I called this new algorithm Competitive Particle Swarm Optimization (CPSO). Indeed, PSO is particularly prone to stagnation and premature convergence (i.e. convergence toward a local minimum) like any evolutionary algorithm. Therefore, I suggest a simple yet efficient modification of the original implementation that improves the diversity of the swarm. I demonstrate on several benchmark test functions that CPSO is more robust in terms of convergence and sensitivity to its parameters compared to PSO. Besides, I show on a highly multi-modal function that CPSO can be used for rapid uncertainty quantification.

Estimation of the Probability Density Function is done by sampling the model parameter space through several independent inversions. The methodology is applied to a first arrival traveltime tomography problem using a real 3D microseismic data set acquired in the context of induced seismicity and is compared to a conventional MCMC sampler. The results demonstrate that CPSO is able to reach the stationary regime much faster and provides uncertainty estimates consistent with the ones obtained with the MCMC sampler. Finally, I analyze the scalability of CPSO for this tomographic problem by evaluating its parallel performance. Note that the results obtained in this work have been further used in Appendix C;

- Chapter 4 aims at studying the feasibility of evolutionary algorithms to solve a problem of larger dimensionality ($> 10^2$). To this end, I extend the methodology introduced in Chapter 3 to the highly non-linear and multi-modal problem of refraction tomography that could potentially find an interest in geothermal reservoir characterization. I apply the methodology on the Marmousi velocity model that presents a complex geological structure and compare the performances of DE, CPSO and CMA-ES. The synthetic traveltime data are generated considering a surface acquisition geometry that consists of two hundred shots and four hundred receivers which results in poor ray coverage in depths. The velocity model is parametrized using 2D cardinal B-splines. I investigate the influences of the initial velocity models, the population size and the maximum number of iterations. Evolutionary algorithm are stochastic optimization methods and should ideally be insensitive to the initial population. However, I show that using a realistic initial population instead of fully random models can significantly improves the convergence of these algorithms on the refraction tomography problem. Finally, I assess the benefits and shortcomings of each algorithm by performing scalability and statistical analysis over the results obtained after several runs;

- Microseismic monitoring requires an efficient automated phase onset picking algorithm for real-time microseismic event locations induced by hydraulic fracturing (Calvez *et al.* (2007)). To a lesser extent, errors in arrival times can also contribute to errors in locations. Common automated phase onset pickers are not designed to estimate arrival time uncertainties. In Chapter 5, I describe an automated phase onset picking algorithm based on a multi-attributes neural network. It is noteworthy that this chapter is the result of a project in collaboration with ENS Paris and ENSG, the main emphasis of this thesis remains the application of evolutionary algorithms to seismic tomography. Principles of neural networks are merely described and only important notions required to the understanding of the chapter are introduced. The chapter is written as a description of a computer software implemented in Python. I describe the methodology by applying the workflow on a real data set acquired at the laboratory scale. Optimal attributes and their parameters are selected thanks to a scatter-plot matrix. Then, a neural network can be trained using either a derivative-based optimizer or an evolutionary algorithm. The probability map output by the neural network model is used to predict a phase onset, assess its error or reject a false positive. Finally, the picker model is applied to the whole data set to predict the arrival times and relocate the microseismic events. The results obtained are promising in terms of accuracy and quantification of arrival time picking errors;

- In Chapter 6, I summarize the main conclusions drawn throughout this manuscript. I also propose several prospects of improvement, such as the *island models* to enhance the performance of evolutionary algorithms, using neuroevolution to find the optimal neural network architecture, and what to do with velocity model uncertainties once quantified;

- In Appendix A, I detail the derivation of the elasto-dynamic equation to obtain the Eikonal equation underlying the main forward modeling involved in this thesis (i.e. computation of first arrival traveltimes);

- In Appendix B, I apply the same methodology as described in Chapter 3 to a surface wave tomography problem. The forward modeling is first briefly detailed and consists in estimating the frequency-dependent modal dispersion curves using the Thomson-Haskell propagator method. Finally, I show an application to a real data set that consists of three dispersion curves obtained by the joint use of ambient noise correlation and Multiple signal characterization algorithm (MUSIC);

- Gesret *et al.* (2015) has shown that accurate hypocenter locations with more reliable uncertainties can be retrieved by accounting for the velocity model uncertainties during the location procedure. This is achieved by locating the events in all the velocity models sampled which can be computationally inefficient. In Appendix C, I propose to use an unsupervised learning algorithm (Mini-batch K-Means) to find a subset of velocity models for the location. The methodology is applied to the velocity models sampled in Chapter 3.

1.4 Contributions

As my work mainly focuses on computation time performance, I first cleaned and optimized the existing finite-difference Eikonal solver proposed in Noble, Gesret and Belayouni (2014). I added several functions for parallel traveltime grid computation using OpenMP and a posteriori ray-tracing based on the work of Podvin and Lecomte (1991). The code is originally written in Fortran and has been wrapped into a Python package for faster prototyping. I also implemented different evolutionary algorithms such as DE, PSO and CMA-ES, and developed CPSO, a new evolutionary algorithm based on PSO that improves its robustness. All the evolutionary algorithms are parallelized using MPI. The codes are available on my GitHub page as Python packages or Fortran modules:

- FTeikPy: object-oriented module that computes accurate first arrival traveltimes in 2D and 3D heterogeneous isotropic velocity model. Available at github.com/keurfonluu/FTeikPy;

- Forlab: Fortran library I wrote that contains more that one hundred polymorphic basic functions inspired by Matlab and NumPy. All the Fortran softwares I wrote have core dependency to this library. Available at github.com/keurfonluu/Forlab;

- StochOPy: user-friendly routines to sample or optimize objective functions with the most popular evolutionary algorithms, as described in Chapter 2. For completeness and out of curiosity, MCMC sampling algorithms such as Metropolis-Hastings and Hamiltonian Monte Carlo have been implemented in this package. A Graphical User Interface (GUI) is available to test the different algorithms and their parameters on several benchmark test functions. Available at github.com/keurfonluu/StochOPy;

- StochOptim: same as StochOPy (without GUI) but in Fortran. This module has been used in Chapter 3 and Chapter 4 to solve the tomographic problems. Available at github.com/keurfonluu/StochOptim.

During my thesis, I also worked with other students on other inverse problems. In the frame of the GEOTREF project, I collaborated with the Laboratoire de Géologie at the ENS Paris and ENSG to study the geomechanical properties of an andesite sample from la Guadeloupe. My work consisted in relocating the acoustic emissions recorded using a conventional tri-axial cell. Therefore, I implemented an automated neural network phase onset picker to pick the arrival times for event relocations. In addition, I worked with a colleague on dispersion curve inversion in the context of ambient noise surface wave tomography. I implemented the forward modeling in Fortran and wrapped it into a Python module, the dispersion curves are inverted using StochOPy. The computer codes used for the two collaborations have also been made available on GitHub:

- AIPycker: pythonic object-oriented module for automated phase onset picking using a multi-attributes neural network, as described in Chapter 5. It depends on NumPy, SciPy, ObsPy, Pandas, Matplotlib, Scikit-learn and StochOPy. It includes a GUI for manual onset picking and neural network training through a wizard. The package is in early development and the methodology still has to be improved. However, AIPycker is more user-friendly and has shown superior results compared to the commercial software used at ENS Paris. Available at github.com/keurfonluu/AIPycker;

- EvoDCinv: surface wave dispersion curve inversion using evolutionary algorithms. The package can handle inversion of multi-modal dispersion curves and both Rayleigh and Love waves, as described in Appendix B. Available at github.com/keurfonluu/EvoDCinv.

The results obtained presented in this manuscript have been subject to oral presentations in two international conferences, and submitted for publication:

- Zhi Li, **Keurfon Luu**, Aurélien Nicolas, Jérôme Fortin and Yves Guéguen, 2016. “Fluid-induced rupture on heat-treated andesite.” *4th International Workshop on Rock Physics*;

- **Keurfon Luu**, Mark Noble, Alexandrine Gesret, 2016. “A competitive particle swarm optimization for nonlinear first arrival traveltime tomography.” *2016 SEG International Exposition and Annual Meeting. Society of Exploration Geophysicists*. doi: 10.1190/segam2016-13840267.1;

- François Bonneau, **Keurfon Luu**, Aurélien Nicolas and Zhi Li, 2017. “Toward an understanding of the relationship between fracturing process and microseismic activity: study at the laboratory scale.” *2017 RING Meeting*;

- **Keurfon Luu**, Mark Noble, Alexandrine Gesret, Nidhal Belayouni and Pierre-François Roux, 2017. “Propagation of velocity uncertainties to Microseismic locations using a competitive Particle Swarm Optimizer.” *79th EAGE Conference and Exhibition 2017*;

- **Keurfon Luu**, Mark Noble, Alexandrine Gesret, Nidhal Belayouni and Pierre-François Roux, 2018. “A parallel competitive Particle Swarm Optimization for non-linear first arrival traveltime tomography and uncertainty quantification.” *Computers and Geosciences* 113 (August 2017). Elsevier Ltd: 81–93. doi: 10.1016/j.cageo.2018.01.016;

- Marc Peruzzetto, Alexandre Kazantsev, **Keurfon Luu**, Jean-Philippe Métaxian, Frédéric Huguet and Hervé Chauris, 2018. “Broadband ambient noise characterization by joint use of cross-correlation and MUSIC algorithm.” *Geophysical Journal International* 215(2): 760-779 (November 2018). doi: 10.1093/gji/ggy311;

- **Keurfon Luu**, Mark Noble, Alexandrine Gesret and Philippe Thierry, 2018. “Toward large scale stochastic refraction tomography: a comparison of three evolutionary algorithms.” *Geophysical Prospecting*. *Under review*.

Chapter 2

Introduction to evolutionary algorithms

Contents

Ce chapitre est une introduction aux algorithmes évolutionnistes dans le contexte de l'optimisation dite *boîte-noire*. Ces algorithmes sont des méthodes stochastiques opérant sur une population de modèles représentés par des individus, et réputés pour être plus robustes que les approches

basées sur les méthodes de Monte-Carlo en termes d'optimisation. Les algorithmes évolutionnistes sont notamment très adaptés au calcul sur architecture parallèle, ce qui leur permet de mieux tirer partie des ressources computationnelles offertes par les super-calculateurs modernes par rapport aux méthodes de Monte-Carlo. J'introduis en premier lieu le concept d'optimisation *boîte-noire* avant de parler des algorithmes évolutionnistes. Bien que de nombreux algorithmes évolutionnistes aient été proposés dans la littérature, je ne m'intéresserai qu'aux trois algorithmes les plus efficaces et populaires, à savoir l'évolution différentielle (Differential Evolution, DE), l'optimisation par essaim particulaire (Particle Swarm Optimization, PSO), et la stratégie d'évolution par adaptation de la matrice de covariance (Covariance Matrix Adaptation - Evolution Strategy, CMA-ES). Pour chaque méthode, je détaille l'algorithme ainsi que les équations sous-jacentes, et j'essaie de donner une explication plus intuitive de leur fonctionnement. Je fournis notamment un exemple de code (écrit en Python) pour chacun des algorithmes. Enfin, j'introduis le parallélisme hybride adopté dans cette thèse pour la résolution des problèmes de tomographie sismique par des algorithmes évolutionnistes avec des temps de calcul raisonnables.

La PSO est plus amplement étudiée dans le Chapitre 3 avec une application sur un jeu de données microsismiques réel. J'y propose un nouvel algorithme que j'ai appelé la PSO Compétitive (CPSO) basé sur la PSO qui résout son principal défaut à savoir la convergence prématurée. La même méthodologie est notamment appliquée à un problème de tomographie des ondes de surface en Annexe B où seuls le problème direct et les données diffèrent. J'étends ensuite l'étude à un problème de tomographie des ondes réfractées dans le Chapitre 4 dont le but est d'analyser la faisabilité des algorithmes évolutionnistes pour la résolution d'un problème à grand nombre de paramètres en comparant les trois algorithmes évolutionnistes (DE, CPSO, CMA-ES) introduits dans ce chapitre.

Tous les algorithmes décrits dans ce chapitre sont disponibles en libre accès sur ma page GitHub :

- StochOPy : routines faciles à utiliser pour échantillonner ou optimiser une fonction d'objectif à l'aide des algorithmes évolutionnistes les plus populaires. Certains algorithmes d'échantillonnage MCMC sont aussi disponibles tels que l'algorithme de Metropolis-Hastings et le Monte-Carlo Hamiltonien. Une interface utilisateur graphique a notamment été implémentée pour tester les différents algorithmes ainsi que leurs paramètres sur de nombreuses fonctions de test. Disponible à l'adresse github.com/keurfonluu/StochOPy;

- StochOptim : même chose que StochOPy (sans interface graphique utilisateur) mais écrit en Fortran. Ce module a été utilisé dans le Chapitre 3 et Chapitre 4 pour l'inversion des deux problèmes de tomographie. Disponible à l'adresse github.com/keurfonluu/StochOptim.

2.1 Black-box optimization

2.1.1 Misfit function

An optimization problem consists in minimizing an *objective* function (or *cost* function) $f(\mathbf{x})$:

$$\min_{\mathbf{x}\in\Omega} f(\mathbf{x}) \tag{2.1}$$

where $\Omega \subset \mathbb{R}^d$ is the search space (or *feasible* space), d denoting the dimension of the problem. It aims to find one or several candidate solutions $\mathbf{x} \in \Omega$ that yield the lowest possible objective

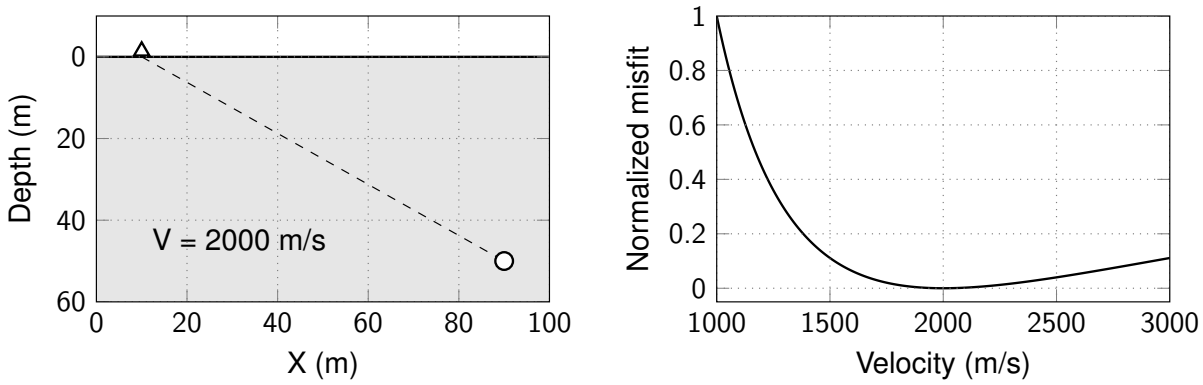

Figure 2.1: (Left) Synthetic earth model. The source and the receiver are respectively represented by the white disk and white triangle. (Right) Misfit function for different values of velocity ($V \in [1000, 3000]$ m/s). The global minimum of the misfit function (at 2000 m/s) indicates the *true* velocity of the earth model.

function value $f(\mathbf{x})$. In geophysics, the objective function measures the *misfit* between the observed data (e.g. traveltimes) and the data generated by a physical model (e.g. velocity model), and is thus usually referred to as the *misfit* function.

For instance, let us consider a simple traveltime tomography problem as depicted in Figure 2.1 (left). The earth model is characterized by a homogeneous velocity model of $V = 2000$ m/s. A signal is emitted by a source point and recorded by a single receiver at a time T. In traveltime tomography, the inputs (e.g. data) are the traveltime T and the positions of the source point and receiver, and the unknown is the velocity V. The physical relationship that links the traveltime to the velocity is written

$$T(V) = \frac{D}{V} \tag{2.2}$$

where D denotes the distance between the source point and the receiver. The misfit function E is usually defined in the least-square sense following

$$E(V) = |T - T(V)|^2. \tag{2.3}$$

Figure 2.1 (right) shows the misfit function *landscape* built by systematically evaluating the function for different velocity values. The global minimum indicates the *true* velocity of the earth model.

Note that the problem illustrated in Figure 2.1 is unidimensional and well-posed, and presents a convex misfit function with a trivial solution. The velocity can be found by a simple *grid* (i.e. exhaustive) search over a range of feasible velocities. This kind of search procedure can only be achieved when dealing with few variables to optimize and is inefficient in higher dimensions (> 5). Several other properties of a misfit function can make it difficult to solve, such as

- multi-modality: the misfit function presents several local minima that can easily trap a local optimization method, as shown in Figure 2.2 (left);
- non-separability: a function is said to be separable if each variable x_i can be minimized independently of the values of the other variables, a separable function can thus be minimized by d one-dimensional optimizations, as shown in Figure 2.2 (middle);
- ill-conditioning: the condition number is the ratio of the largest to the smallest eigenvalues of the Hessian. A problem is said to be ill-conditioned if its condition number is large

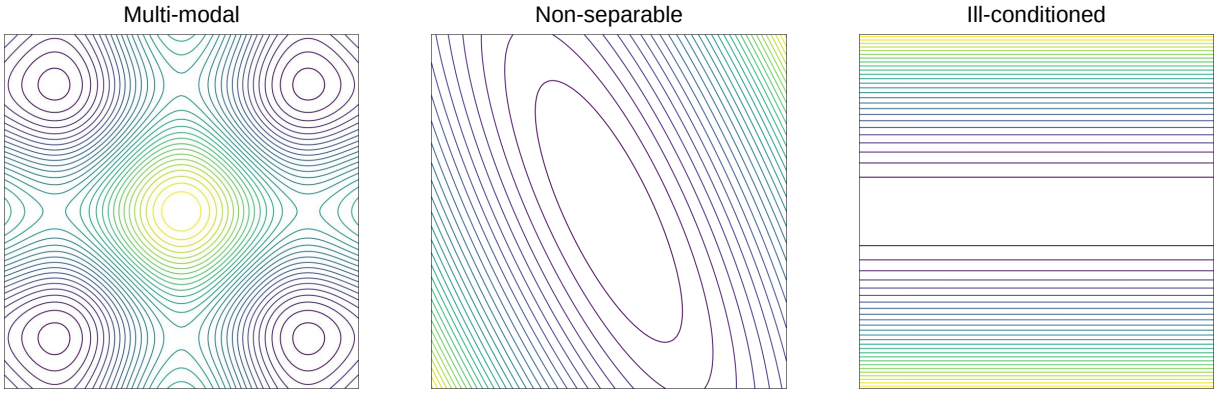

Figure 2.2: (Left) Multi-modal function with four local minima. (Middle) Non-separable function represented by a rotated ellipsoid. (Right) Ill-conditioned function with one undetermined parameter.

(typically > 10^5). Ill-conditioning results in variables that exhibit significant discrepancies in the sensitivity to their contributions to the misfit function value, as shown in Figure 2.2 (right).

Real-world optimization problems are generally ill-posed, multi-modal, non-separable and often ill-conditioned, and require advanced optimization algorithms to be solved. In the following and for consistency with the remainder of the manuscript, the misfit function is denoted by E and the variables to optimize are referred to as *model parameters* consequently denoted by $\mathbf{m}$.

2.1.2 Derivative-free algorithms

In the context of *black-box* optimization, no assumption is made on the misfit function such as whether it is continuous and/or differentiable. Black-box optimization algorithms can only query the misfit function values $E(\mathbf{m})$ of any search point $\mathbf{m}$ of the feasible space Ω (i.e. zero order information). Thus, the algorithms do not have access to higher order information such as the gradient of the misfit function. However, it is still possible to approximate the gradient by finite-difference in the case where the misfit function is computationally *cheap* (i.e. fast to compute), and derivative-based optimization algorithms can thus be used in the black-box scenario. Real-world optimization problems deal with computationally expensive misfit functions (especially in geophysics) and numerical approximation of the gradient is not feasible in practice.

Methods taylored for black-box optimization are the derivative-free optimization algorithms. These algorithms are called zero order methods as they only exploit the first term of the Taylor

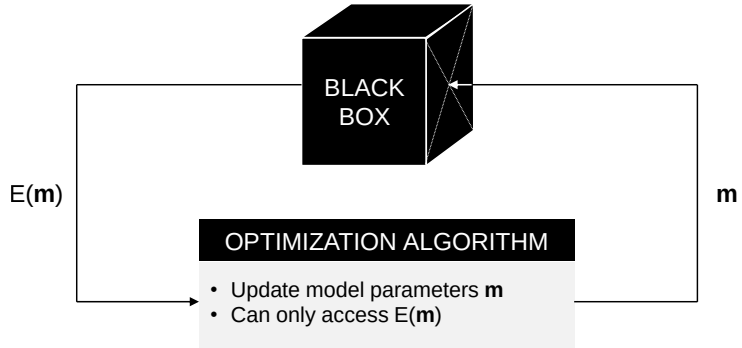

Figure 2.3: Black-box optimization. The optimization algorithm has only access to zero order information.

expansion of the misfit function (analogously, derivative-based methods are first or second order methods). While derivative-based methods are strictly *deterministic* in the sense that the solution is fully determined by the parameters and initial conditions, derivative-free methods can either be deterministic or *stochastic* (i.e. randomized). Deterministic derivative-free algorithms iteratively evaluate a set of points around the current point, and save the one that yields a lower misfit function value than the current point. It includes Simplex or Nelder-Mead's method (Nelder and Mead (1965), McKinnon (1998)), pattern search (Hooke and Jeeves (1961), Torczon (1997)) or Powell's method (Fletcher and Powell (1963), Powell (1964)). However, similarly to derivative-based methods, these algorithms are local optimization methods that depend on the initial conditions. On the other hand, evolutionary algorithms are stochastic derivative-free algorithms inspired by the natural evolution of species. This thesis mainly focuses on this type of black-box optimization algorithm.

2.2 Evolutionary algorithms

> As many more individuals of each species are born than can possibly survive; and as, consequently, there is a frequently recurring struggle for existence, it follows that any being, if it vary however slightly in any manner profitable to itself, under the complex and sometimes varying conditions of life, will have a better chance of surviving, and thus be naturally selected. From the strong principle of inheritance, any selected variety will tend to propagate its new and modified form.
>
> — Charles Darwin, On the Origin of Species (1859)

Natural evolution of species has been first theorized by Charles Darwin in his book *On the Origin of Species* published in 1859. He states that all the species have evolved from a common ancestor as a result of a process he called *natural selection*. According to Darwin's theory of evolution, natural selection operates following "one general law leading to the advancement of all organic beings, namely, multiply, vary, let the strongest live and the weakest die". Although his theory has been well received by other scientists, it has also raised several criticisms. The main criticism concerned the *blending* inheritance principle that would alter any beneficial characteristics and thus eliminate them after several generations. Blending inheritance has been discarded later in favor of the *Mendelian* inheritance discovered by Gregor Mendel in 1865 and presented in *Experiments on Plant Hybridization* (Mendel (1866), Hewlett and Mendel (1966)). Following the Mendelian inheritance principle, characteristics from parents are discretely passed to the offspring with a certain probability instead of being averaged. Modern population genetics is based on the combination of the principles brought by Darwin's theory of evolution and Mendel's principles of inheritance. All in all, evolution of populations is driven by three main mechanisms:

- Mutation: alteration of the characteristics of each individual within a population,
- Recombination: production of offspring with characteristic combinations that differ from those of either parent,
- Selection: pick offspring with characteristics that are beneficial and dispose of disadvantageous ones.

Evolutionary algorithms are stochastic derivative-free optimization algorithms inspired by the natural evolution of species. The intrinsic operations are based on the genetic inheritance principle and the natural selection mechanism proposed respectively by Gregor Mendel and Charles Darwin. Broadly speaking, evolutionary algorithm consequently refers to all the optimization methods that operate on a concurrent population of individuals (i.e. models) and generate new populations through genetic operations (mutation, recombination, selection). Evolutionary algorithms include Genetic algorithm (Holland (1973), Davis (1991)), Genetic programming (Koza (1992), Koza (1994)), Evolutionary programming (Fogel (1993), Fogel (1999)), Evolution strategies (Rechenberg (1973), Schwefel (1984)) and Swarm intelligence (Bishop (1989), Dorigo, Maniezzo and Colorni (1996)). Strictly speaking, Swarm intelligence is not a class of evolutionary algorithm as these methods are rather inspired by the social group behavior of certain species. However, they still fall into the broad definition of evolutionary algorithm and are usually considered as such. Likewise, the well-known Neighborhood algorithm (Sambridge (1999)) – at least among the geophysical community – can also be considered as an evolutionary algorithm.

From an optimization point of view, evolution starts by selecting the *fittest* individuals in a population for reproduction to form the next generation. The chances of survival of the produced offspring depend on the quality of the characteristics they inherited from their parents. This process is iterated until a population with the fittest individuals is found. An evolutionary algorithm can be described by the following general template:

1. Given a parametrized distribution $P(\mathbf{m}|\boldsymbol{\theta})$, initialize $\boldsymbol{\theta}^0$,
2. For generation $k = 1, 2, ...$:
    a. Sample n new candidates from distribution $P\left(\mathbf{m}|\boldsymbol{\theta}^{k-1}\right) \rightarrow \mathbf{m}_1, ..., \mathbf{m}_n$,
    b. Evaluate the candidates on E: $E(\mathbf{m}_1), ..., E(\mathbf{m}_n)$,
    c. Update parameters $\boldsymbol{\theta}^k$,

in which sampling (2a.) is the mutation and recombination mechanisms that create new individuals, selection of individuals (2b.) evaluates the fitness of a new individual by assigning a scalar misfit value, and parameters update (2c.) corresponds to the remainder of the random phenomena that contribute to the *genetic drift*.

These simple mechanisms provide evolutionary algorithms interesting properties for optimization. They have demonstrated great flexibility and adaptability to solve a given task, robust performance and global search capabilities (Back, Hammel and Schwefel (1997)). Besides, evolutionary algorithms seem to be particularly suitable to solve multi-objective optimization problems as they are able to capture several Pareto-optimal solutions in a single run (Fonseca and Fleming (1995), Fonseca and Fleming (1998), Zitzler (1999), Deb (2001)). Evolutionary algorithms are also closely related to neuroevolution – a subfield of research in Artificial Intelligence – that consists in mutating and selecting the best neural networks to solve simple tasks (Ronald and Schoenauer (1994), Angeline, Saunders and Pollack (1994), Stanley and Miikkulainen (2002), Kassahun and Sommer (2005)). More recently, evolutionary algorithms have been successfully applied to train deep neural networks, a task usually reserved to derivative-based algorithms due to its high computational cost (Such *et al.* (2017), Conti *et al.* (2017), Lehman *et al.* (2017)). This has been made possible by exploiting the intrinsic parallel property of evolutionary algorithms that enables a better handling of all the computational power provided by modern supercomputers. This property can also find practical applications in geophysical inverse problems which is the main emphase of this thesis.

In the last decades, many new nature-inspired algorithms have been proposed such as Bees Algorithm (Pham *et al.* (2006)), Glowworm Swarm Optimization (Krishnanand and Ghose

(2006)), Cuckoo Search (Yang and Deb (2009)), Bat Algorithm (Yang (2010)), Flower Pollination Algorithm (Yang (2012)) or Grey Wolf Optimization (Mirjalili, Mirjalili and Lewis (2014)). This has attracted criticism among the research community whose main concern is the lack of novelty hidden behind a metaphor and argues that researchers should rather emphasize their works on analyzing existing algorithms (Sörensen (2015)). For instance, Weyland (2010) has shown that Harmony Search (Zong Woo Geem, Joong Hoon Kim and Loganathan (2001)) – an algorithm mimicking the improvisation of musicians – is simply a special case of $(\mu + 1)$-ES (Rechenberg (1973)) introduced about 30 years earlier. In a recent paper, Saka, Hasançebi and Geem (2016) performed a thorougher analysis of both methods, and concluded that the two algorithms are substancially different conceptually and operationally. Discussing who is right or wrong, whether one should stop proposing new nature-inspired algorithms or not, is beyond the scope of this thesis. This chapter only introduces the three most popular evolutionary algorithms, namely Differential Evolution (DE), Particle Swarm Optimization (PSO) and Covariance Matrix Adaptation - Evolution Strategy (CMA-ES). In the following, if not explicitly stated, the size of the population is denoted by n, the dimension of the problem by d and the iteration number (i.e. generation) by k.

2.2.1 Differential Evolution

Differential Evolution (DE) is a genetic programming algorithm introduced by Price (1996) and Storn and Price (1997). DE begins with a set of individuals called a population where each individual is a model solution to the problem to optimize. The population is initially sampled according to a uniform distribution. It differs from the well-known Genetic algorithm by its mutation and crossover (i.e. recombination) mechanisms.

Mutation

Unlike Evolution strategies, DE does not sample candidate solutions using predetermined probability distribution functions. DE perturbs a random existing population vector by adding to it the weighted difference between two different random population vectors. This mutation process is repeated for each individual in the population. For each target vector $\mathbf{m}_i^k$, DE generates a mutant vector $\mathbf{v}_i^k$ following

$$\mathbf{v}_i^k = \mathbf{m}_{r_1}^{k-1} + F\left(\mathbf{m}_{r_2}^{k-1} - \mathbf{m}_{r_3}^{k-1}\right) \tag{2.4}$$

where $r1, r2, r3 \in \{1, 2, ..., n\}$ are three distinct random indices different from i, $F \in [0, 2]$ is the mutation factor that weighs the differential variation $\left(\mathbf{m}_{r_2}^{k-1} - \mathbf{m}_{r_3}^{k-1}\right)$. Mutation is an important process that maintains diversity within the population and prevents premature convergence. Large mutation factor increases the search radius (i.e. diversity) at the expense of speed of convergence. The mutation mechanism is sketched in Figure 2.4.

Crossover

In order to further increase diversity in the population of mutant vectors, the authors have introduced a crossover mechanism (i.e. recombination). It consists in mixing information from the current vector $\mathbf{m}_i^{k-1}$ with information from the mutant vector $\mathbf{v}_i^k$ to produce a trial vector $\mathbf{u}_i^k$. Crossover is achieved by randomly picking a parameter from either the current or mutant vectors accordingly to a *binomial* distribution with probability defined by the crossover rate $CR \in [0, 1]$,

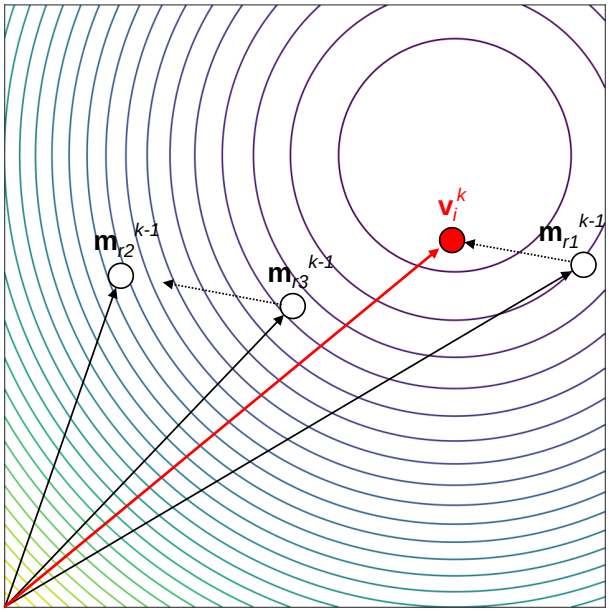

Figure 2.4: Mutation in DE on a 2D misfit function represented by the contour lines. $\mathbf{v}_i^k$ is generated by adding the weighted differential variation $\left(\mathbf{m}_{r_2}^{k-1} - \mathbf{m}_{r_3}^{k-1}\right)$ to the individual $\mathbf{m}_{r_1}^{k-1}$, with $\mathbf{m}_{r_1}^{k-1}$, $\mathbf{m}_{r_2}^{k-1}$ and $\mathbf{m}_{r_3}^{k-1}$ three random individuals chosen in the population.

which is written

$$u_{ji}^k = \begin{cases} v_{ji}^k & \text{if } r_j \leq CR \text{ or } j = R \\ m_{ji}^{k-1} & \text{otherwise} \end{cases} \tag{2.5}$$

with j being the parameter index, $r_j \sim \mathcal{U}(0,1)$ a uniform random number, $R \in \{1, 2, ..., d\}$ a random parameter index. The condition $j = R$ ensures that the trial vector gets at least one mutated parameter. Crossover is illustrated in Figure 2.5.

According to Das and Suganthan (2011), a low CR value (< 0.1) is beneficial for non-separable functions as it results in a search that changes separately each parameter of the mutated vectors. Nonetheless, as previously explained, real-world problems are usually non-separable and it is recommended to set $CR > 0.1$.

DE was originally designed for unconstrained optimization problems. Consequently, the trial

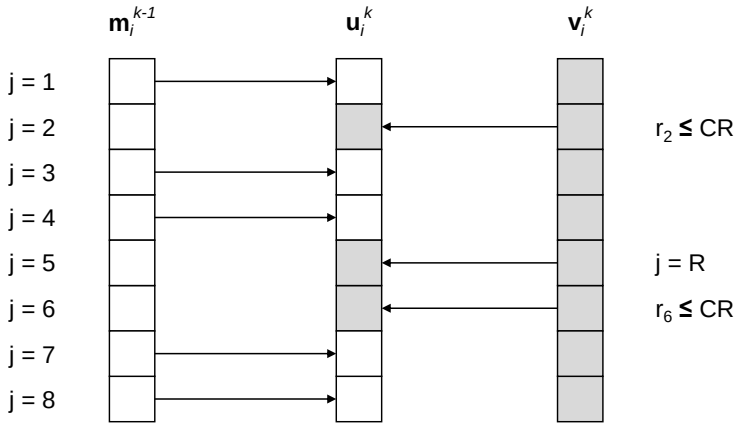

Figure 2.5: Crossover in DE for $d = 8$ parameters. For each parameter, the trial vector $\mathbf{u}_i^k$ receives a parameter from either the current or mutant vectors accordingly to a binomial distribution with probability defined by CR.

vector resulting from crossover is likely to fall outside of the feasible space. Price, Storn and Lampinen (2006) suggests that the most *unbiased* constraints handling approach is to randomly reinitialize any infeasible solution which also helps to maintain diversity within the population.

Selection and termination

Finally, selection applies the greedy criterion to determine which individuals to preserve based on their misfit function values. If the trial vector $\mathbf{u}_i^k$ yields a lower misfit, it replaces the target vector $\mathbf{m}_i^k$, otherwise, the previous model $\mathbf{m}_i^{k-1}$ is retained.

The algorithm stops when the population has converged:

1. The population is unable to produce better offspring different from the previous generation;
2. The maximum number of iterations is reached.

Mutation and crossover strategies

The strategy presented in this section is default and is known as *DE/rand/1/bin* as only one differential weight is added to a randomly chosen vector, and crossover is due to independent binomial experiments. Many other strategies are available in the literature and can be classified following the notation *DE/x/y/z* where

- *x* denotes the individual to be mutated $\mathbf{m}_{r_1}$ and can be *rand* (randomly chosen) or *best* (individual that yields the lowest misfit);
- *y* specifies the number of difference vectors to add to $\mathbf{m}_{r_1}$. The current variant is *1*. In case of *2*, the weighted differential variation is written $F\left(\mathbf{m}_{r_2} - \mathbf{m}_{r_3} + \mathbf{m}_{r_4} - \mathbf{m}_{r_5}\right)$ with r_2, r_3, r_4 and r_5 being four random indices;
- *z* characterizes the crossover distribution law and can be *bin* (binomial) or *exp* (exponential).

The default strategy has demonstrated excellent performances in many real-world problems including geophysical problems (Barros *et al.* (2015), Storn (2017)). Nevertheless, as well as the differential weight, the crossover rate and the population size, the strategy has to be chosen dependently to the problem to optimize.

2.2.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a nature-inspired population-based optimization algorithm. As a Swarm intelligence algorithm, PSO is not an evolutionary algorithm strictly speaking but is usually considered as such as collective knowledge is channeled within the population. It has been first introduced by Kennedy and Eberhart (1995) to study the social behavior of fishes and birds in a flock.

PSO algorithm

In PSO, the first step is to randomly position a swarm composed of several particles in the misfit landscape (i.e. model parameter space). Each particle represents a model and can be seen as *flying* through the misfit landscape while interacting with its neighborhood to find the global minimum of the misfit function. Neighborhood topologies are more thoroughly described in Section 2.2.2. All along the optimization process, each particle remembers the best position it has visited so far in addition to the best position achieved by the entire swarm. More specifically, a particle i is defined by a position vector $\mathbf{m}_i^k$ and a velocity vector $\mathbf{v}_i^k$ which is adjusted according to its own personal best position and the global best position of the whole swarm. The velocity and the position of each particle are updated following

$$\mathbf{v}_i^k = \omega \mathbf{v}_i^{k-1} + \phi_p \mathbf{r}_p^k \left(\mathbf{m}_{p,i} - \mathbf{m}_i^{k-1} \right) + \phi_g \mathbf{r}_g^k \left(\mathbf{m}_g - \mathbf{m}_i^{k-1} \right) \tag{2.6}$$

$$\mathbf{m}_i^k = \mathbf{m}_i^{k-1} + \mathbf{v}_i^k \tag{2.7}$$

where $\mathbf{m}_{p,i}$ and $\mathbf{m}_g$ are respectively the personal best position of particle i and the global best position of the population, $\mathbf{r}_p^k$ and $\mathbf{r}_g^k$ are uniform random number vectors drawn at iteration k, ω is an inertia weight, ϕ_p and ϕ_g are two acceleration parameters that respectively control the cognition and social interactions of the particles. Principle of PSO is illustrated in Figure 2.6.

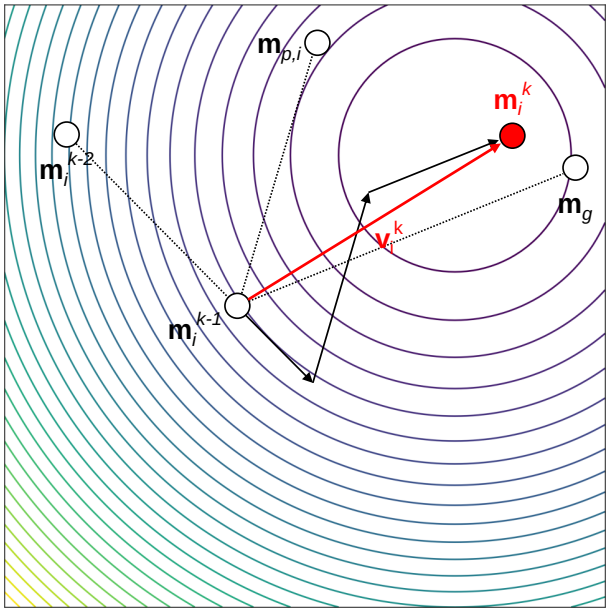

Figure 2.6: Principle of PSO on a 2D misfit function represented by the contour lines. Particle velocity $\mathbf{v}_i^k$ is constructed by adding three weighted terms: the previous velocity $\mathbf{v}_i^{k-1}$ that acts as an inertial term, the cognition term $\left(\mathbf{m}_{p,i} - \mathbf{m}_i^{k-1} \right)$ that accounts for the particle's personal knowledge, and the sociability term $\left(\mathbf{m}_g - \mathbf{m}_i^{k-1} \right)$ that involves the knowledge of the entire swarm.

The inertia weight ω has been introduced by Shi and Eberhart (1998) to help the particles to dynamically adjust their velocities and refine the search near a local minimum. Another formulation using a constriction coefficient based on Clerc (1999) to insure the convergence of the algorithm can be found in the literature. However, Eberhart and Shi (2000) showed that the inertia and constriction approaches are equivalent since the parameters are connected.

Empirical works have concluded that the performance of PSO is sensitive to its control parameters, namely the swarm size n, the maximum number of iterations $k_{\max}$, ω, ϕ_p and ϕ_g. Yet,

these studies have provided some insights on the initialization of some parameters (Van Den Bergh and Engelbrecht (2006)). Eberhart and Shi (2000) empirically found that $\omega = 0.7298$ and $\phi_p = \phi_g = 1.49618$ are good parameter choices that lead to convergent behavior.

The swarm size and the maximum number of iterations have to be carefully chosen depending on the problem and the computer resources available. These two parameters are related since a smaller swarm requires more iterations to converge, while a bigger swarm converges more rapidly. In real-world optimization problems, the computation cost is mainly dominated by the forward modeling. Therefore, the optimization is usually stopped when a predefined number of forward modelings (i.e. computations of misfit functions) is performed. The desired number of forward modelings is controlled by both the swarm size and the maximum number of iterations. Trelea (2003) has studied the effect of the swarm size on several benchmark test functions in 30 dimensions. He found that a medium number of particles ($\approx$ 30 particles) gives the best results in terms of number of misfit function evaluations. Too few particles ($\approx$ 15 particles) gives a very low success rate while too many particles ($\approx$ 60 particles) results in much more misfit function evaluations than needed although it increases the success rate. Piccand, O'Neill and Walker (2008) came to the same conclusion with problems of higher dimensions (up to 500).

Constraints handling

Empirical studies have shown that the particles are prone to go beyond the search space boundaries early throughout the optimization process. This behavior can raise several possible problems such as

- Infeasible optimal solution: as personal best positions are pulled outside of the search space, the global best position is pulled outside of the feasible space as well;
- Wasted effort: if particles fail to find a better solution outside, they are eventually pulled back into the feasible space;
- Divergence: particle's velocity increases drastically as the particle is distant from its personal best and global best.

Several strategies that act directly upon the velocity have been proposed to address this behavior. For instance, the velocity vector can be clamped to lie in $[-\mathbf{V}_{\max}, \mathbf{V}_{\max}]$ to keep the particle from moving too far from the model parameter space (Clerc and Kennedy (2002), Van Den Bergh and Engelbrecht (2006)). However, Angeline (1998) showed that velocity clamping is not sufficient to properly control the step size. Besides, the optimal clamping threshold $\mathbf{V}_{\max}$ is problem-dependent. Another factor that can potentially influence the performance of PSO is how velocities are initialized. Engelbrecht (2012) demonstrated that particles tend to leave the search space independently of the velocity initialization approach and that the best approach is to initialize particles to zero.

Therefore, it may be required to constrain the particles to stay within the feasible space by *repairing* infeasible solutions. Several approaches are available in the literature of evolutionary algorithms, such as:

- Random: this approach is the most common and simply consists in resampling each parameter of a solution that violates the feasible space boundaries, as shown in Figure 2.7 (left);

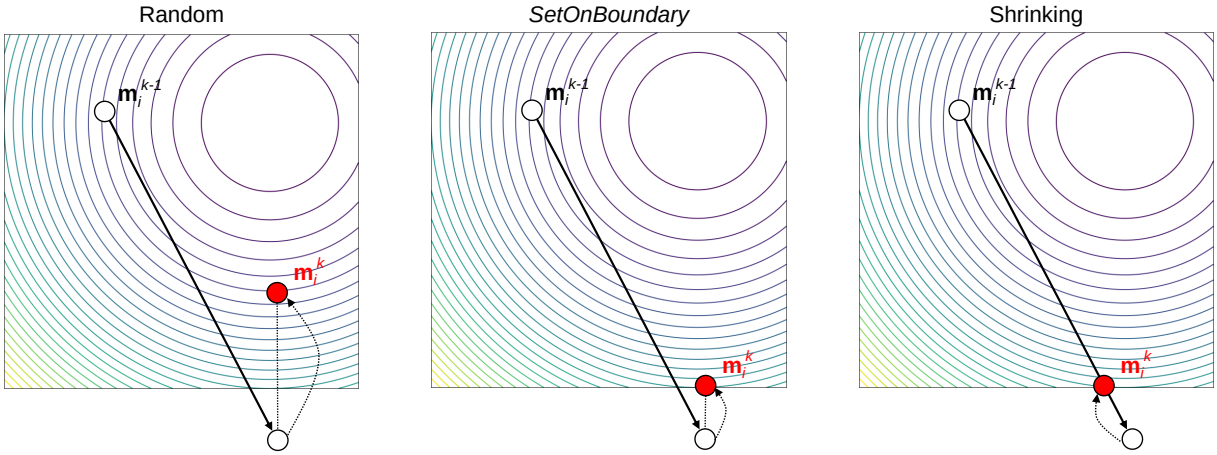

Figure 2.7: Constraints handling approaches: Random, *SetOnBoundary* and Shrinking.

- *SetOnBoundary*: any solution outside of the feasible space is reset on the bound of the parameters it violates, as shown in Figure 2.7 (middle);

- Shrinking: the infeasible solution is set on the intersection of the line joining the parent position to the child position and the violated boundary, as shown in Figure 2.7 (right).

Random approach explicitly maintains diversity within the population and is the one used in DE (Section 2.2.1). However, it cannot be applied to PSO as it disrupts particle dynamics and can potentially lead to divergent behavior. The *SetOnBoundary* and Shrinking approaches are alike but only the latter preserves the particle's initial trajectory. Padhye, Mittal and Deb (2015) showed that Shrinking approach is the most versatile method for PSO.

Neighborhood topologies

In the presented PSO, each particle is connected to every other particle in the swarm following a *star* topological structure. This variant of PSO is known as *gbest* (global best) PSO. It has been shown that the structure of a social network strongly affects the communication and performance within a social group (Bavelas (1950)). Likewise, performance of PSO is strongly affected by the topology defining the interactions between the particles within a swarm. The most commonly used topologies are

- *Star* (known as *gbest* PSO): the swarm is fully connected and every particle communicates with every other particles in the swarm. Each particle is thus attracted by the best position ever visited by the swarm. Information is more rapidly spread within the swarm which results in faster convergence compared to other topologies but with higher chance of premature convergence;

- *Ring* (known as *lbest* PSO): each particle is connected to its K adjacent neighbors. Therefore, each particle is attracted by the best particle in its neighborhood instead of the global best. This topology has the advantage to explore different regions of the search space simultaneously at the cost of slower convergence;

- *Wheel*: a single particle is connected to every other particles which are only connected to that one;

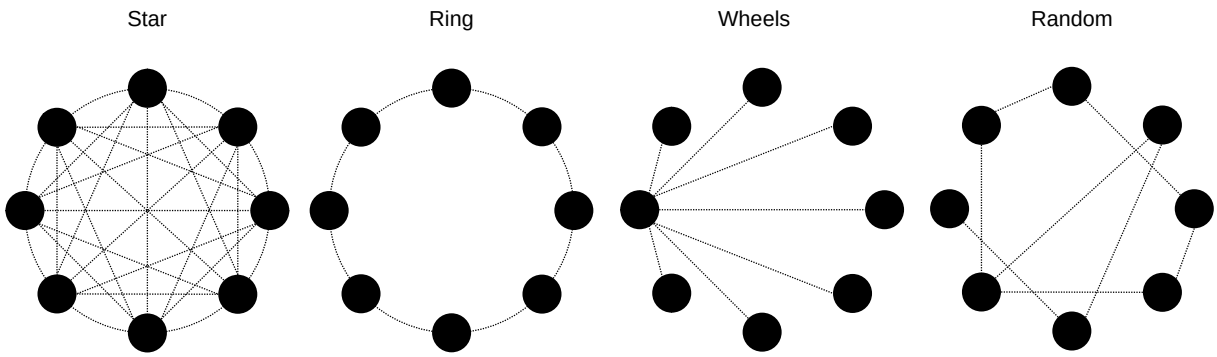

Figure 2.8: Topologies of PSO: Star, Ring, Wheels and Random.

- *Random*: for *n* particles, *n* random symmetrical connections are assigned between each pair of particles.

The aforementioned topologies are illustrated in Figure 2.8. Kennedy (1999) has studied these topologies and concluded that the *star* topology performed better than the others. A more recent study compared other topologies and also came to the conclusion that *gbest* PSO converges faster but with lower diversity (Figueiredo and Ludermir (2014)).

2.2.3 Covariance Matrix Adaptation - Evolution-Strategy

The Covariance Matrix Adaptation - Evolution Strategy (CMA-ES) (Hansen, Müller and Koumoutsakos (2003)) is considered as the state-of-the-art method for stochastic numerical optimization and is derived from the natural gradient. It is a second-order method similar to Quasi-Newton methods (yet randomized) that has been designed to approximate the contour lines of the objective function by adapting the covariance matrix of a multivariate gaussian distribution to the landscape of the misfit function. It belongs to the class of Evolution strategies (ES) that has been introduced in the early seventies (Rechenberg (1973)). First ES were designed to perform the search without self-learning of the parameter dependencies and have been gradually improved over time which has led to the CMA-ES.

Mutation: sampling

In CMA-ES, a population consists of λ models called offspring sampled from a multivariate gaussian distribution, following

$$\forall i \in [1, \lambda], \mathbf{m}_i^k \sim \bar{\mathbf{m}}^{k-1} + \sigma^{k-1} \mathcal{N}\left(\mathbf{0}, \mathbf{C}^{k-1}\right) \tag{2.8}$$

where $\mathcal{N}\left(\mathbf{0}, \mathbf{C}^{k-1}\right)$ denotes a multivariate gaussian distribution with zero mean and covariance matrix $\mathbf{C}^{k-1}$, $\bar{\mathbf{m}}^{k-1}$ is the mean vector of the distribution and σ^{k-1} is the step size. The default population size grows logarithmically with dimensionality following $\lambda = 4 + 3 \log d$ and is recommended to be much larger for multi-modal misfit functions.

The performance of ES depends on the adjustments of their internal parameters and no improvement in the search can be expected otherwise (Beyer *et al.* (2002)). In ES, mutation is regarded as the main operator and its distribution should be adapted during the optimization process to favor the production of better offspring in the next generations. In CMA-ES, the mutation distribution is constantly adjusted by moving its mean and adapting its covariance matrix.

Selection and recombination: moving the mean

The λ newly sampled offspring are evaluated on the misfit function and ranked according to their values. Then, the mean vector is moved toward the weighted mean formed by the recombination of the μ best individuals selected among the λ offspring, following

$$\bar{\mathbf{m}}^k = \bar{\mathbf{m}}^{k-1} + \sum_{i=1}^{\mu} \omega_i \left(\mathbf{m}_{i:\lambda}^k - \bar{\mathbf{m}}^{k-1} \right), \quad \text{with} \quad \sum_{i=1}^{\mu} \omega_i = 1, \quad \omega_1 \geq \omega_2 \geq \ldots \geq \omega_\mu > 0 \tag{2.9}$$

where $i{:}\,\lambda$ denotes the index of the i^{th} best individual offspring, and the weights $\boldsymbol{\omega} = \{\omega_1, \ldots, \omega_\mu\}$ control the influence of each selected individual (higher for better ranked parents). The number of parents μ is usually set to $\lambda/2$.

Adaptation of the covariance matrix

First, consecutive moves of the mean $\frac{\bar{\mathbf{m}}^k - \bar{\mathbf{m}}^{k-1}}{\sigma^{k-1}}$ are tracked in time using an *evolution path* $\mathbf{p}_c^k$ (i.e. the path the population takes over a number of generations), and reads

$$\mathbf{p}_c^k = (1-c)\,\mathbf{p}_c^{k-1} + \sqrt{c\,(2-c)\,\mu_{\text{eff}}}\,\frac{\bar{\mathbf{m}}^k - \bar{\mathbf{m}}^{k-1}}{\sigma^{k-1}} \tag{2.10}$$

where $c = \frac{d}{d+4}$ and $\mu_{\text{eff}} = \frac{1}{\sum_{i=1}^{\mu} \omega_i^2}$ act as normalization coefficients. The CMA-ES adapts the covariance matrix of the mutation distribution by performing two critical updates, namely the rank-one and the rank-μ updates. The rank-one update reinforces the likelihood of steps in the vicinity direction of the evolution path $\mathbf{p}_c^k$ while the rank-μ update exploits information from the distribution of the current population by computing a covariance matrix as a weighted sum of covariances of the consecutive steps of the μ selected parents, which is written

$$\mathbf{C}^k = \left(1 - c_1 - c_\mu \sum \omega_i\right) \mathbf{C}^{k-1} + c_1 \underbrace{\mathbf{p}_c^k \mathbf{p}_c^{k\top}}_{\text{rank-one update}} \tag{2.11}$$

$$+ \underbrace{c_\mu \sum_{i=1}^{\lambda} \frac{\omega_i}{\sigma^{k-1}} \left(\mathbf{m}_{i:\lambda}^k - \bar{\mathbf{m}}^{k-1} \right) \left(\mathbf{m}_{i:\lambda}^k - \bar{\mathbf{m}}^{k-1} \right)^\top}_{\text{rank-}\mu\text{ update}} \tag{2.12}$$

with $c_1 \leq 1$ and $c_\mu \leq 1$ being the learning rates, and the first term that accounts for the information from the previous covariance matrices decaying exponentially with time. The principle of CMA-ES is illustrated in Figure 2.9.

Step size control

The adaptation of the covariance matrix does not control the overall scale of the distribution, only the directions and lengths of its principal axis. Therefore, a second evolution path $\mathbf{p}_\sigma^k$ inherited from Hansen and Ostermeier (1996) is considered to cumulate the lengths of the consecutive step sizes, following

$$\mathbf{p}_\sigma^k = (1-c_\sigma)\,\mathbf{p}_\sigma^{k-1} + \sqrt{c_\sigma\,(2-c_\sigma)\,\mu_{\text{eff}}}\,\mathbf{C}^{k-1^{-\frac{1}{2}}}\,\frac{\bar{\mathbf{m}}^k - \bar{\mathbf{m}}^{k-1}}{\sigma^{k-1}} \tag{2.13}$$

where $c_\sigma = \frac{\mu_{\text{eff}}+2}{d+\mu_{\text{eff}}+3}$ is a relaxation coefficient that decays the contribution of previous steps with time. This second evolution path contains information on the correlations between consecutive

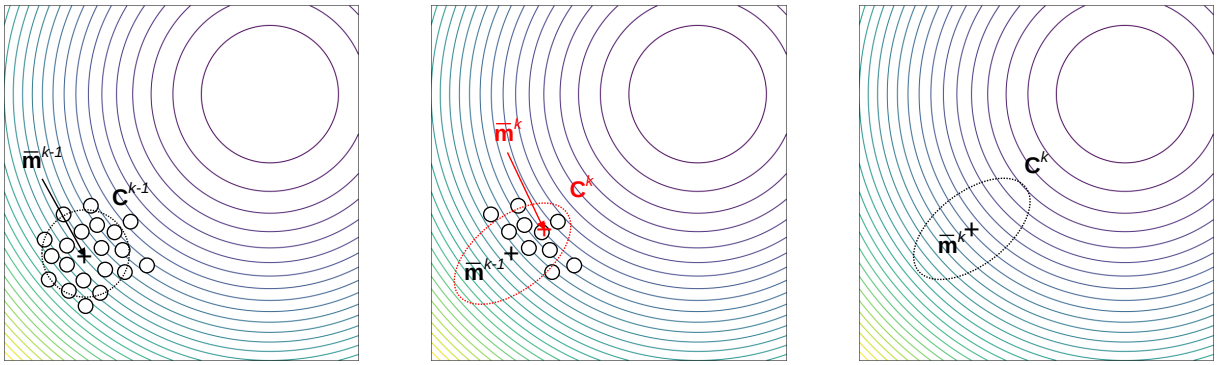

Figure 2.9: Principle of CMA-ES on a 2D misfit function represented by the contour lines. The population should move toward the upper right corner. (Left) Sample of $\lambda = 20$ offspring distributed accordingly to $\mathcal{N}\left(\bar{\mathbf{m}}^{k-1}, \mathbf{C}^{k-1}\right)$. (Middle) $\mu = 10$ best individuals selected to update the mean and covariance matrix. (Right) Mutation distribution for the next generation. Adapted from Hansen (2011).

steps. If the consecutive steps are parallel correlated, it means that they are going into the same direction which could have been done with fewer but longer steps. If they are anti-parallel correlated, they are canceling each other out. Therefore, consecutive steps must have no correlation between them to improve the mutation process. Geometrically, it means that consecutive steps should be perpendicular to each other (see Figure 2.10).

Thus, the length of the second evolution path $\mathbf{p}_\sigma^k$ is compared to its expected length upon random selection which removes the correlations between consecutive steps as they are uncorrelated under random selection, which reads

$$\sigma^k = \sigma^{k-1} \left(\frac{c_\sigma}{d_\sigma} \left(\frac{\left\| \mathbf{p}_\sigma^k \right\|}{E\left[\left\| \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right) \right\| \right]} - 1 \right) \right) \tag{2.14}$$

with $d_\sigma \approx 1$ being a damping parameter. If the actual evolution path is longer than expected due to parallel correlation, the step size is increased. On the other hand, if the actual evolution path is shorter than expected, the step size is decreased.

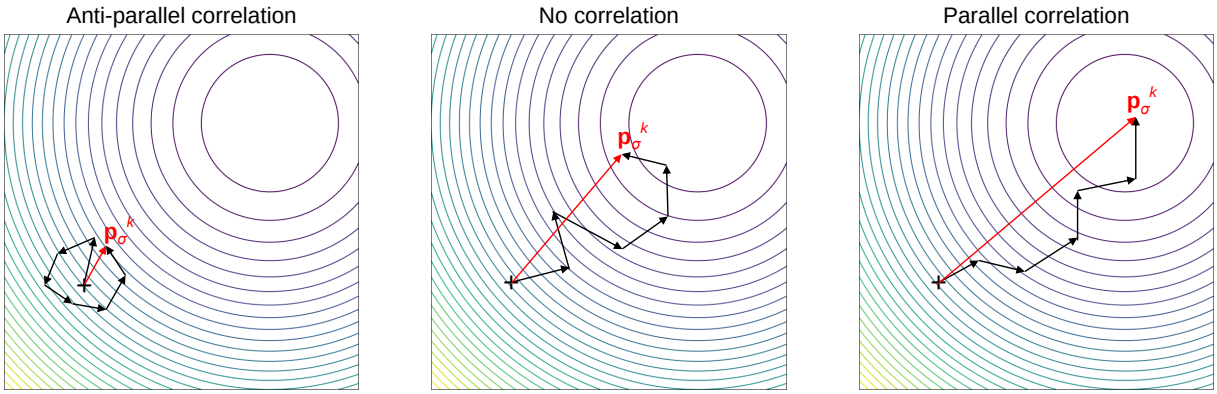

Figure 2.10: Step size adaptation in CMA-ES. The lengths of each single step size are comparable. (Left) Anti-parallel correlation: the consecutive steps cancel each other out resulting in a short cumulation path. (Middle) No correlation: the consecutive steps are perpendicular and the length of the cumulation path is ideal. (Right) Parallel correlation: the consecutive steps are pointing to the same direction resulting in a long cumulation path. Adapted from Hansen (2011).

Termination criterions

In CMA-ES, the optimization process is stopped when one of the following termination criterion is met:

1. *NoEffectAxis*: stop if adding a 0.1-standard deviation vector in any principal axis direction of the covariance matrix does not change the mean;
2. *NoEffectCoord*: stop if adding 0.2-standard deviations to any single parameter does not change the mean;
3. *ConditionCov*: stop if the condition number of the covariance matrix exceeds 10^{14};
4. *EqualFunValues*: stop if the range of the best misfit function values of the last $10 + 30d/\lambda$ generations is zero;
5. *TolX*: stop if the standard deviation of the gaussian distribution and $\sigma \mathbf{p}_c$ are smaller than *TolX* for all parameters. By default, *TolX* is set to 10^{-12} times the initial σ;
6. The maximum number of iterations is reached.

Termination criterions 1 to 5 can be useful to restart an independent optimization process with twice the population size (Auger and Hansen (2005)).

Constraints handling

The standard approach for constraints handling consists in penalizing the misfit function values of infeasible models. Hansen (2010) describes a constraints algorithm where each evaluated model is guaranteed to lie within the feasible space. The misfit of an infeasible model $\mathbf{m}$ is calculated by evaluating the misfit of the closest feasible model $\mathbf{m}^{\text{feas}}$ (i.e. on the boundary of the feasible space) to which is added a penalty term weighted and scaled parameter wise that depends on the distance to the feasible space. For each parameter, the weights are written

$$\forall i \in [1, d], \gamma_i = \frac{2\delta_{\text{fit}}}{\sigma^2 \times \frac{1}{d}\text{diag}(\mathbf{C})} \tag{2.15}$$

where δ_{fit} is the median of the interquartile range of the unpenalized misfit function values from the last $20 + 3d/\lambda$ generations.

In order to prevent the mean from moving too far away from the feasible space, for each parameter, the weight γ_i is multiplied by a factor $1.1^{\max(1,\mu_{\text{eff}}/(10d))}$ if the mean is outside of the feasible space and its distance to the bound is larger than $3 \times \sigma\sqrt{C_{ii}} \times \max\left(1, \sqrt{d}/\mu_{\text{eff}}\right)$ (as the distance to the optimum on the sphere function is proportional to $\sigma\sqrt{d}/\mu_{\text{eff}}$). Finally, the misfit function of the infeasible model $\mathbf{m}$ is penalized according to

$$E(\mathbf{m}) = E\left(\mathbf{m}^{\text{feas}}\right) + \frac{1}{d}\sum_{i=1}^{d} \gamma_i \frac{\left(m_i^{\text{feas}} - m_i\right)^2}{\xi_i} \tag{2.16}$$

in which $\xi_i = \exp\left(0.9\left(\log(C_{ii}) - \frac{1}{d}\text{diag}(\log(\mathbf{C}))\right)\right)$ scales the distance for each parameter with respect to the covariance matrix.

2.3 Sample codes

In the following, I provide sample codes written in Python for the three evolutionary algorithms described in this chapter. For consistency with the notation adopted, a population of models is denoted by `M`, `n` and `d` respectively represent the population size and the dimensionality, `kmax` is the maximum number of iterations, `E` is the misfit function to minimize, `m_min` and `m_max` are the lower and upper boundaries of the model parameter space, respectively. The sample codes extensively use the vectorization capability of NumPy for faster sequential computation.

```
import numpy as np

# Parameters
n = 10                                          # Population size
d = 5                                           # Number of dimensions
kmax = 2000                                     # Maximum number of iterations

# Misfit function to minimize (here Rosenbrock)
E = lambda x: 100*np.sum((x[1:]-x[:-1]**2)**2)+np.sum((1-x[:-1])**2)
m_min = np.full(d, -5.12)                       # Lower boundary
m_max = np.full(d, 5.12)                        # Upper boundary
```

2.3.1 Differential Evolution

```
# Parameters
F = 0.9                                         # Differential weight
CR = 0.5                                        # Crossover probability

# Initialize variables
M = np.random.uniform(m_min, m_max, (n, d)) # Initialize models
pfit = np.array([ E(m) for m in M ])        # Initialize model misfits
mg = M[np.argmin(pfit)]                     # Initialize global best model

# Repeat until stopping criterion is met
k = 1                                           # Initialize iteration number
converge = False                                # Initialize stopping criterion
while not converge:
    k += 1

    # Mutation
    idx = [ [ j for j in range(n) if j != i ] for i in range(n) ]
    r = np.transpose([ np.random.choice(i, 3, replace = False) for i in idx ])
    V = M[r[0]] + F * ( M[r[1]] - M[r[2]] )

    # Recombination
    r1 = np.random.rand(n, d)
    irand = np.random.randint(d, size = n)
    mask = np.eye(d, dtype = bool)[irand]
    mask = np.logical_or(mask, r1 <= CR)
    U = np.where(mask, V, M)
```

```
    # Selection
    fit = np.array([ E(m) for m in U ])      # Evaluate misfit functions
    idx = fit < pfit                         # Locate improved individuals
    pfit[idx] = fit[idx]                     # Update model misfits
    M[idx] = U[idx]                          # Update population
    mg = M[np.argmin(pfit)]                  # Update global best model

    # Test convergence (here only maximum number of iterations)
    converge = k == kmax
```

2.3.2 Particle Swarm Optimization

```
# Parameters
w = 0.729                                    # Inertia weight
c1 = 1.49618                                 # Cognition parameter
c2 = 1.49618                                 # Sociability parameter

# Initialize variables
V = np.zeros((n, d))                         # Initialize velocities
M = np.random.uniform(m_min, m_max, (n, d)) # Initialize models
pfit = np.array([ E(m) for m in M ])         # Initialize personal best misfits
pbest = np.array(M)                          # Initialize personal best models
mg = M[np.argmin(pfit)]                      # Initialize global best model

# Repeat until stopping criterion is met
k = 1                                        # Initialize iteration number
converge = False                             # Initialize stopping criterion
while not converge:
    k += 1
    r1 = np.random.rand(n, d)
    r2 = np.random.rand(n, d)
    V = w*V + c1*r1*(pbest-M) + c2*r2*(mg-M)# Update velocities
    M = M + V                                # Update models
    fit = np.array([ E(m) for m in M ])      # Evaluate misfit functions
    idx = fit < pfit                         # Locate improved particles
    pfit[idx] = fit[idx]                     # Update personal best misfits
    pbest[idx] = M[idx]                      # Update personal best models
    mg = M[np.argmin(pfit)]                  # Update global best model

    # Test convergence (here only maximum number of iterations)
    converge = k == kmax
```

2.3.3 Covariance Matrix Adaptation - Evolution Strategy

```
# Parameters
xmean = np.random.uniform(m_min, m_max)      # Initial mean
```

```
sigma = ( m_max - m_min ) / 3.                  # Step size

# Selection strategy parameters
mu = int(0.5 * n)                               # Number of parents
weights = np.log(mu + 0.5) - np.log(np.arange(1, mu+1))
weights /= np.sum(weights)
mueff = np.sum(weights)**2 / np.sum(weights**2)

# Adaptation strategy parameters
cc = ( 4. + mueff / d ) / ( d + 4. + 2. * mueff / d )
cs = ( mueff + 2. ) / ( d + mueff + 5. )
c1 = 2. / ( ( d + 1.3 )**2 + mueff )
cmu = min(1. - c1, 2. * ( mueff - 2. + 1. / mueff ) / ( ( d + 2. )**2 + mueff ))
damps = 1. + 2. * max(0., np.sqrt( ( mueff - 1. ) / ( d + 1. ) ) - 1.) + cs

# Initialize dynamic (internal) strategy parameters and constants
pc = np.zeros(d)
ps = np.zeros(d)
B = np.eye(d)
D = np.ones(d)
C = np.eye(d)
invsqrtC = np.eye(d)
chind = np.sqrt(d) * ( 1. - 1. / ( 4. * d ) + 1. / ( 21. * d**2 ) )

# Repeat until stopping criterion is met
eigeneval = 0                                   # Initialize eigeneval
n_eval = 0                                      # Initialize n_eval
k = 1                                           # Initialize iteration number
converge = False                                # Initialize stopping criterion
while not converge:
    k += 1

    # Generate lambda offspring
    arx = np.array([ xmean + sigma * np.dot(B, D*np.random.randn(d))
                     for i in range(n) ])
    arfitness = np.array([ E(m) for m in arx ])
    n_eval += n

    # Sort by fitness and compute weighted mean into xmean
    arindex = np.argsort(arfitness)
    xold = np.array(xmean)
    xmean = np.dot(weights, arx[arindex[:mu]])

    # Cumulation
    ps = ( 1. - cs ) * ps \
         + np.sqrt( cs * ( 2. - cs ) * mueff ) \
         * np.dot(invsqrtC, xmean - xold) / sigma
    hsig = np.linalg.norm(ps) / np.sqrt( 1. - ( 1. - cs )**(2.*n_eval/n) ) \
           / chind < 1.4 + 2. / ( d + 1. )
    pc = ( 1. - cc ) * pc \
         + hsig * np.sqrt( cc * ( 2. - cc ) * mueff ) \
```

```
        * ( xmean - xold ) / sigma

    # Adapt covariance matrix C
    artmp = ( arx[arindex[:mu]] - np.tile(xold, (mu, 1)) ) / sigma
    C = ( 1. - c1 - cmu ) * C \
        + c1 * ( np.outer(pc, pc) + ( 1. - hsig ) * cc * ( 2. - cc ) * C ) \
        + cmu * np.dot(np.dot(artmp.transpose(), np.diag(weights)), artmp)

    # Adapt step size sigma
    sigma *= np.exp( ( cs / damps ) * ( np.linalg.norm(ps) / chind - 1. ) )

    # Diagonalization of C
    if n_eval - eigeneval > n / ( c1 + cmu ) / d / 10.:
        eigeneval = n_eval
        C = np.triu(C) + np.triu(C, 1).transpose()
        D, B = np.linalg.eigh(C)
        idx = np.argsort(D)
        D = D[idx]
        B = B[:,idx]
        D = np.sqrt(D)
        invsqrtC = np.dot(np.dot(B, np.diag(1./D)), B.transpose())

    # Test convergence (here only maximum number of iterations)
    converge = k == kmax
```

2.4 Parallel implementation

Evolutionary algorithms operate on a set of concurrent models represented by individuals within a population, and are therefore intrinsically parallel. They have been used to solve geophysical inverse problems since the early nineties (Sambridge and Drijkoningen (1992), Boschetti, Dentith and List (1996)). However, due to the large number of forward modelings required to solve a given optimization problem, these algorithms turned out to be impractical and have been disregarded in favor of derivative-based approaches that are computationally more efficient. Nevertheless, with the rise in computational power mainly brought by modern supercomputers that we have witnessed in the recent years, evolutionary algorithms are becoming an interesting alternative as the forward modelings can be performed in parallel. In this thesis, I principally exploit this characteristics of evolutionary algorithms to solve seismic tomography problems in reasonable computation time.

2.4.1 About supercomputers

In order to better understand the parallel programming paradigms introduced in Section 2.4.2, it is essential to first describe the different elements involved in a supercomputer. Nowadays supercomputers are systems made of Symmetric Multi-Processor machines (SMP) interconnected to each other to form a unified computing resource. A SMP machine is a hardware system composed of two or more identical processors connected to a shared main memory. A diagram of a SMP machine is sketched in Figure 2.11 followed by a brief definition of the different terms related to supercomputers.

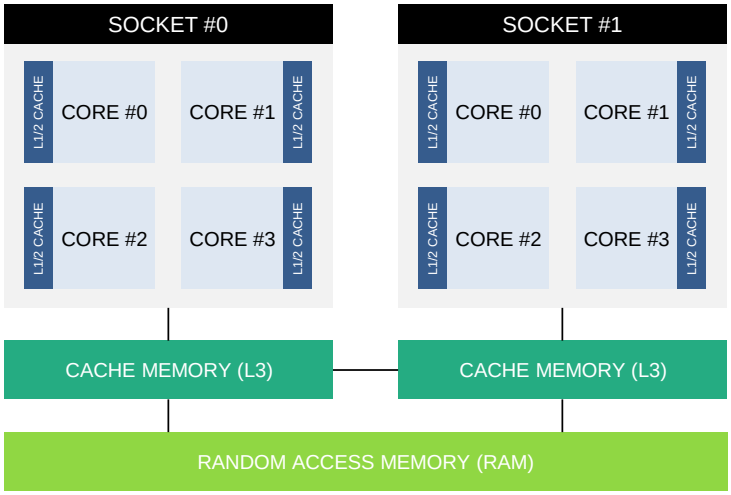

Figure 2.11: Diagram of a SMP machine made of 2 multi-core CPUs. A supercomputer is composed of several interconnected SMP machines.

- *Process*: a process is an independent program with its own memory space that runs on a computer;

- *Thread*: a thread is a child process tied to a parent process and runs on a shared memory space. A process may have several threads (at least one thread);

- *Core*: a core is a logical processing unit that can independently execute process threads;

- *CPU*: CPU stands for *Central Processing Unit* and is also called *processor*. The CPU is the physical chip that executes the instructions of a process. A CPU may have several cores and is connected on a *socket* to the computer motherboard;

- *Random Access Memory (RAM)*: the RAM is the main memory of a computer that stores the data currently used by different processes on a machine. A *shared memory* typically refers to a block of RAM that can be accessed by different process threads. A *distributed memory* refers to a computer system where each process has its own local memory and message passing is required to exchange data between processes;

- *Cache memory*: the cache memory is a smaller memory which copies and stores frequently used main memory data and instructions. The cache memory is closer to the CPU and CPU can access faster to *cached* data than data in the RAM. Modern CPUs have several levels of cache organized hierarchically (L1, L2, L3). On the one hand, the L1 and L2 levels of cache are tied to and only accessible by a single core. On the other hand, cores on the same chip are all connected to the shared L3 cache memory.

On a supercomputer made of several SMP nodes, parallel programming can be achieved using different paradigms that describe how different processes interact with each other. The next section describes the parallel programming paradigm that will be used throughout this thesis to efficiently solve seismic tomography problems using evolutionary algorithms.

2.4.2 Hybrid parallel programming

Theoretically, parallel programming simply consists in using more than one core to execute a program. In practice, it requires the parallel programming paradigm that describes the interactions between each process to be specified. Two libraries are most often used depending on the paradigm adopted:

- On a shared memory architecture, *OpenMP* is the standard library which allows multi-threaded execution of a process. OpenMP is principally employed for loop parallelization (with no dependencies) and can only be used for shared memory programming;

- On a distributed memory architecture, message passing is needed to exchange data between each process, in particular with the *Message Passing Interface* library (*MPI*). In the message passing paradigm, a master-slave scheme is commonly adopted where a master process sends new task data to slave processes. Besides, MPI is applicable to a wider range of problems than OpenMP and can be run on both shared and distributed memory architectures. However, its main performance bottleneck is the speed of communication between each process (e.g. the bandwidth and latency of the network between the SMP nodes).

Modern supercomputers combine features of both shared and distributed memory systems, and parallel programming can be achieved using a hybrid paradigm where the memory is shared at the node level and distributed above. More specifically, MPI can be employed for message passing between multi-threaded processes and OpenMP for loop parallelization inside each process. The hybrid parallel scheme is considered in this thesis as it offers greater flexibility and is particularly adapted to the tomographic problems to solve.

Only independent tasks where the order of execution does not matter can be parallelized. In evolutionary algorithms, the individuals within a population are independent and their misfit function values can therefore be evaluated in parallel. In my implementation, p processes are spawned using MPI each in charge of evaluating the misfit function values of n/p models. A master process is responsible for the population update and manages the distribution of the models to the slave processes that send back only the misfit function values to the master, as depicted in Figure 2.12. Such implementation requires the population size n to be a multiple of the number of available cores for maximum efficiency to avoid idle cores.

It is noteworthy that *asynchronous* implementations of DE and PSO exist where the individuals of a generation are updated one after the other. However, such implementation is not straightforward to parallelize and only the *synchronous* implementations (i.e. simultaneous update of a population) of DE and PSO are considered in this manuscript.

In addition, the computation of the misfit function can also be parallelized depending on the problem to solve. For instance, in traveltime tomography problems such as described in Chapter 3 and Chapter 4, the forward modeling consists in computing a traveltime grid for each source (i.e. shot) using an Eikonal solver. Obviously, the sources are independent and the traveltime grids can be computed in parallel. Within the framework of hybrid parallel programming, t process threads for each MPI process are created using OpenMP to parallelize the computation of the traveltime grids, as shown in Figure 2.13. In the surface wave tomography problem described in Appendix B, the computation of the dispersion functions for each frequency is parallelized using OpenMP.

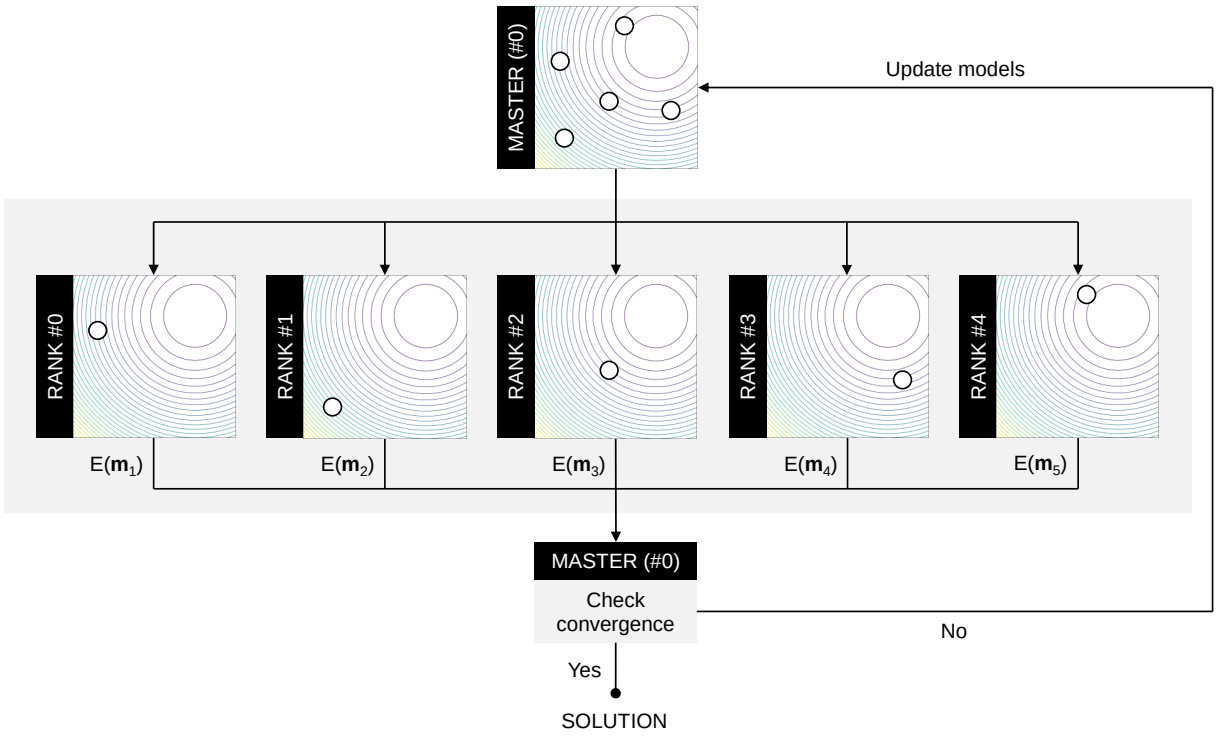

Figure 2.12: Parallel computation of the misfit function values using MPI. A population of models is generated by a master process that evenly scatters the models over the slave processes for concurrent misfit function evaluations. The misfit function values are finally sent back to the master process to assess convergence or to update the model population otherwise.

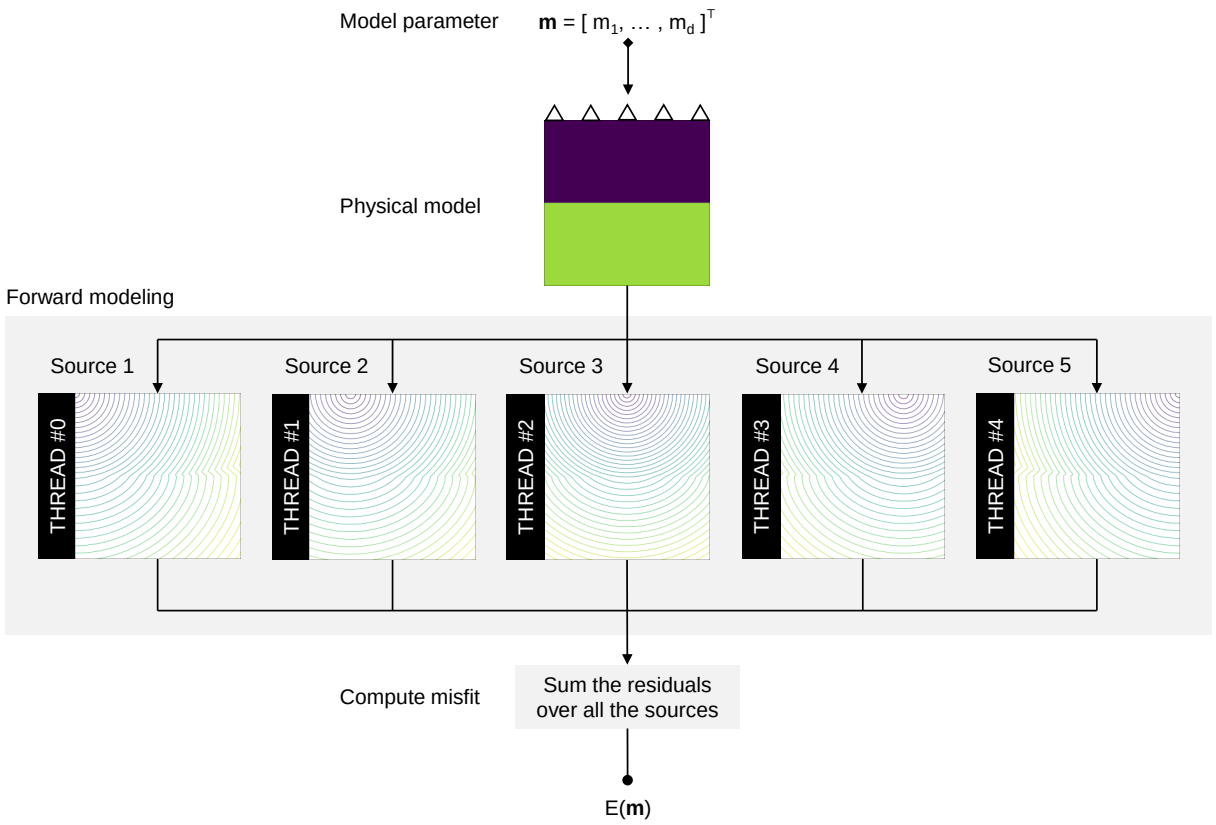

Figure 2.13: Parallel computation of the traveltime grids using OpenMP. A model vector $\mathbf{m}$ defined by d parameters is transformed into a physical velocity model that can be used by an Eikonal solver. The computation of the traveltime grids for each source is scattered over the threads with OpenMP.

2.5 Conclusion

I introduced the principals of evolutionary algorithms in the context of black-box optimization. In its broader sense, evolutionary algorithm refers to all the optimization methods that operate on a concurrent population of independent models which makes them intrinsically parallel. Many evolutionary algorithms that try to mimic phenomena observed in the nature (e.g. evolution, social behavior) have been proposed. Yet, I chose to focus on the three most popular and efficient algorithms known in the literature, namely Differential Evolution (DE), Particle Swarm Optimization (PSO) and Covariance Matrix Adaptation - Evolution Strategy (CMA-ES). In this chapter, I gave a comprehensive description of these algorithms along with sample computer codes written in Python. Because the main interest of evolutionary algorithms in this thesis is their intrinsic parallelisms, I also described the hybrid parallel programming paradigm that will be used throughout this thesis.

The next chapters are devoted to the application of evolutionary algorithms to seismic tomography problems within the hybrid parallel programming framework. PSO is more extensively studied in Chapter 3 for its efficiency and ease of implementation. I present a new optimization algorithm based on PSO that I called Competitive Particle Swarm Optimization (CPSO) which improves its robustness. The same methodology is applied to a surface wave tomography problem in Appendix B with only a different forward modeling and data set. In Chapter 4, the study is broadened to a refraction tomography problem that aims at analyzing the feasibility of evolutionary algorithms to solve a problem in larger dimensions by comparing the performances of the three algorithms described hereinabove.

Chapter 3

1D traveltime tomography

Contents

Dans ce chapitre, je m'intéresse principalement à l'optimisation par essaim particulaire (PSO) pour son efficacité et sa simplicité d'implémentation. J'y présente un nouvel algorithme d'optimisation basé sur la PSO qui résout son principal défaut. J'ai nommé cet algorithme la PSO Compétitive (Competitive Particle Swarm Optimization, CPSO). En tant qu'algorithme évolutionniste, la PSO est particulièrement sujette au problème de stagnation et de convergence prématurée (c'est-à-dire convergence vers un minimum local). Je propose donc une modification de l'algorithme original simple mais efficace qui permet d'améliorer la diversité de l'essaim. Je démontre sur plusieurs fonctions de test que la CPSO est plus robuste que la PSO en termes de convergence et de sensibilité à ses paramètres. De plus, je montre sur une fonction multi-modale que la CPSO peut être utilisée pour une quantification rapide des incertitudes. L'estimation de la densité de probabilité a posteriori est effectuée par échantillonnage de l'espace des paramètres du modèle après plusieurs inversions indépendantes. La méthode est appliquée à un problème de tomographie des temps de première arrivée avec des données réelles 3D de microsismicité acquises dans le contexte de la sismicité induite, et comparée à un échantillonneur conventionnel de type Monte-Carlo par Chaînes de Markov (Markov Chain Monte Carlo, MCMC). Les résultats démontrent que la CPSO est capable d'atteindre un régime stationnaire beaucoup plus rapidement et permet d'obtenir des incertitudes cohérentes avec celles obtenues par un échantillonneur MCMC. Enfin, j'analyse la scalabilité de la CPSO sur ce problème de tomographie en évaluant ses performances parallèles. Les résultats obtenus dans ce chapitre ont notamment été utilisés dans l'Annexe C pour propager les incertitudes liées aux modèles de vitesse aux localisations des évènements microsismiques.

Ce chapitre est écrit sous la forme d'un article publié et a notamment fait l'objet d'une présentation sous forme de poster à la SEG 2016 (Dallas) :

- **Keurfon Luu**, Mark Noble, Alexandrine Gesret, 2016. "A competitive particle swarm optimization for nonlinear first arrival traveltime tomography." *2016 SEG International Exposition and Annual Meeting. Society of Exploration Geophysicists*. doi: 10.1190/segam2016-13840267.1;

- **Keurfon Luu**, Mark Noble, Alexandrine Gesret, Nidhal Belayouni and Pierre-François Roux, 2018. "A parallel competitive Particle Swarm Optimization for non-linear first arrival traveltime tomography and uncertainty quantification." *Computers and Geosciences* 113 (August 2017). Elsevier Ltd: 81–93. doi: 10.1016/j.cageo.2018.01.016.

Abstract

Seismic traveltime tomography is an optimization problem that requires large computational efforts. Therefore, linearized techniques are commonly used for their low computational cost. These local optimization methods are likely to get trapped in a local minimum as they critically depend on the initial model. On the other hand, global optimization methods based on MCMC are insensitive to the initial model but turn out to be computationally expensive. Particle Swarm Optimization (PSO) is a rather new global optimization approach with few tuning parameters that has shown excellent convergence rates and is straightforwardly parallelizable, allowing a good distribution of the workload. However, while it can traverse several local minima of the evaluated misfit function, classical implementation of PSO can get trapped in local minima at later iterations as particles inertia dim. We propose a Competitive PSO (CPSO) to help particles to escape from local minima with a simple implementation that improves the diversity of the swarm. The model space can be sampled by running the optimizer multiple times and by keeping all the

models explored by the swarms in the different runs. A traveltime tomography algorithm based on CPSO is successfully applied to a real 3D data set in the context of induced seismicity.

Keywords: microseismic, traveltime tomography, particle swarm optimization, uncertainty quantification, high performance computing

3.1 Introduction

Tomographic inversion schemes aiming at reconstructing the subsurface structures from seismic traveltime data are widely used (e.g. Rawlinson, Pozgay and Fishwick (2010)). The obtained wave propagation velocity distribution is usually a starting point for further analysis at various scales, from near surface to global scale. For a reliable and quantitative interpretation of the tomographic solution, an accurate velocity model with its associated uncertainties are required.

In spite of the fact that the inversion for the velocities is a totally non-linear problem, very often it is solved with iterative linearized approaches that minimize a misfit function. The misfit function usually measures the difference between observed and computed traveltimes as a function of the velocity model parameters. The linearization makes the implicit assumption of a unique solution which is chosen thanks to a regularization procedure that reduces the solution non-uniqueness (Menke (2012)). This is generally achieved by imposing the solution to be somehow similar or close to an initial a priori model.

The data-model (traveltimes-velocities) relationship can be highly non-linear and requires the use of global optimization methods. In addition, the linearized approaches are not really adapted to provide reliable uncertainties. From a theoretical point of view, to address these two issues, methods based on Markov Chain Monte Carlo (MCMC) that sample the velocity model parameter space are required, such as reversible-jump MCMC (Green (1995), Bodin and Sambridge (2009)), Parallel Tempering (Sambridge (2014)), or Interactive MCMC (Bottero *et al.* (2016)). These global optimization methods can be applied to non-smooth and non-convex functions as they are derivative-free and produce results independent of the initial model. However, they cannot be parallelized and turn out to be prohibitive in terms of computation time.

Another class of global optimization methods has shown growing interest in the last decades. These methods, known as evolutionary algorithms (EA), are inspired by the natural evolution of species and have demonstrated very good convergence rates. While MCMC methods sample the model parameter space by perturbing iteratively a single model, EA work with a population of simultaneous models that evolve toward better models through stochastic processes. This simultaneous evaluation of independent models implies that it is straightforward to parallelize and thus can significantly reduce the computation time. EA include Genetic Algorithm (Sambridge and Drijkoningen (1992), Whitley (1994)), Differential Evolution (Storn and Price (1997), Barros *et al.* (2015)), and Covariance Matrix Adaptation - Evolution Strategy (Hansen, Müller and Koumoutsakos (2003), Grayver and Kuvshinov (2016)).

In this work, we propose to overcome the non-linearity using a rather new EA known as Particle Swarm Optimization (PSO) for its ease of implementation and the low number of control parameters required. PSO has been introduced to study birds flocking and fish schooling (Kennedy and Eberhart (1995)). While it has been extensively used in other engineering domains (e.g. biomedical, signal processing...) for years, PSO has been fairly ignored by the geophysical community until recently. In seismics, PSO has been applied in history matching for reservoir characterization (Mohamed *et al.* (2010), Fernández Martínez *et al.* (2012)), traveltime tomography (Tronicke, Paasche and Böniger (2012), Rumpf and Tronicke (2015), Poormirzaee, Moghadam and Zarean (2015)), and surface wave tomography (Wilken and Rabbel

(2012), Poormirzaee (2016)). Yet, PSO may suffer from premature convergence, in particular for functions with complex landscape. Therefore, we propose and describe a simple modification of PSO to tackle premature convergence and improve its robustness. Although PSO is mainly used as a global optimization method, we show that our implementation not only demonstrates better convergence rates, but also samples correctly the model parameter space, allowing more reliable uncertainty quantification. We apply the method on a real 3D microseismic example and sample the model parameter space which allows us to derive reliable velocity model uncertainties.

3.2 Theory and method

Geophysical inverse problems are underdetermined optimization problems that can be solved by either linear or non-linear techniques (Tarantola and Valette (1982)). Calculated data $\mathbf{d}^{calc}$ are generated by applying the forward modeling operator g, most often non-linear, on the model vector $\mathbf{m} = [m_1, \ldots, m_d]^\top$, with d the number of parameters defining the model

$$\mathbf{d}^{calc} = g(\mathbf{m}). \tag{3.1}$$

Inverse problems consist in determining the model vector $\mathbf{m}$ that minimizes the misfit between the observed data $\mathbf{d}^{obs}$ and the calculated data $\mathbf{d}^{calc}$

$$\mathbf{e}(\mathbf{m}) = \mathbf{d}^{obs} - \mathbf{d}^{calc} = \mathbf{d}^{obs} - g(\mathbf{m}). \tag{3.2}$$

Given an error vector $\mathbf{e}$ (Equation (3.2)), the misfit function is usually defined with an ℓ_p-norm. In geophysical inverse problems, even though other norms can be found in the literature, the ℓ_2-norm is often used

$$\|\mathbf{e}(\mathbf{m})\|_2 = \left[\left(\mathbf{d}^{obs} - g(\mathbf{m})\right)^\top \left(\mathbf{d}^{obs} - g(\mathbf{m})\right)\right]^{\frac{1}{2}}. \tag{3.3}$$

The non-linearity can be addressed by global optimization methods that explore the model parameter space. In this section, we first describe the PSO algorithm before introducing a more robust implementation based on PSO that tackles its shortcomings.

3.2.1 Particle Swarm Optimization

For consistency in the notation, the so-called position vector usually denoted by $\mathbf{x}$ in the literature is denoted by $\mathbf{m}$. Consequently, we only speak in terms of models instead of position vector.

In PSO, the first step is to generate a swarm composed of several models in the model parameter space. The initial models can either be defined a priori or generated given a random distribution (usually uniform). Each model is represented by a particle that interacts with its neighborhood to find the global minimum of the misfit function. Kennedy (1999) has studied several neighborhood topologies and concluded that the global best topology (all the particles are connected to each other) performed better than the others. Thus, we here only consider the global best topology where the neighborhood of each particle is the entire swarm.

At iteration k, a particle i is defined by a model vector $\mathbf{m}_i^k$ and a velocity vector $\mathbf{v}_i^k$ and is adjusted according to its own personal best model and the global best model of the whole swarm. The velocity vector controls how a particle moves in the model parameter space and is initialized to zero (Engelbrecht (2012)). The velocity and the position of each particle are updated following

$$\mathbf{v}_i^k = \omega \mathbf{v}_i^{k-1} + \phi_p \mathbf{r}_p^k \left(\mathbf{m}_{p,i} - \mathbf{m}_i^{k-1}\right) + \phi_g \mathbf{r}_g^k \left(\mathbf{m}_g - \mathbf{m}_i^{k-1}\right) \tag{3.4}$$

$$\mathbf{m}_i^k = \mathbf{m}_i^{k-1} + \mathbf{v}_i^k \tag{3.5}$$

where $\mathbf{m}_{p,i}$ and $\mathbf{m}_g$ are respectively the personal best model of particle i and the global best model of the swarm, $\mathbf{r}_p^k$ and $\mathbf{r}_g^k$ are uniform random number vectors drawn at iteration k, ω is an inertia weight, ϕ_p and ϕ_g are two acceleration parameters that respectively control the cognition and social interactions of the particles.

The inertia weight ω has been introduced by Shi and Eberhart (1998) to help the particles to dynamically adjust their velocities and refine the search near a local minimum. Another formulation using a constriction coefficient based on Clerc (1999) to insure the convergence of the algorithm can be found in the literature. However, Eberhart and Shi (2000) showed that the inertia and constriction approaches are equivalent since the parameters are connected.

Empirical studies have concluded that the performance of PSO is sensitive to its control parameters, namely the swarm size n, the maximum number of iterations $k_{\max}$, ω, ϕ_p and ϕ_g. Yet, these studies have provided some insights on the initialization of some parameters (Van Den Bergh and Engelbrecht (2006)). Eberhart and Shi (2000) empirically found that $\omega = 0.7298$ and $\phi_p = \phi_g = 1.49618$ are good parameter choices that lead to convergent behavior. Although these parameters have shown good results in previous studies, be aware that they can also be tuned according to the optimization problem. The sensitivity of PSO to these parameters is analyzed in Section 3.3.1. Unless explicitly stated, we set $\omega = 0.7298$ and $\phi_p = \phi_g = 1.49618$.

The swarm size and the maximum number of iterations have to be carefully chosen dependently on the problem and the computer resources available. These two parameters are related since a smaller swarm requires more iterations to converge, while a bigger swarm converges more rapidly. In real-world optimization problems, the computation cost is mainly dominated by the forward modeling. Therefore, the optimization is usually stopped when a predefined number of forward modelings (i.e. computations of misfit function values) is performed. The desired number of forward modelings is controlled by both the swarm size and the maximum number of iterations. Trelea (2003) has studied the effect of the swarm size on several benchmark test functions in 30 dimensions. He found that a medium number of particles ($\approx$ 30 particles) gives the best results in terms of number of misfit function evaluations. Too few particles ($\approx$ 15 particles) gives a very low success rate while too many particles ($\approx$ 60 particles) results in much more misfit function evaluations than needed although it increases the success rate. Piccand, O'Neill and Walker (2008) came to the same conclusion with problems of higher dimensions (up to 500).

In the original PSO, the global best position of the swarm is updated in a synchronous fashion. In other words, $\mathbf{m}_g$ is updated at the end of an iteration once the misfit functions of the entire swarm have been evaluated. Carlisle and Dozier (2001) has shown that PSO yields better performance when the particles are evaluated asynchronously (i.e. $\mathbf{m}_g$ is evaluated after each individual misfit evaluation). Synchronous PSO is intrinsically parallel and performs well if the individual misfit function evaluations require the same amount of time, while a parallel asynchronous PSO is not straightforward but could reduce wasted CPU cycles (Schutte *et al.* (2004), Koh *et al.* (2006)). This work only deals with optimization problems with constant misfit computation time, therefore only the synchronous PSO is considered. The algorithm is described in Algorithm 3.3.

3.2.2 Premature convergence

The classical implementation of PSO suffers from premature convergence and stagnation as it can be trapped in a local minimum, particularly when the evaluated misfit function has a complex landscape. The swarm is said to have stagnated when the particles keep moving in the close vicinity of the global best $\mathbf{m}_g$ as particles momentum have faded. We illustrate this statement with the 2D Rastrigin function.

The Rastrigin function is a highly multi-modal function commonly used to benchmark global optimization methods due to its complex landscape containing several local minima. The function is usually defined in $[-5.12, 5.12]^d$ and its global minimum lies in $(0)^d$, d being the number of dimensions.

While PSO is able to find the global minimum with $n = 10$ particles, we apply PSO on the 2D Rastrigin function with $n = 5$ particles in order to illustrate the premature convergence (Figure 3.1). The particles are initialized uniformly in the model parameter space. At iteration 73, PSO has converged prematurely and is trapped in the local minimum $(2, 0)$. At this state, particles inertia have almost vanished and the swarm is stagnating.

3.2.3 Competitive Particle Swarm Optimization

To address premature convergence and stagnation state, Van Den Bergh (2001) proposed to restart the whole swarm anew. Evers and Ben Ghalia (2009) enhanced the method by adapting the search space at each restart. However, after restarting the whole swarm, the particles may either converge to the same/equivalent or even a worse solution. Therefore, we propose hereby a Competitive PSO (CPSO) to avoid such situations. The idea is to improve the diversity of the swarm by renewing part of the population and keeping the "best" particles only. "Worst" particles are reset, allowing a better exploration of the model parameter space. The newly generated particles will try to look for a better minimum while the "best" particles keep searching around the current global best position. If the newest particles find a better minimum, the swarm will gather around the new global best, otherwise they will come back to the current global best until being reset again. We call a reset a "competition". Competition is triggered only when premature convergence is detected. Van Den Bergh (2001) proposed several methods to detect such state:

- Cluster analysis: a percentage of the particles is at a specified Euclidean distance from the global best;
- No improvement of the misfit function: the misfit function does not improve significantly over the last iterations;
- Swarm maximum radius: the distance of the farthest particle from the global best reaches a certain threshold.

We find that the latter works better in our case. Thus, at the iteration k, competition is triggered if the swarm maximum radius δ^k is smaller than a threshold ε. This condition is written as

$$\delta^k = \max_{1 \leq i \leq n} \left(\frac{\left\| \mathbf{m}_i^k - \mathbf{m}_g \right\|}{\|\mathbf{m}_{\max} - \mathbf{m}_{\min}\|} \right) < \varepsilon = \frac{\log(1 + 0.003n)}{\max(0.2, \log(0.01 k_{\max}))} \tag{3.6}$$

where $\|\cdot\|$ denotes the Euclidean norm. The expression of ε has been empirically[1] determined and works well for competition triggering since it prevents the swarm from stagnating for too long in a local minimum for any swarm size n. In order to preserve the convergence property

[1] Van Den Bergh (2001) found that a constant swarm maximum radius threshold $\varepsilon = 10^{-6}$ produced acceptable results on a test set of benchmark functions. However, we noticed poor performance with bigger swarm size since the threshold was hardly achieved (i.e. no competition is triggered). We decided to define the threshold as a function of the swarm size n and chose a logarithmic scale to limit the growth of the threshold ε with bigger swarm size which would otherwise trigger too much competitions. The denominator acts as a scaling factor that squeezes the threshold when the maximum number of iterations $k_{\max}$ is big which allows competition triggering even when $k_{\max}$ is small. The different parameters have been empirically tweaked to obtain good performance on several benchmark functions (mainly Rastrigin and Rosenbrock) with different combinations of swarm sizes and numbers of iterations.

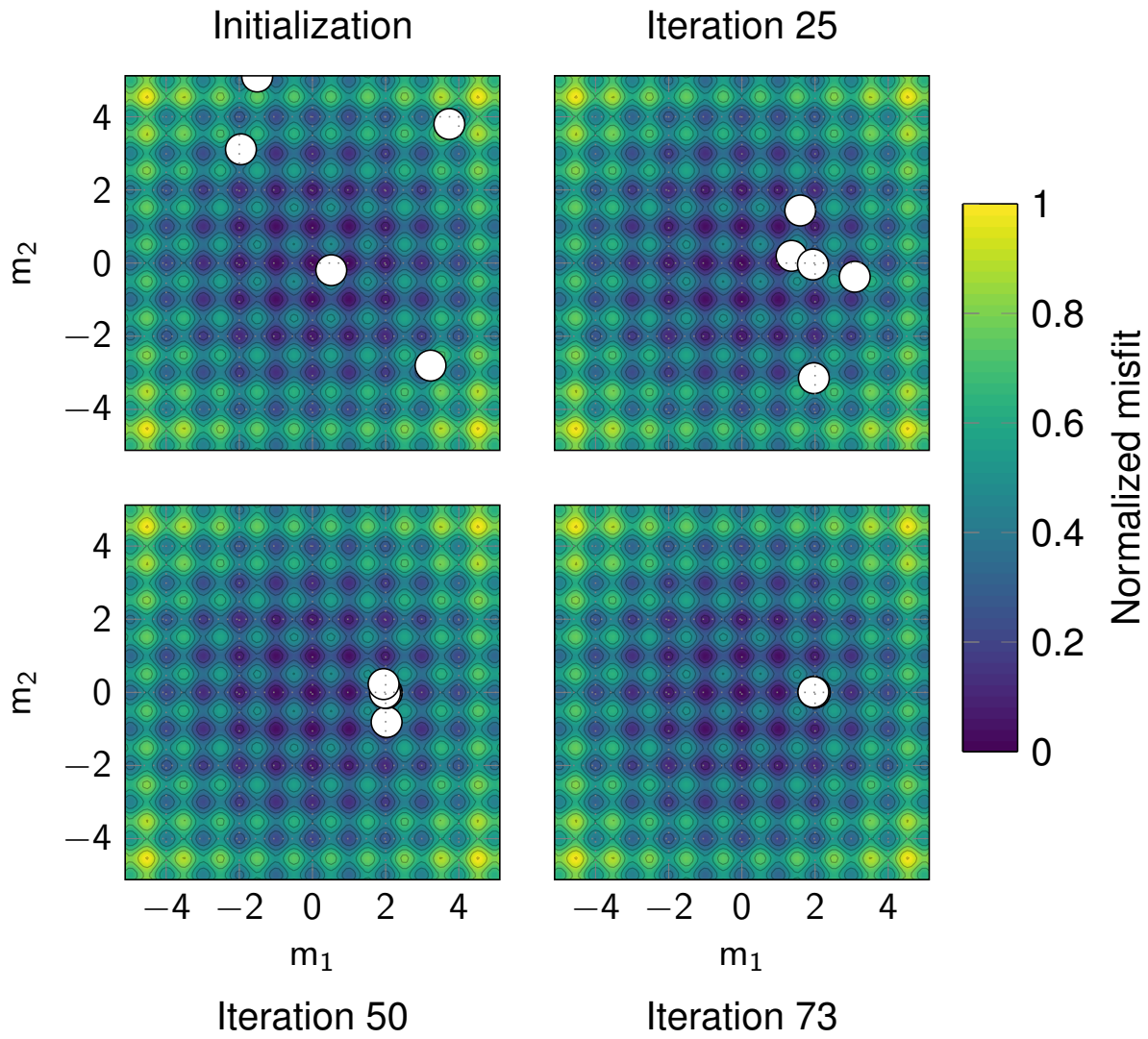

Figure 3.1: Illustration of the premature convergence of PSO on the 2D Rastrigin function with $n = 5$ particles. The 5 particles are uniformly distributed in the model parameter space. Then, the particles converge toward the local minimum $(2, 0)$. From iteration 73, the swarm is trapped in the local minimum.

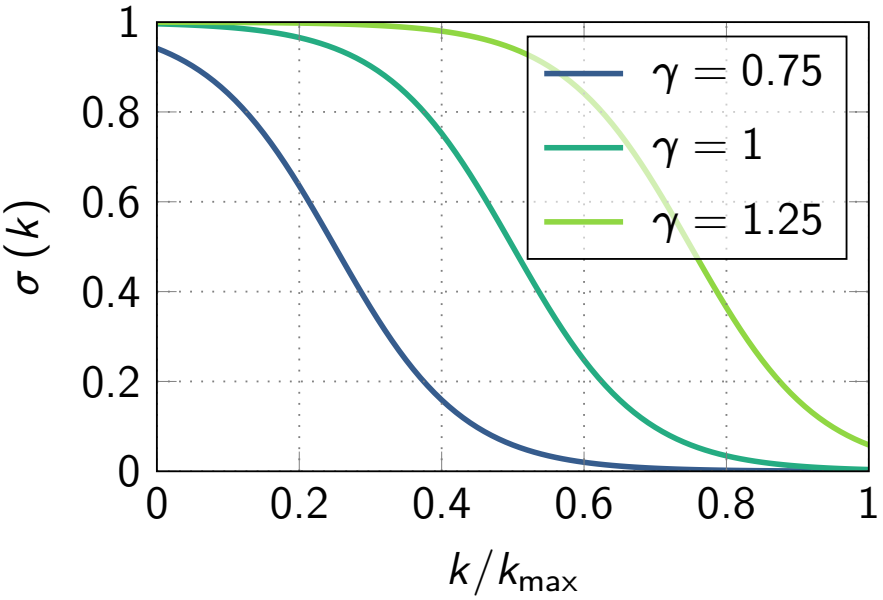

Figure 3.2: Logistic function with different values of competitivity parameter γ. Increasing γ improves the exploration ability (i.e. diversity) of the swarm as more particles are reset. Decreasing γ results in faster convergence with higher chance of entrapment in a local minimum.

of PSO, the proportion of particles to reset should decrease over time. Therefore, we define the proportion of particles to reset at iteration k by a logistic function $\sigma(k)$ (Equation (3.7)) parametrized such that it decreases non-linearly with the iteration number. We introduce a competitivity parameter $\gamma \in [0, 2]$ that controls the position of its inflection point, following

$$\sigma(k) = \left(1 + e^{\frac{1}{0.09}\left(\frac{k}{k_{\max}} - \gamma + 0.5\right)}\right)^{-1} \tag{3.7}$$

with $k_{\max}$ the maximum number of iterations. The logistic function is shown in Figure 3.2 for different values of γ. The parameters of the logistic function have been empirically[2] tweaked in a way that at early iterations, the swarm is competitive and many particles are reset; at later iterations, we let the swarm stagnate in the last minimum found to refine the solution. γ is a problem-dependent parameter. Although we found that $\gamma = 1$ works well in most problems, the parameter can be tuned for faster convergence. A high value of γ increases competitivity between particles, resulting in slower convergence. Low competitivity is recommended for unimodal functions (e.g. Spherical function). When $\gamma = 0$ (no competition), CPSO actually behaves like the original PSO since zero particle are reset for any iteration number. The algorithm is presented in Algorithm 3.4. The competitivity parameter γ is set to 1 from now on.

We apply CPSO on the previous example (Figure 3.3). Premature convergence is detected at iteration 74 and competition is triggered. A particle finds a better local minimum at iteration 81 which allows the whole swarm to escape from the local minimum and to finally gather around the true global minimum. The global minimum is finally found at iteration 186 (the misfit value is lower than a specified threshold).

Figure 3.4 displays the global best misfit value as a function of the iteration number. The black

[2] In the initial form of the equation (Luu, Noble and Gesret (2016)), the number of particles to reset was decreased linearly with time. It was not ideal as several particles were still reset even at the end of the optimization when we would prefer refining the current solution. We chose a logistic function as it allows to model the expected behavior. The different parameters have been visually tweaked in order that the logistic function tends to 1 at early iterations and to 0 at later iterations.

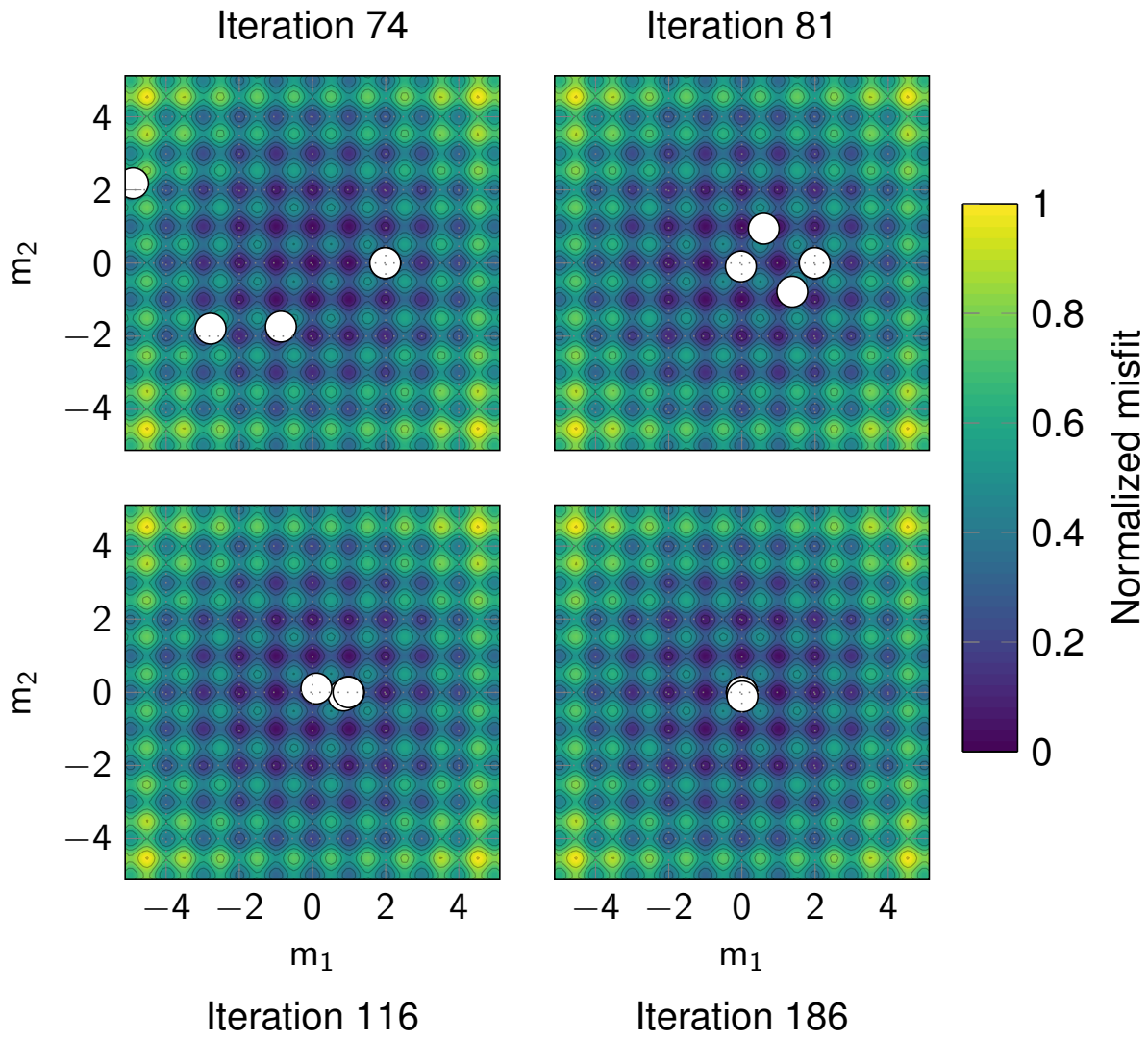

Figure 3.3: Example of competition triggering. Three particles are redistributed uniformly in the model parameter space. At iteration 81, one particle has found the central mode which allows the swarm to escape from the previous local minimum. Finally, the swarm has found the global minimum.

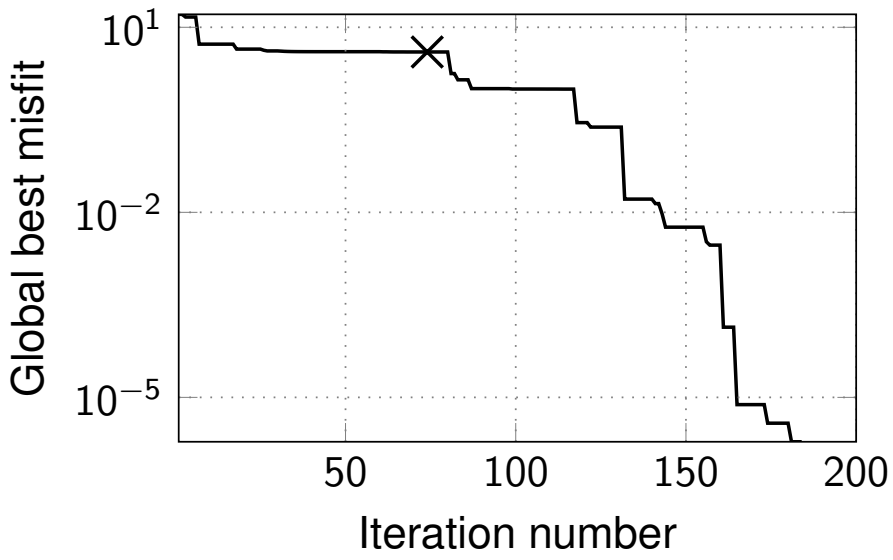

Figure 3.4: Global best misfit as a function of iteration number. Competition triggering is marked by the black cross (iteration 74). Only one reset has been required for the swarm to escape from a local minimum and eventually find the global minimum.

cross marks the iteration when competition has been triggered. The algorithm has required only 1 reset (iteration 74) to be able to find the global minimum.

In this example, CPSO has been able to find the global minimum while PSO fails most of the time with a swarm size of 5 particles. Even though only 1 reset was required for the particles to escape from a local minimum, it should be pointed out that the competition triggering mechanism does not guarantee the swarm to always find a better minimum at each reset, it only improves its chance to escape by improving the diversity of the swarm. The next sections are dedicated to the analysis of the robustness of CPSO with respect to its tuning parameters in higher dimensions.

3.3 Robustness testing

3.3.1 Sensitivity analysis

We analyze the sensitivity of both PSO and CPSO with respect to the variation of the inertia weight ω, and the two acceleration parameters ϕ_p and ϕ_g on the Rastrigin function in 5, 10 and 20 dimensions. We vary ω in the range $[0, 1]$. In order to facilitate the analysis, we set $\phi_p = \phi_g = \phi$ and vary ϕ in the range $[0, 3]$. The swarm size is set to 5 times the dimension. For each couple of parameters (ω, ϕ), 50 trials of 1000 iterations each are performed. The sensitivity with respect to a couple of parameters (ω, ϕ) is characterized by the success rate (SR) defined as the percentage of trials that yield a misfit lower than a specified threshold. The results of the sensitivity analysis are summarized in Figure 3.5.

PSO performs well for a wide range of ω and ϕ with a narrow region of high SR. This means that ω has to be chosen dependently on the parameter ϕ. On the other hand, the high SR region is much wider for CPSO which is therefore more flexible in the choice of ω and ϕ. Notice that the control parameters recommended by Eberhart and Shi (2000) ($\omega = 0.7298$ and $\phi = 1.49618$) are included in the high SR region. Additional results on the Rosenbrock function are shown in Appendix 3.7.4.

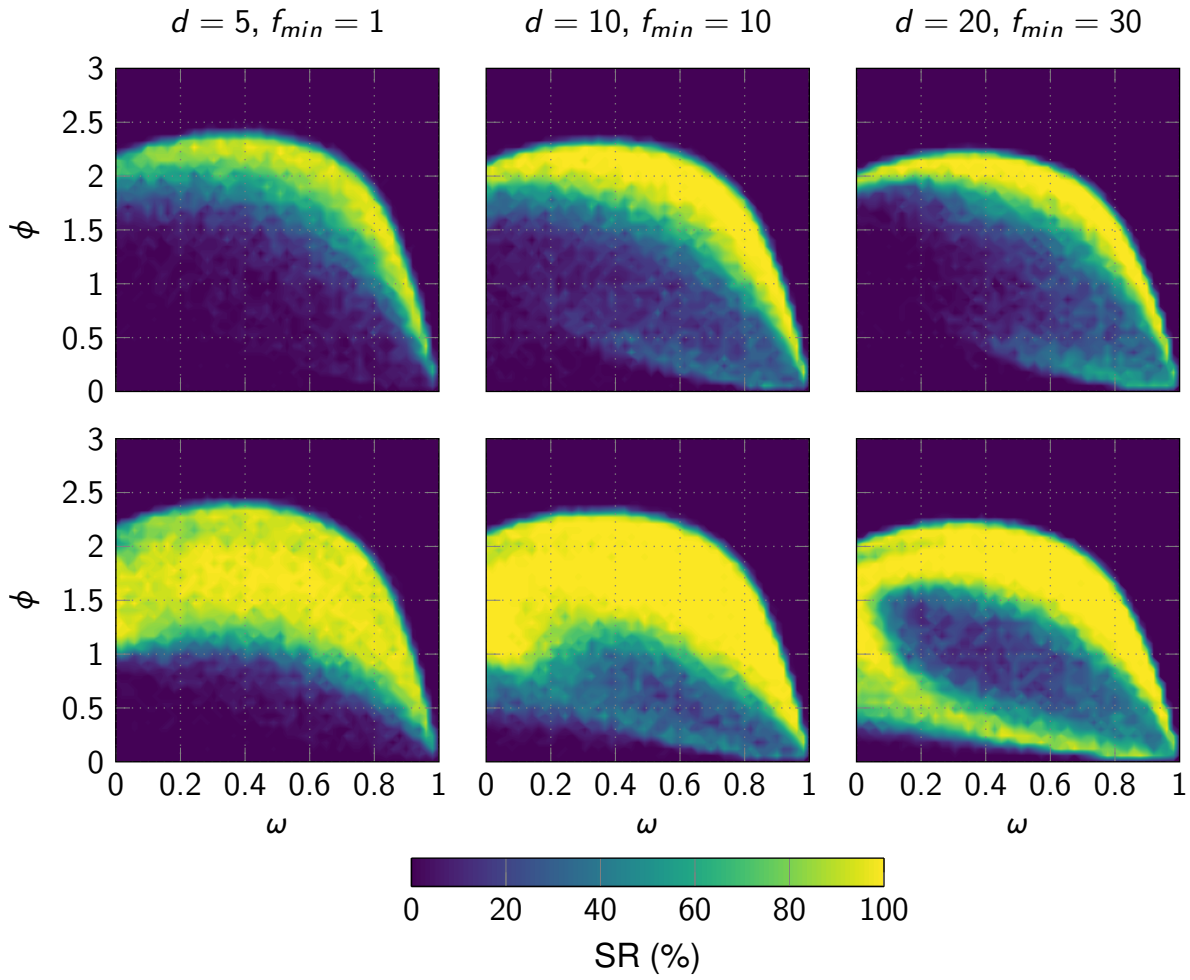

Figure 3.5: Results of the sensitivity analysis to parameters ω and ϕ for PSO (top) and CPSO (bottom) on the Rastrigin function in 5, 10 and 20 dimensions. The swarm size is set to 5 times the dimension and the goal to achieve is indicated by f_{min}. CPSO is more flexible in the choice of these parameters as the high SR region is wider than for PSO.

3.3.2 Benchmark

We compare the performances of PSO and CPSO on six classical benchmark test functions (Appendix 3.7.3) in $d = 30$ dimensions. Most of the test functions are either unimodal, multi-modal or dynamic. Therefore, these benchmark functions are a rather good proxy for assessing the reliability of an optimization method. For both PSO and CPSO, we use $n = 30$ particles and a maximum of $k_{\max} = 2000$ iterations. The global minimum misfit is 0 for all the functions tested. The minimum, median and maximum misfit values over 100 independent trials are presented in Table 3.1. We recall that the results have been obtained using synchronized particles.

Table 3.1: Results of the benchmark of PSO and CPSO with $n = 30$ particles in $d = 30$ dimensions on six benchmark test functions. The global minimum misfit is 0 for all the functions.

Function	PSO			CPSO		
	Min.	Median	Max.	Min.	Median	Max.
Ackley	1.377E-13	2.814	9.539	9.766E-13	2.092E-10	1.502
Griewank	0	2.951E-02	0.244	0	1.232E-02	8.096E-02
Quartic (noise)	4.538E-03	1.639E-02	7.131E-02	3.081E-03	9.060E-03	1.888E-02
Rastrigin	32.83	68.16	135.31	12.93	28.85	50.74
Rosenbrock	1.472E-02	18.36	81.30	6.537E-03	18.77	83.06
Styblinski-Tang	56.54	141.4	226.2	1.074E-03	56.54	113.1

For Quartic (noise) and Rosenbrock (unimodal functions), PSO and CPSO yield similar performances. For the four other functions tested, CPSO has outperformed PSO since the minimum, median and maximum misfit values are lower. While PSO fails to recover the global minimum for the highly multi-modal functions Rastrigin and Styblinski-Tang, CPSO shows rather good performances with median misfit much lower compared to PSO. These benchmarks indicate that CPSO is a more robust optimizer than PSO given the same parameters.

3.3.3 Importance sampling

Inversed models obtained through optimization are usually subject to uncertainties introduced by errors in data measurements, in the forward modeling and by the use of a finite number of parameters to describe the model. Therefore, the solution of an optimization problem is not unique as several models can explain the observed data. Uncertainties can be quantified by accounting for all sources of errors in a Bayesian framework and by generating acceptable models that fit the data equally well in terms of misfit function. The posterior Probability Density Function (PDF) is given by the Bayes theorem following

$$P\left(\mathbf{m} \,\middle|\, \mathbf{d}^{obs}\right) \propto P\left(\mathbf{m}\right) P\left(\mathbf{d}^{obs} \,\middle|\, \mathbf{m}\right) \tag{3.8}$$

where the first term $P\left(\mathbf{m}\right)$ is the prior PDF and contains all the information we know about the model $\mathbf{m}$ prior to the measurement of data $\mathbf{d}^{obs}$. The likelihood $P\left(\mathbf{d}^{obs} \,\middle|\, \mathbf{m}\right)$ links the model to the observed data considering that a model that does not explain exactly the data is acceptable to a certain extent, and is written

$$P\left(\mathbf{d}^{obs} \,\middle|\, \mathbf{m}\right) \propto \exp\left(-E\left(\mathbf{m}\right)\right) \tag{3.9}$$

Assuming theoretical normally distributed errors, E is the misfit function and is expressed as

$$E\left(\mathbf{m}\right) = \frac{1}{2}\left(\mathbf{d}^{obs} - g\left(\mathbf{m}\right)\right)^{\top} \mathbf{\Sigma}^{-1}\left(\mathbf{d}^{obs} - g\left(\mathbf{m}\right)\right) \tag{3.10}$$

where $\mathbf{\Sigma}$ is the covariance matrix (diagonal for independent observations) that accounts for data measurement and modeling errors Tarantola (2005).

Ideally, the model parameter space needs to be explored as a whole by estimating $P\left(\mathbf{m} \mid \mathbf{d}^{obs}\right)$ for each model $\mathbf{m}$. However, it is not feasible when the number of parameters to estimate is large. Therefore, importance sampling is usually performed through Markov Chain Monte Carlo (MCMC) that generates models distributed according to the PDF when the number of iterations tends toward infinity. We show how to use CPSO to obtain a rather good approximation of $P\left(\mathbf{m} \mid \mathbf{d}^{obs}\right)$.

A single run of PSO does not provide a proper sampling of the model parameter space since it is designed to rapidly locate and exploit a single minimum. Sen and Stoffa (1996) suggests to perform several independent runs of a global optimizer with different starting solutions to sample different parts of the model parameter space. Figure 3.6 (top) shows 100000 models sampled after 50 runs of 200 iterations of PSO and CPSO on the 2D Rastrigin function. The whole model parameter space is correctly sampled for both PSO and CPSO with particles mainly focused on the central part of the function corresponding to low misfit values (i.e. high probability).

We estimate the distributions for the 100000 previously sampled models by multiple runs of PSO and CPSO with gaussian kernels as shown in Figure 3.6 (bottom). Even though PSO detects the principal modes, it fails at identifying the central mode as the principal one. For each run, PSO seems to converge prematurely in different local minima without being able to escape from it. On the other hand, CPSO successfully samples the 9 principal modes and the central mode is correctly identified. Therefore, the frequency distribution from multiple runs of CPSO directly provides a rather good approximation of the PDF with no further computation.

In this section, we have demonstrated the robustness of CPSO as a global optimizer compared to classical implementation of PSO, but also as a more reliable method for uncertainty quantification. We now apply CPSO on a real first arrival traveltime tomography.

3.4 Numerical example

In this section, we apply the CPSO tomography algorithm on a real data set recorded in the context of induced seismicity.

3.4.1 Acquisition

Tomography is a geophysical inverse problem that consists in retrieving the background velocity model from observed first arrival traveltimes of seismic waves recorded at a set of receivers. The geometry used for the acquisition is represented in Figure 3.7 (left) in a relative Cartesian coordinate system. The monitoring network consists of two wireline arrays of forty 3C-receivers that were deployed in two different vertical wells (white triangles) ranging between 50 and 750 meters depth and regularly spaced every 15 meters. Fifteen perforation shots have been performed along two horizontal wells (green circles) at a depth of 1050 meters. For each of the fifteen perforation shots, almost all 80 first P-wave and approximately 40 S-wave arrivals could be picked. The average P- and S- wave picking uncertainties are estimated to be around 1 ms and 4 ms, respectively.

Acoustic logs for both P- and S- waves have been acquired in one well using a sonic wireline tool. These acoustic logs have been used to derive a layered velocity model consisting of 15 layers (Figure 3.7 (right)) that will serve as a reference model.

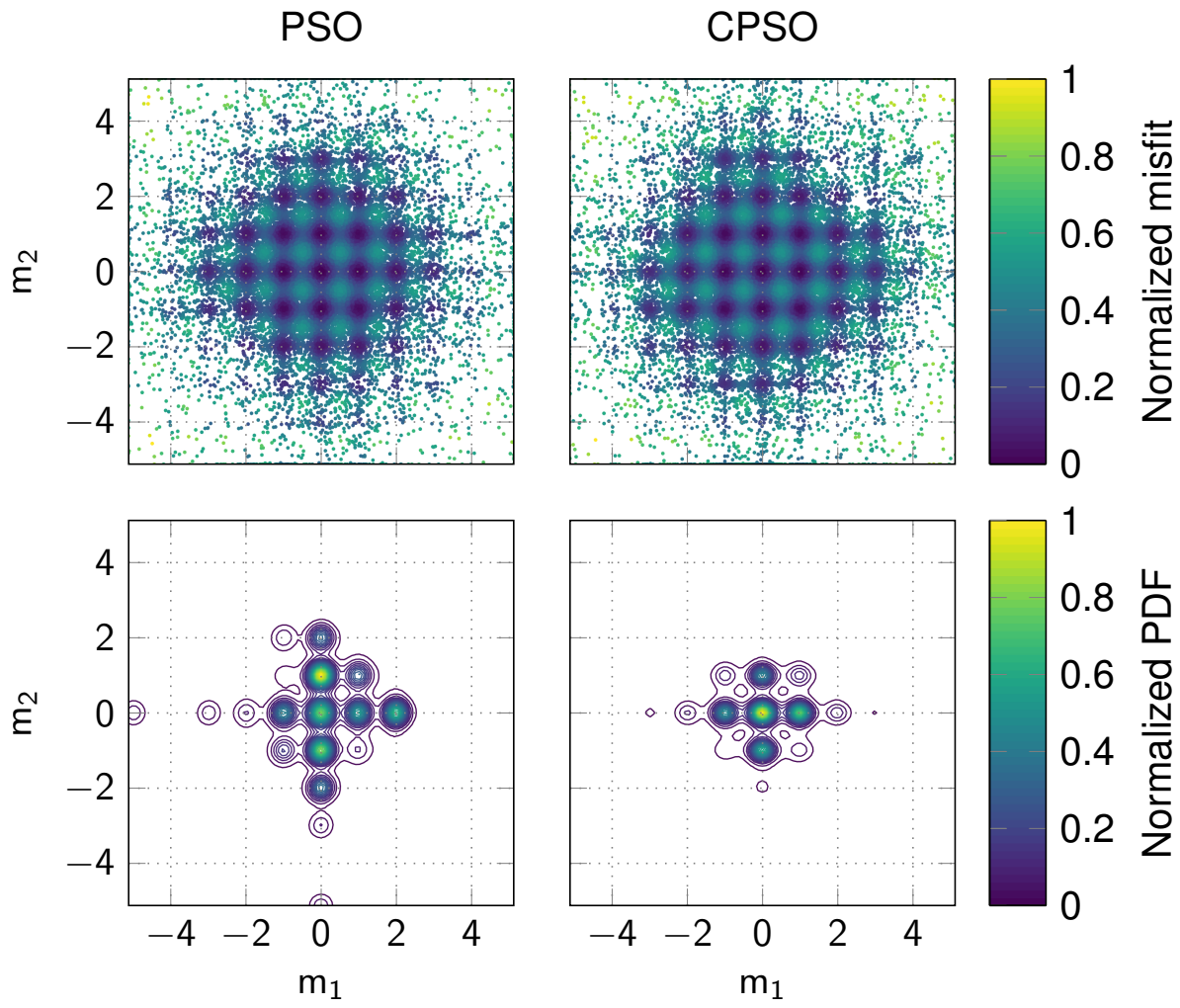

Figure 3.6: (Top) 100000 models sampled after 50 runs of PSO and CPSO on the 2D Rastrigin function with 5 particles and 200 iterations. The low misfit part of the function is correctly explored by the particles. Similar results can be obtained with more particles. (Bottom) Frequency distributions of 100000 models sampled by multiple runs of PSO and CPSO on the 2D Rastrigin function. PSO fails at identifying the central mode as the principal one while CPSO correctly identified the 9 central modes.

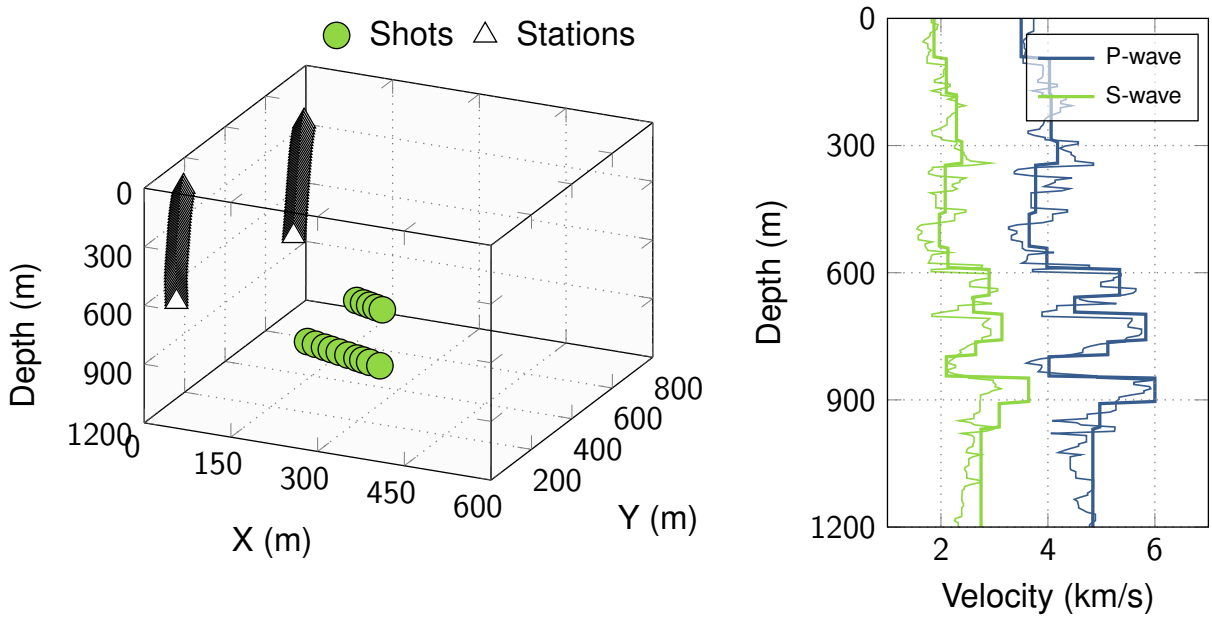

Figure 3.7: (Left) 3D acquisition geometry with perforation shot locations (green circles) and receiver locations (white triangles). (Right) Acoustic logs and 1-D reference calibrated velocity models for P-wave (blue) and S-wave (green).

The acquisition geometry is poorly constrained and requires the solution to be represented in terms of PDF through a Bayesian formulation. We also expect to obtain greater uncertainties for the S-wave velocity model since we have fewer arrival times associated to S-waves.

3.4.2 Inversion results

The velocity model is parametrized with 15 layers based on the reference model, and we invert for the P-wave velocity V_p, the ratio V_p/V_s and the interface depth of each layer. Therefore, the model consists of 45 unknown parameters. The lower and upper boundaries of each layer parameter are summarized in Table 3.2.

Traveltimes are calculated using an Eikonal solver that generates traveltime grids for each perforation shot (Noble, Gesret and Belayouni (2014)). We run the CPSO algorithm 50 times to sample the model parameter space sufficiently for uncertainty quantification with different swarm sizes (16, 32, 64 and 128 particles). A run is stopped when 200 iterations are performed. The four tomographies lasted respectively 1.8 minutes, 3.6 minutes, 7.2 minutes and 14.3 minutes using a total of 96 cores out of 104 (four sockets platform made of 4 Intel® Xeon® Platinum 8164 CPU, 26 cores @ 2.00 GHz each). To allow comparison with MCMC, we also run a Monte Carlo tomography that consists of 1 million models sampled using the Metropolis-Hastings algorithm which lasted 1.3 hours using 15 cores for the parallelization of the forward problem.

Figure 3.8 represents the evolution of the misfit with respect to the iteration number for Monte Carlo tomography (left) and the four CPSO tomographies (right). It shows that despite the 45 parameters to invert, a swarm size of $n = 16$ was actually enough to recover a good velocity model. We can also notice that increasing the swarm size allows the swarm to converge faster which was expected. Besides, Monte Carlo tomography required more than 10000 iterations to reach its stationary regime, while CPSO tomographies needed less than 100 iterations (i.e. less than 1600 concurrent misfit function evaluations in the best case) to converge. In terms of optimization performance, this means a gain of 2 orders of magnitude compared to MCMC.

Table 3.2: Lower and upper boundaries of each layer parameter. Particles are uniformly initialized in the search space.

Layer #	Vp (m/s)		Vp/Vs		Depth (m)	
	Min.	Max.	Min.	Max.	Min.	Max.
1	2500	5000	1.5	2.2	90	110
2	2500	5000	1.5	2.2	172	192
3	2500	5000	1.5	2.2	273	305
4	3000	5500	1.5	2.2	340	360
5	2500	5000	1.5	2.2	450	480
6	2500	5000	1.5	2.2	530	550
7	2500	5000	1.5	2.2	580	600
8	3500	6500	1.5	2.2	650	670
9	3000	6000	1.5	2.2	690	710
10	4000	7000	1.5	2.2	750	775
11	3500	6500	1.5	2.2	790	810
12	3000	6000	1.5	2.2	840	860
13	3500	7000	1.5	2.2	900	920
14	3500	6500	1.5	2.2	960	980
15	3500	6500	1.5	2.2	1200	1200

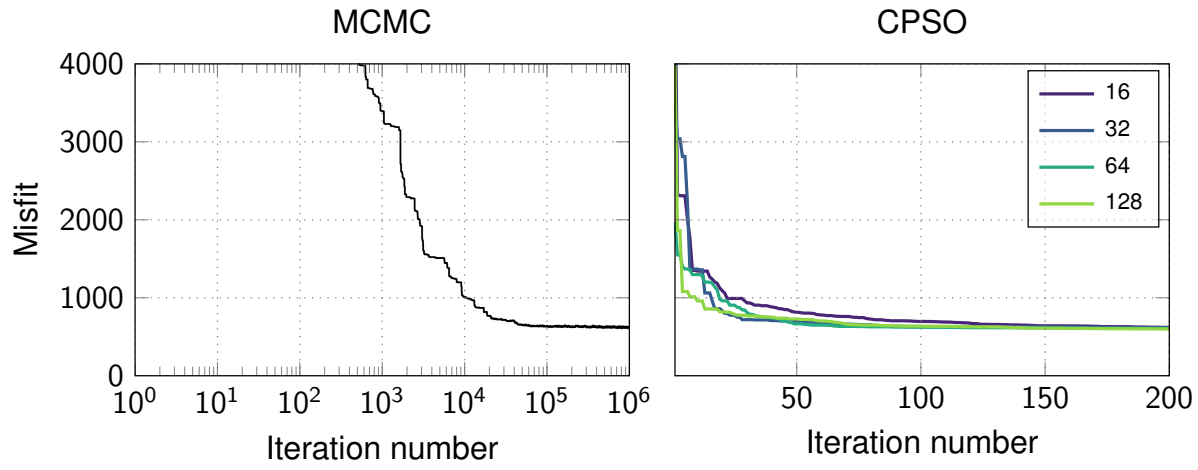

Figure 3.8: (Left) Energy (or misfit) of the Markov Chain as a function of the iteration number. The algorithm required more than 10000 iterations to reach equilibrium. (Right) Global best misfits for the best models with respect to the iteration number for different swarm sizes. In the four cases, the misfit function values are equivalent at the last iteration.

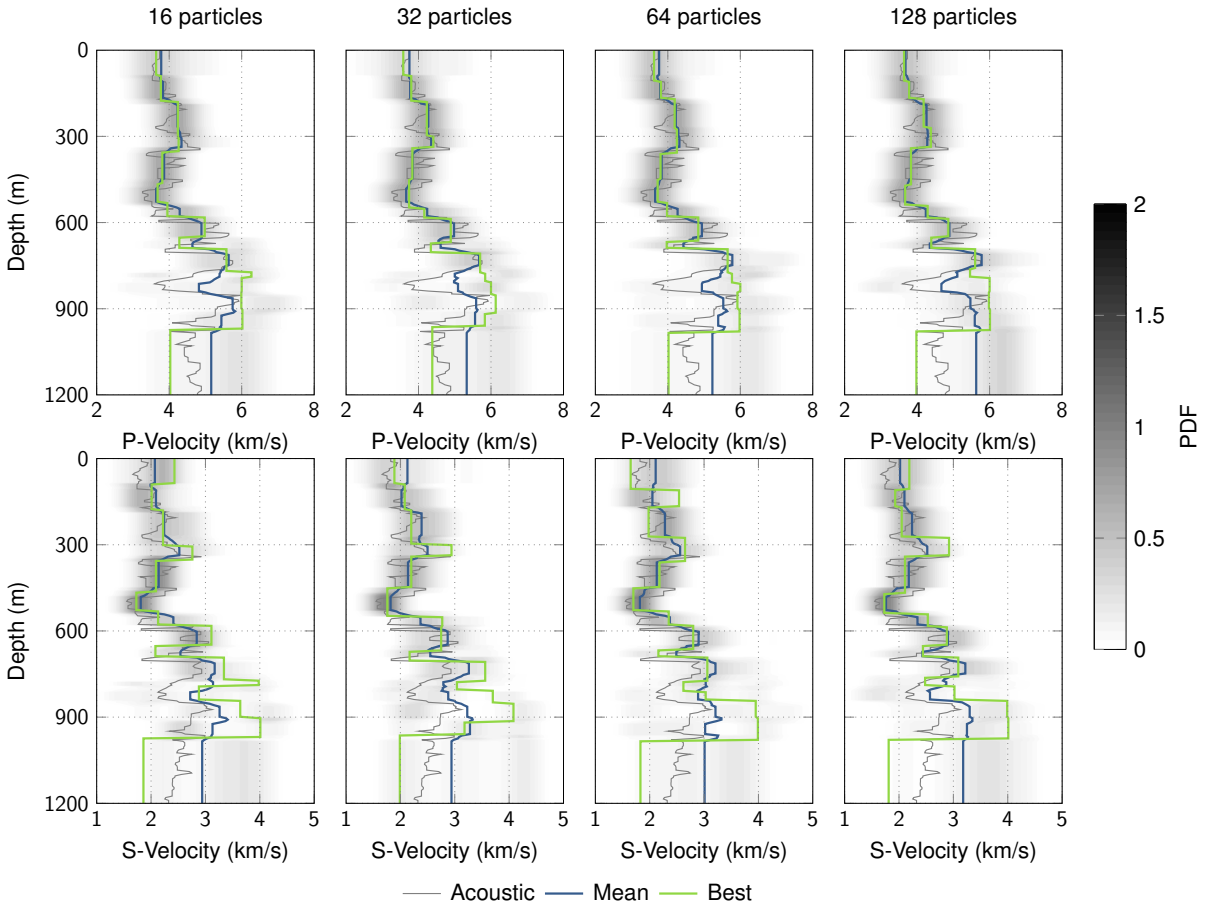

Figure 3.9: P- and S- wave velocity models obtained with 16 particles, 32 particles, 64 particles and 128 particles. The acoustic logs are represented in black, the best velocity models in green, the mean velocity models in blue, and the density plots in gray scale, darker colors indicating higher probabilities. Results are remarkably similar in the four cases.

The best and mean models are respectively represented in green and blue in Figure 3.9. The mean model is calculated as the weighted sum of all the models sampled and is written as

$$\hat{\mathbf{v}} = \frac{\sum_i P\left(\mathbf{m_i} \mid \mathbf{d}^{obs}\right) \mathbf{v_i}}{\sum_i P\left(\mathbf{m_i} \mid \mathbf{d}^{obs}\right)} \tag{3.11}$$

where $\mathbf{v_i}$ is the continuous velocity model transformed from the parameters $\mathbf{m_i}$. The velocity models obtained from the different tomographies are remarkably similar and seem consistent with the acoustic logs for both P- and S- waves.

The velocity uncertainties are represented by the density plots, darker colors indicating higher probabilities. We display in Figure 3.10 the marginal probabilities at three different depths (300, 600 and 900 meters). The marginal probabilities are narrow at 300 and 600 meters depth which corresponds to the constrained region where the receivers are deployed, and are almost uniform at 900 meters depth (below the receivers). Also, the marginal probabilities obtained for the S-wave velocity models are thicker than those of P-wave velocity models as we have fewer arrival time picks associated to S-waves. These marginal probabilities are consistent with the acquisition geometry and the ray coverage between the sources and the receivers and are in agreement with the ones obtained with a MCMC sampler. Besides, the velocity uncertainties obtained for the four tomographies do not differ much from each other, which means that uncertainty quantification in first arrival traveltime tomography using CPSO is relatively insensitive to the swarm size.

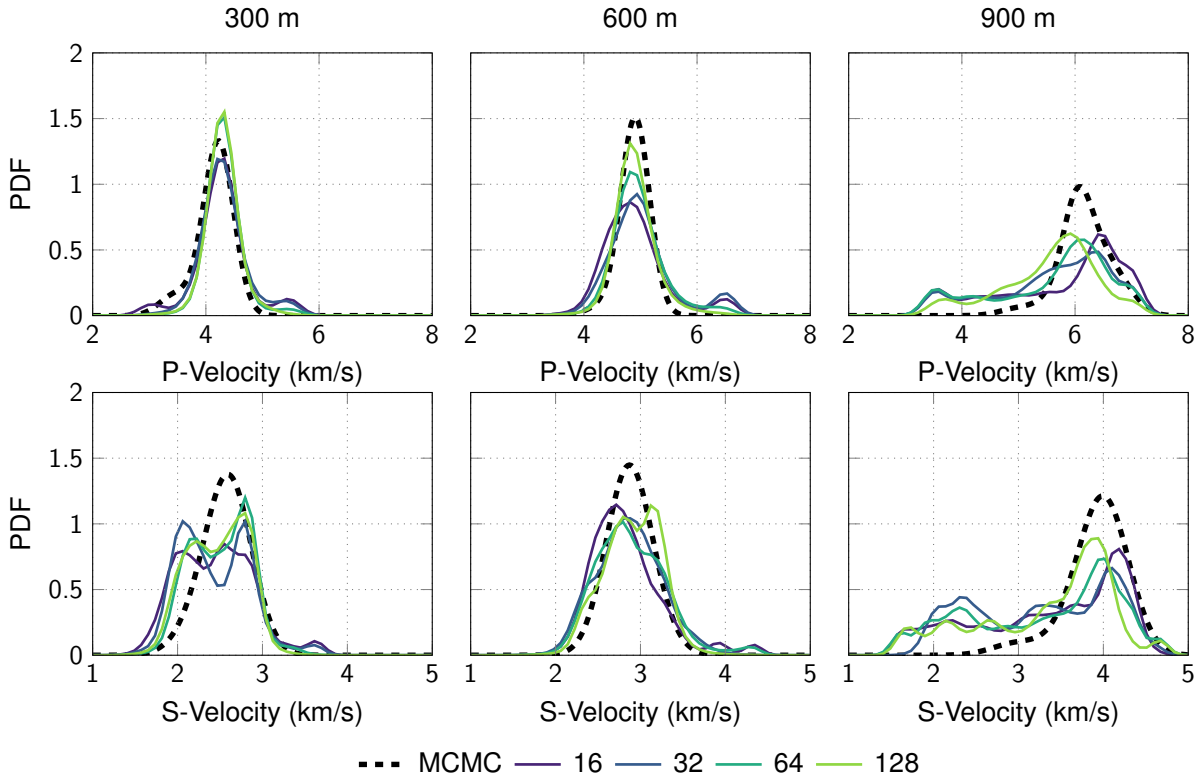

Figure 3.10: Marginal probabilities at 300, 600, and 900 meters depth obtained with different swarm sizes (16, 32, 64, 128 particles) and MCMC. Probabilities are narrow at 300 and 600 meters depth where the receivers are deployed, and wide at 900 meters below the receivers. Marginal probabilities obtained with CPSO are in agreement with the ones obtained with MCMC.

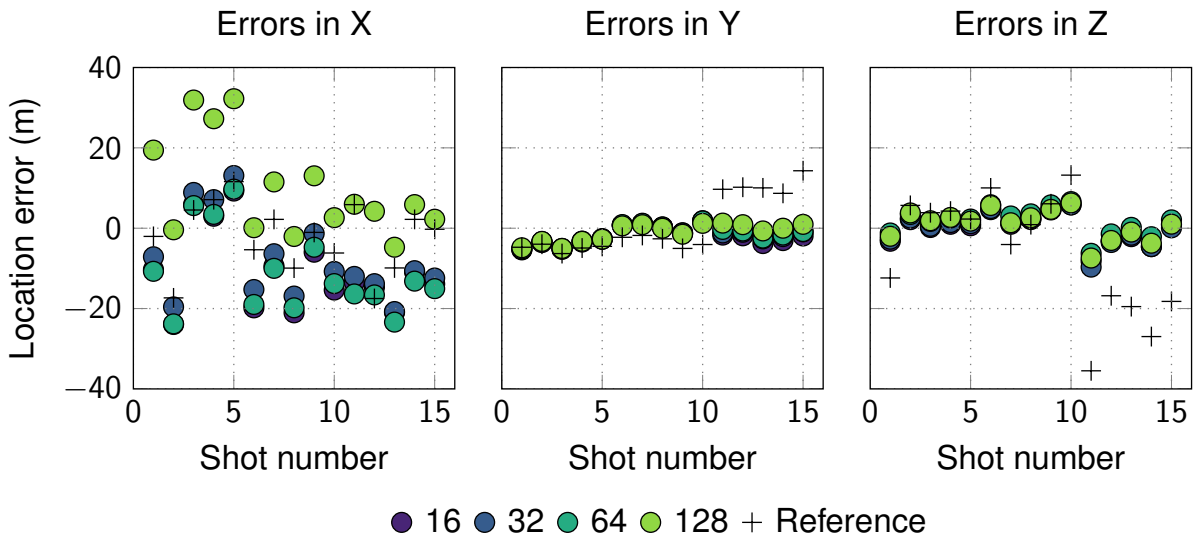

Figure 3.11: Location errors in X, Y and Z directions for the four best velocity models obtained with different swarm sizes (16, 32, 64, 128 particles). The black crosses represent the location errors in the reference model. The shots are accurately relocated in the Y and Z directions but the mean absolute location error in the X direction is about 10 meters.

Finally, we relocate the fifteen perforation shots using an hybrid approach in the four best velocity models obtained with different swarm sizes, and in the calibrated reference velocity model as well. First, we perform a grid search over the hypocenter parameter space. Then, we refine the search around the solution found by the grid search using CPSO. Figure 3.11 represents the location errors for each perforation shot in the X, Y and Z directions. In the reference model, we observe a mean absolute location error of 10, 7 and 12 meters in the X, Y and Z directions, respectively. In our four velocity models, the fifteen shots are accurately relocated in the Y and Z directions with a mean absolute error of 2 and 3 meters. Yet, the mean absolute location error in the X direction is about 10 meters. This may be due to the poor ray coverage in this direction and by the fact that we did not take into account a potential anisotropy in the medium.

All in all, the locations obtained from our four velocity models are more accurate, especially in depth, which indicates that velocity models obtained from CPSO tomography can be reliably used in microseismic monitoring.

3.5 Hybrid parallel implementation

In the last decades, High Performance Computing (HPC) has gain growing interest in geophysics as it provides an integrated solution to solve large scientific and engineering problems. Today's supercomputers are systems made of Symmetric Multi-Processor machines (SMP) and therefore combine features of shared and distributed memory architectures (Rabenseifner, Hager and Jost (2009), Drosinos and Koziris (2004)). This allows multi-level hybrid parallel programming where message passing model (MPI) is employed between processes (or ranks), and shared memory model (OpenMP) is used for each process.

Non-linear traveltime tomography is a heavy computational problem that requires a lot of forward modeling (i.e. computation of traveltimes). In our microseismic example, a traveltime grid is computed for each perforation shot. Each grid can be computed independently which allows a first level of parallelism. Besides, unlike MCMC methods usually applied in geophysical problems,

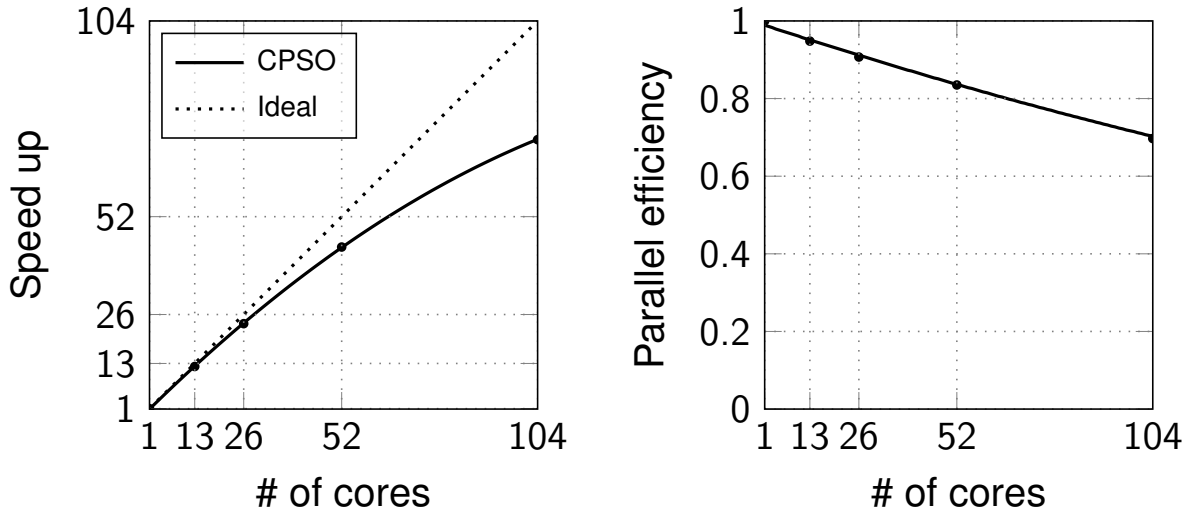

Figure 3.12: Parallel performance of CPSO on a real tomography problem. (Left) Speed up. (Right) Parallel efficiency.

EA such as PSO have the advantage to evaluate concurrently a population of models, which add another level of parallelism. In our hybrid parallel implementation, we evenly distribute the particles to the MPI processes. Therefore, our implementation requires the swarm size to be a multiple of the number of available cores for maximum efficiency to avoid idle cores. For each process, given a velocity model defined by the particle, the computation of the traveltime grids for each perforation shot is scattered over the threads with OpenMP.

We evaluate the parallel performance of our implementation on our real tomography example by calculating the speed up and parallel efficiency for different number of cores. The speed up is defined as the ratio of sequential computation time to parallel computation time. Parallel efficiency is the ratio of speed up to the number of cores. Ideally, speed up and parallel efficiency should equal the number of cores and 1, respectively. We solve our tomography problem with 104 particles and 100 iterations on the 104 cores of a four sockets SMP machine (each processor having 26 cores), and perform 5 runs using 1, 13, 26, 52 and 104 cores with only 1 OpenMP thread, which corresponds to a strong scaling analysis. Results of parallel performance are reported in Figure 3.12.

Overall, increasing the number of cores reduces the computation time. However, speed up is not ideal (72 at 104 cores) and parallel efficiency decreases almost linearly with increasing number of cores. Therefore, the algorithm becomes less efficient by adding more and more cores. This is due to communication overhead and/or overhead implied by parallel decomposition of the algorithm. An asynchronous parallel implementation could improve PSO scalability and allow close to ideal parallel performance (Koh *et al.* (2006), Venter and Sobieszczanski-Sobieski (2006), Mussi, Nashed and Cagnoni (2011)).

3.6 Discussion and conclusion

With the rise in computational power that we have witnessed in the recent years – in particular with the emergence of multi-core machines – it is important to implement algorithms that are able to handle efficiently all the available CPU resources. EA such as PSO are global optimization methods that evaluate simultaneously a set of independent solutions, implying a straightforward parallelization of these algorithms.

In this paper, we have introduced a new algorithm based on PSO for non-linear first arrival traveltime tomography that we called CPSO. We have demonstrated its robustness in finding a good fitting solution on several benchmark test functions, and even in appraising reliably uncertainties when run multiple times. CPSO has been successfully applied to a real 3D microseismic example in the context of induced seismicity with reliable uncertainties indicating which parts of the velocity models are well constrained. The velocity models sampled by multiple runs of CPSO can be used to propagate velocity uncertainties to microseismic event locations. Indeed, Gesret *et al.* (2015) showed that more accurate locations with more reliable uncertainties can be obtained by taking into account both traveltime picking and velocity errors.

We have shown that CPSO is less sensitive to its main tuning parameters in addition to the swarm size. Unfortunately, there is no rule-of-thumb for this parameter as the swarm size is highly dependent on the problem (i.e. the dimension, the landscape). In our tomography example, we have shown that 16 particles are enough in spite of the number of unknowns to invert (45 parameters). Increasing the swarm size improves the speed of convergence as it enhances the exploration capability of the swarm, yet it does not improve much the final model.

In our implementation, we have parallelized the computation of the forward problem at two levels: the individuals are evaluated simultaneously among different MPI processes, and the traveltime grids of each perforation shot are calculated concurrently by different OpenMP threads. Another approach of parallelization are the *island models* where the EA are parallelized themselves (Gong and Fukunaga (2011), Gong *et al.* (2015)). In other words, the initial population is divided into subpopulations that evolve independently on a subset of cores (the islands) with a periodical exchange of individuals (migration). It has been shown that island model EA yield better performance as each population follows a different evolution path (Whitley, Rana and Heckendorn (1999)). The migration process is analogous to the exchange of states in Parallel Tempering and Interactive MCMC. One can also consider that each subpopulation evolves following different stochastic processes, which has led to hybrid island model based evolutionary algorithms. The advantage of these algorithms is that by combining different EA, the shortcomings of one can be compensated by another.

In our tomography algorithm, we assumed that the picking errors were known. In practice, observation picking uncertainties are usually poorly determined. Malinverno and Briggs (2004) addressed this issue with a Hierarchical Bayesian formulation that assigns the measurement errors as unknowns of the inverse problem, which allows the algorithm to infer the level of noise that fit the data the best. Our algorithm can be extended to such formulation with very little modification.

CPSO can also be applied to other geophysical inversion problems that require a global optimization method (e.g. earthquake location, estimation of focal mechanism, inversion of dispersion curves). The competitivity concept introduced in this paper can, as well, be adapted to other EA that suffer from premature convergence such as Differential Evolution.

3.7 Appendice

3.7.1 PSO algorithm

Detailed PSO algorithm is summarized in Algorithm 3.3 where $\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_n] \in \mathbb{R}^{n \times d}$ and $\mathbf{M} = [\mathbf{m}_1, ..., \mathbf{m}_n] \in \mathbb{R}^{n \times d}$ respectively denote particle velocities and model matrices, $\mathbf{p}_{fit} \in \mathbb{R}^n$ is the vector of particle personal best misfits, and $\mathbf{P}_{best} = [\mathbf{m}_{p,1}, ..., \mathbf{m}_{p,n}] \in \mathbb{R}^{n \times d}$ is a vector that contains the personal best models of every particles. The algorithm is stopped when:

1. The global best model changes less than a specified threshold ε_1 and its misfit function value is lower than a threshold ε_2;

2. The maximum number of iterations is reached.

Table 3.3: PSO algorithm.

	Pseudocode	Comment
1:	$\mathbf{V} \leftarrow \mathbf{0}$	▷ Initialize velocities
2:	$\mathbf{M} \sim \mathcal{U}(\mathbf{m}_{\min}, \mathbf{m}_{\max})$	▷ Initialize models
3:	$\mathbf{p}_{fit} \leftarrow E(\mathbf{M})$	▷ Initialize personal best misfits
4:	$\mathbf{P}_{best} \leftarrow \mathbf{M}$	▷ Initialize personal best models
5:	$\mathbf{m}_g \leftarrow \text{argmin}(E(\mathbf{P}_{best}))$	▷ Initialize global best model
6:	**repeat**	
7:	**for** $i = 1$ to n **do**	
8:	$\mathbf{v}_i \leftarrow \omega\mathbf{v}_i + \phi_p\mathbf{r}_p(\mathbf{m}_{p,i} - \mathbf{m}_i) + \phi_g\mathbf{r}_g(\mathbf{m}_g - \mathbf{m}_i)$	▷ Update velocity
9:	$\mathbf{m}_i \leftarrow \mathbf{m}_i + \mathbf{v}_i$	▷ Update model
10:	**if** $E(\mathbf{m}_i) < \mathbf{p}_{fit,i}$ **then**	
11:	$\mathbf{p}_{fit,i} \leftarrow E(\mathbf{m}_i)$	▷ Evaluate misfit function
12:	$\mathbf{P}_{best,i} \leftarrow \mathbf{m}_i$	▷ Update personal best model
13:	**end if**	
14:	**end for**	
15:	$\mathbf{m}_g \leftarrow \text{argmin}(E(\mathbf{P}_{best}))$	▷ Update global best model
16:	**until** stopping criterion is met	

3.7.2 CPSO algorithm

Algorithm 3.4 describes CPSO competition triggering with ε the swarm maximum radius threshold as defined in Equation (3.6). The maximum swarm radius δ^k at iteration k is evaluated right after the update of the swarm global best model (line 15 of Algorithm 3.3). The "worst" particles can be determined using a quickselect method that finds only the n_w^k individuals yielding the highest misfit values. Note that reset particle personal best misfits are set to $+\infty$ instead of their true misfits to avoid extra misfit computations that would deteriorate parallel performance.

Table 3.4: CPSO competition triggering algorithm.

	Pseudocode	Comment
1:	**if** $\delta^k < \varepsilon$ **then**	
2:	**for** $i = 1$ to n **do**	
3:	**if** $\mathbf{m}_i$ is worse **then**	
4:	$\mathbf{v}_i \leftarrow \mathbf{0}$	▷ Reset velocity
5:	$\mathbf{m}_i \sim \mathcal{U}(\mathbf{m}_{\min}, \mathbf{m}_{\max})$	▷ Reset model
6:	$\mathbf{p}_{fit,i} \leftarrow +\infty$	▷ Reset personal best misfit
7:	$\mathbf{P}_{best,i} \leftarrow \mathbf{m}_i$	▷ Reset personal best model
8:	**end if**	
9:	**end for**	
10:	**end if**	

3.7.3 Benchmark test functions

Several test functions used to benchmark global optimization methods are summarized in Table 3.5, together with their application ranges, minimum positions and values.

Table 3.5: Benchmark test functions.

Function		Range	Solution	Minimum
Ackley	$20 + e - 20e^{-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}} - e^{\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)}$	[-32.768,32.768]	$(0)^d$	0
Griewank	$1 + \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right)$	[-600,600]	$(0)^d$	0
Quartic (noise)	$\sum_{i=1}^{d} i x_i^2 + \text{rand}(0,1)$	[-1.28,1.28]	$(0)^d$	0
Rastrigin	$10d + \sum_{i=1}^{d} \left(x_i^2 - 10\cos(2\pi x_i)\right)$	[-5.12,5.12]	$(0)^d$	0
Rosenbrock	$\sum_{i=1}^{d-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2\right)$	[-5.12,5.12]	$(1)^d$	0
Styblinski-Tang	$\frac{1}{2}\sum_{i=1}^{d} \left(x_i^4 - 16x_i^2 + 5x_i\right) + 39.16599d$	[-5,5]	$(-2.903534)^d$	0

3.7.4 Sensitivity analysis

We conducted the same sensitivity analysis as in Section 3.3.1 on the Rosenbrock function in 5, 10 and 20 dimensions. The Rosenbrock function is a classical benchmark function that presents a very flat valley similar to what can be observed in traveltime tomography. The results are shown in Figure 3.13. For both PSO and CPSO, the high SR region is comparable to the one obtained on the Rastrigin function, which means that a couple (ω, ϕ) lying in this region can be used for different type of functions. Yet, since the high SR region is wider, CPSO is less sensitive to the choice of the couple (ω, ϕ).

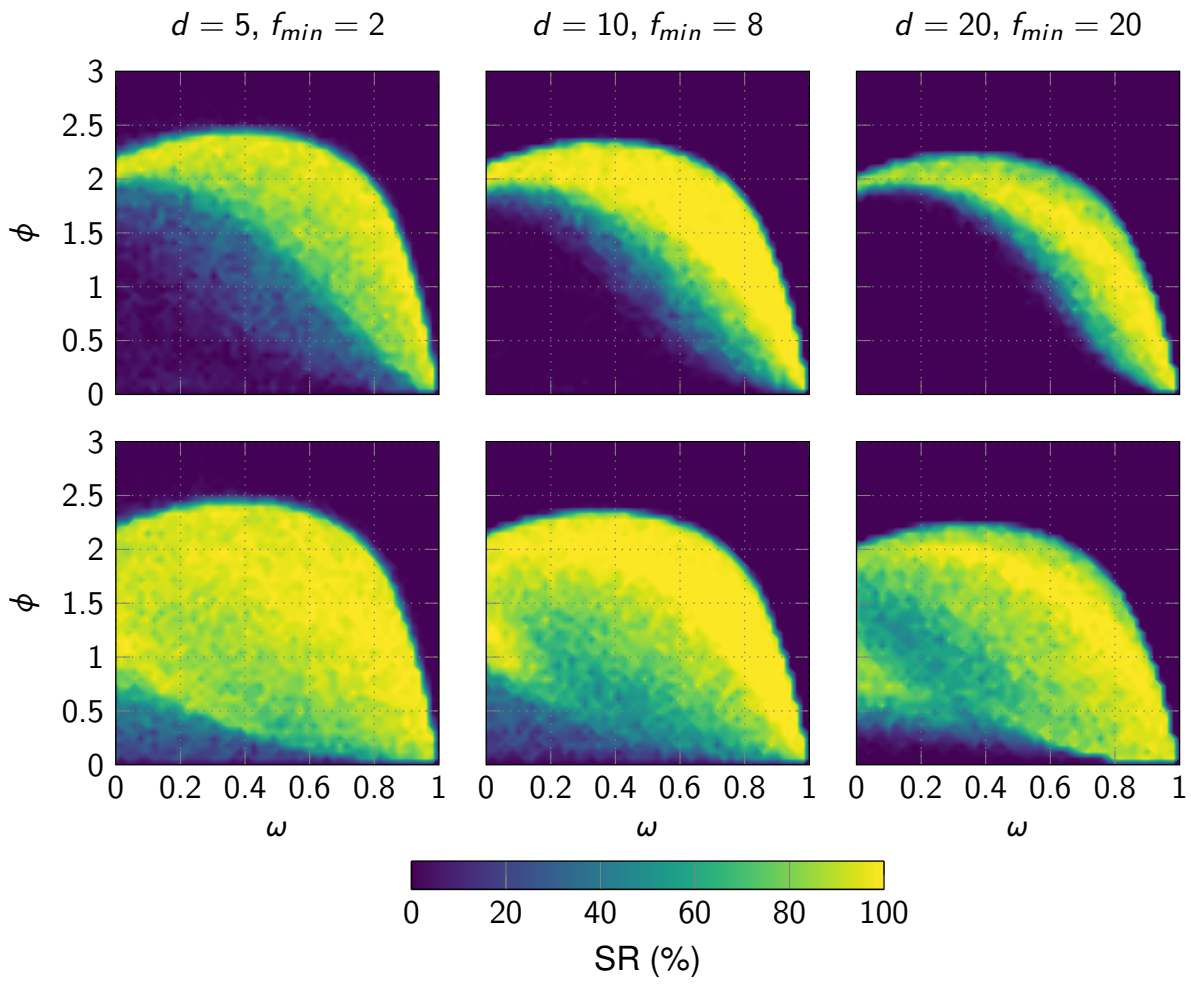

Figure 3.13: Results of the sensitivity analysis to parameters ω and ϕ for PSO (top) and CPSO (bottom) on the Rosenbrock function in 5, 10 and 20 dimensions. The swarm size is set to 5 times the dimension and the goal to achieve is indicated by f_{min}.

Chapter 4

Refraction traveltime tomography

Contents

Dans le Chapitre 3, je me suis intéressé à l'application d'un algorithme évolutionniste à un problème de tomographie des temps de première arrivée en milieu tabulaire. Cette paramétrisation permet de représenter un modèle de vitesse avec un faible nombre de paramètres à inverser. Cependant, ce type de paramétrisation n'est pas adapté pour décrire des milieux géologiques plus complexes, et des paramétrisations moins parcimonieuses sont nécessaires pour caractériser ces milieux. Les algorithmes évolutionnistes sont connus pour être sujets au *fléau de la dimension* (*curse of dimensionality* en anglais), c'est-à-dire qu'ils perdent en efficacité avec la dimension de l'espace de recherche qui augmente ($> 10^2$). Ce chapitre s'intéresse à la faisabilité des algorithmes évolutionnistes pour l'optimisation d'un problème mal-posé à

grand nombre de paramètres. J'applique les trois méthodes décrites dans le Chapitre 2 au problème non-linéaire et multi-modal de tomographie des ondes réfractées sur le modèle de vitesse Marmousi qui présente une structure géologique complexe. Les données de temps de trajet synthétiques sont générées en considérant une géométrie d'acquisition stationnaire en surface. Elle consiste en 200 sources et 400 récepteurs résultant en une faible illumination du modèle de vitesse en profondeur. Le modèle de vitesse est paramétrisé par des courbes B-splines cardinaux 2D. J'examine notamment l'influence des modèles de vitesse initiaux, la taille de la population et le nombre maximal d'itérations. En tant que méthodes d'optimisation stochastiques, les algorithmes évolutionnistes doivent être idéalement insensibles à la population initiale. Cependant, je montre que leur convergence sur un problème de tomographie des ondes réfractées peut être nettement améliorée en définissant une population initiale ayant un sens physique. Enfin, j'évalue les avantages et inconvénients de chacun des algorithmes testés à l'aide d'analyses statistiques des résultats de convergence et d'une analyse de scalabilité.

Ce chapitre est écrit sous la forme d'un article soumis :

- **Keurfon Luu**, Mark Noble, Alexandrine Gesret and Philippe Thierry, 2018. "Toward large scale stochastic refraction tomography: a comparison of three evolutionary algorithms." *Geophysical Prospecting*.

Abstract

The main goal of this study is to assess the potential of evolutionary algorithms to solve highly non-linear and multi-modal tomography problems (such as first arrival traveltime tomography) and their abilities to estimate reliable uncertainties. Classical tomography methods apply derivative-based optimization algorithms that require the user to determine the value of several parameters (such as regularization level and initial model) prior to the inversion as they strongly affect the final inverted model. In addition, derivative-based methods only perform a local search dependent on the chosen starting model. Global optimization methods based on Markov Chain Monte Carlo that thoroughly sample the model parameter space are theoretically insensitive to the initial model but turn out to be computationally expensive. Evolutionary algorithms are population-based global optimization methods and are thus intrinsically parallel, allowing these algorithms to fully handle available computer resources. We apply three evolutionary algorithms to solve a refraction traveltime tomography problem, namely the Differential Evolution, the Competitive Particle Swarm Optimization and the Covariance Matrix Adaptation - Evolution Strategy. We apply these methodologies on a smoothed version of the Marmousi velocity model and compare their performances in terms of optimization and estimates of uncertainty. By performing scalability and statistical analysis over the results obtained with several runs, we assess the benefits and shortcomings of each algorithm.

Keywords: evolutionary algorithms, global optimization, uncertainties, non-linear inversion, tomography

4.1 Introduction

Building a macro-velocity model is a key step in seismic imaging. Generally speaking, first arrival traveltime tomography based on direct, diffracted or refracted waves is applied to invert for a velocity model that explains the observed data. Such obtained velocity models are often used

as background models in migration or as initial models in Full Waveform Inversion (FWI) whose results depend on the quality of the velocity models. First arrival traveltime tomography is a typical non-linear and ill-posed problem encountered by the geophysical community. Acquisition aperture is often limited and the sources and receivers are located only at the surface which causes the resolution of velocity to decrease and error to increase with depth. To assess properly resolution and errors, a reliable estimation of uncertainties is mandatory.

In spite of its non-linearity, many refraction tomography methods apply a derivative-based optimization method to solve the linearized problem iteratively (White (1989), Zelt and Barton (1998), Zhang and Toksöz (1998)). The linearization makes the implicit assumption of a unique solution obtained by a regularization procedure, and only allows to obtain a local solution dependent on the initial solution (Menke (2012)). The level of damping required by the regularization procedure has to be determined prior to the inversion by the user and strongly influences the inversion result. From a computational point of view, derivative-based methods are very attractive since the gradient can be efficiently calculated using the adjoint-state method (e.g. Taillandier *et al.* (2009), Noble *et al.* (2010)) making them suitable for large scale optimization problems (i.e. 3D velocity models containing millions of parameters whatever the type of parametrization).

Besides, derivative-based approaches are under gaussian hypothesis and require additional computation to quantify uncertainties by assessing the quality of the inverted solution using bootstrapping or evaluation of the sensitivity kernel for example. On the one hand, derivative-free methods based on Markov Chain Monte Carlo (MCMC) that sample the velocity model parameter space provide reliable estimates of uncertainty. These methods can be applied to non-smooth and non-convex functions and produce results independent of the initial model. Although these algorithms are sequential, each Markov chain can be run in parallel either independently or interactively (Sambridge (2014), Bottero *et al.* (2016)). Other recent studies have proposed methods to parallelize MCMC based algorithms (Neiswanger, Wang and Xing (2013), Goudie *et al.* (2017)). However, some approaches such as interactive MCMC are not straightforward to implement. On the other hand, evolutionary algorithms (EA) are global optimization methods inspired by the natural evolution of species that are intrinsically and straightforwardly parallel as they operate on a population of models. Each model is represented by an independent individual within a population that can be evaluated in parallel.

Global optimization methods have been successfully applied to various inversion problems in geophysics such as earthquake location (Sambridge and Gallagher (1993), Billings (1994), Růžek and Kvasnička (2001)), surface wave dispersion curve inversion (Socco and Boiero (2008), Song *et al.* (2012), Wilken and Rabbel (2012)), history matching (Mohamed *et al.* (2010), Mohamed *et al.* (2012)), or traveltime tomography (Bodin and Sambridge (2009), Tronicke, Paasche and Böniger (2012), Bottero *et al.* (2016), Belhadj *et al.* (2018)). However, few studies have applied this kind of methods to solve a typical refraction tomography problem due to the higher dimensionality and multi-modality of the problem. Indeed, the *curse of dimensionality* that inherently affects all global optimization algorithms can exponentially increase the number of local minima with the number of model parameters which makes them not suitable for large scale optimization. Boschetti, Dentith and List (1996) applied a genetic algorithm on a synthetic refraction tomography problem and tackled the *curse of dimensionality* by adopting a multi-scale strategy (Bunks *et al.* (1995)) that iteratively increases the dimensionality of the search space. Improta *et al.* (2002) applied an hybrid scheme based on a Monte Carlo approach and the simplex optimization technique to derive an initial background velocity model from first arrival traveltime data. Rumpf and Tronicke (2015) used a particle swarm optimizer to solve the tomographic problem and quantify uncertainties using a 1D layer-based model parametrization. More recently, Ryberg and Haberland (2018) directly applied a bayesian MCMC formalism to invert refraction data and parametrized the velocity model using Voronoi tesselation and

triangulated meshes. To our knowledge, no study has been published on the application of EA to the seismic tomography problem for a large number of model parameters.

In this work, we propose to study the feasibility of a medium to large scale (300 parameters) stochastic refraction tomography by comparing three EA, namely the Differential Evolution (DE, Storn and Price (1997)), the Competitive Particle Swarm Optimization (CPSO, Luu *et al.* (2018)) and the Covariance Matrix Adaptation - Evolution Strategy (CMA-ES, Hansen, Müller and Koumoutsakos (2003)). These three EA are becoming more and more popular in the geophysical community because their implementations are straightforward, they require less tuning, they are by nature parallel algorithms and their convergence rates is much higher compared to classical global optimization methods (Angeline (1998), Mohamed, Christie and Demyanov (2010)). These algorithms were originally designed for optimization and are now used to quantify uncertainties (Fernández Martínez *et al.* (2012), Tronicke, Paasche and Böniger (2012), Rumpf and Tronicke (2015)). Luu *et al.* (2018) showed on a simple real data example that CPSO does sample properly the velocity model parameter space to provide reliable estimates of uncertainty, results were similar to those obtained by MCMC at a much lower computational cost. It is worth mentioning that EA like any global optimizers do not guarantee to find the global minimum. Nevertheless, they are global in the sense that they are less dependent on their initial conditions. The main objective of this work is to evaluate the performances and robustness of the three EA to solve a medium to large scale highly non-linear tomography problem. The ability of DE and CMA-ES to quantify uncertainties are compared to CPSO. First, we briefly describe the three algorithms before showing the practical implementations for refraction tomography and applying them to reconstruct a smoothed version of the Marmousi velocity model. We choose a 2D cardinal B-splines parametrization to obtain a smooth and realistic velocity model. Finally, we perform a scalability analysis to assess the parallel performance of each algorithm on our problem.

4.2 Theory and method

Many geophysical problems are underdetermined optimization/sampling problems that can be solved using either local or global algorithms (Tarantola and Valette (1982)). An optimization problem consists in minimizing the objective function E under the constraint $\mathbf{m}_{\text{min}} \leq \mathbf{m} \leq \mathbf{m}_{\text{max}}$ (feasible space). In other words, we want to find the optimal model $\mathbf{m}_{\text{opt}}$ so that

$$\mathbf{m}_{\text{opt}} = \text{argmin}\left(E\left(\mathbf{m}\right)\right), \quad \mathbf{m}_{\text{min}} \leq \mathbf{m} \leq \mathbf{m}_{\text{max}}. \tag{4.1}$$

In geophysical inverse problems, the objective function measures the difference between the experimental data and the synthetic data calculated by a theoretical (often numerical) model, and is hence usually referred to as the misfit function. Let us define the discrete data vector $\mathbf{d}^{obs}$. For a model $\mathbf{m}$, the misfit can be measured by the well-known root-mean-square (RMS) error written

$$E\left(\mathbf{m}\right) = \left[\frac{1}{N}\left(\mathbf{d}^{obs} - g\left(\mathbf{m}\right)\right)^{\top} \mathbf{C_D}^{-1}\left(\mathbf{d}^{obs} - g\left(\mathbf{m}\right)\right)\right]^{\frac{1}{2}} \tag{4.2}$$

with N the number of data points, $\mathbf{C_D}$ the data covariance matrix that accounts for the noise present in the observed data, and $g\left(\mathbf{m}\right)$ the data calculated by the forward operator g – often non-linear – on the model $\mathbf{m}$. The non-linearity can be addressed by global optimization methods that explore the model parameter space. In this paper, we focus on evolutionary algorithms where the misfit function value of each model within the population can be independently evaluated.

4.2.1 Evolutionary algorithms

Evolutionary algorithms (EA) are population-based stochastic optimization methods that present mechanisms inspired by the natural evolution of species such as mutation, recombination and selection. A candidate model to the optimization problem is represented by an individual with its misfit determining the quality of the model. EA are intrinsically parallel as each individual is independent and their misfit values can be evaluated concurrently. This property allows EA to benefit from all the computational resources available.

In this section, we describe the three evolutionary algorithms that will be used to solve the refraction tomography problem, namely the Differential Evolution, the Competitive Particle Swarm Optimization and the Covariance Matrix Adaptation - Evolution Strategy. As EA are nature-inspired optimization algorithms, they may present some discrepancies in the vocabulary. For the sake of consistency, we speak in terms of models, population, parameters and iterations to respectively designate candidate solutions, ensemble of candidate solutions, variables of solution vectors and successive solution updates. Besides, unless explicitly stated, the population size is denoted by n, the dimensionality by d, the iteration number by superscript k and the subscript i refers to the individual i of the population.

Differential Evolution

Differential Evolution (DE) is a genetic programming algorithm introduced by Storn and Price (1997). It has shown excellent performances in many real-world problems including geophysical problems (Storn (2017)). DE starts by generating a population of models uniformly distributed in the model parameter space. For each target model $\mathbf{m}_i^k$, DE generates a mutant model $\mathbf{v}_i^k$ by adding the weighted difference between two population models to a third model such that

$$\mathbf{v}_i^k = \mathbf{m}_{r_1}^{k-1} + F\left(\mathbf{m}_{r_2}^{k-1} - \mathbf{m}_{r_3}^{k-1}\right) \tag{4.3}$$

where $r_1, r_2, r_3 \in \{1, 2, ..., n\}$ are three distinct random indices different from i, $F \in [0, 2]$ is the mutation factor that weighs the differential variation $\left(\mathbf{m}_{r_2}^{k-1} - \mathbf{m}_{r_3}^{k-1}\right)$. This operation is called mutation. In order to increase diversity within the population of mutant models, crossover has been introduced to produce a trial model $\mathbf{u}_i^k$ following

$$u_{ji}^k = \begin{cases} v_{ji}^k & \text{if } r_j \leq CR \text{ or } j = R \\ m_{ji}^{k-1} & \text{otherwise} \end{cases} \tag{4.4}$$

with j being the parameter index, $r_j \sim \mathcal{U}(0, 1)$ a uniform random number, $R \in \{1, 2, ..., d\}$ a random parameter index and $CR \in [0, 1]$ the crossover rate that controls the likelihood to receive a parameter from a mutant model with the condition $j = R$ ensuring that it gets at least one mutated parameter. Finally, selection applies the greedy criterion to determine which models to preserve based on their misfit function values. If the trial model $\mathbf{u}_i^k$ yields a lower misfit, it replaces the target model $\mathbf{m}_i^k$, otherwise, its previous value is retained. The variant presented in this section that will be used in this paper is known as DE/*rand*/*1*/*bin* as only one differential weight is added to a randomly chosen model, and crossover is due to independent binomial experiments (Storn and Price (1997)). The mutation and crossover mechanisms are illustrated in Figure 4.1.

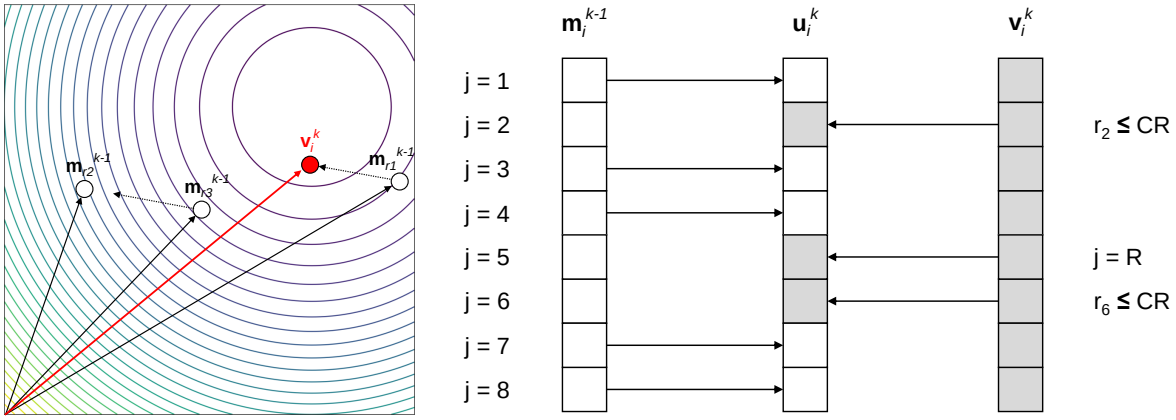

Figure 4.1: (Left) Mutation in DE on a 2D misfit function represented by the contour lines. $\mathbf{v}_i^k$ is generated by adding the weighted differential variation $\left(\mathbf{m}_{r_2}^{k-1} - \mathbf{m}_{r_3}^{k-1}\right)$ to the individual $\mathbf{m}_{r_1}^{k-1}$, with $\mathbf{m}_{r_1}^{k-1}$, $\mathbf{m}_{r_2}^{k-1}$ and $\mathbf{m}_{r_3}^{k-1}$ three random individuals chosen in the population. (Right) Crossover in DE for $d = 8$ parameters. For each parameter, the trial vector $\mathbf{u}_i^k$ receives a parameter from either the current or mutant vectors accordingly to a binomial distribution with probability defined by CR.

Competitive Particle Swarm Optimization

Particle Swarm Optimization (PSO) has been introduced by Kennedy and Eberhart (1995) to study the social behavior of fishes and birds. It belongs to the EA subclass of Swarm intelligence where collective knowledge is channeled within the population. In PSO, the first step is to generate a population randomly distributed in the model parameter space. Each model is represented by a particle that interacts with its neighborhood to find the global minimum of the misfit function.

At iteration k, a particle i is defined by a model vector $\mathbf{m}_i^k$ and a velocity vector $\mathbf{v}_i^k$ and is adjusted according to its own personal best model and the global best model of the whole population. The velocity vector controls how a model moves in the model parameter space and is initialized to zero (Engelbrecht (2012)). The velocity and the position of each model are updated following

$$\mathbf{v}_i^k = \omega \mathbf{v}_i^{k-1} + \phi_p \mathbf{r}_p^k \left(\mathbf{m}_{p,i} - \mathbf{m}_i^{k-1}\right) + \phi_g \mathbf{r}_g^k \left(\mathbf{m}_g - \mathbf{m}_i^{k-1}\right) \tag{4.5}$$

$$\mathbf{m}_i^k = \mathbf{m}_i^{k-1} + \mathbf{v}_i^k \tag{4.6}$$

where $\mathbf{m}_{p,i}$ and $\mathbf{m}_g$ are respectively the personal best model of particle i and the global best model of the population, $\mathbf{r}_p^k$ and $\mathbf{r}_g^k$ are uniform random number vectors drawn at iteration k, ω is an inertia weight, ϕ_p and ϕ_g are two acceleration parameters that respectively control the cognition and social interactions of the particles. Principle of PSO is illustrated in Figure 4.2.

The classical implementation of PSO suffers from premature convergence and stagnation as it can be trapped in a local minimum, particularly when the evaluated function has a complex landscape. Therefore, Luu *et al.* (2018) proposed a Competitive PSO (CPSO) to help the models to escape from a local minimum whenever the population stagnates. The idea is to improve the diversity of the population by renewing part of the population and keeping the "best" models only. "Worst" models are reset, allowing a better exploration of the model space. A reset is called a "competition" and is triggered whenever premature convergence is detected. The population is considered to be stagnating when its maximum radius δ^k is lower than a threshold ε following

$$\delta^k = \max_{1 \leq i \leq n} \left(\frac{\left\| \mathbf{m}_i^k - \mathbf{m}_g \right\|}{\left\| \mathbf{m}_{\max} - \mathbf{m}_{\min} \right\|} \right) < \varepsilon = \frac{\log\left(1 + 0.003n\right)}{\max\left(0.2, \log\left(0.01 k_{\max}\right)\right)} \tag{4.7}$$

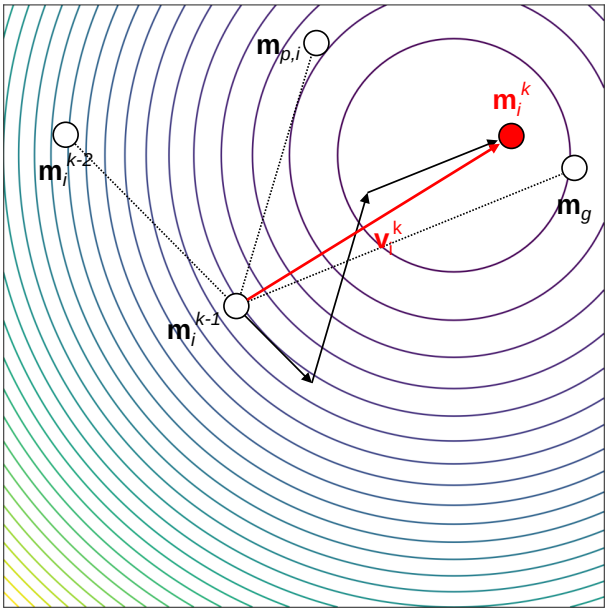

Figure 4.2: Principle of PSO on a 2D misfit function represented by the contour lines. Particle velocity $\mathbf{v}_i^k$ is constructed by adding three weighted terms: the previous velocity $\mathbf{v}_i^{k-1}$ that acts as an inertial term, the cognition term $\left(\mathbf{m}_{p,i} - \mathbf{m}_i^{k-1}\right)$ that accounts for the particle's personal knowledge, and the sociability term $\left(\mathbf{m}_g - \mathbf{m}_i^{k-1}\right)$ that involves the knowledge of the entire swarm.

where $\|\cdot\|$ denotes the Euclidean norm, and $k_{\max}$ is the maximum number of iterations. Besides, a competitivity parameter $\gamma \in [0, 2]$ is introduced to control the proportion of models to reset following

$$\sigma(k) = \frac{1}{1 + e^{\frac{1}{0.09}\left(\frac{k}{k_{\max}} - \gamma + 0.5\right)}}. \tag{4.8}$$

The logistic function defined in Equation (4.8) has been tweaked so that the population is competitive at early iterations and many models are reset; at later iterations, the population stagnates in the last minimum found to refine the solution. CPSO has demonstrated better convergence rates compared to classical implementation of PSO, and is more robust to the choice of its control parameters.

Covariance Matrix Adaptation - Evolution Strategy

The Covariance Matrix Adaptation - Evolution Strategy (CMA-ES) (Hansen, Müller and Koumoutsakos (2003)) is considered as the state-of-the-art method for stochastic numerical optimization and is derived from the natural gradient. It is a second-order method similar to Quasi-Newton methods yet randomized, that has been designed to approximate the contour lines of the objective function by adapting the covariance matrix of a multivariate gaussian distribution to the objective function topography. It belongs to the class of Evolution strategies that has been introduced in the early seventies (Rechenberg (1973)). In CMA-ES, a population consists of λ models called offspring sampled from a multivariate gaussian distribution :

$$\forall i \in [1, \lambda], \mathbf{m}_i^k \sim \bar{\mathbf{m}}^{k-1} + \sigma^{k-1}\mathcal{N}\left(\mathbf{0}, \mathbf{C}^{k-1}\right) \tag{4.9}$$

where $\mathcal{N}\left(\mathbf{0}, \mathbf{C}^{k-1}\right)$ denotes a multivariate gaussian distribution with zero mean and covariance matrix $\mathbf{C}^{k-1}$, $\bar{\mathbf{m}}^{k-1}$ is the mean vector of the distribution and σ^{k-1} is the step size. Then, μ

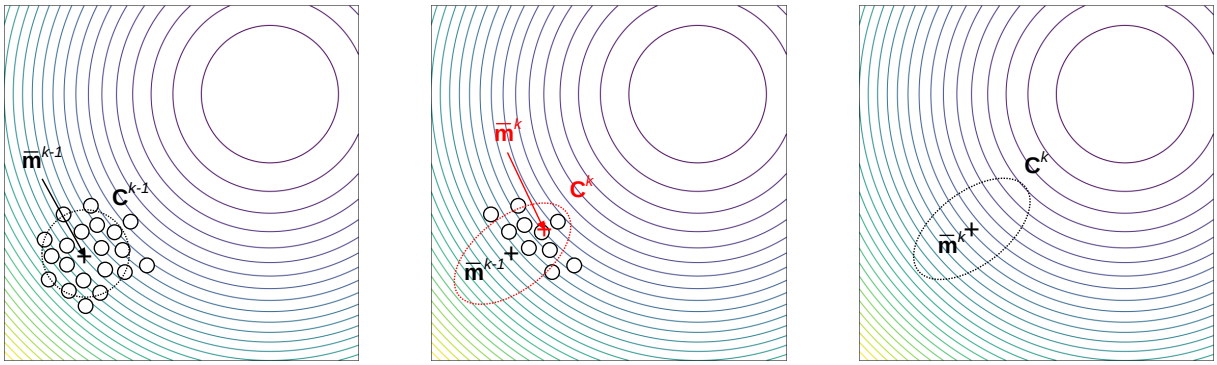

Figure 4.3: Principle of CMA-ES on a 2D misfit function represented by the contour lines. The population should move toward the upper right corner. (Left) Sample of $\lambda = 20$ offspring distributed accordingly to $\mathcal{N}\left(\bar{\mathbf{m}}^{k-1}, \mathbf{C}^{k-1}\right)$. (Middle) $\mu = 10$ best individuals selected to update the mean and covariance matrix. (Right) Mutation distribution for the next generation. Adapted from Hansen (2011).

models are selected as parents among the model offspring that yield the lowest misfit function values, and recombined to form the mean of the distribution for the next iteration, following

$$\bar{\mathbf{m}}^{k} = \bar{\mathbf{m}}^{k-1} + \sum_{i=1}^{\mu} \omega_i \left(\mathbf{m}_{i:\lambda}^{k} - \bar{\mathbf{m}}^{k-1}\right), \quad \text{with} \quad \sum_{i=1}^{\mu} \omega_i = 1, \quad \omega_1 \geq \omega_2 \geq \ldots \geq \omega_\mu > 0 \tag{4.10}$$

where $i:\lambda$ denotes the index of the i^{th} best model offspring. The CMA-ES then adapts the covariance matrix of the distribution by performing two critical updates, namely the rank-one and rank-μ updates, and reads

$$\mathbf{C}^{k} = \left(1 - c_1 - c_\mu \sum \omega_i\right) \mathbf{C}^{k-1} + c_1 \underbrace{\mathbf{p}_c^k \mathbf{p}_c^{k\top}}_{\text{rank-one update}} \tag{4.11}$$

$$+ c_\mu \underbrace{\sum_{i=1}^{\lambda} \frac{\omega_i}{\sigma^{k-1}} \left(\mathbf{m}_{i:\lambda}^{k} - \bar{\mathbf{m}}^{k-1}\right) \left(\mathbf{m}_{i:\lambda}^{k} - \bar{\mathbf{m}}^{k-1}\right)^{\top}}_{\text{rank-}\mu\text{ update}} \tag{4.12}$$

with $c_1 \leq 1$ and $c_\mu \leq 1$ being the learning rates. The first term accounts for the information from the previous covariance matrices whose contributions decay exponentially with time. The rank-one update reinforces the likelihood of steps in the vicinity direction of the evolution path $\mathbf{p}_c^k$ (sum of consecutive steps of the mean in the previous iterations) while the rank-μ update exploits information from the distribution of the current population.

The adaptation of the covariance matrix does not control the overall scale of the distribution, only the directions and lengths of its principal axis. Thus, the step size is also independently adapted by comparing the length of a second evolution path $\mathbf{p}_\sigma^k$ (sum of consecutive step lengths) to its expected length upon random selection, which is written

$$\sigma^{k} = \sigma^{k-1} \left(\frac{c_\sigma}{d_\sigma} \left(\frac{\left\|\mathbf{p}_\sigma^k\right\|}{E\left[\left\|\mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)\right\|\right]} - 1\right)\right) \tag{4.13}$$

where $c_\sigma < 1$ is the time horizon of the evolution path and $d_\sigma \approx 1$ is a damping parameter. Therefore, the step size is decreased whenever it is too short, and increased whenever it is too long. The principle of CMA-ES is illustrated in Figure 4.3.

4.2.2 Control parameter values

EA are designed to be easy to use with few control parameters that are robust and easy to choose. In addition to the population size and the maximum number of iterations, DE is controlled by two main user parameters, namely the mutation factor F and the crossover rate CR. CPSO is controlled by four parameters, namely the inertia weight ω, the cognition and sociability parameters ϕ_p and ϕ_g, and the competitivity parameter γ. From a user point of view, CMA-ES requires only two parameters to be set by the user, namely the initial mean and the step size, the remaining parameters introduced are usually fixed independently of the problem. Empirical studies have shown that performances of EA are sensitive to the input control parameters, especially DE and CPSO. However, these studies have provided some insights on the initialization of these parameters (Iwan *et al.* (2012), Van Den Bergh and Engelbrecht (2006), Eberhart and Shi (2000)).

In practice, we followed these guidelines that turn out to be very robust and summarize in Table 4.1 the parameter values for each EA. Note that this table only shows one parameter for CMA-ES (i.e. the initial step size) as the choice of the initial mean is explained in Section 4.3.3. In general, an algorithm is stopped when the misfit value threshold or the maximum number of iterations specified by the user are reached. In the following, we choose the latter stopping criterion to allow the algorithms to sample around the final solution.

Table 4.1: Default control parameter values.

	Symbol	Value
DE		
Mutation factor	F	0.9
Crossover rate	CR	0.5
CPSO		
Inertia weight	ω	0.7298
Cognition	ϕ_p	1.49618
Sociability	ϕ_g	1.49618
Competitivity	γ	1
CMA-ES		
Initial step size	σ	$\frac{\mathbf{m}_{\max}-\mathbf{m}_{\min}}{3}$

4.3 Numerical example

4.3.1 Synthetic data and parametrization

In this paper, we invert for the Marmousi velocity model using the three EA presented in Section 4.2.1. The Marmousi velocity model was designed to be geologically realistic yet complex with strong vertical and lateral velocity variations (Versteeg (1994)). Therefore, reconstructing this velocity model by tomographic methods is challenging. We smooth the original model using a gaussian filter with a 50 meters standard deviation in both directions to generate synthetic traveltimes considering a stationary acquisition geometry that consists of two hundred shots and four hundred receivers spaced every 50 and 25 meters, respectively. The maximum offset reaches 10 km just for the end shots, so we expect to retrieve the velocities fairly accurately down to a depth of approximately 1 km that corresponds to 1/10 of the maximum offset. Beyond this depth, accuracy of the retrieved velocity model will decrease more and more with depth.

We parametrize the velocity model using a cubic cardinal B-spline surface (order 4). Such a representation reduces the number of unknowns and provides by nature a smooth velocity model which thus avoids introducing an additional regularization term. A B-spline surface is a parametric surface defined by $n_z \times n_x$ control points $\mathbf{P} \in \mathbb{R}^{n_z \times n_x}$, its orders k_z and k_x respectively corresponding to functions of degree $k_z - 1$ and $k_x - 1$. Any B-spline surface of order (k_z, k_x) can be expressed as a linear combination of B-splines of order (k_z, k_x) following

$$S(u,v) = \sum_{i=1}^{n_z} \sum_{j=1}^{n_x} P_{ij} B_i^{k_z}(u) B_j^{k_x}(v), \quad u, v \in [0,1] \tag{4.14}$$

where the B-spline basis functions $B_i^{k_z}$ and $B_j^{k_x}$ can be calculated with de Boor's recursion (Boor (1972)). In our refraction tomography problem, $\mathbf{P}$ is a set of control points that contains the velocity information of the model. The velocity model is finally interpolated on a finer 120 by 400 cartesian grid (i.e. grid spacing of 25 meters) and traveltimes are calculated using a 2D Eikonal solver (Noble, Gesret and Belayouni (2014)) that generates traveltime grids for each shot location. Such a grid spacing offers a good trade-off between computation time and traveltime accuracy for this data set. It should be mentioned that because of the B-spline parametrization, we will only be able to retrieve the long wavelengths of the velocity model. Therefore, in order to facilitate the comparison, we generate a low-frequency target velocity model by further smoothing the original model using a gaussian filter with respectively a 200 and 75 meters standard deviation in X and Z directions.

Figure 4.4 shows the velocity model used to generate the travaltime data (top), the low-frequency target velocity model that will be used to compare the results (middle), and the ray density map for this acquisition geometry (bottom). The ray density map exhibits some area that are not illuminated (e.g. between 4 and 7 km at 2 km depth, and the lower corners of the velociy model). High uncertainties should thus be expected in these areas. The data being noise-free, we set the data covariance matrix in Equation (4.2) to $\mathbf{C_D} = \mathbf{I}$ with $\mathbf{I}$ the identity matrix.

4.3.2 Weighted mean model and standard deviation

Because refraction tomography is an ill-posed and highly multi-modal problem, the solution is not unique and the global minimum is not guaranteed to be found by an evolutionary optimizer, especially in high dimensions. Typically, tomographic solutions are appraised using several statistical measures, in particular the mean and the standard deviation (Bodin and Sambridge (2009), Bodin *et al.* (2012)). The mean velocity model is usually considered as a reference model while the standard deviation is interpreted as an error map, under the assumption that each model parameter is normally distributed. In this study, the weighted mean velocity model obtained by averaging all the models sampled is also considered alternatively to the best velocity model. In addition, the associated errors (i.e. standard deviation) are estimated to assess the ability of an EA to quantify uncertainties.

A single run of EA does not provide a proper sampling of the model parameter space since they are designed to rapidly locate and exploit a single minimum. Sen and Stoffa (1996) (in Section Multiple MAP estimation) suggests to perform several independent runs of a global optimizer with different starting solutions to sample different parts of the model parameter space. Assuming that the posterior density distribution $P\left(\mathbf{m} \,\middle|\, \mathbf{d}^{obs}\right)$ can be approximated using all the models sampled by several runs, the mean model is calculated as the weighted sum of all the models sampled and is written as

$$\hat{\mathbf{v}} = \frac{\sum_i P\left(\mathbf{m_i} \,\middle|\, \mathbf{d}^{obs}\right) \mathbf{v_i}}{\sum_i P\left(\mathbf{m_i} \mid \mathbf{d}^{obs}\right)} \tag{4.15}$$

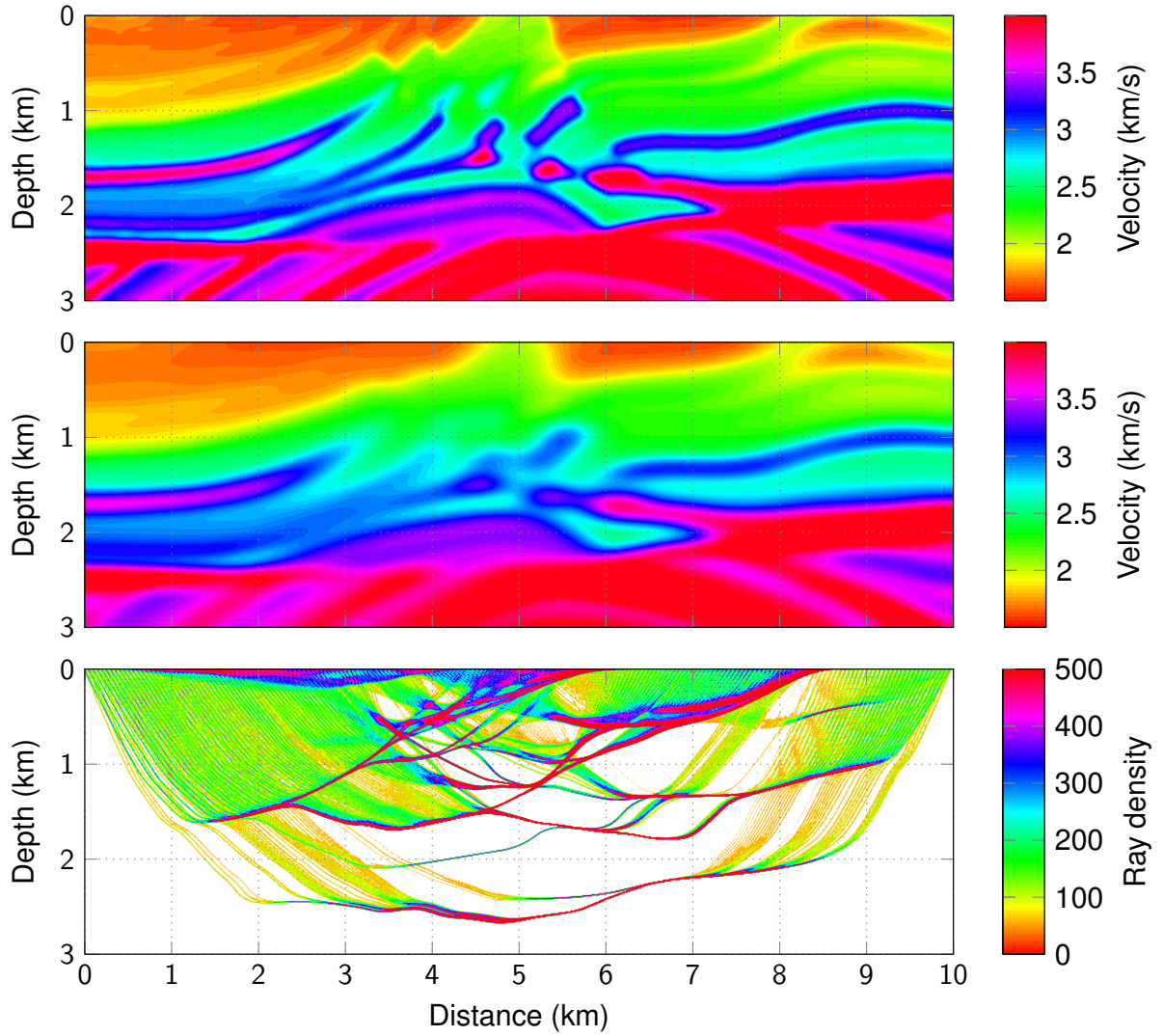

Figure 4.4: Marmousi velocity model. (Top) Velocity model used to generate the traveltime data. (Middle) Low-frequency target velocity model. (Bottom) Ray density map.

where $\mathbf{v_i}$ is the 2D cartesian velocity model interpolated from the B-spline grid defined by the model parameters $\mathbf{m_i}$. More specifically, the contribution of each model is weighted by its posterior probability. Because bad fitting models have very low probability, their contributions to the mean are nil. Likewise, the unbiased standard deviation reads

$$\sigma = \sqrt{\frac{N}{N-1} \frac{\sum_i P\left(\mathbf{m_i} \mid \mathbf{d}^{obs}\right) \left(\mathbf{v_i} - \hat{\mathbf{v}}\right)^2}{\sum_i P\left(\mathbf{m_i} \mid \mathbf{d}^{obs}\right)}} \tag{4.16}$$

with N the total number of models.

From a formal point of view, the estimation of uncertainties should be obtained by computing the density plots or ideally the quantiles based on the marginals. Even if EA are very efficient in terms of CPU resources, we still end up with a very large number of models (i.e. the number of independent runs times the number of iterations times the population size), and these quantities can become prohibitive to calculate. The standard deviation is a pragmatic approach to quantify uncertainties. It is also in agreement with the mean.

4.3.3 Initial models

From a general point of view, global optimization methods are ideally insensitive to the initial models under the condition that the algorithm is run for a very large number of iterations (e.g. Bodin and Sambridge (2009) required 1 million iterations for a velocity model parametrized with a small number of unknowns). Whenever using EA, initial models are usually randomly generated in the model parameter space, which ideally should not influence the output of the optimization. However, refraction tomography is a highly non-linear and multi-modal optimization problem. While DE, CPSO and CMA-ES have been designed to deal with premature convergence, random velocity models may contain strong local velocity heterogeneities that can mislead the optimization process and require more iterations to ensure convergence. Therefore, one can help the search by introducing more realistic initial models (yet randomized).

In order to investigate the influence of the initial models on the quality of the inverted models, we run 20 inversions of 5000 iterations using CPSO with a population size of 26 individuals and different types of model initializations, namely fully random, homogeneous and vertically increasing gradient velocity models. The velocity model is parametrized by a cardinal B-spline grid of 8 by 13 ($d = 104$) i.e. 8 and 13 spline nodes are regularly spaced every 429 m in the Z direction and every 833 m in the X direction, respectively. Such parametrization offers a good trade-off between the vertical resolution and the dimensionality of the problem. For each type of model initialization, the velocities of the B-spline nodes are randomly initialized according to a uniform distribution which is more explicitly described in Table 4.2.

Table 4.2: Initialization of the velocities of the B-spline nodes for each type of model initialization. The initialization procedures are independently applied to every model in the population.

Method	Description
Random	Velocities are randomly sampled from $\mathcal{U}(1500, 5000)$.
Homogeneous	Velocities are all equal to a single random value sampled from $\mathcal{U}(1500, 5000)$.
Gradient	Velocity at the surface is sampled from $\mathcal{U}(1500, 3250)$. Velocity at the bottom is sampled from $\mathcal{U}(3250, 5000)$. Velocities for intermediate nodes are linearly interpolated between surface and bottom velocity nodes.

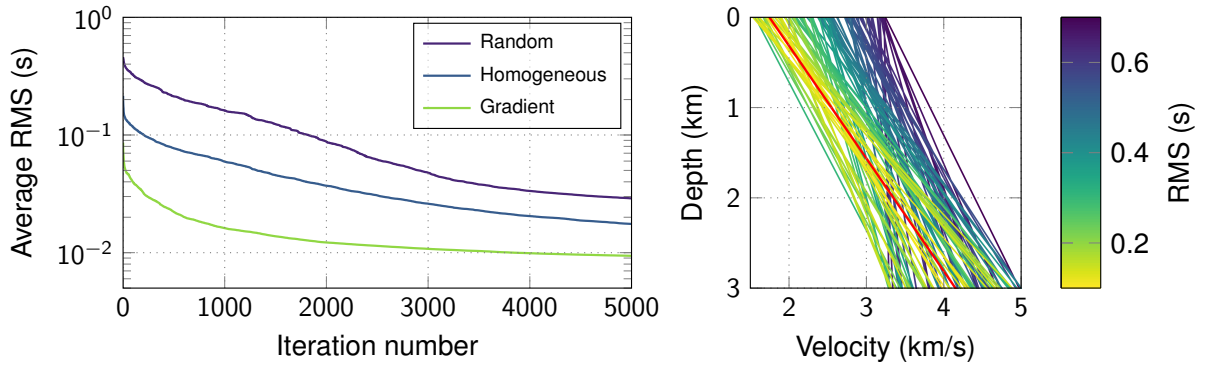

Figure 4.5: (Left) Average RMS over 20 runs as a function of iteration number. When using random vertically increasing gradient initialization, the optimizers converge faster toward low RMS velocity models. (Right) Example of 100 random vertically increasing gradient velocity models, the color scale indicating their RMS values. For CMA-ES, the model that yields the lowest RMS is chosen as the initial mean vector (red).

We recall that the B-spline grid is interpolated on a 120 by 400 cartesian grid for the computation of traveltimes. The influence of the population size with dimensionality will be further studied in the next section. Figure 4.5 shows the average evolution of the RMS over the 20 runs for the different types of initializations (left) and example of 100 random vertically increasing gradient 1D profiles (right). Initializing the models with increasing gradient velocity models allows the optimizers to start the search in a lower RMS space than the two other types of initializations. The model that yields the lowest RMS will serve as the global best model for DE and CPSO, while it will be used as the initial mean of the gaussian distribution for CMA-ES.

Figure 4.6 displays the mean velocity model obtained for the three different types of initialization. While the shallow structure of the weighted mean velocity models fit the target velocity model, the optimizers are not able to retrieve the strong refractor at 2.5 km depth when using fully random and homogeneous initial models. On the other hand, the mean velocity model for gradient initialization fits well the long wavelengths of the target velocity model at all depths.

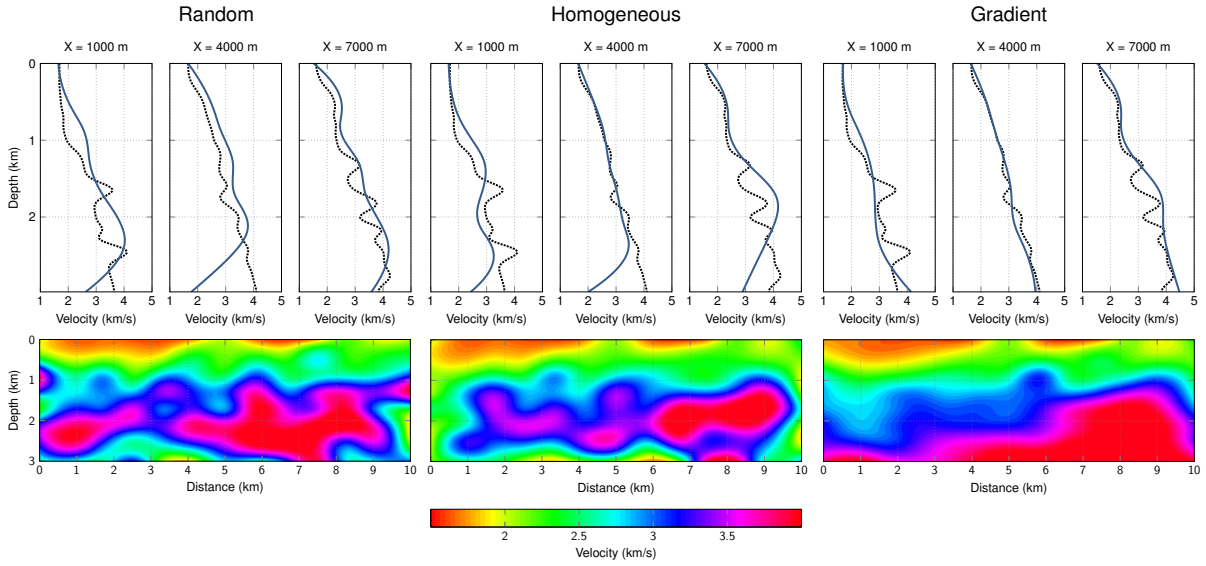

Figure 4.6: 1D profiles (top) and 2D models (bottom) for different initializations. (Left) Fully random. (Middle) Homogeneous. (Right) Vertically increasing gradient. The mean velocity model (blue) fits the long wavelengths of the target velocity model (black) at all depths for gradient initialization. The results have been obtained using CPSO.

Note that the results presented were obtained using CPSO only. However, comparable results and conclusions can be obtained with DE or CMA-ES as the initial population starts in a better search space which allows the algorithms to converge faster.

4.3.4 Results

In the previous section, we investigated the influence of the initialization by inverting for a 8 by 13 B-spline grid ($d = 104$) and obtained a very low resolution mean velocity model. In this section, we invert for a velocity model of higher resolution parametrized by a 15 by 20 B-spline grid ($d = 300$) i.e. 15 and 20 spline nodes are regularly spaced every 214 m in the Z direction and every 526 m in the X direction, respectively. We investigate the influences of the population size n (26, 52 and 104) and the maximum number of iterations $k_{\max}$ (10000 and 20000).

We use a parallel hybrid implementation where the computation of the misfit function values is distributed to the MPI processes and the computation of the traveltime grids for each shot is scattered over the threads with OpenMP. Our implementation requires the population size to be a multiple of the number of available cores for maximum efficiency to avoid idle cores. The inversions are performed using 520 cores (26 Symmetric Multi-Processor machines with two sockets made of 2 Intel® Xeon® CPU E5-2640, 10 cores @ 2.40 GHz each).

For each EA, we conduct three different experiments consisting of 50 independent runs with different population sizes and maximum numbers of iterations. The parameters and the computation times per run for all the experiments on the supercomputer are reported in Table 4.3. The first experiment principally aims to determine which algorithm is the most robust with smaller population. For the second and third experiments, we fix the number of forward modelings per run to determine whether it is better to have a larger population or to perform more iterations given the same computation time. Indeed, the computation time is the main limiting factor when dealing with real-world problems and is directly proportional to the population size and the maximum number of iterations. For all the experiments, the population is initialized using increasing gradient velocity models as previously described in Table 4.2.

Table 4.3: Parameters and computation times per run (in hours). MPI and OMP respectively indicate the number of processes and threads used for each experiment.

	Experiment 1	Experiment 2	Experiment 3
Parameter			
Population size	26	52	104
Number of iterations	10000	20000	10000
MPI / OMP	26 / 20	52 / 10	104 / 5
Computation time			
DE	0.21	0.75	0.75
CPSO	0.20	0.74	0.74
CMA-ES	0.38	1.27	1.26

Figure 4.7 displays the average evolution of the RMS and its standard deviation with respect to the iteration number for the 50 independent runs. The RMS standard deviation corresponds to the repeatability of an algorithm and illustrates how the initial populations affect the final solutions. Generally, increasing the population size improves the repeatability of the algorithms (lower RMS deviation). Increasing the population size from 26 to 52 noticeably improves their speeds of convergence. However, increasing the population size from 52 to 104 does not improve much

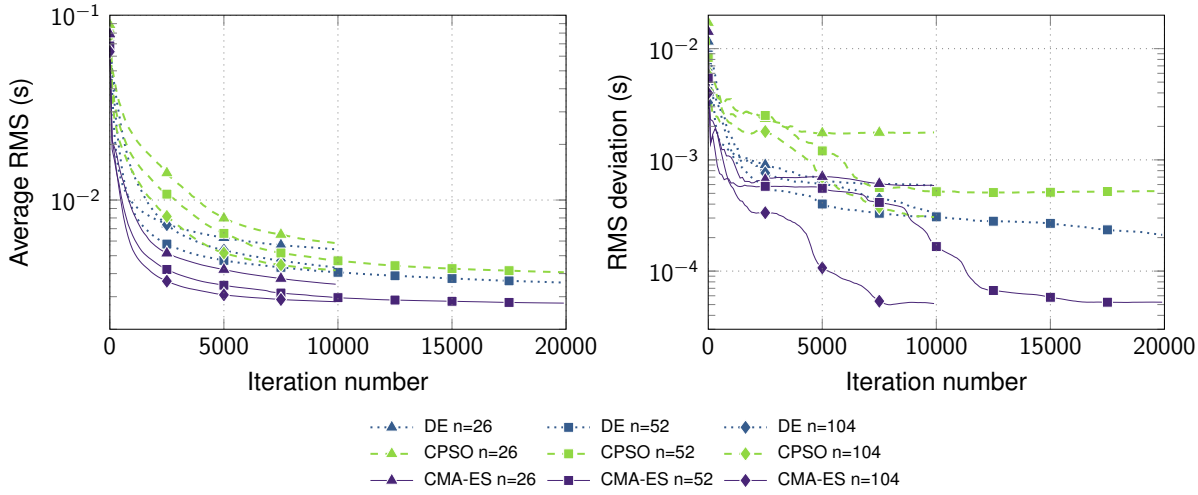

Figure 4.7: Evolution of average RMS (left) and RMS deviation (right) with respect to iteration number for the 3 experiments with the 3 EA.

their convergence behaviors. On the other hand, more iterations allow the algorithm to slowly refine the solutions. Indeed, CPSO and CMA-ES were able to reach misfits comparable to the ones obtained with a smaller population but with twice more iterations. These results shows that larger population improves the speed of convergence but is not necessarily required when considering the computation time. Indeed, performing more iterations can also improve the convergence of the algorithms. Nonetheless, experiment 2 for CPSO produces worse solutions in terms of repeatability than in experiment 3.

From an optimization point of view, CMA-ES clearly outperforms both DE and CPSO for its robustness and speed of convergence as it is able to reach lower RMS even with a smaller population size of 26. In the following, we only discuss the results for experiment 3 but similar conclusions can be drawn with the other experiments. Figures 4.8 and 4.9 respectively display the vertical and horizontal velocity profiles of the target, the mean and best models at different locations and depths. For each EA, the best velocity models are of higher frequency yet none of them, even for CMA-ES, match the target velocity model in depth (below 1000 m). All the best velocity models are probably trapped in a local minimum due to the non-linearity and high dimensionality of the problem. On the other hand, despite their lower resolutions, the mean velocity models obtained by the three EA fit the long wavelengths of the target velocity model at all depths. This can be explained by the beneficial effect of sampling and averaging many models in the ensemble similarly to MCMC based methods (Bodin and Sambridge (2009)).

We also display in Figures 4.10 and 4.11 the cross-sections of the differences between the target velocity model and the mean velocity models for each method. CMA-ES exhibits worse results compared to DE and CPSO with higher absolute errors in both the vertical and horizontal cross-sections. CPSO seems to perform slightly better than DE since the average errors are closer to zero. More generally, the model fit for the three algorithms deteriorates with increasing depth which is consistent with the acquisition geometry on surface.

We show in Figure 4.12 the mean velocity models for experiment 3 and the associated uncertainties. We superimpose over the results the main structure and the ray coverage of the target velocity model to quality control the results. All the mean velocity models are fairly consistent with the target structure, except for DE that underestimates the velocity below 2.5 km depth. Nonetheless, when considering the associated relative standard deviations, only CPSO is in agreement with the target ray coverage. Indeed, high uncertainties are retrieved in areas that are not illuminated by the rays, while both DE and CMA-ES produce low uncertainties in these

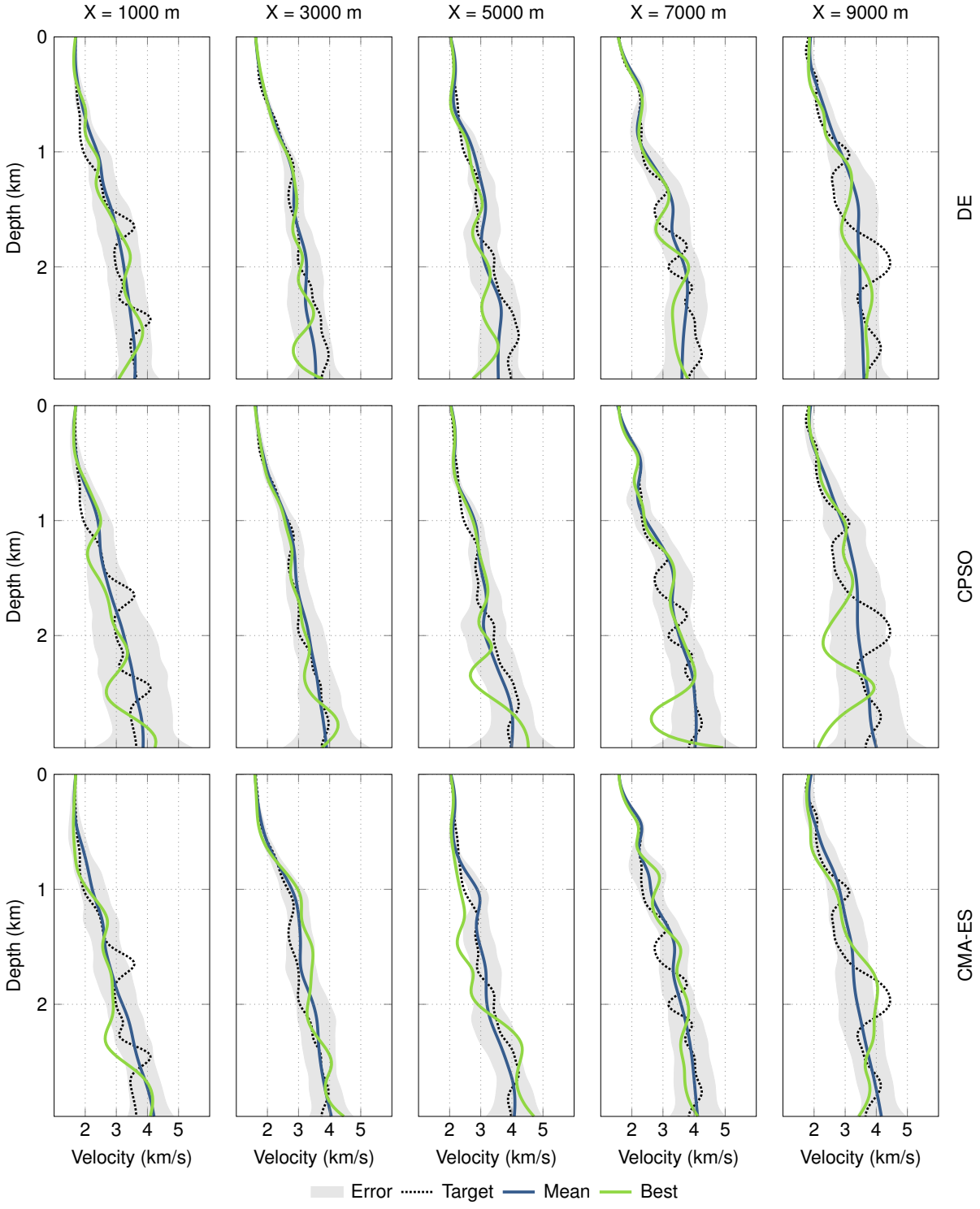

Figure 4.8: Comparison of vertical profiles between the target (black), the mean (blue) and the best (green) velocity models at different locations for the 3 EA. The errors are indicated in gray shade.

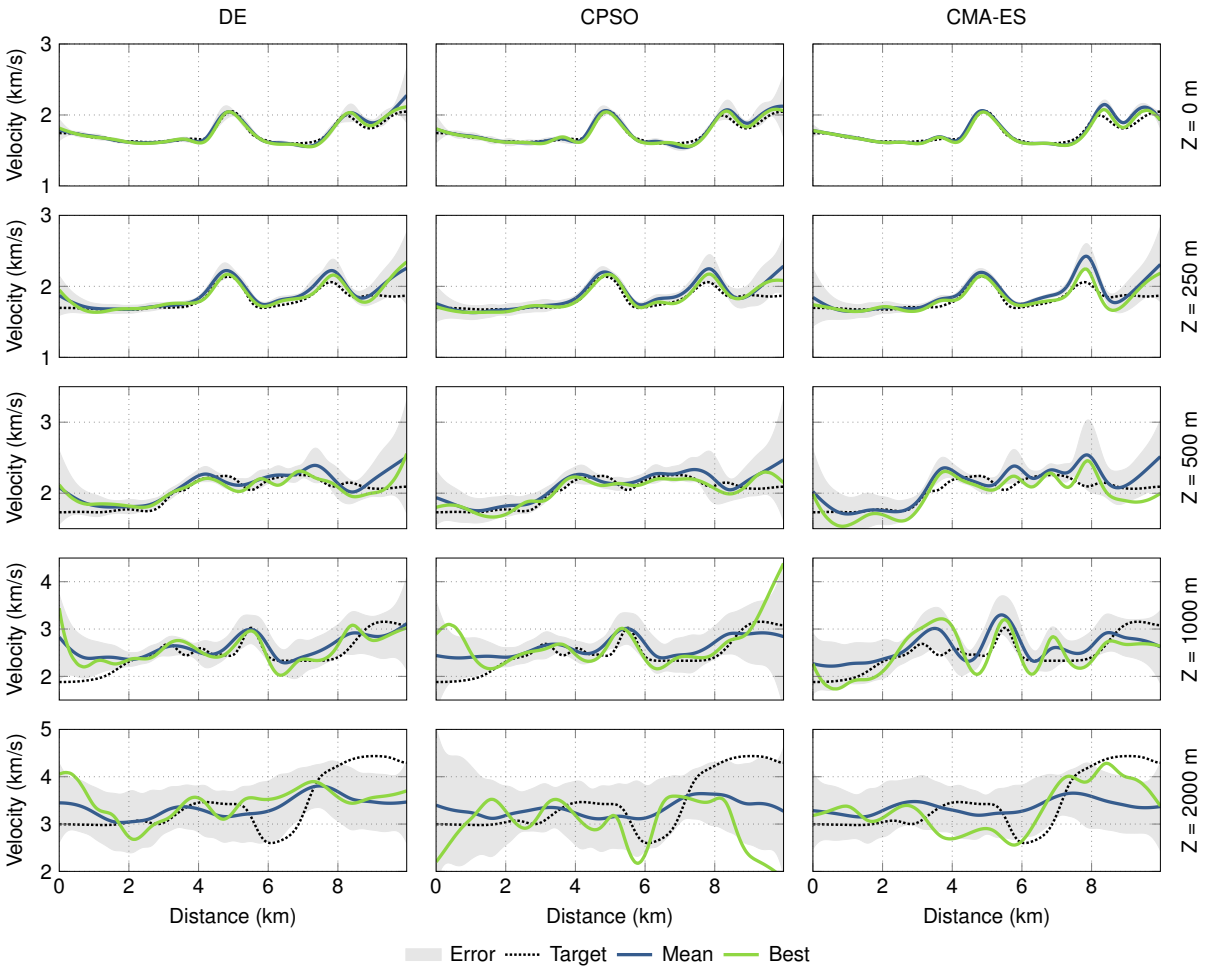

Figure 4.9: Comparison of horizontal profiles between the target (black), the mean (blue) and the best (green) velocity models at different depths for the 3 EA. The errors are indicated in gray shade.

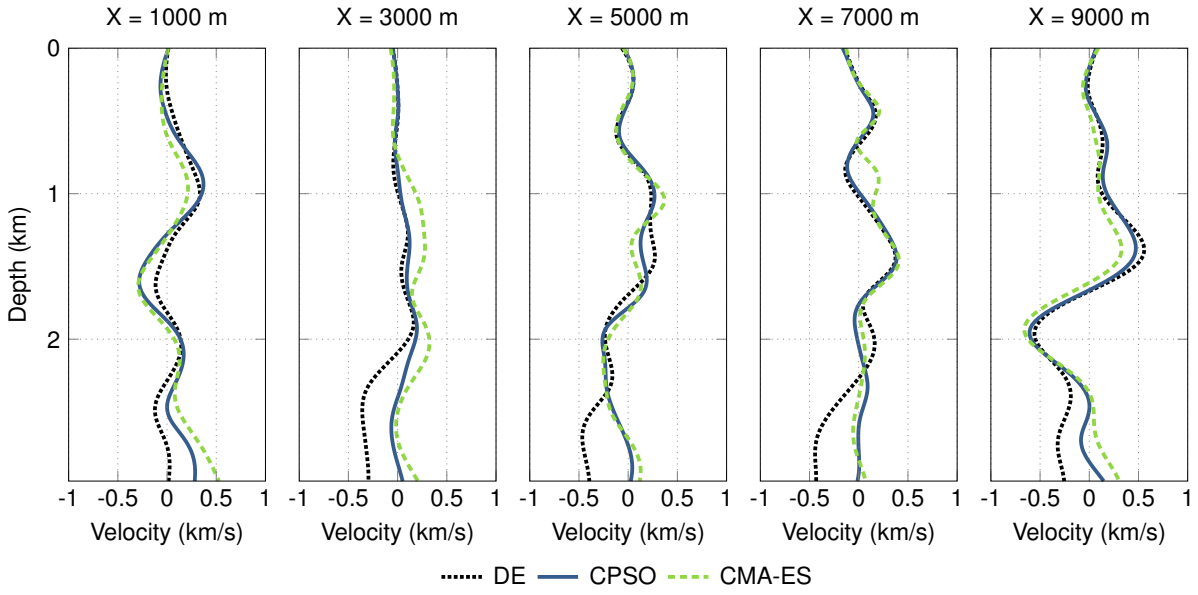

Figure 4.10: Vertical cross-sections of the difference between the target and mean velocity models.

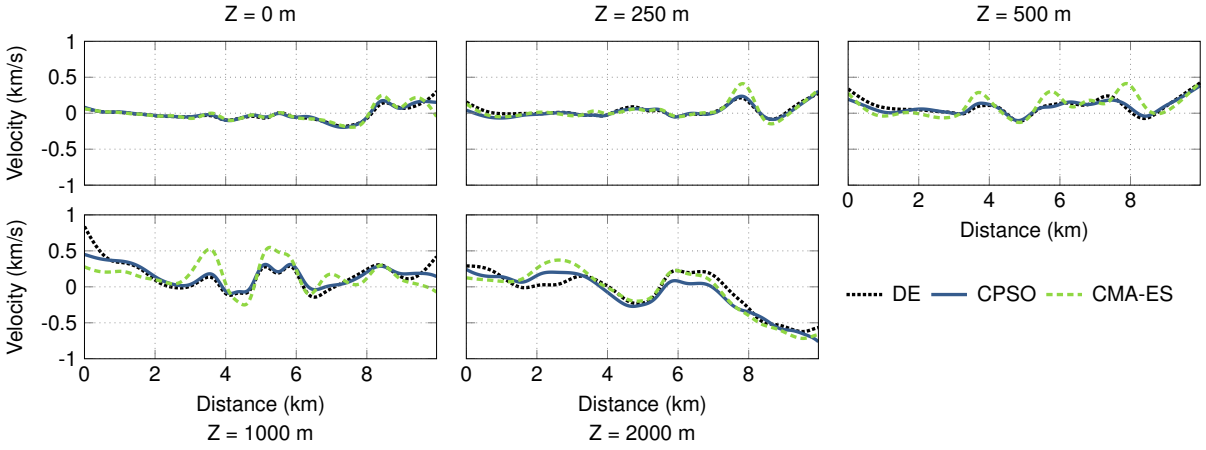

Figure 4.11: Horizontal cross-sections of the difference between the target and mean velocity models.

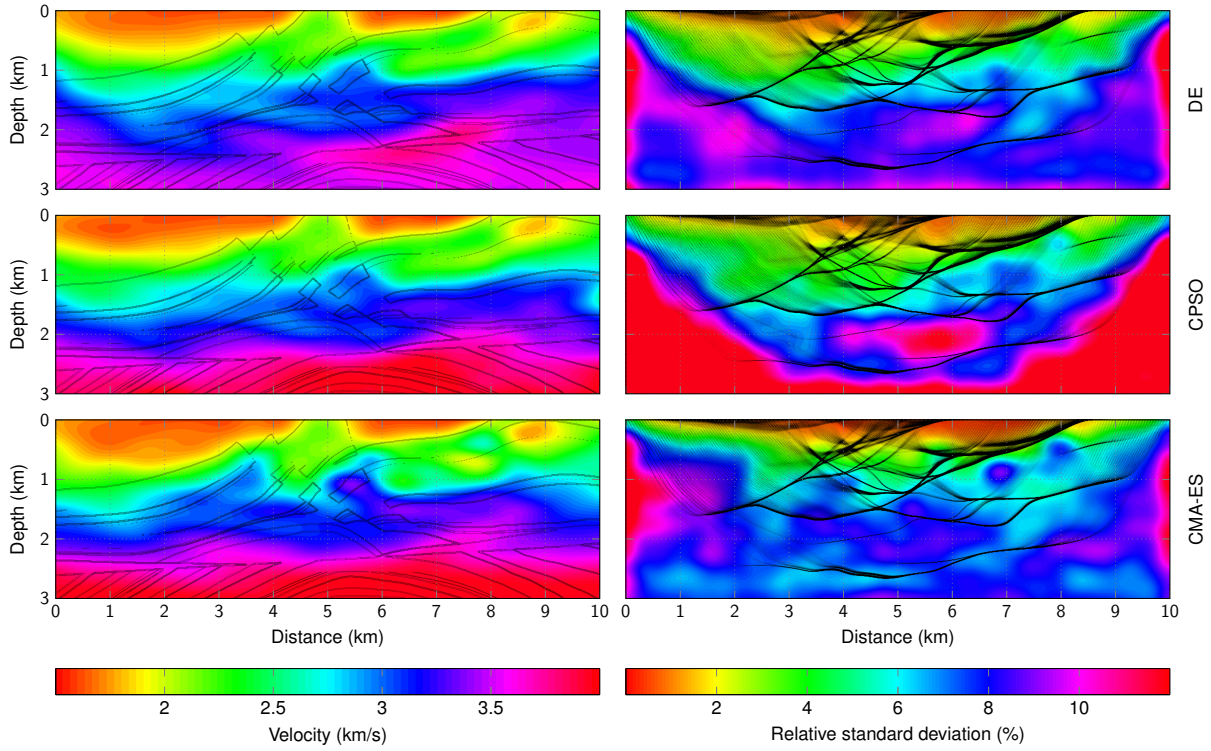

Figure 4.12: Inversion results for experiment 3. Weighted mean velocity models and associated uncertainties for (top) DE, (middle) CPSO and (bottom) CMA-ES. The main structure and the ray coverage of the target velocity model are superimposed over the results.

areas. Such a discrepancy is due to the better repeatability of DE and CMA-ES compared to CPSO. Indeed, great repeatability means that each independent run has produced a final model with comparable misfit value. The lower repeatability obtained by CPSO indicates that it also samples models of lower misfit values similarly to MCMC based methods. Therefore, despite being less repeatable, CPSO can be used to approximate the posterior density distribution, and thus is more reliable to quantify uncertainties compared to the other algorithms.

4.3.5 Scalability

EA are intrinsically parallel as they can simultaneously evaluate the quality of the models in a population. However, their parallel performances are mainly limited by the fraction of computational load that is not parallel (Amdahl (1967)). In our implementation, data management, generation of new populations and internal variable initializations and updates are all performed sequentially, only the evaluations of the models (i.e. computations of the misfit function values) are parallelized.

We evaluate the scalability of each algorithm on three refraction tomography problems of different dimensionality with various B-spline grids, namely a 10 by 15 grid ($d = 150$), a 15 by 20 grid ($d = 300$) and a 20 by 30 grid ($d = 600$). The population size is fixed to 104. As the algorithms are mainly CPU-bounded, we perform a strong scale analysis and measure the sequential and parallel computation time using one core only. We evaluate the parallel performance of each optimizer by calculating the theoretical speed up and parallel efficiency that can be achieved by the algorithms. The theoretical speed up S can be described by Amdahl's law stating that the computation can be decomposed into sequential and parallel tasks and the speed up is only

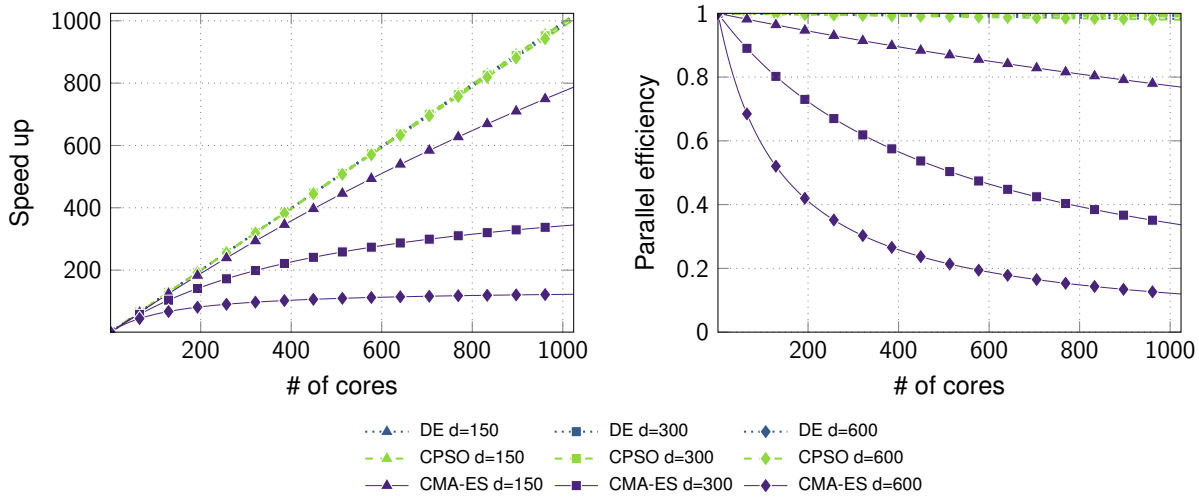

Figure 4.13: Maximum parallel performances of DE, CPSO and CMA-ES on a refraction tomography problem with a population size of 104. (Left) Speed up. (Right) Parallel efficiency.

limited by the time required by the sequential part, following

$$S(p, N) \leq \frac{1}{1 - p + \frac{p}{N}} \tag{4.17}$$

where p denotes the parallel fraction of the computation and N the number of cores working in parallel. The theoretical parallel efficiency is the ratio of the theoretical speed up to the number of cores. An algorithm is considered to be embarassingly parallel if the speed up equals the number of cores and the parallel efficiency is 1. The results of the scalability analysis are shown in Figure 4.13.

The theoretical speed up and parallel efficiency that can be achieved by DE and CPSO are almost ideal. However, parallel performance of CMA-ES strongly degrades with increasing processing power and dimensionality. This discrepancy is tied up with the fact that both CPSO and DE have an internal time complexity on the order of $\mathcal{O}(nd)$, while CMA-ES has a time complexity on the order of $\mathcal{O}(nd^3)$ due to the eigenvalue decomposition of the covariance matrix. While DE and CPSO seem to have an ideal scalability, in practice, this is not true as good scalability is harder to achieve with increasing number of cores due to the communication overhead and/or overhead implied by the parallel decomposition of the algorithms that are not accounted for in Amdahl's law.

4.4 Discussion and conclusion

The main goal of this paper is to study the feasibility of EA to solve the highly non-linear and multi-modal refraction tomography problem in high dimensions. With the rise in computational power, it is important to implement algorithms that are able to handle efficiently all the available CPU resources. The methodology presented is promising in terms of computational cost as EA are intrinsically parallel and are well adapted to supercomputers. We applied and compared three EA to solve the refraction tomography problem, namely the Differential Evolution (DE), the Competitive Particle Swarm Optimization (CPSO), and the Covariance Matrix Adaptation - Evolution Strategy (CMA-ES).

While global optimization methods should ideally be insensitive to the initial models, we showed that using realistic models (i.e. random increasing gradient) as initial population helps the EA to

start the search in a space of lower misfit which significantly improves the convergence of EA. The main advantage of using EA for refraction tomography lies in the fact that this initialization procedure does not require the user to explicitly specify a good starting model and works well even with uninformative but realistic feasible space boundaries. In our paper, we set the lower and upper boundaries to 1500 and 5000 m/s, respectively.

From a pure optimization point of view, CMA-ES clearly outperforms both DE and CPSO, being more robust in terms of repeatability and able to reach lower misfits even with a smaller population size. However, it comes at the cost of poor scalability with increasing dimensionality mainly due to the eigenvalue decomposition to adapt the covariance matrix. Akimoto, Auger and Hansen (2014) proposed a modification of CMA-ES called VDCMA that uses a diagonal matrix $\mathbf{D}$ and a principal rotation vector $\mathbf{v}$ to parametrize the covariance matrix following

$$\mathbf{C} = \mathbf{D}\left(\mathbf{I} + \mathbf{v}\mathbf{v}^\top\right)\mathbf{D}^\top \tag{4.18}$$

where $\mathbf{I}$ is the identity matrix. Such parametrization reduces its internal time complexity from $\mathcal{O}\left(nd^3\right)$ to $\mathcal{O}\left(nd\right)$ which should improve its scalability and feasibility for problems of larger dimension. On the other hand, DE and CPSO both exhibit good scalability but converge more slowly as larger population sizes and/or more iterations are required. Among the three methods tested, CPSO is the least repeatable algorithm.

Because of the multi-modal nature of refraction tomography, although the three EA are able to converge toward models that explain the data, the final inverted models fit the target velocity model at shallow depths only. Nevertheless, as stochastic algorithms, each run of EA explores different subspaces of the model parameter space. Therefore, one can use different statistical estimates to reconstruct the velocity model and appraise uncertainties. We demonstrated that the weighted mean velocity model over all the models explored by every runs of an EA is able to approximate the long wavelengths of the target velocity model. However, only CPSO was able to produce uncertainties consistent with the target ray coverage. This can be explained by the lower repeatability of CPSO implying that it samples subspaces of lower likelihood as well, similarly to MCMC based methods.

In this study, we investigated the influences of the population size and the maximum number of iterations. We tested different population sizes lower than the dimensionality ($n < d$). The *curse of dimensionality* implies an exponential growth of local optima. The different tests performed showed that increasing either the population size or the number of iterations both improve the convergence of the algorithms. Therefore, we would recommend to use larger populations instead of performing more iterations, especially since we can parallelize the individuals but not the iterations. However, although using a population larger than the number of dimensions ($n > d$) is common in problems of low dimensionality, we do not believe that increasing the population size to that extent would help EA to find the global minimum in acceptable time, but would rather deteriorate their performances (Chen, Montgomery and Bolufé-Röhler (2015)). In our experience, at least in solving tomographic problems, the population size should be set logarithmically with the dimensionality. Besides, more thorough analysis should be carried out on the different control parameters of EA. Indeed, we used the default parameter values that have been empirically tweaked in the literature with optimization in mind (i.e. good convergence and repeatability). For better estimates of uncertainty, these parameters should be tweaked to decrease the repeatability of the algorithms as we speculate that good repeatability is not beneficial for sampling. Nonetheless, in our case, default parameters for CPSO indeed perform well for sampling and uncertainty quantification.

The methodology is not limited to refraction tomography only and can be easily applied to crosshole tomography or stereotomography with no modification. Extension to a 3D tomographic problem is also straightforward.

4.5 List of symbols

Table 4.4: Symbol definitions.

Symbol	Definition
Inverse problem	
$\mathbf{m}$	Vector of model parameters
$\mathbf{m}_{\text{min}}, \mathbf{m}_{\text{max}}$	Vectors of model parameter space boundaries
$\mathbf{d}^{obs}$	Vector of observed data
$\mathbf{C_D}$	Data covariance matrix
g	Forward operator linking model parameters to data
E	Misfit function
n	Population size
d	Dimensionality, number of parameters to optimize
k, k_{max}	Iteration number and maximum number of iterations
DE	
$\mathbf{v}_i$	Vector of mutant model parameters
$\mathbf{u}_i$	Vector of trial model parameters
CPSO	
$\mathbf{v}_i$	Velocity vector
$\mathbf{m}_{p,i}$	Vector of personal best parameters of model i
$\mathbf{m}_g$	Vector of global best parameters of the population
$\mathbf{r}_p$, $\mathbf{r}_g$	Vectors of uniform random numbers
δ, ε	Swarm maximum radius and threshold
σ	Proportion of models to reset
CMA-ES	
λ, μ	Numbers of offspring and parents
σ^k	Step size at iteration k
$\bar{\mathbf{m}}$, $\mathbf{C}$	Mean vector of and covariance matrix of multivariate gaussian distribution
c_1, c_μ	Learning rates
$\mathbf{p}_c$, $\mathbf{p}_\sigma$	Evolution paths
c_σ, d_σ	Horizon and damping parameters
Mathematical notation	
$\mathcal{U}$	Uniform distribution
$\mathcal{N}$	Multivariate gaussian distribution
$E[\cdot]$	Expected value
i	Lower case subscript i denotes the individual number
j	Lower case subscript j denotes the parameter index
k	Lower case superscript k denotes the iteration number

Chapter 5

Neural network automated phase onset picking

Contents

Dans les chapitres précédents, les jeux de données que j'ai utilisés consistaient en des temps d'arrivée déjà pointés manuellement par un opérateur ou calculés synthétiquement. Le monitoring microsismique nécessite un algorithme de pointé automatique des phases efficace pour permettre la relocalisation en temps réel des évènements microsismiques induits par la fracturation hydraulique (Calvez *et al.* (2007)). À moindres mesures, les erreurs sur les temps de trajet contribuent aux erreurs sur les localisations. Les algorithmes de pointé automatique classiques n'ont pas été conçus pour estimer ces erreurs. Dans ce chapitre, je décris un prototype d'algorithme de pointé automatique des phases basé sur un réseau de neurones

(artificiel) multi-attributs. Il est important de rappeler que les réseaux de neurones ne sont pas le sujet principal de cette thèse. Le principe des réseaux de neurones est introduit en début de chapitre, et je me contente de présenter brièvement quelques notions importantes pour la compréhension de la méthodologie. Le chapitre est présenté comme une description d'un logiciel écrit en Python. Je décris la méthodologie en l'appliquant à un jeu de données réelles acquis au Laboratoire de Géologie de l'ENS Paris. Les attributs optimaux ainsi que leurs paramètres sont sélectionnés à l'aide d'une matrice de nuages de points. Un réseau de neurones peut ensuite être entraîné en utilisant soit une méthode de gradient, soit un algorithme évolutionniste. À partir de la carte de probabilité produite par le modèle de réseau de neurones entraîné, il est alors possible de prédire les temps d'arrivée, d'estimer les incertitudes du pointé et de rejeter les faux positifs. Une fois validé, le modèle entraîné peut enfin être appliqué à l'ensemble du jeu de données pour pointer les temps d'arrivées et relocaliser les évènements microsismiques. Les résultats obtenus sont prometteurs en termes de précision et d'estimation des erreurs de pointé.

Dans le cadre du projet GEOTREF, j'ai collaboré avec une étudiante du Laboratoire de Géologie de l'ENS Paris et un ingénieur de l'ENSG sur l'étude des propriétés géomécaniques d'un échantillon d'andésite issu de la Guadeloupe. Mon travail consistait à relocaliser les émissions acoustiques générées à l'aide d'une cellule triaxiale. Le logiciel décrit dans ce chapitre a été utilisé dans ce cadre et est notamment disponible en libre accès sur GitHub :

- AIPycker : module orienté objet pour le pointé automatique des phases à l'aide d'un réseau de neurones multi-attributs. Il utilise les librairies NumPy, SciPy, ObsPy, Pandas, Matplotlib, Scikit-learn et StochOPy (cf. Chapitre 2). Il comprend notamment une interface utilisateur graphique pour le pointé manuel et l'entraînement d'un réseau de neurones par l'intermédiaire d'un assistant. Le module est à un stage préléminaire de son développement et la méthodologie peut encore être améliorée. Cependent, AIPycker est plus facile à utiliser et a montré des résultats supérieurs à ceux obtenus à l'aide du logiciel commercial utilisé au Laboratoire de Géologie de l'ENS Paris. Disponible à l'adresse github.com/keurfonluu/AIPycker.

Les résultats issus de cette collaboration ont aussi fait l'objet de deux présentations orales en conférences :

- Zhi Li, **Keurfon Luu**, Aurélien Nicolas, Jérôme Fortin and Yves Guéguen, 2016. "Fluid-induced rupture on heat-treated andesite." *4th International Workshop on Rock Physics*;

- François Bonneau, **Keurfon Luu**, Aurélien Nicolas and Zhi Li, 2017. "Toward an understanding of the relationship between fracturing process and microseismic activity: study at the laboratory scale." *2017 RING Meeting*.

Abstract

Phase onset picking is a key step in microseismic monitoring as accurate traveltimes are required to reliably locate microseismic events induced by the fracturing process. We introduce the package AIPycker to automatically identify and pick seismic trace phase onsets using a multi-attributes based neural network. AIPycker is written in Python 3 and consists of several object-oriented modules that correspond to each step of the processing workflow. It depends on several popular Python packages such as NumPy, ObsPy and Scikit-learn. Optimal attributes

and their parameters can be easily selected using a scatter-plot matrix. Then, a neural network is trained using either a derivative-based optimizer or an evolutionary algorithm. We use the probability map output by a neural network model to predict a phase onset, assess its error or flag it as a false positive. We test the methodology and the numerical implementation of AIPycker by applying the workflow on a real data set acquired at the laboratory scale. The predicted phase onsets are finally validated by relocating the events with results consistent with the observations for this type of experiment. Attributes selection, neural network training and prediction on unknown data set can all be performed either using the available modules or through an intuitive Graphical User Interface.

Keywords: microseismic, automated phase picking, neural network, signal processing

5.1 Introduction

Microseismic monitoring is one of the main tools to estimate hydraulic fracture geometry and failure mode. It consists in detecting and locating very small events induced by the fracturing process (Cipolla *et al.* (2011)). The location problem is usually solved by first arrival traveltime inversion (Lomax, Michelini and Curtis (2009), Maxwell (2009)), therefore accurate arrival times are required to avoid misinterpretation of resulting hypocenter locations. Nowadays seismic networks can produce very large volumes of data, thus manual arrival time picking is not feasible in real-time monitoring.

In the last decades, many algorithms have been proposed to automatically identify and pick phase onsets. Most of these methods are based on the computation of a characteristic function that acts as an energy detector either by using sliding windows such as short-time/long-time average ratio (Allen (1982), Baer and Kradolfer (1987)), modified Coppen's method (Sabbione and Velis (2010)), using higher order statistics (Saragiotis, Hadjileontiadis and Panas (2002), Küperkoch *et al.* (2010), Baillard *et al.* (2014)), waveform cross-correlation (Molyneux and Schmitt (1999), De Meersman, Kendall and Baan (2009)), or auto-regressive methods like Akaike Information Criterion (Maeda (1985), Sleeman and Eck (1999)). An extensive review of many algorithms and their parameters has been carried out by Akram and Eaton (2016).

Another approach to automated phase onset picking is to adopt a data-driven learning scheme using a neural network where picking is treated as a pattern recognition problem. The use of neural networks for phase onset picking is not novel and have been applied since the early nineties (Veezhinathan and Wagner (1990), McCormack, Zaucha and Dushek (1993), Dai and MacBeth (1997)), yet they only started to draw attention recently within the geophysical community with the advent of machine learning and artificial intelligence that we have witnessed in the last years. Veezhinathan and Wagner (1990) first described a first-break picking method with a selection of attributes. Murat and Rudman (1992) introduced a multi-attributes neural network to determine whether each half-cycle of a seismic trace is a phase onset with false positives rejection based on a confidence level. McCormack, Zaucha and Dushek (1993) fed a neural network using a rasterized seismic trace as input. Dai and MacBeth (1997) used the absolute value of the full waveform with a sliding window of 40 samples to feed a multi-layer perceptron. Hart (2003) demonstrated the importance of using the cross-entropy cost function to be able to interpret the output of a neural network in terms of probability. Gentili and Michelini (2006) trained a neural tree that adapts its network architecture for P- and S- phase picking. More recently, Maity, Aminzadeh and Karrenbach (2014) showed the robustness of an automated neural network onset picker on signals with low signal-to-noise ratio, and Akram, Ovcharenko and Peter (2017) applied the concept of weight-based saliency to select an optimal model. Nevertheless, no automated phase onset picking software based on neural network has been

made available so far.

In this paper, we introduce an Application Programming Interface (API) as a package, AIPycker, written in Python 3 for neural network automated phase onset picking. Its core relies on popular Python packages such as NumPy and SciPy for fast numerical computations, ObsPy for reading seismic traces, Pandas for output data handling, and Matplotlib and Seaborn for graphical outputs (Jones, Oliphant and Peterson (2001), Oliphant (2006), Hunter (2007), Beyreuther *et al.* (2010), McKinney (2010)). It also interfaces with Scikit-learn (Pedregosa *et al.* (2012)) as well as a custom package for neural network classification. Throughout the development of AIPycker, we emphasized on the runtime performance, usability and consistency by writing user-friendly pythonic functions with detailed documentations. AIPycker includes both object-oriented modules that can be used for scripting and a Graphical User Interface (GUI) for manual onset picking and neural network training. In the following, we first briefly describe the concept of (artificial) neural network and present several useful attributes. Then, we explain and validate the methodology by using AIPycker on a real data set acquired at the laboratory scale.

5.2 Description

AIPycker is a multi-attributes phase onset automated picker using a neural network that consists of several modules corresponding to each step of the workflow presented in Section 5.3. In this section, we mainly describe the principle of neural networks and the three attributes that will be used throughout this paper.

5.2.1 Artificial neural network

An artificial neural network is an ensemble of interconnected processing units (called neurons) that return a response given an input signal. They are usually organized in layers with at least an input layer and an output layer. There may be one or more intermediate layers – called hidden layers – between the input and the output layers. The number of hidden layers and their structures are to be specified by the user depending on the desired level of complexity between the input features and the outputs. Each neuron is passed through an activation function to produce an output signal that serves as an input feature for the neurons in the next layer. Neural networks classify data by fitting a decision boundary that correctly separates the trained class from the other classes.

Onset picking is a binary classification problem where the samples of a given seismic trace are either labeled phase onset (class 1) or not phase onset (class 0). For the sake of simplicity, we refer to all the samples that are not phase onsets as noise samples. Most automated phase onset pickers are based on the computation of an attribute for each sample with the onset determined by an extremum of the attribute function. This kind of onset pickers can actually be seen as a simple neural network with a single input feature, no hidden layer, and activated by a linear function as shown in Figure 5.1 (left). Therefore, several attributes can naturally be fed into a neural network to train an automated phase onset picker. We display in Figure 5.1 (right) an example of neural network based picker with four attributes as input features, one hidden layer and one output.

The memory of a neural network is stored in the neural weights that are simultaneously optimized (i.e. trained) by minimizing the cross-entropy cost function (Nielsen (2015)) defined as

$$E = -\sum_{i=1}^{m} y_i \log\left(h_\theta\left(x_i\right)\right) + \left(1 - y_i\right) \log\left(1 - h_\theta\left(x_i\right)\right) \tag{5.1}$$

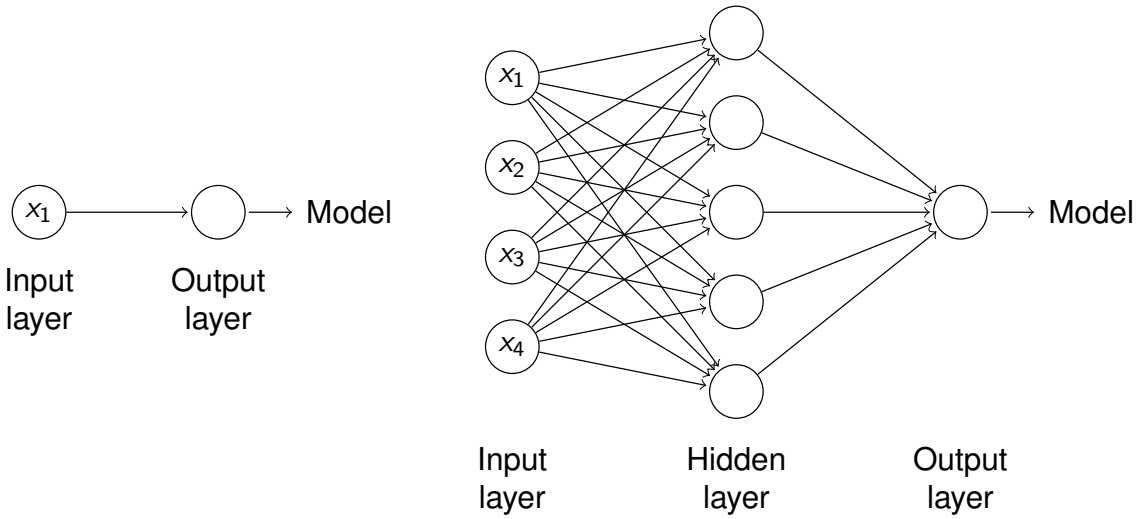

Figure 5.1: (Left) Attribute based automated picker seen as a neural network. (Right) Example of multi-attributes onset picker based on a neural network with four input features, one hidden layer and one output.

where m is the number of input data x_i, $y_i \in \{0, 1\}$ is its corresponding label (i.e. class), and h_θ denotes the activation function. The activation function is applied to determine the output response of a neural network by mapping the values between specific ranges using linear or non-linear functions. Neural networks are optimization problems that can thus be solved using either derivative-based algorithms with the gradient computed by the backpropagation method, or derivative-free algorithms. In our package AIPycker, a neural network can be trained using derivative-based methods such as L-BFGS or stochastic gradient as we provide compatibility with the popular Scikit-learn classifiers, it can also be trained using evolutionary algorithms such as Differential Evolution (Storn and Price (1997)), Competitive Particle Swarm Optimization (Luu *et al.* (2018)) or Covariance Matrix Adaptation - Evolution Strategy (Hansen and Ostermeier (1996)) through a custom neural network classifier derived from Scikit-learn.

5.2.2 Attributes

In this section, we present three attributes that can be applied to discriminate phase onsets from noise samples. It is important to note that we do not claim these attributes to be the most suitable attributes for neural network automated phase onset picking. However, these attributes are easy to use as they only require the user to specify the sliding window length and have demonstrated good results on different data sets. Many other attributes are available in the literature and can be easily implemented by the user with few modifications to the codes. The attributes presented in this section are already available in the initial release of AIPycker. In the following, a seismic trace is represented by a time serie $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ of length n, k represents a sample index, and $\mathbf{x}_{i:j}$ denotes a slice of the time serie from sample i to sample j.

Signal-to-noise ratio

The signal-to-noise ratio (SNR) is an attribute that acts as an energy detector. It is a modification of the energy ratio with the noise window at sample k starting from the first sample of the seismic

trace, following

$$SNR(k) = \frac{RMS\left(\mathbf{x}_{k:k+\Delta t}\right)}{RMS\left(\mathbf{x}_{1:k}\right)} \tag{5.2}$$

where Δt is a time window and RMS is the root-mean-square value defined as

$$RMS\left(\mathbf{x}\right) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}. \tag{5.3}$$

Figure 5.2 shows the SNR attribute for an example trace. As the SNR attribute is computed as the ratio of a postsample and presample windows, the phase onset is characterized by the index of the maximum of the function.

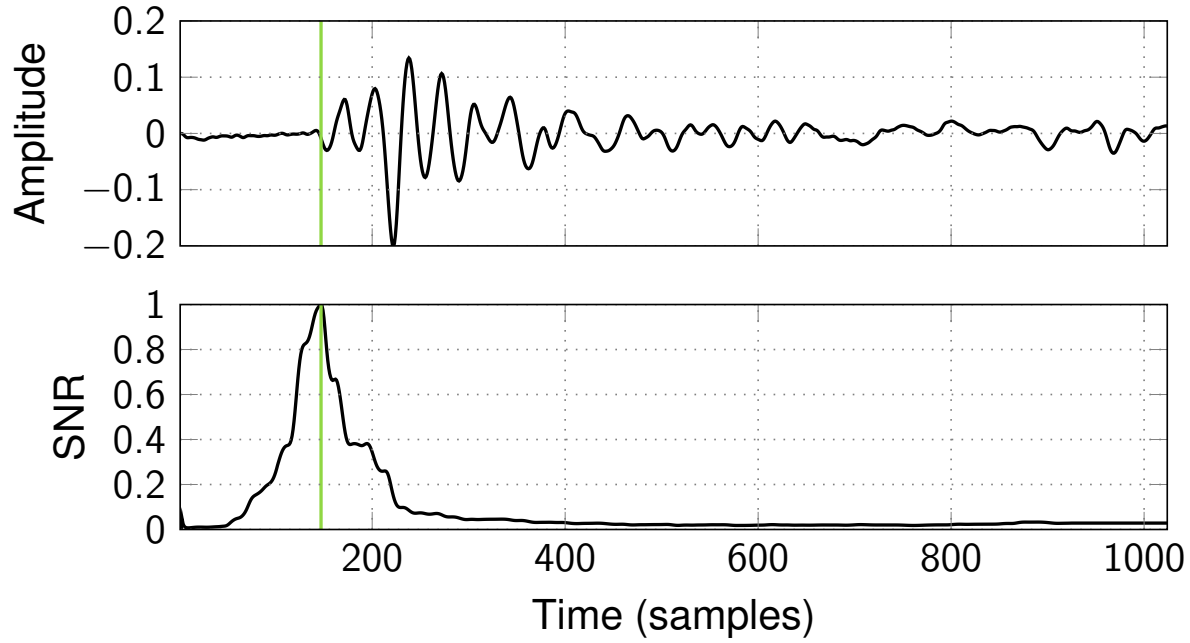

Figure 5.2: SNR attribute. (Top) Example trace. (Bottom) SNR attribute with $\Delta t = 50$ samples. The vertical line corresponds to the phase onset given by the global maximum of the SNR attribute. Attribute values are normalized.

Akaike Information Criterion

Onset picking based on the Akaike Information Criterion (AIC, Akaike (1974)) has been used in many picking algorithms (Leonard (2000) Zhang, Thurber and Rowe (2003)). The AIC function is based on the principle that seismic signals are non-stationary and can be divided into two locally stationary time series each fitted by an auto-regressive model (Sleeman and Eck (1999)). The AIC function computed using the estimated auto-regressive model order provides the fitness of the model. The index that yields the global minimum of the AIC function indicates the optimal onset that separates the seismic trace into noise and signal time series. Maeda (1985) estimated the AIC function directly from the seismic trace without using auto-regressive models following

$$AIC(k) = k\log\left(\mathrm{var}\left(\mathbf{x}_{1:k}\right)\right) + (n-k-1)\log\left(\mathrm{var}\left(\mathbf{x}_{k+1:n}\right)\right) \tag{5.4}$$

with k ranging through all the samples of the seismic trace. As the phase onset corresponds to the global minimum of the AIC function, the phase is difficult to identify if the seismic trace contains several arrivals. Therefore, a time window for which the AIC function will be estimated

must be specified in order to correctly identify the first-break onset (Zhang, Thurber and Rowe (2003)). Ideally, the time window should span from the noise part to the signal part of the seismic trace. Following Sedlak *et al.* (2008), we estimate the AIC function on a time window defined by $k \in [1, t + \Delta t]$ where t corresponds to the time that yields the maximum absolute amplitude of the signal (i.e. $x(t) = \max(|\mathbf{x}|)$) and Δt defines the time delay from t. Basically, we expect the beginning of the trace to be non-informative which is thus unchanged. On the other hand, $t + \Delta t$ should correspond to the end of the informative part of the seismic trace which removes undesired local minima in the AIC function. Figure 5.3 shows the AIC and the windowed AIC (AIC-W) functions for the example seismic trace. The first-break onset is a local minimum of the AIC function while it corresponds to the global minimum of the AIC-W function.

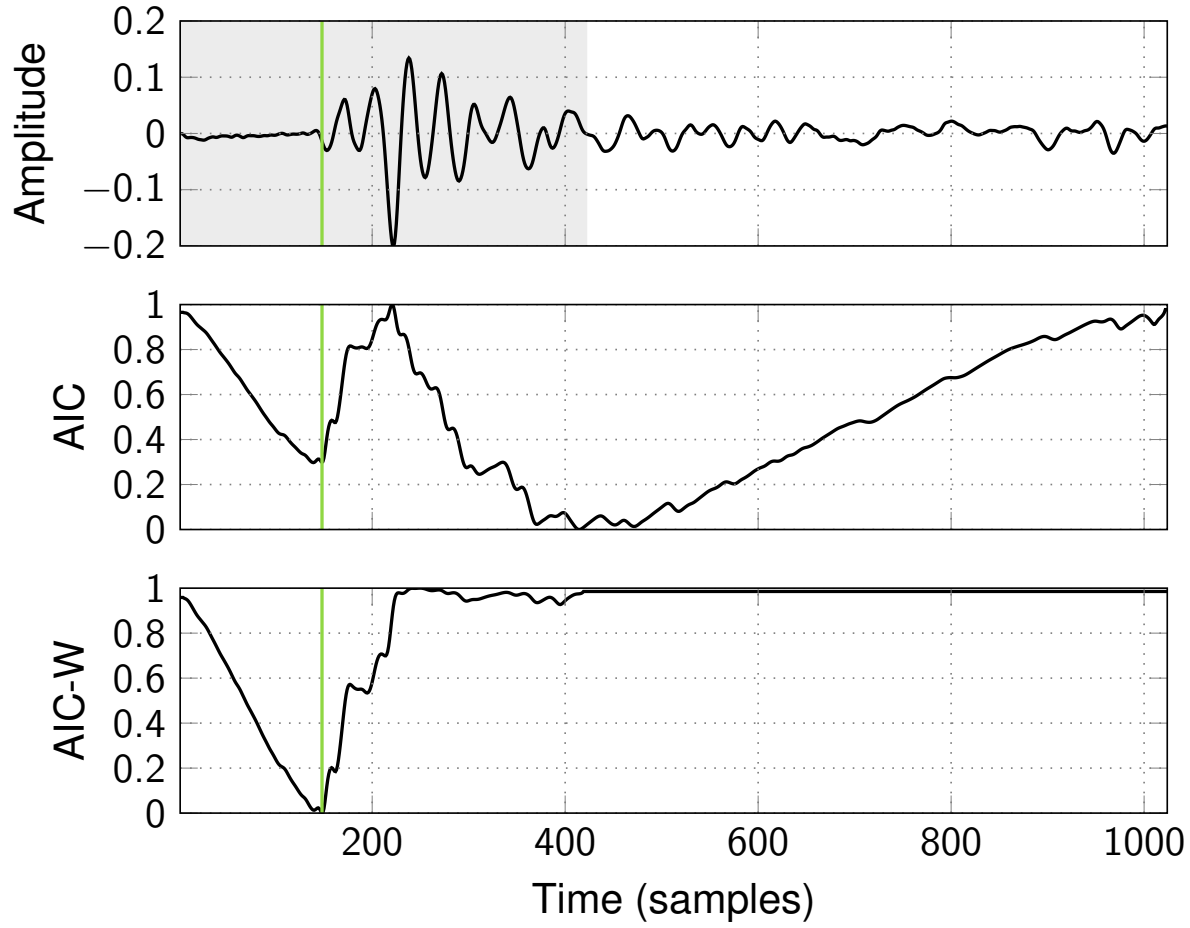

Figure 5.3: AIC attribute function. (Top) Example trace. The shaded area indicates the time window with Δt = 200 samples. (Middle) AIC function. (Bottom) Windowed AIC function with $\Delta t = 200$ samples. The vertical line corresponds to the phase onset given by the global minimum of the AIC-W function. Attribute values are normalized.

Kurtosis

The kurtosis is a higher-order statistics that has been applied to identify phase onsets (Saragiotis, Hadjileontiadis and Panas (2002), Küperkoch *et al.* (2010), Baillard *et al.* (2014)). In statistics, the kurtosis is a measure of the heaviness of the tails of a distribution and is zero for a gaussian distribution. Seismic waves generate non-gaussian wavefields presenting high values that appear in the tails of the distributions. Therefore, the kurtosis rapidly increases in the presence of seismic signal and can be used to accurately pick phase onsets. We follow Baillard *et al.*

(2014) and calculate the kurtosis over a sliding window of Δt samples following

$$F_1(k) = \frac{\hat{m_4}(k)}{\hat{m_2}(k)^2} - 3 \tag{5.5}$$

in which $\hat{m_d}$ is the central statistic moment of order d defined as

$$\hat{m_d}(k) = \frac{1}{\Delta t} \sum_{i=1}^{\Delta t} (x_{k-i+1} - \bar{x_k})^d \tag{5.6}$$

with $\bar{x_k}$ the mean of the time window $\mathbf{x}_{k-\Delta t+1:k}$.

Baillard *et al.* (2014) applied a succession of transformations to the initial characteristic function F_1 to identify the strongest onsets. The first transformation consists in removing all the negative slopes that represent the transition from coherent signal to noise, which is written

$$F_2(k+1) = F_2(k) + \delta(k) \times dF_1(k), \quad \text{with} \quad \begin{cases} F_2(1) & = F_1(1) \\ dF_1(k) & = F_1(k+1) - F_1(k) \\ \delta(k) & = 1, \quad \text{if } dF_1(k) \geq 0 \\ \delta(k) & = 0, \quad \text{otherwise} \end{cases}. \tag{5.7}$$

We do not apply the next transformations suggested as the final characteristic function presents narrow peaks. Instead, we integrate F_2 with a sliding window and compute the squared variance. The Kurtosis attribute is defined as

$$\text{Kurtosis}(k) = \text{var}\left(\mathbf{F_2}_{k-\frac{\Delta t}{2}:k+\frac{\Delta t}{2}}\right)^2. \tag{5.8}$$

We display in Figure 5.4 the different transformations to obtain the Kurtosis attribute for the example seismic trace. The first-break onset corresponds to the global maximum of the attribute.

5.3 Methodology

In this section, we describe the general workflow to train a neural network for automated phase onset picking by applying the methodology on a real data set. The workflow is summarized in Figure 5.5. The study of the best attributes, network architecture or input training hyperparameters is beyond the scope of this paper as we believe that it mostly depends on the data set. The values of all these parameters can be controlled and are left to the user in our package.

5.3.1 Real data set

Synthetic data sets with gaussian noise usually do not exhibit enough discrepancies between traces to validate the robustness of an automated picking algorithm. We make the original choice to test and validate our implementation by applying the methodology on a real data set acquired at the laboratory scale.

The acquisition has been performed on a cylindric sample of andesite from la Guadeloupe using a conventional tri-axial cell installed in the Laboratoire de Géologie at the ENS Paris (Brantut, Schubnel and Guéguen (2011), Nicolas *et al.* (2016)). The sample has a diameter and length of 4 cm and 8 cm, respectively. Sixteen piezoelectric transducers (PZT) with a sampling rate of 10 MHz are directly glued onto the sample surface. Twelve PZTs are sensitive to P-waves and four

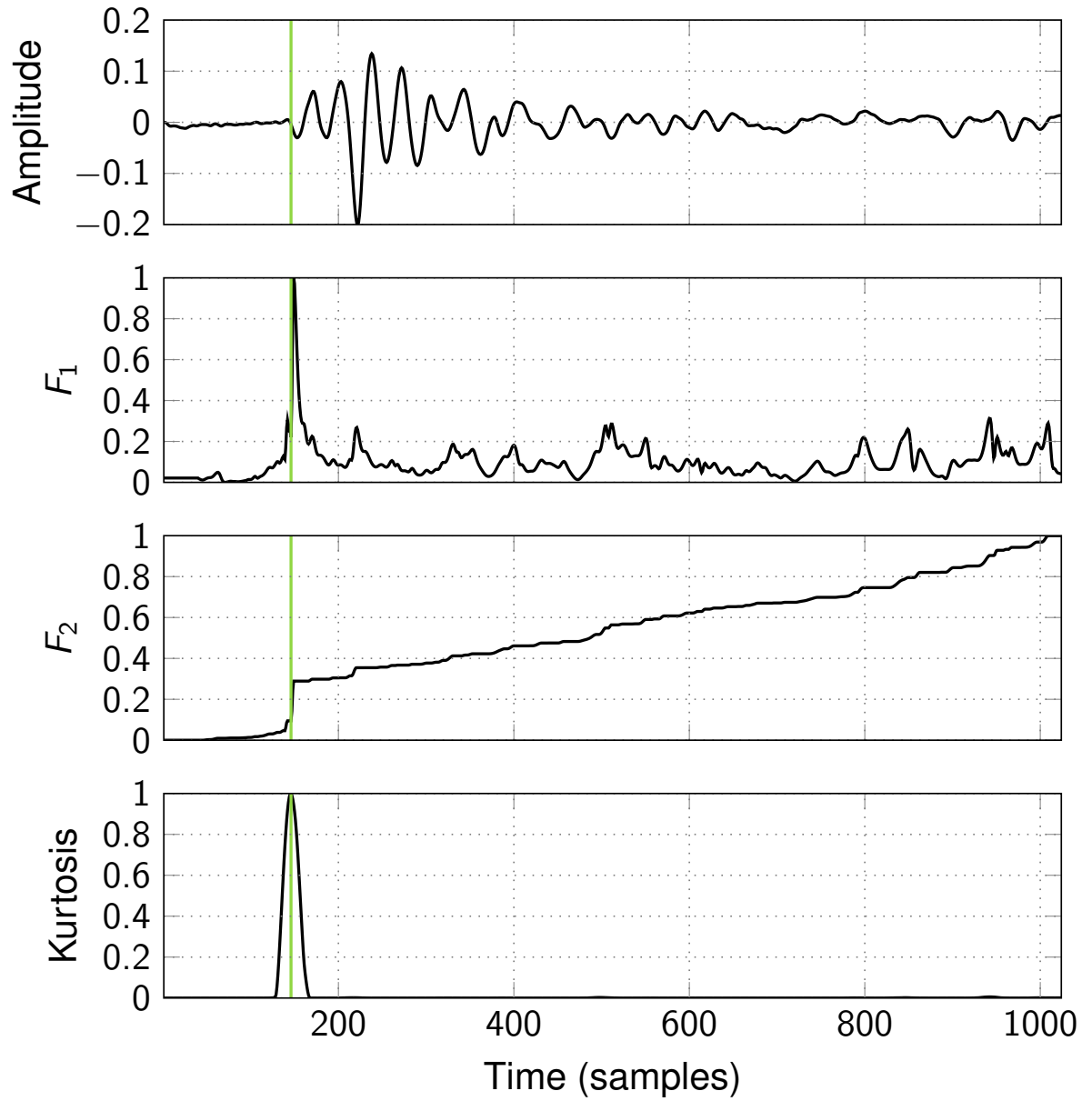

Figure 5.4: Kurtosis attribute function. (Top) Example trace. (Middle) Kurtosis statistics F_1 and removal of negative slopes F_2. (Bottom) Kurtosis attribute with $\Delta t = 40$ samples. The vertical line corresponds to the phase onset given by the global maximum of the Kurtosis attribute. Attribute values are normalized.

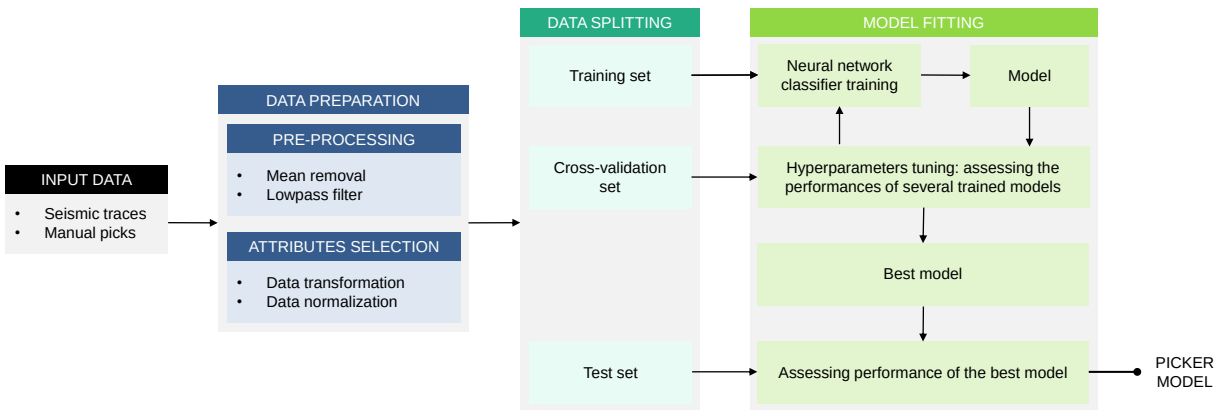

Figure 5.5: Neural network automated phase onset picking workflow.

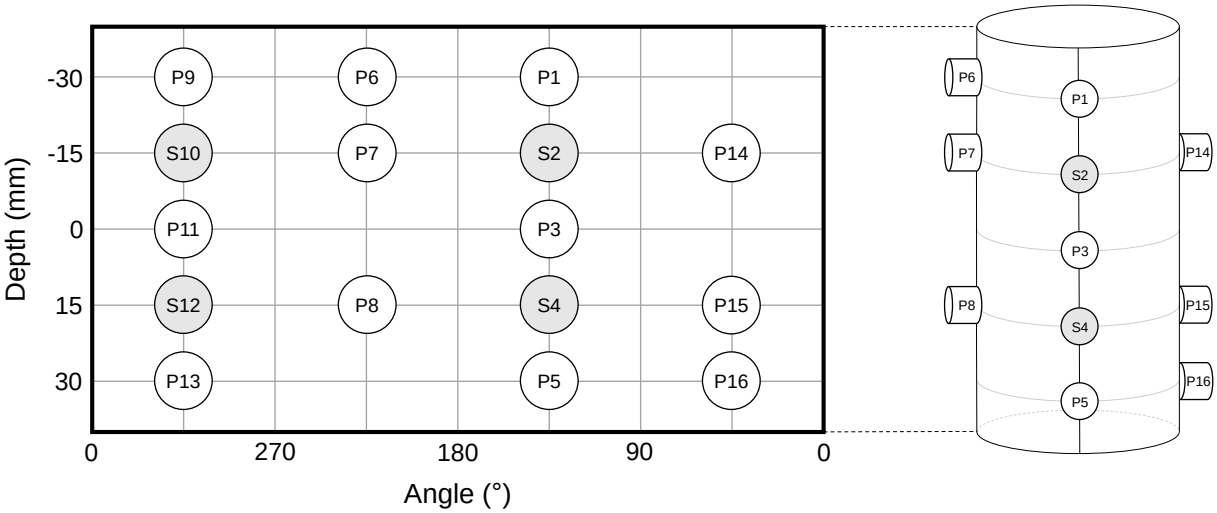

Figure 5.6: The acquisition geometry consists of sixteen piezoelectric transducers.

PZTs to S-waves. However, all the S-waves and 1 P-wave receivers were not working properly. Events are automatically detected based on an amplitude threshold that triggers the recording. The acquisition geometry used for the experiment is sketched in Figure 5.6.

The whole data set consists of 1145 events detected for a total of 18320 seismic traces to pick. We pre-process each seismic trace by removing the mean and applying a lowpass filter with a cut-off frequency of 1 MHz to remove the noise. The snippet below shows an example code to pre-process the data using AIPycker with `X` refering to the input data array.

```
preprocess = [
    PreProcessor("demean"),
    PreProcessor("lowpass", sampling_rate = 1e7, cutoff = 1e6),
    ]
for p in preprocess:
    X = p.transform(X)
```

We manually picked the first-break onsets for 300 seismic traces with different signal-to-noise ratios in order to build our input data set. Figure 5.7 shows a sample data for one event.

The data set is finally split into three subsets:

- the training set (150 traces) to model output values;
- the cross-validation set (75 traces) to tune the input training hyperparameters so as to obtain an unbiased model;
- the test set (75 traces) to assess the performance of the final selected model by measuring the error between the true and the predicted classes.

Note that we arbitrarily manually picked 300 seismic traces. More or fewer manual picks may be sufficient for the training depending on the data set to predict.

5.3.2 Attributes selection

Feature extraction and selection is a key and common step in machine learning and consists in determining a subset of features that have a predictive value. In neural network automated

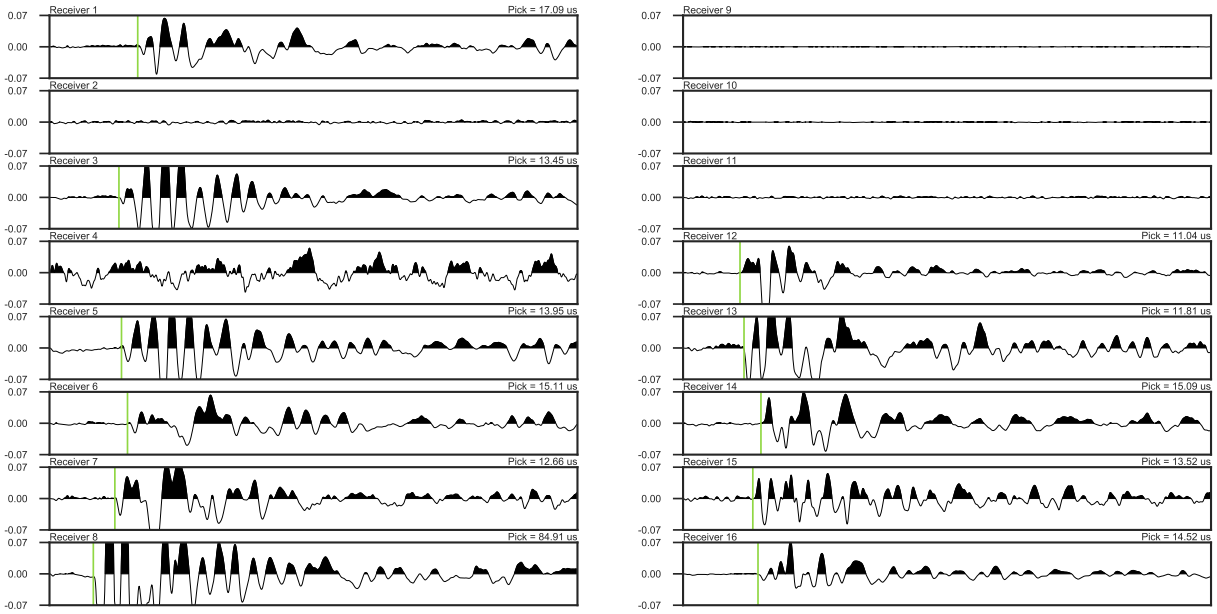

Figure 5.7: Sample data for one event. Receivers 2, 4, 9, 10 and 11 were not working properly. The vertical lines indicate the manual picks.

onset picking, the features are the attribute function values for a given sample. Even though an attribute that is not predictive should be automatically weighted out during the training, it is important from a computational point of view to only select predictive attributes to help the training and avoid unnecessary computation of non-predictive attributes during the prediction.

The training set is composed of 150 seismic traces with various signal-to-noise ratios so as to train a model capable of predicting the phase onset on any seismic trace. In order to account for all the possible signal types in the training, we also randomly pick a sample different from the manually picked phase onset for each trace of the training set in order to characterize noise samples.

Attributes and their parameters (such as the window length) must be selected so that their values can accurately distinguish a first-break onset from a noise sample. This task can be done visually by displaying the distributions of the attribute values for the two classes (phase onset and noise sample) through a scatter-plot matrix. Scatter plots are useful to visualize the distributions and correlations between each pair of features. In the following, we select the AIC-W, the Kurtosis and the SNR attributes as input features of the neural network, respectively using window lengths of 200, 40 and 50 samples. The attributes for the whole data set are then independently normalized following

$$\mathbf{x}_{\text{norm}} = \frac{\mathbf{x} - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \tag{5.9}$$

where $\mathbf{x}$ is the attribute vector to normalize, x_{max} and x_{min} its maximum and minimum values, respectively. We show in Figure 5.8 a scatter-plot matrix for the three attributes applied to the training set. The distributions of the phase onsets and the noise samples for the selected attributes do not overlap significantly, which means that they are predictive for this data set.

In AIPycker, the user can specify the attributes to use by declaring a list of `Attribute` objects with their window lengths `n_len` (in samples). An attribute-features extractor can finally be initialized by passing the list to an object `TraceFeatures`. Attributes and their parameters can be selected by displaying a scatter-plot matrix simply calling the method `scatter_matrix`.

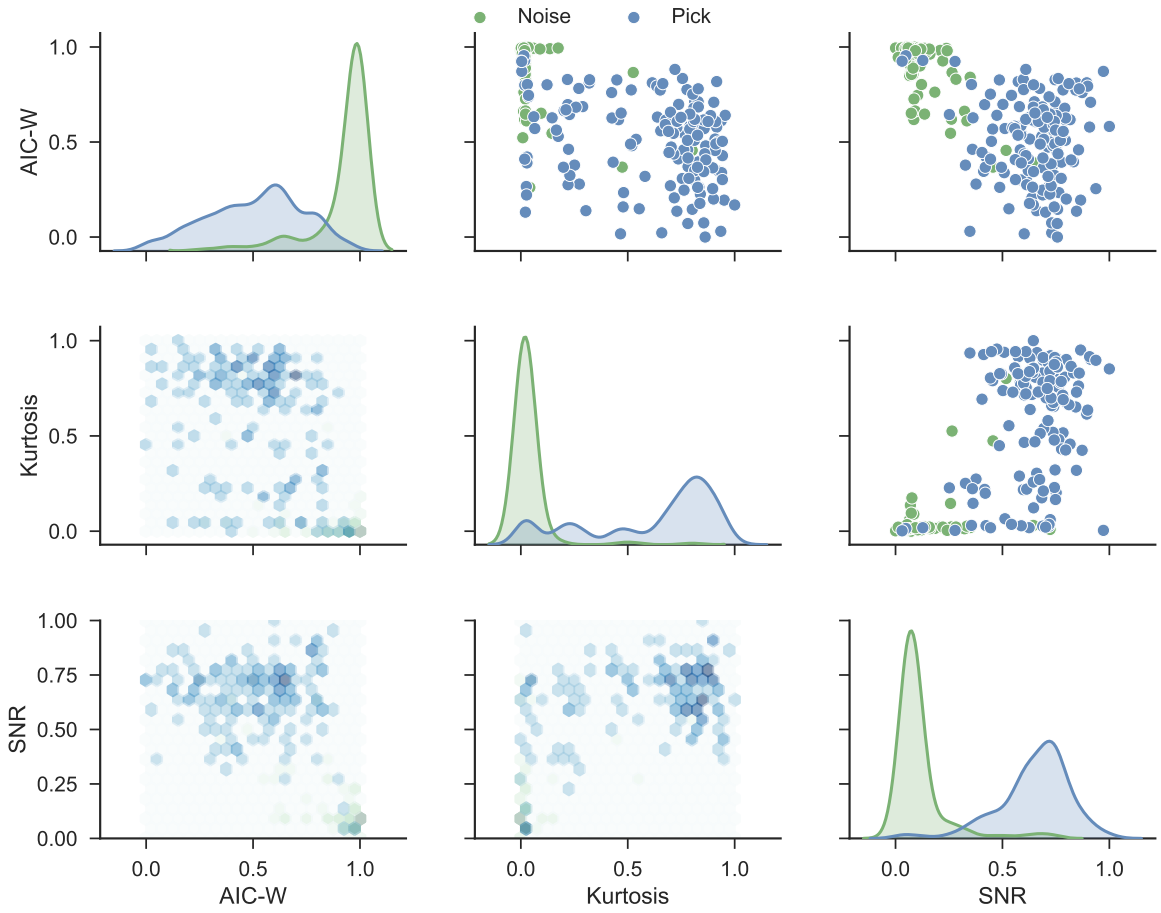

Figure 5.8: Scatter-plot matrix for AIC-W, Kurtosis and SNR. The diagonal shows the KDE plots for each individual attribute, the lower and upper off-diagonals respectively display the pairwise hexagonal binning plots and scatter plots of the attributes.

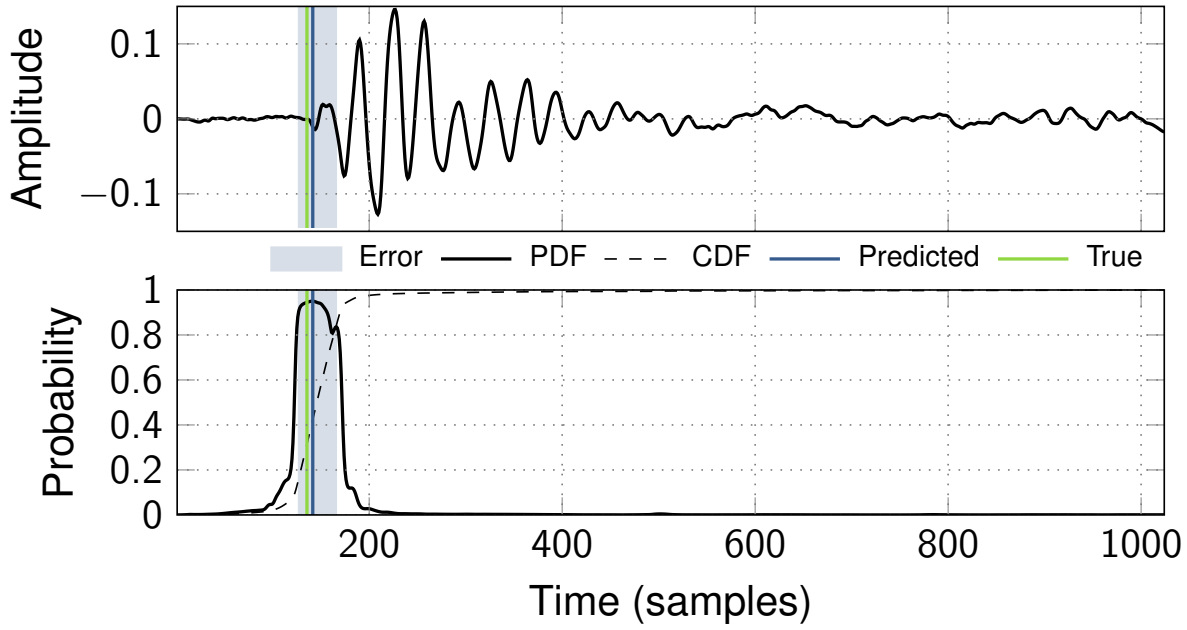

Figure 5.9: (Top) Example trace. (Bottom) Predicted probability map. The manually picked and predicted phase onsets are indicated by the green and blue vertical lines, respectively. The prediction error is shown in blue shade.

```
features = [
    Attribute("aicw", n_len = 200),
    Attribute("kurtosis", n_len = 40),
    Attribute("snr", n_len = 50),
    ]
tf = TraceFeatures(features)
tf.transform(X, y)
tf.scatter_matrix()
```

5.3.3 Training

Once the attributes and their parameters have been selected, a neural network can be trained by optimizing the neural weights to minimize the cross-entropy cost function (Equation (5.1). Our neural network is composed of three input features and a single hidden layer of 5 neurons, and is trained using an evolutionary algorithm (CPSO with a population size of 50 and 500 iterations) with hyperbolic tangent as activation function (LeCun *et al.* (1998)). The trained model maps the input attributes to an output probability (between 0 and 1) to belong to a given class. In other words, for each seismic trace, the output probability map can be used to determine whether a sample is a noise sample or a phase onset, and assess the prediction error in the latter case. We simply define the phase onset as the sample that yields the maximum probability. A predicted phase onset is flagged as false positive (i.e. rejected) if its probability is lower than a threshold specified by the user. In order to assess the prediction error, we compute the cumulative function (CDF) of the probability map and define the error as the half-distance between the samples that yield 0.16 and 0.84 on the CDF (similarly to one standard deviation for a gaussian distribution).

We evaluate the performance of a model using the root-mean-square metric following

$$RMS\left(\mathbf{t}^{pred}\right) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(t_i^{man} - t_i^{pred}\right)^2} \tag{5.10}$$

where $\mathbf{t}^{pred} = \left\{t_1^{pred}, ..., t_N^{pred}\right\}$ is a vector containing the phase onsets predicted by the model trained, and $\mathbf{t}^{man} = \{t_1^{man}, ..., t_N^{man}\}$ is the target vector of manual picks. The hyperparameters of the neural network (such as its architecture) should be tuned to achieve a low RMS on the cross-validation set only. The performance of the chosen model can finally be assessed on the test set.

As we previously mentioned, AIPycker is compatible with Scikit-learn neural network classifier `MLPClassifier`. We also provide a custom neural network classifier that can be trained using an evolutionary algorithm. A phase onset picker is initialized by passing both the classifier and the list of attributes to an object `AIPicker` and trained using the method `fit`. The model performance can be measured by the RMS metric (Equation (5.10)) with the method `score`.

```
clf = MLPClassifier(hidden_layer_sizes = (5,),
                    max_iter = 500,
                    solver = "lbfgs",
                    activation = "tanh")
picker = AIPicker(classifier = clf, features = features)
picker.fit(X, y)
picker.score(X, y)
```

In this code snippet, `y` is an array of manual pick indices. Note that any classifier that has a `predict_proba` method can be used as input classifier of `AIPicker`.

5.3.4 Skewing the training set

In neural network based phase onset picking, a single noise sample is usually picked (here randomly) for each seismic trace in the training set so as to account for all types of signal. However, such a balanced data set (1 noise sample for 1 phase onset) is not representative of a seismic trace. Indeed, a seismic trace contains very few phase onsets, every other samples can be considered as noise. Therefore, we recommend to purposely skew the training set by introducing more noise samples than phase onsets. In addition, we duplicate the phase onsets to rebalance the data set in order to avoid overfitting of the major class (i.e. noise samples) during the training. To demonstrate the influence of the number of noise samples on the training, we train 20 neural networks for different numbers of noise samples using two attributes only, namely the Kurtosis and the SNR. Figure 5.10 displays the mean decision boundaries in the attribute space along with their standard deviations in gray shade. The lower standard deviations in the last two plots shows that increasing the number of noise samples stabilizes the decision boundary, and thus helps constraining the training of a neural network. On the other hand, this procedure also increases the training time as we artificially increased the size of the training set. We empirically found that a ratio of 9 noise samples for 1 manual pick is a good compromise between decision boundary constraining and training time.

Note that we used a single hidden layer with 5 neurons, more complex decision boundaries (convex or closed) can be obtained by adding more hidden layers and/or neurons.

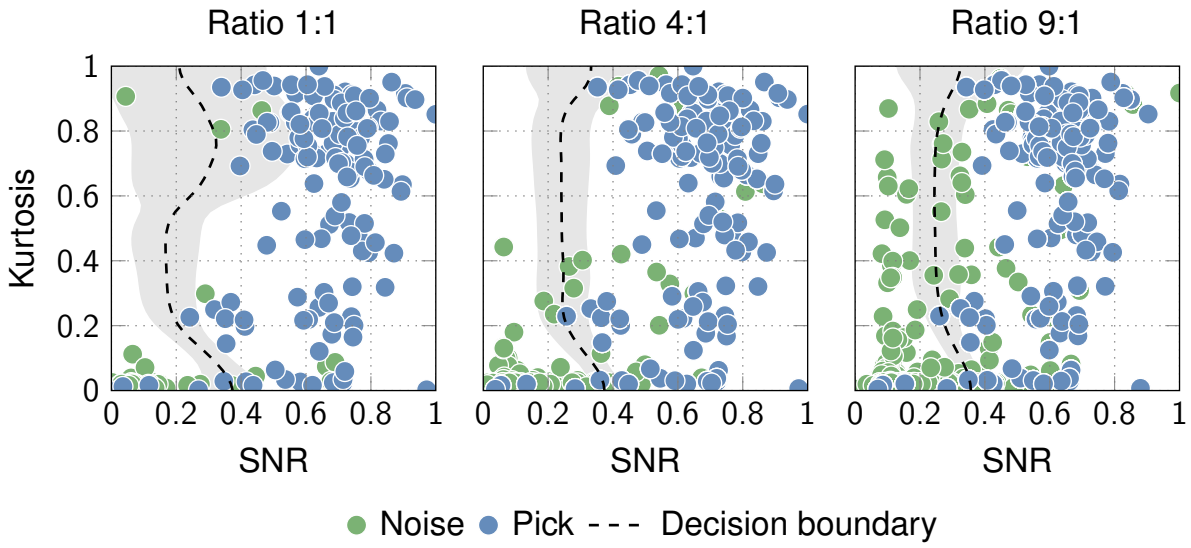

Figure 5.10: Influence of the number of noise samples on the decision boundary (black). The standard deviations are represented in gray shade.

5.3.5 Prediction

The trained neural network can finally be used to predict the phase onsets on the whole data set. We set the threshold at a sufficiently high level (0.8) to reject false positives. We show on Figure 5.11 the result of the prediction for one event that has not been manually picked (i.e. not used for the training). All the predictions are accurate and picking uncertainties are consistent with the waveform. We should note that the five seismic traces associated to the non-working PZTs have not been picked which demonstrates the ability of our neural network automated onset picker to reject false positives.

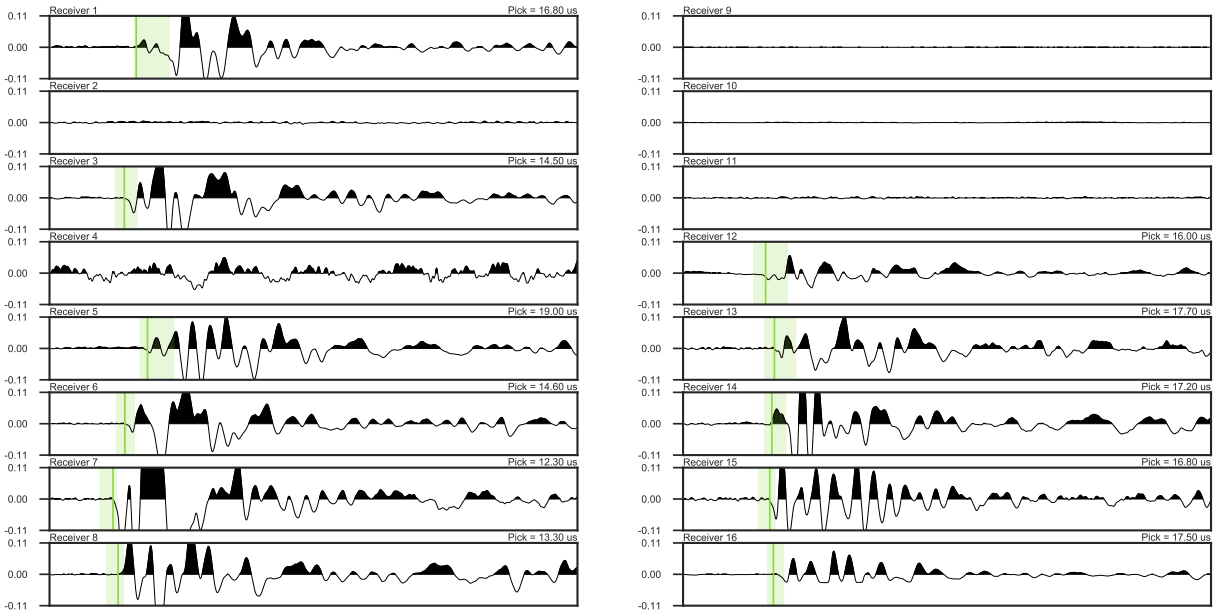

Figure 5.11: Prediction of phase onsets for one event. The vertical lines indicate the predicted picks along with the picking errors in green shade. The seismic traces recorded by receivers 2, 4, 9, 10 and 11 have been rejected by the trained neural network.

In AIPycker, prediction on unknown data set can simply be performed by using the method `transform` of an already trained picker model `AIPicker`.

```
picks = picker.transform(X)
```

To validate the phase onsets predicted by our picker, we relocate the acoustic emissions recorded during the failure of the rock sample. The evolution of the velocity within the rock sample is measured using an active seismic acquisition. Every 5 minutes during the experiment, a 250 V high frequency signal is pulsed on each PZT while the other receivers are recording. Arrival times are known with an accuracy of 0.1 microseconds, which leads to an accuracy on velocity of the order of 5 per cent (Figure 5.12 (left)). Location is a non-linear inverse problem consisting in minimizing the misfit function

$$E(\mathbf{m}) = \frac{1}{2}\sum_{i=1}^{n} \frac{\left(t_i^{obs} - t_i^{calc}(\mathbf{m})\right)^2}{\sigma_i^2} \tag{5.11}$$

where $\mathbf{m} = \{x, y, z, t_0\}$ represents the hypocenter parameters, $n = 16$ is the number of receivers, t_i^{obs} and $t_i^{calc}(\mathbf{m})$ denote respectively the observed (i.e. picked) and calculated traveltimes at receiver i, and σ_i is the associated picking error. For each receiver, we compute accurate traveltime grids using an Eikonal solver (Noble, Gesret and Belayouni (2014)), and relocate the acoustic emissions using an evolutionary optimizer (Differential Evolution with population size of 10 and 100 iterations). Results of the location are shown in Figure 5.12 (right) with the color scale indicating the relative origin time. Most of the events are localized along two fracture plans which is consistent with what is usually observed in this type of experiment. The origin times of the events indicate how the two plans were formed. The fracture starts from the lower left corner and propagates toward the upper right corner of the sample. An auxiliary plan is formed at the end of the experiment starting from the lower right to the upper left corner of the sample.

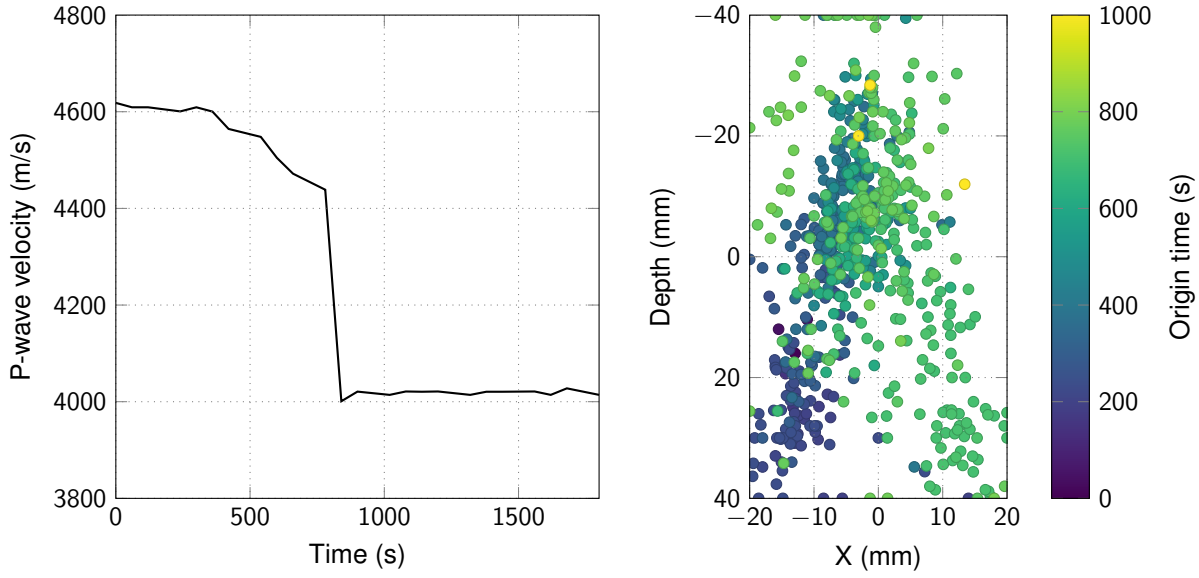

Figure 5.12: (Left) Evolution of the acoustic wave velocity during the experiment. (Right) Acoustic event locations. The color scale indicates the relative origin time.

5.4 Conclusion

Phase onset picking is a common and important task encountered in microseismic monitoring as it can be used for earthquake location or traveltime tomography. Even though neural network has been used for automated picking since the early nineties, it has only recently gained in popularity within the geophysical community with the advent of artificial intelligence. AIPycker has been designed to provide a unified framework for automated phase onset picking based on a neural network. Given a data set, optimal attributes and their parameters (e.g. window length) can easily be selected based on a scatter-plot matrix. In this paper, we only presented three attributes that worked well for our example, but other attributes are available in AIPycker. We also proposed to purposely skew the data set to better constrain the training by introducing more noise samples than manual picks. In addition, the probability map outputs by a neural network model is used to predict a phase onset with its associated picking error, or to reject a pick if its probability is lower than a threshold specified by the user. We applied the workflow on a real data set acquired at the laboratory scale to validate the methodology and its numerical implementation in AIPycker. Even though we only showed results on a passive data set, it should be mentioned that AIPycker can be used in active seismic as well. The initial version of AIPycker only handles single channel data set (e.g. vertical component) and thus is only able to identify one phase. However, extension to 3-C data set is straightforward and will be introduced in a future update of AIPycker. The presented methodology can also be run through a Graphical User Interface (GUI) that allows the user to manually pick phase onsets, train a neural network picker following a training wizard, and predict on a data set using an already trained picker. However, the GUI is more limited in functionalities compared to the API (e.g. fewer hyperparameters to tune, no custom attributes).

Chapter 6

Conclusions and perspectives

Contents

Durant cette thèse, je me suis intéressé aux méthodes d'optimisation numérique dites évolutionnistes. Ces méthodes sont des approches stochastiques qui s'inspirent de la théorie de Darwin et le principe d'hérédité mendélienne. De manière analogue à l'évolution naturelle des espèces, les algorithmes évolutionnistes opèrent sur une population (de modèles) où les individus se reproduisent/coopèrent pour trouver la solution optimale d'un problème d'optimisation. La tomographie sismique des temps de première arrivée est un problème d'optimisation mal-posé et multi-modal du fait de la relation non-linéaire entre les données du problème (les temps d'arrivée) et les modèles physiques (le modèle de vitesse). Une autre difficulté rencontrée lors de la résolution de ce problème inverse est la non-unicité de la solution dans le sens où plusieurs modèles de vitesse peuvent expliquer les temps observés. Les méthodes de Monte Carlo par Chaînes de Markov sont le plus souvent employées pour résoudre les problèmes de non-linéarité et de non-unicité. Cependant, en tant qu'algorithmes séquentiels, ces méthodes ne sont pas capables de tirer partie de l'intégralité des ressources computationnelles fournies par les super-calculateurs modernes. Dans ce manuscrit, je me suis proposé de résoudre le problème de tomographie sismique à l'aide d'algorithmes évolutionnistes car intrinsèquement parallèles et donc très adaptés à l'architecture des super-calculateurs. Dans un second temps,

j'ai décrit un algorithme de pointé automatique des temps d'arrivée, nécessaires en tomographie sismique, à l'aide d'un réseau de neurones multi-attributs.

6.1 Conclusions

During this thesis, I have taken an interest in evolutionary numerical optimization methods. These methods are stochastic approaches inspired by Darwin's theory of evolution and Mendel's principles of inheritance. Analogously to natural evolution of species, evolutionary algorithms operate on a population (of models) where the individuals breed/cooperate to find the optimal solution for a given optimization problem. Seismic first arrival traveltime tomography is an ill-posed and multi-modal optimization problem due to the non-linear relationship between the data (i.e. arrival times) and the physical models (i.e. velocity model). Another issue encountered in inverse problems is the non-uniqueness of the solution as many other velocity models can explain the observed arrival time data. Methods based on Markov Chain Monte Carlo are usually applied to address both the non-linearity and non-uniqueness issues. Given that these methods are sequential algorithms, they cannot fully handle the computational resources provided by modern supercomputers. In this manuscript, I proposed to solve the seismic tomography problem by using evolutionary algorithms as they are intrinsically parallel and thus well suited to supercomputer architectures. Secondly, I described an algorithm to automatically pick the arrival times required in seismic tomography by using a multi-attributes neural network.

6.1.1 1D traveltime tomography

In Chapter 3, my work was mainly focused on applying PSO on a 1D first arrival traveltime tomography problem. I chose PSO for its robustness and ease of implementation. I showed on a 2D highly multi-modal function that PSO suffers from premature convergence and therefore proposed a new evolutionary algorithm based on PSO that I called Competitive PSO (CPSO). CPSO has been designed to address premature convergence by reinitializing the worse performing particles to improve the diversity of the swarm. I carried a thorough analysis and compared the performance of both algorithms on several benchmark test functions. CPSO outperformed PSO and appeared to be less sensitive to the choice of its control parameter. CPSO has also demonstrated to be more reliable for rapid uncertainty quantification. I applied CPSO on a real tomographic problem in the context of hydraulic fracturing and compared the results with the ones obtained using a conventional MCMC sampler. I performed four inversions using CPSO with different swarm sizes. In all cases, CPSO has been able to reach the stationary regime much faster and the uncertainties estimated by CPSO are consistent with the ones obtained with MCMC. Besides, for this specific problem, a smaller swarm showed similar performance compared to a bigger swarm which invalidates some conclusions that can be found in the geophysical literature (e.g. a swarm size of 300 particles is used to relocate microseismic events in Lagos, Sabbione and Velis (2014)). Finally, I assess the parallel performance of CPSO on this tomographic problem by performing a strong scale analysis. Speed up and parallel efficiency were not ideal due to the communication overhead and/or overhead implied by parallel decomposition of the algorithm. An asynchronous parallel implementation would be required to improve its scalability and obtain close to ideal parallel performance.

6.1.2 Refraction traveltime tomography

In Chapter 4, I extended the study to the highly non-linear and multi-modal problem of refraction tomography in 2D by comparing the performances of DE, CPSO and CMA-ES. The aim of this comparison is to assess the feasibility of evolutionary algorithms to solve a real-world problem in high dimension (typically > 10^2 parameters). First, I generated synthetic traveltimes on the Marmousi velocity model that presents a complex geology structure considering a stationary surface acquisition. The geometry of the acquisition consists of two hundred shots and four hundred receivers which results in poor ray coverage in depth. I chose to parametrize the velocity model using 2D cardinal B-splines that provide a smooth and sparse model. I investigated the influences of the initial velocity models, the population size and the maximum number of iterations. Ideally, evolutionary algorithms as stochastic methods should be rather insensitive to the initial population. Therefore, I tested several types of initialization using random and more physical models. Random increasing gradient initial population demonstrated to significantly improve the convergence of evolutionary algorithms on the refraction tomography problem. The retained initialization procedure only requires the user to specify the lower and upper boundaries of the feasible space. I then applied each algorithm on the same problem but using a 15 by 20 B-spline grid (i.e. 300 parameters), and assessed their benefits and shortcomings by statistically analyzing their performances and scalabilities. From a pure optimization point of view, CMA-ES clearly outperformed both DE and CPSO at the expense of poor parallel scalability. On the other hand, DE and CPSO demonstrated comparable parallel performances, yet CPSO appeared to be less robust in terms of repeatability. Nevertheless, CPSO turned out to be the only method able to retrieve uncertainties consistent with the target ray coverage.

6.1.3 Neural network automated phase onset picking

In Chapters 3 and 4, traveltime data were either already manually picked by an operator or synthetically generated. In microseismic monitoring, an efficient automated phase onset picking algorithm is required for real-time microseismic event locations induced by hydraulic fracturing. In Chapter 5, I described an automated phase onset picking algorithm based on a multi-attributes neural network. It is noteworthy that this chapter is the result of a project in collaboration with ENS Paris and ENSG, the main emphase of this thesis remains the application of evolutionary algorithms to seismic tomography. Nonetheless, the relevancy of this chapter is two-fold: (1) automated phase picking is an important step when dealing with traveltimes; (2) evolutionary algorithms are inherently related to neural networks, in particular to neuroevolution. The chapter is written as a description of a computer software implemented in Python called AIPycker. The acquisition has been performed on a cylindric sample of andesite from la Guadeloupe using a conventional tri-axial cell installed in the Laboratoire de Géologie at the ENS Paris. The methodology is based on several selected attributes able to distinguish a phase onset sample from a noise sample. Optimal attributes and corresponding parameters can be selected using a scatter-plot matrix. In this chapter, I only retained three attributes namely SNR, AIC-W and Kurtosis. In order to better constrain the decision space, I proposed to skew the training set by purposely introducing more noise samples than onset samples which are then duplicated to rebalance the data set. In AIPycker, a neural network can be trained either by a derivative-based method or an evolutionary algorithm. The probability map output by the trained neural network model indicates the likelihood of a sample to be a phase onset. The phase onset is chosen as the sample that yields the maximum probability and is flagged as false positive if its probability is lower than a confidence threshold specified by the user. The error associated to the predicted phase onset is also estimated using the probability map as the half-distance between the samples that yield 0.16 and 0.84 on the CDF. Finally, the picker model is applied to

the whole data set to predict the arrival times. The results are promising in terms of accuracy and estimated picking errors. The microseismic events relocated using the picked arrival times are also consistent with what is usually observed in this type of experiment.

6.2 Perspectives

6.2.1 Improving parallelism and convergence: Island models

Throughout this manuscript, the computation of the forward problem has been parallelized at two levels: (1) the models represented by different individuals within a population have been evenly spread accross several MPI processes for concurrent evaluation; (2) the traveltime grids for each seismic source have been calculated simultaneously on several OpenMP threads. Note that the different runs required for uncertainty quantification have been performed sequentially due to the lack of computer resources. However, the runs are independent and can be easily parallelized given more computation power.

Migration is a mechanism of population genetics that is usually not considered in basic evolutionary algorithms. It consists in transfering individuals that carry the genetic variations from one population to another. This mechanism can be reproduced for optimization purpose through the so-called *island models*. In island models evolutionary algorithms, the initial population is divided into subpopulations that evolve independently on a subset of cores (the islands) with a periodical exchange of individuals (Gong and Fukunaga (2011), Gong *et al.* (2015)). The migration process is comparable to the exchange of states introduced in some algorithms based on MCMC such as Parallel Tempering (Sambridge (2014)) and Interactive MCMC (Romary (2010)). Likewise parallel MCMC algorithms, island model evolutionary algorithms have demonstrasted better performance as each population follows a different evolution path which naturally improves diversity within the whole population (Whitley, Rana and Heckendorn (1999)). Within the island model framework, the subpopulations can evolve following different stochastic processes (i.e. different parameters and/or algorithms) which has led to hybrid island model based evolutionary algorithms. The main interest of this hybrid approach is that the combination of different evolutionary algorithms can compensate their individual shortcomings.

6.2.2 Improving phase onset picking: Bayesian neural network and neuroevolution

In Chapter 5, I described an algorithm for automatic phase onset picking based on a neural network. My initial idea was to use a Bayesian neural network to quantify the errors associated to the neural weights (Neal (1992), Neal (1996)). In Bayesian neural networks, the final probability map is the product of the probability maps from all the models sampled. I sampled the neural weights using a Hamiltonian Monte Carlo sampler (Duane *et al.* (1987), Neal (2011)) to significantly reduce the number of models required to correctly approximate the PDF as the gradient can be efficiently calculated using the backpropagation algorithm. However, many models were still required for a correct approximation of the PDF which results in a non-reasonable computation time for the given task. However, implementing the methodology using a lower level language (e.g. C or Fortran) and/or parallelize the computation of the probability map using General-purpose GPU would drastically reduce the computation time.

The neural network implemented in this manuscript is the simplest form of neuroevolution where only the neural weights are adjusted using an evolutionary algorithm (Ronald and Schoenauer

(1994)). Recent developments in neuroevolution have led to algorithms capable of evolving both the neural weights and the network structure. These algorithms are known as *TWEANN* (Topology and Weights Evolving Artificial Neural Network) and include NeuroEvolution of Augmenting Topologies (NEAT, Stanley and Miikkulainen (2002)), Evolutionary Acquisition of Neural Topologies (EANT, Kassahun and Sommer (2005), Siebel and Sommer (2007)) and Covariance Matrix Adaptation with Hypervolume Sorted Adaptive Grid Algorithm (CMA-HAGA, Rostami and Neri (2017), Shenfield and Rostami (2017)). Automated neural network onset picking algorithms that can be found in the geophysical literature all deal with predetermined network architectures. Therefore, applying TWEANN on phase onset picking is certainly an interesting prospect for improvement.

6.2.3 Use of velocity model uncertainties

Evolutionary algorithms are optimization algorithms designed to rapidly locate and sample a single (hopefully good) mode. I introduced a methodology to quantify uncertainties using evolutionary algorithms by analyzing all the models sampled by different runs. In Chapter 5, I showed that only CPSO was able to recover uncertainties consistent with the ray coverage. However, the estimated velocity model uncertainties have not been used so far. In the context of hydraulic fracturing, Gesret *et al.* (2015) described a methodology to obtain more accurate locations with reliable uncertainties by accounting for uncertainties associated to velocity models. This is done by locating the microseismic events in all the velocity models sampled and summing the resulting PDFs. I further improve the methodology and suggest to perform a cluster analysis using unsupervised learning algorithms to find a reliable subset of representative velocity models for the location, as described in Appendix C.

In addition, I recommend not to use the mean model from several runs of CPSO as such but to further refine the solution using a derivative-based algorithm with the solution as prior model $\mathbf{m}_{\text{prior}}$ following the misfit function

$$E(\mathbf{m}) = \left(\mathbf{d}^{obs} - g(\mathbf{m})\right)^{\top} \mathbf{\Sigma_D}^{-1} \left(\mathbf{d}^{obs} - g(\mathbf{m})\right) + (\mathbf{m} - \mathbf{m}_{\text{prior}})^{\top} \mathbf{\Sigma_M}^{-1} (\mathbf{m} - \mathbf{m}_{\text{prior}}) \quad (6.1)$$

where $\mathbf{\Sigma_D}$ and $\mathbf{\Sigma_M}$ are respectively the data and model covariance matrices (Tarantola (2005)). Typically, the regularization procedure applied when using derivative-based optimization algorithms treats the model covariance matrix as a constant damping coefficient that needs to be tuned by the user. Nevertheless, I believe that better results can be obtained by using the uncertainties estimated by CPSO to directly define the model covariance matrix. Note that this idea is not specific to CPSO and the mean model and uncertainties obtained by conventional MCMC samplers (Ryberg and Haberland (2018), Belhadj *et al.* (2018)) can theoretically be used as well.

Appendix A

Eikonal equation

First arrival traveltimes can be accurately computed by solving the Eikonal equation that links the velocities of the propagation medium to the traveltimes. In this appendix, I derive the Eikonal equation from the elasto-dynamic equation by applying the high frequency approximation that consists in considering that the signal wavelength is neglectable with respect to the characteristic wavelength of the medium. A full and detailed understanding of the dynamic of acoustic waves can be found in Chapman (1985), Sheriff and Geldart (1995) and Červený (2001) and is beyond the scope of this thesis.

The theory is based on the elasto-dynamic equation in an isotropic and homogeneous medium that links the strain to the stress applied, written as

$$\rho \frac{\partial^2 \mathbf{u}}{\partial t^2} - (\lambda + \mu) \nabla (\nabla \cdot \mathbf{u}) - \mu \nabla^2 \mathbf{u} = \mathbf{0} \tag{A.1}$$

where $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$ is the displacement vector field, λ and μ are Lamé's parameters, $\nabla = (\partial_x, \partial_y, \partial_z)$ denotes the gradient operator and $\nabla\cdot$ the divergence operator.

Helmholtz's theorem states that any vector field can be decomposed into the gradient of a scalar field Φ with zero curl and the curl of a vector field $\mathbf{\Psi}$ with zero divergence following

$$\mathbf{u} = \nabla \Phi + \nabla \times \mathbf{\Psi} \tag{A.2}$$

with $\nabla \times (\nabla \Phi) = \mathbf{0}$ and $\nabla \cdot (\nabla \times \mathbf{\Psi}) = 0$. Therefore, we decompose the displacement vector field into a longitudinal wavefield $\mathbf{u}_p$ and a transverse wavefield $\mathbf{u}_s$:

$$\mathbf{u} = \mathbf{u}_p + \mathbf{u}_s \qquad \text{with} \begin{cases} \nabla \times \mathbf{u}_p = \mathbf{0} \\ \nabla \cdot \mathbf{u}_s = 0 \end{cases} \tag{A.3}$$

where $\mathbf{u}_p$ and $\mathbf{u}_s$ respectively represent the compressional wave (P-) and shear wave (S-) displacement fields as depicted in Figure A.1.

Substituting Equation (A.3) into (A.1), we obtain

$$\rho \left(\frac{\partial^2 \mathbf{u}_p}{\partial t^2} + \frac{\partial^2 \mathbf{u}_s}{\partial t^2} \right) - (\lambda + \mu) \nabla (\nabla \cdot \mathbf{u}_p) - \mu \left(\nabla^2 \mathbf{u}_p + \nabla^2 \mathbf{u}_s \right) = \mathbf{0}. \tag{A.4}$$

The gradient of the divergence of the compressional vector field $\mathbf{u}_p$ can be expressed as

$$\nabla (\nabla \cdot \mathbf{u}_p) = \nabla \times \underbrace{(\nabla \times \mathbf{u}_p)}_{\mathbf{0}} + \nabla^2 \mathbf{u}_p = \nabla^2 \mathbf{u}_p. \tag{A.5}$$

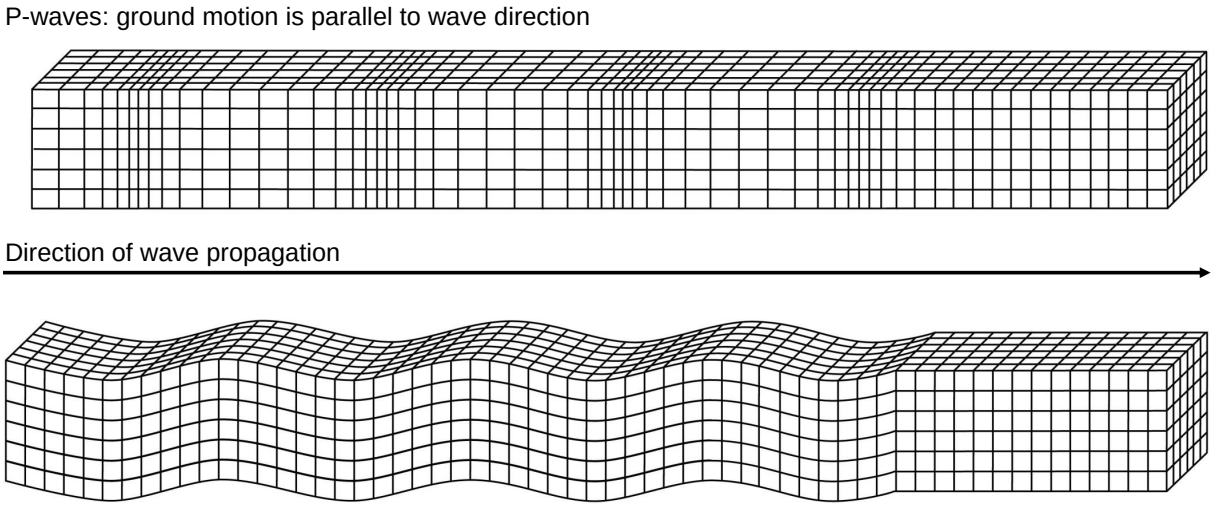

Figure A.1: Displacements for P- and S- waves (adapted from levee.wustl.edu/seismology/book/).

Regrouping the terms dependent on $\mathbf{u}_p$ and $\mathbf{u}_s$, the elasto-dynamic equation (A.1) is rewritten

$$\left(\rho\frac{\partial^2\mathbf{u}_p}{\partial t^2} - (\lambda+2\mu)\,\nabla^2\mathbf{u}_p\right) + \left(\rho\frac{\partial^2\mathbf{u}_p}{\partial t^2} - \mu\nabla^2\mathbf{u}_s\right) = \mathbf{0}. \tag{A.6}$$

The vector fields $\mathbf{u}_p$ and $\mathbf{u}_s$ are independent, thus the two terms of the sum both equal zero, following

$$\begin{cases} \rho\frac{\partial^2\mathbf{u}_p}{\partial t^2} - (\lambda+2\mu)\,\nabla^2\mathbf{u}_p = \mathbf{0} \\ \rho\frac{\partial^2\mathbf{u}_p}{\partial t^2} - \mu\nabla^2\mathbf{u}_s = \mathbf{0} \end{cases}. \tag{A.7}$$

Writing $c_p = \sqrt{\frac{\lambda+2\mu}{\rho}}$ and $c_s = \sqrt{\frac{\mu}{\rho}}$, we obtain the wave equation for the two wavefields, following

$$\begin{cases} \frac{1}{c_p^2}\frac{\partial^2\mathbf{u}_p}{\partial t^2} = \nabla^2\mathbf{u}_p \\ \frac{1}{c_s^2}\frac{\partial^2\mathbf{u}_s}{\partial t^2} = \nabla^2\mathbf{u}_s \end{cases} \tag{A.8}$$

where c_p and c_s are P- and S- wave velocities, respectively. Let us consider the wave equation with no source term in the frequency domain, written as

$$-\frac{\omega^2}{c^2(\mathbf{x})}\mathbf{u}(\mathbf{x},\omega) = \nabla^2\mathbf{u}(\mathbf{x},\omega). \tag{A.9}$$

The solution of the wave equation in the frequency domain is written

$$\mathbf{u}(\mathbf{x},\omega) = S(\omega)\,A(\mathbf{x},\omega)\,e^{i\omega T(\mathbf{x})} \tag{A.10}$$

where $S(\omega)$ is the wave signature, $A(\mathbf{x},\omega)$ is the wave amplitude with $\omega T(\mathbf{x})$ its phase that depends on the frequency ω and the traveltime $T(\mathbf{x})$.

According to the high frequency approximation Ansatz (Sheriff and Geldart (1995), Červený (2001)), the amplitude $A(\mathbf{x},\omega)$ can be written in the form of a Taylor expansion in $\frac{1}{i\omega}$

$$A(\mathbf{x},\omega) = \sum_{n=0}^{+\infty}\frac{A_n(\mathbf{x})}{(i\omega)^n}. \tag{A.11}$$

Substituting the approximation (A.11) and the solution (A.10) into the wave equation (A.9), we obtain

$$\begin{aligned} -\frac{\omega^2}{c^2(\mathbf{x})}\sum_{n=0}^{+\infty}\frac{A_n(\mathbf{x})}{(i\omega)^n} = &-\omega^2\left(\nabla T(\mathbf{x})\right)^2\sum_{n=0}^{+\infty}\frac{A_n(\mathbf{x})}{(i\omega)^n} \\ &+ i\omega\sum_{n=0}^{+\infty}\frac{2\nabla T(\mathbf{x})\cdot\nabla A_n(\mathbf{x})+\nabla^2 T(\mathbf{x})A_n(\mathbf{x})}{(i\omega)^n} \\ &+\sum_{n=0}^{+\infty}\frac{\nabla^2 A_n(\mathbf{x})}{(i\omega)^n}. \end{aligned} \tag{A.12}$$

The equality holds at all frequencies and we have thus the equality for all the coefficients of the expansion. By performing trivial factorizations and simplifications, we obtain the following system of equations

$$\begin{cases} (i\omega)^2 & \left(\nabla T(\mathbf{x})\right)^2 = \frac{1}{c(\mathbf{x})^2} \\ (i\omega)^1 & 2\nabla T(\mathbf{x})\cdot\nabla A_0(\mathbf{x}) + \nabla^2(\mathbf{x})A_0(\mathbf{x}) = 0 \\ (i\omega)^0 & 2\nabla T(\mathbf{x})\cdot\nabla A_1(\mathbf{x}) + \nabla^2(\mathbf{x})A_1(\mathbf{x}) + \nabla^2 A_0(\mathbf{x}) = 0 \\ \dots & \\ (i\omega)^{-n} & 2\nabla T(\mathbf{x})\cdot\nabla A_{n+1}(\mathbf{x}) + \nabla^2(\mathbf{x})A_{n+1}(\mathbf{x}) + \nabla^2 A_n(\mathbf{x}) = 0 \end{cases}. \tag{A.13}$$

The first equation that corresponds to the zero order of the Taylor expansion only involves the traveltime and is known as the Eikonal equation which is written for the P- and S- wavefields

$$\begin{cases} |\nabla T|^2 = \frac{1}{c_p^2} \\ |\nabla T|^2 = \frac{1}{c_s^2} \end{cases}. \tag{A.14}$$

Appendix B

Surface wave tomography

Contents

Dans ce chapitre annexe, j'applique la méthodologie décrite dans le Chapitre 3 au problème de tomographie des ondes de surface. La connaissance du modèle de propagation des ondes de cisaillement est une donnée importante aussi bien en géotechnique qu'en géophysique car elle fournit une information sur les propriétés mécaniques d'un sol. Ce modèle est généralement reconstruit par un procédé d'inversion des courbes de dispersion des ondes de surface. Ce chapitre annexe décrit le problème direct pour générer des courbes de dispersion étant donné un milieu tabulaire, suivi d'une application sur données réelles.

Ces travaux ont fait l'objet d'une publication en tant que co-auteur :

- Marc Peruzzetto, Alexandre Kazantsev, **Keurfon Luu**, Jean-Philippe Métaxian, Frédéric Huguet and Hervé Chauris, 2018. "Broadband ambient noise characterization by joint use of cross-correlation and MUSIC algorithm." *Geophysical Journal International* 215(2): 760-779 (November 2018). doi: 10.1093/gji/ggy311;

Les résultats présentés dans cette publication ont été obtenus avec un logiciel que j'ai développé en Python disponible en libre accès sur ma page GitHub :

- EvoDCinv: inversion des courbes de dispersion des ondes de surface par des algorithmes évolutionnistes. Ce module permet l'inversion de courbes de dispersion multi-modales des ondes de Rayleigh et/ou Love. Disponible à l'adresse github.com/keurfonluu/EvoDCinv.

B.1 Introduction

The knowledge of the shear wave propagation medium is valuable in geotechnics and geophysics as it directly provides information on the mechanical properties of the subsurface. It is usually reconstructed by inverting the dispersion curves of surface waves. Several methods using either active or passive sources are available to acquire dispersion data such as Spatial autocorrelation (SPAC, Aki (1957)), Frequency-wavenumber method (FK, Capon, Greenfield and Kolker (1967)), Spectral analysis of surface waves (SASW, Nazarian *et al.* (1983)), Multi-channel analysis of surface waves (MASW, Park, Miller and Xia (1999)), Refraction microtremor (Re-Mi, Louie (2001)) or Ambient noise correlation (ANC, Shapiro and Campillo (2004)). This short appendix presents the forward modeling to compute synthetic dispersion curves given a layered earth model, and a real application in which the dispersion curves obtained by the joint use of cross-correlation and Multiple signal characterization algorithm (MUSIC, Schmidt (1986), Goldstein and Archuleta (1987)) are inverted. The same methodology as described in Chapter 3 is applied. The description of the acquisition technique is beyond the scope of this thesis.

B.2 Forward problem: Thomson-Haskell propagator

The forward modeling consists in estimating the frequency-dependent dispersion curves for all the modes of a surface wave given a layered earth model. Each layer is characterized by its P- and S- wave velocities, density and thickness. Following the works of Thomson (1950) and Haskell (1953), the dispersion equation derived from the wave equation is elegantly written in the form

$$\det|\mathbf{T}(\omega, c)| = 0 \tag{B.1}$$

where $\mathbf{T}(\omega, c)$ is the so-called Thomson-Haskell matrix and depends on the frequency ω and the phase velocity c. The resulting Thomson-Haskell method evaluates a transfer matrix for each layer and propagates the solution from the top (free surface) to the bottom (half-space). In this appendix, we are only interested in Rayleigh waves, but it should be mentioned that the theory also applies for Love waves.

B.2.1 Rayleigh wave in a layered medium

This section aims to demonstrate the Thomson-Haskell matrix formalism as described in Buchen and Ben-Hador (1996). Surface, interface and radiation conditions can be expressed in terms of displacement-stress state vectors $\mathbf{y}_i(z)$ for each layer i eventually used to form the Thomson-Haskell propagator matrices. In the following, P- and S- wave velocities in layer i are respectively denoted by $c_{p,i}$ and $c_{s,i}$, the layer thickness by d_i, the total number of layers by N and the half-space by $L = N + 1$.

State vectors

Let us rewrite the Helmholtz's decomposition of the displacement vector $\mathbf{u}$ (Equation (A.3)) introduced in Appendix A, following

$$\mathbf{u} = \mathbf{u}_p + \mathbf{u}_s \qquad \text{with} \begin{cases} \mathbf{u}_p = \nabla\Phi \\ \mathbf{u}_s = \nabla \times \boldsymbol{\Psi} \end{cases} \tag{B.2}$$

where $\mathbf{u}_p$ and $\mathbf{u}_s$ respectively denote the P- and S- displacement wavefields. We recall that $\nabla = (\partial_x, \partial_y, \partial_z)$. In the following, we consider a motion in the plane (x, z), hence $\partial_y = 0$.

The P-wave displacement field $\mathbf{u}_p$ associated to the scalar potential Φ is expressed by

$$\mathbf{u}_p = \nabla\Phi = \begin{cases} \partial_x\Phi \\ \partial_y\Phi = 0 \\ \partial_z\Phi \end{cases} . \tag{B.3}$$

The vector potential $\mathbf{\Psi}$ is orthogonal to the direction of displacement of particles, hence $\Psi_x = \Psi_z = 0$. Therefore, the S-wave displacement field $\mathbf{u}_s$ associated to the vector potential $\mathbf{\Psi}$ reads

$$\mathbf{u}_s = \nabla \times \mathbf{\Psi} = \begin{cases} \partial_y\Psi_z - \partial_z\Psi_y &= -\partial_z\Psi_y \\ \partial_z\Psi_x - \partial_x\Psi_z &= 0 \\ \partial_x\Psi_y - \partial_y\Psi_x &= \partial_x\Psi_y \end{cases} . \tag{B.4}$$

As the components of $\mathbf{u}_s$ only depend on Ψ_y, we drop the dependence on y in the notation and write Ψ instead of Ψ_y. Combining Equations (B.2), (B.3) and (B.4), the total displacement field in each layer of the medium is written

$$\begin{cases} u_x^i &= \partial_x\Phi_i - \partial_z\Psi_i \\ u_z^i &= \partial_z\Phi_i + \partial_x\Psi_i \end{cases} . \tag{B.5}$$

We express the wavefield by means of the Helmholtz potential and decompose it into an upgoing (U) and a downgoing (D) wavefield, following

$$\begin{cases} \Phi_i &= \left(A_i^U e^{-kr_i z} + A_i^D e^{kr_i z}\right)\cos\left(kx - \omega t\right) \\ \Psi_i &= \left(B_i^U e^{-ks_i z} + B_i^D e^{ks_i z}\right)\sin\left(kx - \omega t\right) \end{cases} \tag{B.6}$$

in which $r_i = \sqrt{1 - \frac{c^2}{c_{p,i}^2}}$ and $s_i = \sqrt{1 - \frac{c^2}{c_{s,i}^2}}$, k is the wavenumber, A_i^U, A_i^D, B_i^U and B_i^D denote the amplitudes of the upgoing and downgoing P-wave and SV-wave, respectively.

A first system of equations that will be used to define the state vector for layer i is obtained by substituting Equation (B.6) into (B.5), following

$$\begin{cases} u_x^i &= -k\sin\left(kx - \omega t\right)\left(A_i^U e^{-kr_i z} + A_i^D e^{kr_i z} - s_i B_i^U e^{-ks_i z} + s_i B_i^D e^{ks_i z}\right) \\ u_z^i &= -k\cos\left(kx - \omega t\right)\left(r_i A_i^U e^{-kr_i z} - r_i A_i^D e^{kr_i z} - B_i^U e^{-ks_i z} - B_i^D e^{ks_i z}\right) \end{cases} . \tag{B.7}$$

The second system of equations is given by the components of the vertical stress field σ_{xz}^i and σ_{zz}^i. The Hooke's law with Lamé's parameters λ and μ for an isotropic medium is written

$$\sigma_{ij} = 2\mu\varepsilon_{ij} + \lambda\varepsilon_{kk}\delta_{ij} \tag{B.8}$$

where the strain tensor is expressed by

$$\varepsilon_{ij} = \frac{1}{2}\left(\partial_j u_i + \partial_i u_j\right) . \tag{B.9}$$

Therefore, the vertical stress field reads

$$\begin{cases} \sigma_{xz}^i &= 2\mu_i\varepsilon_{xz} \\ \sigma_{zz}^i &= \left(\lambda_i + 2\mu_i\right)\varepsilon_{zz} + \lambda\varepsilon_{xx}. \end{cases} \tag{B.10}$$

After differentiating Equation (B.10) using Equation (B.9) and performing trivial simplifications, we obtain a second system of equations defined by

$$\begin{cases} \sigma_{xz}^{i} = & \mu_i k^2 \sin\left(kx-\omega t\right) \\ & \left(2r_i A_i^U e^{-kr_i z} - 2r_i A_i^D e^{kr_i z} - \left(s_i^2+1\right) B_i^U e^{-ks_i z} - \left(s_i^2+1\right) B_i^D e^{ks_i z}\right) \\ \sigma_{zz}^{i} = & \mu_i k^2 \cos\left(kx-\omega t\right) \\ & \left(\left(s_i^2+1\right) A_i^U e^{-kr_i z} + \left(s_i^2+1\right) A_i^D e^{kr_i z} - 2s_i B_i^U e^{-ks_i z} + 2s_i B_i^D e^{ks_i z}\right) \end{cases}. \tag{B.11}$$

The final system formed by Equations (B.7) and (B.11) can be rewritten in a matrix form as a function of the state vector $\mathbf{y}_i(z) = \left[\mathbf{U}_x^i(z) \quad \mathbf{U}_z^i(z) \quad \mathbf{\Sigma}_{xz}^i \quad \mathbf{\Sigma}_{zz}^i\right]^\top$ according to

$$\left[u_x^i, u_z^i, \sigma_{xz}^i, \sigma_{zz}^i\right]^\top = \mathbf{S}_i \underbrace{\mathbf{M}_i \mathbf{P}_i \mathbf{E}_i(-z)\mathbf{a}_i}_{\mathbf{y}_i(z)} \tag{B.12}$$

with

$$\begin{cases} \mathbf{S}_i & = \begin{bmatrix} -k\sin\left(kx-\omega t\right) & 0 & 0 & 0 \\ 0 & -k\cos\left(kx-\omega t\right) & 0 & 0 \\ 0 & 0 & k^2\sin\left(kx-\omega t\right) & 0 \\ 0 & 0 & 0 & k^2\cos\left(kx-\omega t\right) \end{bmatrix} \\ \mathbf{M}_i & = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \mu_i & 0 \\ 0 & 0 & 0 & \mu_i \end{bmatrix} \\ \mathbf{P}_i & = \begin{bmatrix} 1 & 1 & -s_i & s_i \\ r_i & -r_i & -1 & -1 \\ 2r_i & -2r_i & -\left(s_i^2+1\right) & -\left(s_i^2+1\right) \\ s_i^2+1 & s_i^2+1 & -2s_i & 2s_i \end{bmatrix} \\ \mathbf{E}_i(z) & = \begin{bmatrix} e^{kr_i z} & 0 & 0 & 0 \\ 0 & e^{-kr_i z} & 0 & 0 \\ 0 & 0 & e^{ks_i z} & 0 \\ 0 & 0 & 0 & e^{-ks_i z} \end{bmatrix} \\ \mathbf{a}_i & = \left[A_i^U \quad A_i^D \quad B_i^U \quad B_i^D\right]^\top \end{cases}. \tag{B.13}$$

Boundary conditions

1. Stress-free surface ($i = 0$): $\sigma_{xz}^0 = \sigma_{zz}^0 = 0$

$$\mathbf{y}_0(0) = \left[\mathbf{U}_x^0(0) \quad \mathbf{U}_z^0(0) \quad \mathbf{0} \quad \mathbf{0}\right]^\top \tag{B.14}$$

2. Continuity of displacement and stress at layer interfaces ($i = 1, ..., N$):

$$\mathbf{y}_i(d_i) = \mathbf{y}_{i+1}(0) \tag{B.15}$$

3. Radiation conditions in half-space ($i = L$):

$$\mathbf{a}_L = \left[A_L^U \quad 0 \quad B_L^U \quad 0\right]^\top \tag{B.16}$$

Propagator matrices

We introduce a layer matrix $\mathbf{Q}_i = \mathbf{M}_i\mathbf{P}_i$ and we rewrite the displacement-stress state vector as

$$\mathbf{y}_i(z) = \mathbf{Q}_i\mathbf{E}_i(-z)\mathbf{a}_i. \tag{B.17}$$

The layer propagator (or transfer) matrix $\mathbf{T}_i(z)$ is expressed by

$$\mathbf{T}_i(z) = \mathbf{Q}_i\mathbf{E}_i(z)\mathbf{Q}_i^{-1}. \tag{B.18}$$

We have for any depths z_1 and z_2 within the same layer

$$\begin{cases} \mathbf{y}_i(z_1) &= \mathbf{Q}_i\mathbf{E}_i(-z_1)\mathbf{a}_i \\ \mathbf{y}_i(z_2) &= \mathbf{Q}_i\mathbf{E}_i(-z_2)\mathbf{a}_i \end{cases} \tag{B.19}$$

and we can also show that

$$\begin{cases} \mathbf{T}_i(z_1)\mathbf{T}_i(z_2) = \mathbf{T}_i(z_1+z_2) \\ \mathbf{T}_i^{-1}(z) = \mathbf{T}_i(-z) \end{cases}. \tag{B.20}$$

Isolating $\mathbf{a}_i$ in $\mathbf{y}_i(z_2)$ and substituting it into $\mathbf{y}_i(z_1)$, we obtain

$$\mathbf{y}_i(z_1) = \mathbf{Q}_i\mathbf{E}_i(-z_1)\mathbf{E}_i(-z_2)^{-1}\mathbf{Q}_i^{-1}\mathbf{y}_i(z_2) \tag{B.21}$$

$$= \underbrace{\mathbf{Q}_i\mathbf{E}_i(-z_1)\mathbf{Q}_i^{-1}}_{\mathbf{T}_i(-z_1)}\underbrace{\mathbf{Q}_i\mathbf{E}_i(-z_2)^{-1}\mathbf{Q}_i^{-1}}_{\mathbf{T}_i(z_2)}\mathbf{y}_i(z_2) \tag{B.22}$$

$$= \mathbf{T}_i(z_2-z_1)\mathbf{y}_i(z_2). \tag{B.23}$$

It follows that

$$\mathbf{y}_i(0) = \mathbf{T}_i(d_i)\mathbf{y}_i(d_i). \tag{B.24}$$

The continuity condition of displacement and stress at layer interfaces (Equation (B.15)) leads to the Thomson-Haskell recursion which is written

$$\mathbf{y}_i(0) = \mathbf{T}_i(d_i)\mathbf{y_{i+1}}(0) \tag{B.25}$$

with solution

$$\mathbf{y}_1(0) = \left(\prod_{i=1}^{N}\mathbf{T}_i(d_i)\right)\mathbf{y}_L(0). \tag{B.26}$$

The dispersion relation is obtained by applying the stress-free surface and radiation conditions to the state vectors $\mathbf{y}_1$ and $\mathbf{y}_L$, which leads to a matrix equation of the form

$$\underbrace{\mathbf{U}^\top\left(\prod_{i=1}^{N}\mathbf{T}_i(d_i)\right)\mathbf{V}}_{\mathbf{T}(\omega,c)}\ \mathbf{a}_L = \mathbf{0} \tag{B.27}$$

where $\mathbf{T}(\omega, c)$ is the Thomson-Haskell matrix, $\mathbf{U}$ and $\mathbf{V}$ are the free-surface and the half-space boundary matrices, respectively. The Rayleigh dispersion equation is finally obtained by solving the system which is homogeneous and has a non-trivial solution only when the determinant is zero, following

$$\det|\mathbf{T}(\omega, c)| = 0. \tag{B.28}$$

B.2.2 Roots search

For a given frequency ω, the Rayleigh dispersion equation (B.28) has a finite number of non-trivial solutions only for specific values of phase velocity c. Many techniques have been proposed to find the roots of the dispersion function, the most frequently used being the propagator matrix methods. Each root corresponds to a modal curve that only exists within a limited phase velocity domain $[V_{\text{min}}, V_{\text{max}}]$ such that

$$\begin{cases} V_{\text{min}} & = \min_{1\leq i\leq L}\left(c_{R,i}\right) \\ V_{\text{max}} & = \max_{1\leq i\leq L}\left(c_{s,i}\right) \end{cases} \tag{B.29}$$

where $c_{s,i}$ denotes the S-wave velocity of layer i and $c_{R,i}$ the Rayleigh wave velocity estimated by considering each layer as a homogeneous half-space, following

$$c_{R,i} = \frac{0.87 + 1.12\nu_i}{1+\nu_i} c_{s,i} \tag{B.30}$$

with $\nu_i = \frac{1-2c_{s,i}^2/c_{p,i}^2}{2\left(1-c_{s,i}^2/c_{p,i}^2\right)}$ being the Poisson's ratio. The first and slowest mode is called the fundamental mode and only exists above a cut-off frequency corresponding to the frequency at which the phase velocity equals the maximum S-wave velocity $\max_{1\leq i\leq L}\left(c_{s,i}\right)$. Because the dispersion function oscillates rapidly especially at high frequencies, a root-bracketing method combined with a bisection algorithm is commonly applied to search the roots of the function in spite of the slow convergence of the method (Socco and Strobbia (2004)). In this appendix, we typically deal with problems with low frequency ranges and thus employ a simple grid search with a fine grid size. The roots are then found by interpolating between all the pairs of adjacent grid points with a sign change. Although this approach is less robust, the computation of the dispersion function can be parallelized for each frequency. Figure B.1 shows the modal dispersion curves for a three-layer velocity model with an example of dispersion function that presents numerous roots.

B.3 Inversion

Surface wave tomography is a non-linear optimization problem mostly solved by a linearized iterative least-squares approach (Socco and Strobbia (2004)). In this section, we solve this inverse problem by means of several runs of CPSO to thoroughly sample the model parameter space. The data consists of three manually picked modal dispersion curves (fundamental, first and second) which have been obtained by a joint use of ANC and MUSIC algorithm. The description of this technique is beyond the scope of this thesis.

We consider a velocity model parametrized by nine layers. The forward modeling (i.e. Thomson-Haskell method) requires the P- and S- wave velocities, the density and the thickness of each layer. However, surface waves are mainly sensitive to S-wave velocities, hence the P-wave velocities and the density are usually assumed a priori to reduce the number of model parameter unknowns to optimize. Consequently, we only invert for the S-wave velocities and thicknesses of each layer. The lower and upper search boundaries are summarized in Table B.1.

For the inversion, we apply the same methodology as described in Chapter 3. We run the CPSO algorithm 20 times to sample the model parameter space sufficiently for uncertainty quantification with a swarm size of 50. We set the maximum number of iterations to 500 and CPSO control parameters to their default values (i.e. $\omega = 0.7298$, $\phi_p = \phi_g = 1.49618$, $\gamma = 1$). The 500000 models have been sampled in 15 minutes using a total of 100 cores out of 104 (four

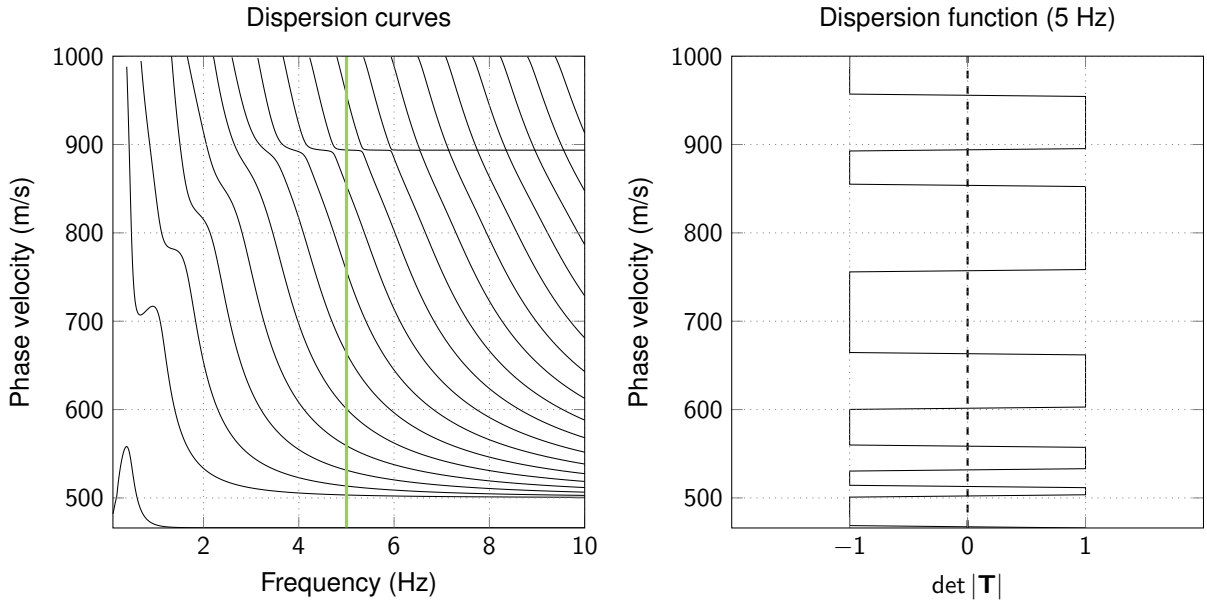

Figure B.1: (Left) Modal dispersion curves for a three-layer model (500 m at 500 m/s, 300 m at 1000 m/s, half-space at 500 m/s). The vertical line (green) indicates a slice at 5 Hz. (Right) Dispersion function at 5 Hz. The positions of the roots (i.e. zeros) correspond to the phase velocities for the different modes. The dispersion function is clipped between -1 and 1.

Table B.1: Lower and upper boundaries of each layer parameter. The last layer corresponds to the half-space with infinite thickness.

Layer #	Vs (m/s)		Thickness (m)	
	Min.	Max.	Min.	Max.
1	100	2000	50	250
2	100	2000	50	250
3	100	2000	50	250
4	100	2000	50	250
5	1000	3000	50	250
6	1000	3000	50	250
7	1000	3000	500	3000
8	1000	3000	500	3000
9	3400	3600	-	-

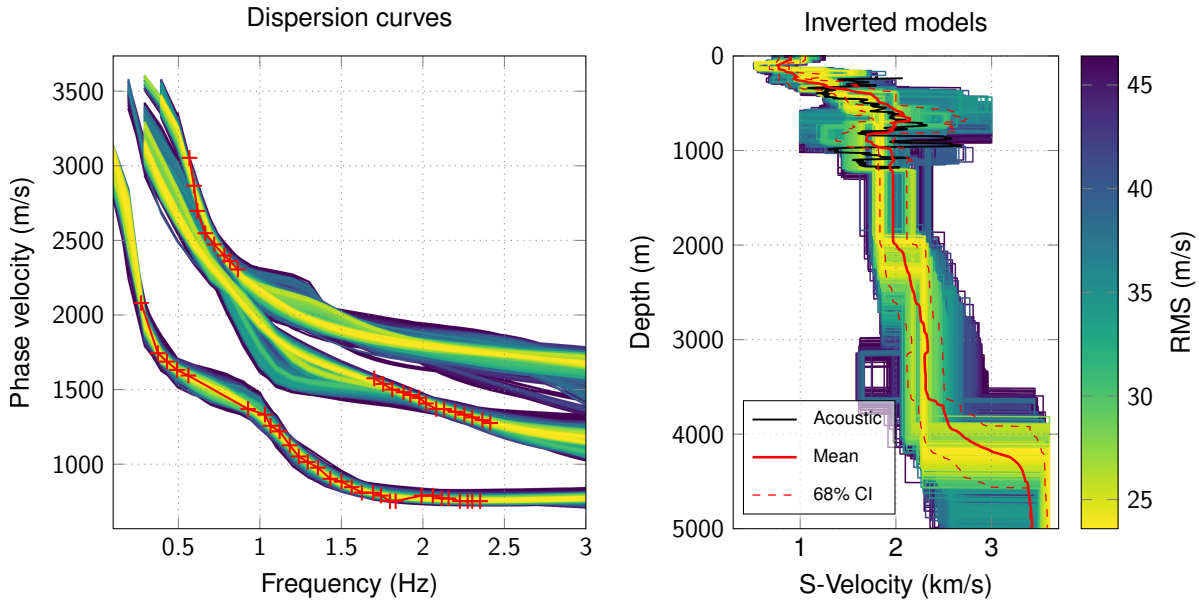

Figure B.2: (Left) Picked (red) and inverted dispersion curves. (Right) Inverted mean velocity profile (red) along with the acoustic log provided by Storengy (black). The velocity models sampled by the different runs of CPSO are represented in the background with the color scale indicating their RMS values. The dashed lines (red) delimit the 68% confidence interval.

sockets platform made of 4 Intel® Xeon® Platinum 8164 CPU, 26 cores @ 2.00 GHz each). The results of the inversion are shown in Figure B.2. The mean velocity model is consistent with the acoustic log provided by Storengy between 230 and 1190 meters and indicates the presence of a rapid layer between 550 and 800 meters. The three inverted modal curves exhibit a good fit with the picked dispersion curves.

B.4 Conclusion

We have described an application of evolutionary algorithms (here CPSO) to another seismic tomography problem. The real case numerical example presented hereinabove shows that the same methodology as adopted in Chapter 3 can be applied to invert surface wave dispersion curves to obtain an accurate shear wave velocity model. Because the dispersion functions are calculated in parallel, this methodology also benefits from the computational resources provided by a supercomputer using a hybrid parallelization approach (OpenMP + MPI).

Appendix C

Propagation of velocity uncertainties to locations

Contents

Plusieurs facteurs peuvent contribuer aux erreurs de localisation des évènements microsismiques dont les erreurs de pointé des temps d'arrivée, une géométrie d'acquisition peu contraignante et une mauvaise connaissance du milieu de propagation. Il est possible de localiser les hypocentres et d'estimer leurs incertitudes de manière plus fiable en propageant les erreurs liées au modèle de vitesse qui peuvent être obtenues en échantillonnant l'espace des modèles de vitesse. Pour ce faire, les évènements microsismiques sont relocalisés dans l'ensemble des modèles de vitesse échantillonnés, ce qui est inefficace d'un point de vue computationnel. Je propose dans ce chapitre annexe une analyse par partitionnements des modèles échantillonnés à l'aide de la CPSO dans le Chapitre 3. L'analyse par partitionnements permet de définir un sous-ensemble de modèles pour la propagation des incertitudes du modèle de vitesse aux localisations microsismiques.

Ce chapitre annexe correspond au résumé étendu (mis à jour) soumis pour une présentation orale à la conférence internationale EAGE 2017 (Paris) :

- **Keurfon Luu**, Mark Noble, Alexandrine Gesret, Nidhal Belayouni and Pierre-François Roux, 2017. “Propagation of velocity uncertainties to Microseismic locations using a competitive Particle Swarm Optimizer.” *79th EAGE Conference and Exhibition 2017*.

Abstract

Microseismic location uncertainties are mainly due to arrival time picking errors, poorly constrained acquisition geometry and the lack of knowledge of the wave propagation medium. More reliable locations of hypocenters with their associated uncertainties can be obtained by propagating velocity model uncertainties which can be obtained by sampling the velocity model space. We propose to use a Competitive Particle Swarm Optimizer (CPSO) to sample the model space by running the algorithm multiple times and keeping all the models that explain the observed data. Then, we perform a cluster analysis on all the acceptable models to define a reliable subset of models to propagate the velocity uncertainties to the microseismic locations. The algorithm is illustrated on a real 3D data set in the context of hydraulic fracturing.

C.1 Introduction

Microseismic monitoring is one of the main tools to estimate hydraulic fracture geometry and failure mode. It consists in detecting and locating very small events induced by the fracturing process (Cipolla *et al.* (2011)). Despite the fact that large volumes of data are being acquired, our understanding of the relation between microseismicity and fracture geometry is still very poor. Very precise location of seismicity is the first step to better understand and delineate hydraulic fracture geometry.

Among many factors such as traveltime picking errors or poorly constrained acquisition geometries that contribute to microseismic location errors, the largest contribution is due to the lack of knowledge of the velocity model. Gesret *et al.* (2015) showed that more reliable locations of hypocenters with their associated uncertainties can be obtained by propagating the velocity model uncertainties to the event locations.

The calibration/inversion for the velocities is a totally non-linear problem and requires the use of global optimization methods. Concerning the estimation of the velocity model uncertainties, from a theoretical point of view, methods based on Markov Chain Monte Carlo (MCMC) that sample the velocity model space are required. However, these algorithms cannot be parallelized and turn out to be prohibitive in terms of computational time. This limitation can be overcome by implementing an Evolutionary Algorithm (EA). EA are algorithms for global optimization inspired by biological evolution that evaluate simultaneously a set of independent models. This simultaneous evaluation of independent models implies that it is straightforward to parallelize and thus can significantly reduce the computation time.

In this appendix, we briefly describe a new EA that samples the model space and enables us to derive reliable velocity model uncertainties in a real 3D microseismic example. We then perform a cluster analysis on the sampled velocity models that allows us to propagate the velocity uncertainties to the hypocenter locations.

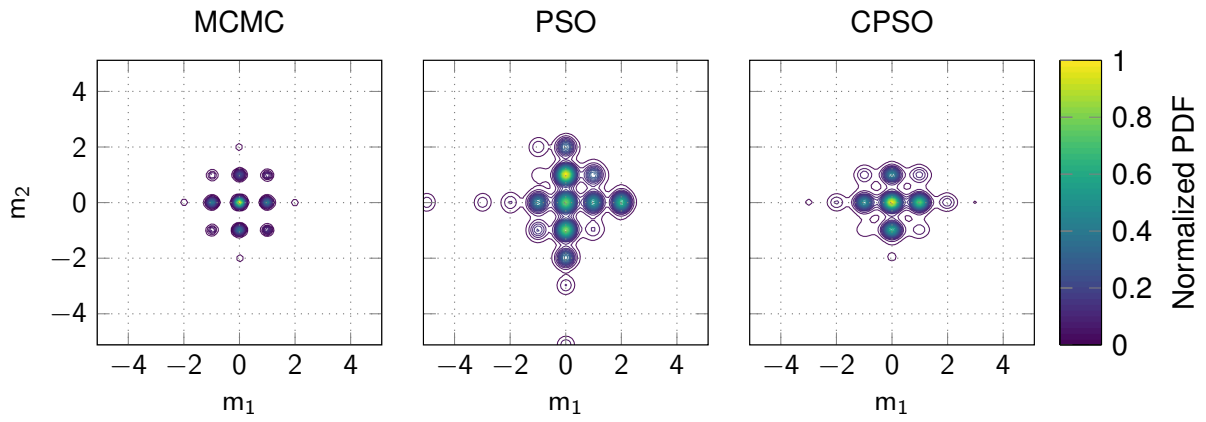

Figure C.1: (Left) 2D Rastrigin PDF sampled by MCMC. Comparison of the sampling capability of (middle) PSO and (right) CPSO on the 2D Rastrigin function.

C.2 From optimization to uncertainty quantification

Our tomography algorithm lies on a new method based on Particle Swarm Optimization (PSO, Kennedy and Eberhart (1995)). In PSO, a swarm composed of several individuals – called particles – is initialized in the model space. At each iteration k, the position of the particle i (i.e. model) is updated following

$$\mathbf{v}_i^k = \omega \mathbf{v}_i^{k-1} + \phi_p \mathbf{r}_p^k \left(\mathbf{m}_{p,i} - \mathbf{m}_i^{k-1} \right) + \phi_g \mathbf{r}_g^k \left(\mathbf{m}_g - \mathbf{m}_i^{k-1} \right) \tag{C.1}$$

$$\mathbf{m}_i^k = \mathbf{m}_i^{k-1} + \mathbf{m}_i^k \tag{C.2}$$

where $\mathbf{v}_i^k$ and $\mathbf{m}_i^k$ are respectively the velocity (displacement) and position vectors, $\mathbf{m}_{p,i}$ and $\mathbf{m}_g$ are the personal best position of particle i and the global best position of the swarm, $\mathbf{r}_p^k$ and $\mathbf{r}_g^k$ are uniform random number vectors drawn at iteration k, ω is an inertial coefficient, ϕ_p and ϕ_g are two constants that respectively control the cognition and social interactions of the particles. However, PSO suffers from premature convergence as it can get trapped in a local minimum. We propose to use a Competitive PSO (Luu *et al.* (2018)) that improves the diversity of the swarm by detecting when the algorithm converges prematurely and by resetting the state of bad fitting particles.

In typical geophysical inverse problems, several models may explain the data very well in terms of misfit function values. This non-unique aspect of the solution is principally caused by the use of a finite number of parameters to describe the earth, and also by the presence of noise in the data. One approach to characterize the non-uniqueness of the solution is to represent the solution in terms of Probability Density Function (PDF). The posterior PDF is usually estimated through Monte Carlo importance sampling or approximated by running global optimization methods multiple times (Sen and Stoffa (1996)).

In order to demonstrate the reliability of CPSO for sampling the PDF compared to classical PSO, we run both algorithms multiple times on the highly multi-modal 2D Rastrigin function. We perform 100 runs for each case and sample the model space with a total of 100000 samples, each run starting with different initial models. The resulting posterior density distributions produced by a Kernel Density Estimator (KDE) are shown in Figure C.1 along with the distribution from a MCMC sampler.

Even though PSO detects the principal modes, it fails at identifying the central mode as the principal one. For each run, PSO seems to converge prematurely in different local minima

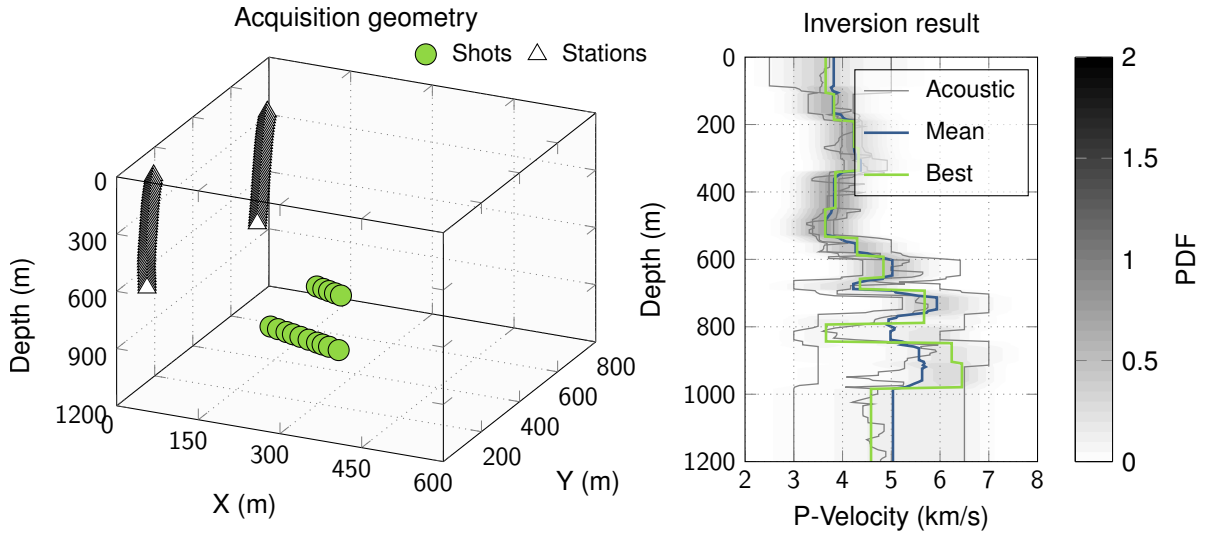

Figure C.2: (Left) 3D acquisition geometry. (Right) P-wave velocity model obtained from CPSO tomography. The 95 percent confidence intervals are represented by the grey lines.

without being able to escape from it. On the other hand, CPSO successfully samples the 9 principal modes and the central mode is correctly identified. Therefore, by running CPSO multiple times, we are able to sample the most significant part of the PDF. The practical consequence is that CPSO is a more reliable method for uncertainty quantification.

C.3 Propagation of velocity uncertainties to locations

C.3.1 Inversion

We apply the CPSO tomography algorithm on a real data set recorded in the context of hydraulic fracturing. The acquisition geometry is represented in Figure C.2 (left) with the green circles being the perforation shots and the white triangles the receivers. The velocity model is parametrized with 15 layers based on the acoustic log, and we invert for the V_p velocities, the V_p/V_s ratios and the interface depths of each layer. We run the inversion for the 45 parameters with 32 particles. A run is stopped when 200 iterations are performed. We run the CPSO algorithm 50 times to sample the model space sufficiently for uncertainty quantification (320000 models sampled). The whole tomography lasted 3.6 minutes using a total of 96 cores out of 104 (four sockets platform made of 4 Intel® Xeon® Platinum 8164 CPU, 26 cores @ 2.00 GHz each). The resulting density plots for the P-wave velocity along with the mean and best models are shown in Figure C.2 (right).

C.3.2 Acceptable models

Let us define the theoretical relationship between the traveltime and the velocity model $\mathbf{v}$, the coordinates of the sources $\mathbf{s}$ and the receivers $\mathbf{r}$, by

$$t^{calc}(\mathbf{s}, \mathbf{r}, \mathbf{v}) = t^{calc}(\mathbf{v}) \tag{C.3}$$

with source and receiver locations being considered as constants. Traveltimes are calculated using an Eikonal solver Noble, Gesret and Belayouni (2014) that generates accurate traveltime grids using a finite-difference scheme.

The velocity model space is sampled by running the CPSO tomography algorithm multiple times. Among all the models visited by the swarm, many models that lie within the uncertainties can be considered acceptable. We assume that the residuals (for P- or S- waves) follow a gaussian distribution with zero mean and variance σ^2, which reads

$$t_i^{obs} - t_i^{calc} \sim \mathcal{N}\left(0, \sigma^2\right). \tag{C.4}$$

Therefore, the standard deviation S of the errors is the root-mean-square-error (RMSE) and follows a $\mathcal{X}^2$ distribution with N degrees of freedom (N being the number of P- or S- wave observations), following

$$\frac{NS^2}{\sigma^2} = \frac{N\frac{1}{N}\sum_{i=1}^{N}\left(t_i^{obs} - t_i^{calc}\right)^2}{\sigma^2} = \frac{N\mathrm{RMSE}^2}{\sigma^2} \sim \mathcal{X}^2_N. \tag{C.5}$$

Given a confidence interval $(1-\alpha)\,100\%$, we define the ensemble of acceptable models as

$$\mathbf{V} = \left\{ \mathbf{v} \mid \mathcal{X}^2_{\frac{\alpha}{2},N} \leq \frac{N\mathrm{RMSE}^2}{\sigma^2} \leq \mathcal{X}^2_{1-\frac{\alpha}{2},N} \right\}. \tag{C.6}$$

C.3.3 Velocity models clustering

In order to propagate the velocity model uncertainties to the microseismic locations, the PDF of the event location should be summed over a subset distributed accordingly to the PDF of the velocity model. As previously mentioned, the solution is not unique and many models can explain the data. Thus, defining a reliable subset of velocity models for the propagation is a difficult task. We propose to use an unsupervised clustering algorithm to automatically find structures in the acceptable models and consequently determine a subset of representative velocity models for the propagation. Given a data set $(mathbf\,m_1, \ldots, mathbf\,m_N)$, the clustering problem consists in minimizing the sum of the distances between the models in the data set and the centroids of clusters to which they are assigned, according to

$$J(\mathbf{c}, \boldsymbol{\mu}) = \sum_{i=1}^{N} D\left(mathbf\,m_i, \mu_{c_i}\right) = \sum_{i=1}^{N}\sum_{j=1}^{d} \omega_j \left(m_{i,j} - \mu_{c_i,j}\right)^2 \tag{C.7}$$

where d is the number of parameters describing a model $mathbf\,m_i$, $\mathbf{c} = (c_1, \ldots, c_N)$ and $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K)$ are respectively the indices and centroids of clusters to which the models are assigned, D is a function that computes the weighted distance between a model and a cluster centroid, ω_j is the weight associated to the parameter j which can be useful to ignore some parameters that are not informative (e.g. the parameters associated to the first and last layers). Most of the unsupervised clustering algorithms require the number of clusters K to be set prior to the clustering. One of the simplest method to determine the optimal K is to calculate the percentage of variance explained for a given number of clusters. The optimal K is chosen so that adding another cluster does not improve the explained variance. Many other methods have been developed to determine the right number of clusters, yet the discussion of the best method is beyond the scope of this appendix. Given the large number of models and dimensions, we use the Mini-batch K-Means algorithm (Sculley (2010)) with a batch size of 20% to clusterize our acceptable models. The parameters are normalized before the clustering.

Figure C.3 (left) shows the percentage of explained variance as a function of the number of clusters. The optimal number of clusters is around $K = 120$ with 98% of the variance being explained. Therefore, we propagate the uncertainties associated to the velocity models by relocating the fifteen perforation shots in the 120 centroid models. The P-wave centroid models

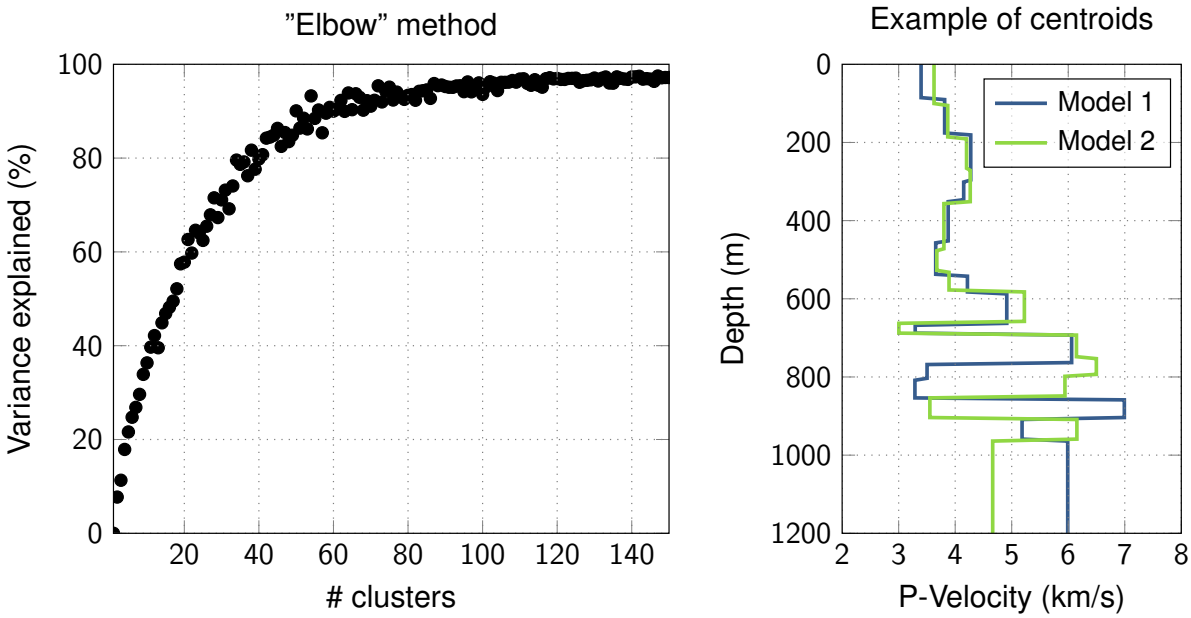

Figure C.3: (Left) "Elbow" method to determine the optimal number of clusters K. (Right) P-wave centroid models of the two most populated clusters.

for the two most populated clusters are shown in Figure C.3 (right). The two models are similar in the constrained zone where the receivers are deployed and differ below with an alternation of fast an slow layers.

For a given hypocenter location **l**, we write the new PDF as

$$P\left(\mathbf{l} \,\middle|\, \mathbf{t}^{obs}, \boldsymbol{\mu}\right) \propto \frac{1}{N} \sum_{i=1}^{K} N_{\mu_i} P\left(\mathbf{t}^{obs} \,\middle|\, \mathbf{l}, \boldsymbol{\mu}_i\right) P\left(\mathbf{l}\right) \tag{C.8}$$

where N_{μ_i} is the number of models assigned to cluster i, $P\left(\mathbf{t}^{obs} \,\middle|\, \mathbf{l}, \boldsymbol{\mu}_i\right)$ is the likelihood that links the event location and the centroid (i.e. velocity model) to the observed traveltimes $\mathbf{t}^{obs}$, $P\left(\mathbf{l}\right)$ describes all the information we know about the location prior to the measurements of the traveltimes. The PDFs of the location for two shots resulting from the propagation in the 120 centroid models is represented in Figure C.4. The true locations are included in the 68 percent confidence intervals which demonstrates the reliability of our tomography algorithm velocity model uncertainty quantification.

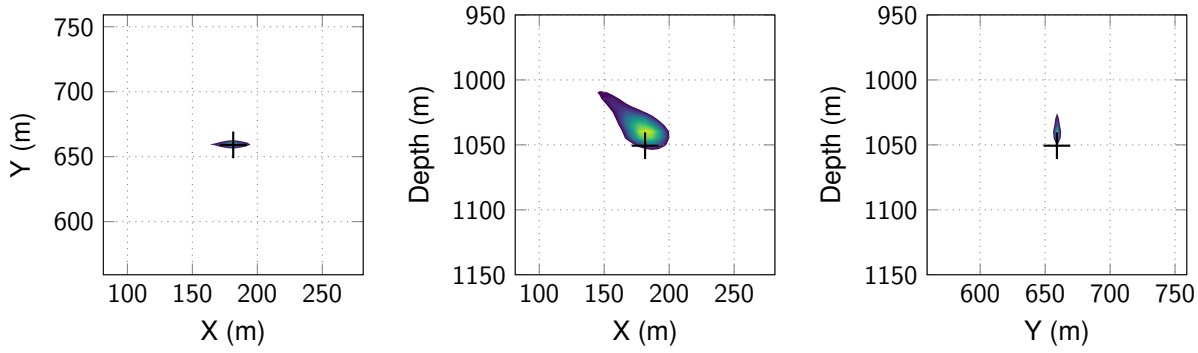

Figure C.4: 68 percent confidence intervals built from the new PDF for the perforation shot 6. The true locations are marked by the black crosses.

C.4 Conclusion

In this appendix, we demonstrated the robustness of CPSO in quantifying uncertainties. We obtained more accurate microseismic locations with reliable uncertainties by propagating the velocity uncertainties. The cluster analysis of the acceptable models not only allows us to define a subset of velocity models, but can potentially speed up the location computation time as it determines the correct number of clusters required to sufficiently represent the sampled PDF.

Bibliography

Akaike H. A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control* 1974;**19**:716–23.

Aki K. Space and time spectra of stationary stochastic waves, with special reference to microtremors. *Bull Earth Res Inst* 1957;**35**:415–56.

Akimoto Y, Auger A, Hansen N. Comparison-based natural gradient optimization in high dimension. In. *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation - Gecco '14*. New York, New York, USA: ACM Press, 2014, 373–80.

Akram J, Eaton DW. A review and appraisal of arrival-time picking methods for downhole microseismic data. *Geophysics* 2016;**81**:KS71–91.

Akram J, Ovcharenko O, Peter D. A robust neural network-based approach for microseismic event detection. In. *SEG Technical Program Expanded Abstracts 2017*. Society of Exploration Geophysicists, 2017, 2929–33.

Allen R. Automatic phase pickers: Their present use and future prospects. *Bulletin of the Seismological Society of America* 1982;**72**:S225–42.

Amdahl GM. Validity of the single processor approach to achieving large scale computing capabilities. In. *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference on - Afips '67 (Spring)*. New York, New York, USA: ACM Press, 1967, 483.

Angeline PJ, Saunders GM, Pollack JB. An evolutionary algorithm that constructs recurrent neural networks. *IEEE transactions on Neural Networks* 1994;**5**:54–65.

Angeline PJ. Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. *Lecture Notes in Computer Science: Evolutionary Programming VII* 1998;**1447**:601–10.

Auger A, Hansen N. A Restart CMA Evolution Strategy With Increasing Population Size. *2005 IEEE Congress on Evolutionary Computation* 2005;**2**:1769–76.

Back T, Hammel U, Schwefel H-P. Evolutionary computation: comments on the history and current state. *IEEE Transactions on Evolutionary Computation* 1997;**1**:3–17.

Baer M, Kradolfer U. An automatic phase picker for local and teleseismic events. *Bulletin of the Seismological Society of America* 1987;**77**:1437–45.

Baillard C, Crawford WC, Ballu V *et al.* An Automatic Kurtosis-Based P- and S-Phase Picker Designed for Local Seismic Networks. *Bulletin of the Seismological Society of America*

2014;**104**:394–409.

Barros T, Ferrari R, Krummenauer R *et al.* Differential evolution-based optimization procedure for automatic estimation of the common-reflection surface traveltime parameters. *Geophysics* 2015;**80**:WD189–200.

Bavelas A. Communication Patterns in Task-Oriented Groups. *The Journal of the Acoustical Society of America* 1950;**22**:725–30.

Belhadj J, Romary T, Gesret A *et al.* New parameterizations for Bayesian seismic tomography. *Inverse Problems* 2018;**34**:065007.

Beyer H-G, Beyer H-G, Schwefel H-P *et al.* Evolution strategies – A comprehensive introduction. *Natural Computing* 2002;**1**:3–52.

Beyreuther M, Barsch R, Krischer L *et al.* ObsPy: A Python Toolbox for Seismology. *Seismological Research Letters* 2010;**81**:530–3.

Billings SD. Simulated annealing for earthquake location. *Geophysical Journal International* 1994;**118**:680–92.

Bishop JM. Stochastic searching networks. In. *Artificial Neural Networks, 1989., First Iee International Conference on (Conf. Publ. No. 313)*. IET, 1989, 329–31.

Bodin T, Salmon M, Kennett BLN *et al.* Probabilistic surface reconstruction from multiple data sets: An example for the Australian Moho. *Journal of Geophysical Research: Solid Earth* 2012;**117**:1–13.

Bodin T, Sambridge M. Seismic tomography with the reversible jump algorithm. *Geophysical Journal International* 2009;**178**:1411–36.

Boor C de. On calculating with B-splines. *Journal of Approximation Theory* 1972;**6**:50–62.

Boschetti F, Dentith MC, List RD. Inversion of seismic refraction data using genetic algorithms. *Geophysics* 1996;**61**:1715–27.

Bottero A, Gesret A, Romary T *et al.* Stochastic seismic tomography by interacting Markov chains. *Geophysical Journal International* 2016;**207**:374–92.

Brantut N, Schubnel A, Guéguen Y. Damage and rupture dynamics at the brittle-ductile transition: The case of gypsum. *Journal of Geophysical Research* 2011;**116**:B01404.

Buchen PW, Ben-Hador R. Free-mode surface-wave computations. *Geophysical Journal International* 1996;**124**:869–87.

Bunks C, Saleck FM, Zaleski S *et al.* Multiscale seismic waveform inversion. *Geophysics* 1995;**60**:1457–73.

Calvez JHL, Craven ME, Klem RC *et al.* Real-Time Microseismic Monitoring of Hydraulic Fracture Treatment: A Tool To Improve Completion and Reservoir Management. *SPE Hydraulic Fracturing Technology Conference* 2007:7.

Capon J, Greenfield RJ, Kolker RJ. Multidimensional maximum-likelihood processing of a large aperture seismic array. *Proceedings of the IEEE* 1967;**55**:192–211.

Carlisle A, Dozier G. An Off-The-Shelf PSO. *Population English Edition* 2001;**1**:1–6.

Cary PW, Chapman CH. Automatic 1-D waveform inversion of marine seismic refraction data.

Geophysical Journal International 1988;**93**:527–46.

Chapman C. Ray theory and its extensions: WKBJ and Maslov seismogram. *Journal of Geophysics* 1985;**58**:27–43.

Chavent G. *Identification of Functional Parameters in Partial Differential Equations*., 1974.

Chen S, Montgomery J, Bolufé-Röhler A. Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution. *Applied Intelligence* 2015;**42**:514–26.

Cipolla C, Maxwell S, Mack M *et al.* A Practical Guide to Interpreting Microseismic Measurements. *SPE North American Unconventional Gas Conference and Exhibition* 2011:1–28.

Clerc M, Kennedy J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation* 2002;**6**:58–73.

Clerc M. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In. *Proceedings of the 1999 Congress on Evolutionary Computation-Cec99 (Cat. No. 99th8406)*. Vol 3. IEEE, 1999, 1951–7.

Conti E, Madhavan V, Such FP *et al.* Improving Exploration in Evolution Strategies for Deep Reinforcement Learning via a Population of Novelty-Seeking Agents. 2017.

Červený V. Ray tracing algorithms in three-dimensional laterally varying layered structures. In. *Seismic Tomography*. Dordrecht: Springer Netherlands, 1987, 99–133.

Červený V. *Seismic Ray Theory*. Cambridge: Cambridge University Press, 2001.

Dai H, MacBeth C. The application of back-propagation neural network to automatic picking seismic arrivals from single-component recordings. *Journal of Geophysical Research: Solid Earth* 1997;**102**:15105–13.

Daniels JL, Waters GA, Le Calvez JH *et al.* Contacting More of the Barnett Shale Through an Integration of Real-Time Microseismic Monitoring, Petrophysics, and Hydraulic Fracture Design. In. *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, 2007.

Das S, Suganthan PN. Differential Evolution: A Survey of the State-of-the-Art. *IEEE Transactions on Evolutionary Computation* 2011;**15**:4–31.

Davis L. Handbook of genetic algorithms. 1991.

De Meersman K, Kendall J-M, Baan M van der. The 1998 Valhall microseismic data set: An integrated study of relocated sources, seismic multiplets, and S-wave splitting. *Geophysics* 2009;**74**:B183–95.

Deb K. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001.

Deichmann N, Giardini D. Earthquakes Induced by the Stimulation of an Enhanced Geothermal System below Basel (Switzerland). *Seismological Research Letters* 2009;**80**:784–98.

Delbos F, Gilbert JC, Glowinski R *et al.* Constrained optimization in seismic reflection tomography: a Gauss-Newton augmented Lagrangian approach. *Geophysical Journal International* 2006;**164**:670–84.

Delprat-Jannaud F, Lailly P. Ill-posed and well-posed formulations of the reflection travel time tomography problem. *Journal of Geophysical Research: Solid Earth* 1993;**98**:6589–605.

Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents.

IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 1996;**26**:29–41.

Drosinos N, Koziris N. Performance comparison of pure MPI vs hybrid MPI-OpenMP parallelization models on SMP clusters. *18th International Parallel and Distributed Processing Symposium, 2004 Proceedings* 2004;**00**:15–24.

Duane S, Kennedy A, Pendleton BJ *et al.* Hybrid Monte Carlo. *Physics Letters B* 1987;**195**:216–22.

Eberhart R, Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization. In. *Proceedings of the 2000 Congress on Evolutionary Computation. Cec00 (Cat. No.00TH8512)*. Vol 1. IEEE, 2000, 84–8.

Eisner L, Duncan PM, Heigl WM *et al.* Uncertainties in passive seismic monitoring. *The Leading Edge* 2009;**28**:648–55.

Ekinci YL, Balkaya Ç, Göktürkler G *et al.* Model parameter estimations from residual gravity anomalies due to simple-shaped sources using Differential Evolution Algorithm. *Journal of Applied Geophysics* 2016;**129**:133–47.

Engelbrecht A. Particle swarm optimization: Velocity initialization. In. *2012 Ieee Congress on Evolutionary Computation*. IEEE, 2012, 1–8.

Evers GI, Ben Ghalia M. Regrouping particle swarm optimization: A new global optimization algorithm with improved performance consistency across benchmarks. In. *2009 Ieee International Conference on Systems, Man and Cybernetics*. IEEE, 2009, 3901–8.

Fernández Martínez JL, Mukerji T, García Gonzalo E *et al.* Reservoir characterization and inversion uncertainty via a family of particle swarm optimizers. *Geophysics* 2012;**77**:M1–M16.

Figueiredo EM, Ludermir TB. Investigating the use of alternative topologies on performance of the PSO-ELM. *Neurocomputing* 2014;**127**:4–12.

Fletcher R, Powell MJD. A Rapidly Convergent Descent Method for Minimization. *The Computer Journal* 1963;**6**:163–8.

Fogel DB. Applying evolutionary programming to selected traveling salesman problems. *Cybernetics and systems* 1993;**24**:27–36.

Fogel DB. An overview of evolutionary programming. In. *Evolutionary Algorithms*. Springer, 1999, 89–109.

Fonseca CM, Fleming PJ. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary computation* 1995;**3**:1–16.

Fonseca CM, Fleming PJ. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. I. A unified formulation. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 1998;**28**:26–37.

Gentili S, Michelini A. Automatic picking of P and S phases using a neural tree. *Journal of Seismology* 2006;**10**:39–63.

Gerhard Pratt, Shin C, Hicks. Gauss-Newton and full Newton methods in frequency-space seismic waveform inversion. *Geophysical Journal International* 1998;**133**:341–62.

Gesret A, Desassis N, Noble M *et al.* Propagation of the velocity model uncertainties to the seismic event location. *Geophysical Journal International* 2015;**200**:52–66.

Goldstein P, Archuleta RJ. Array analysis of seismic signals. *Geophysical Research Letters*

1987;**14**:13–6.

Gong Y, Fukunaga A. Distributed island-model genetic algorithms using heterogeneous parameter settings. In. *2011 Ieee Congress of Evolutionary Computation (Cec)*. IEEE, 2011, 820–7.

Gong YJ, Chen WN, Zhan ZH *et al.* Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. 2015;**34**:286–300.

Goudie RJB, Turner RM, De Angelis D *et al.* MultiBUGS: A parallel implementation of the BUGS modelling framework for faster Bayesian inference. 2017:1–19.

Grandis H, Menvielle M, Roussignol M. Bayesian inversion with Markov chains-I. The magnetotelluricone-dimensional case. *Geophysical Journal International* 1999;**138**:757–68.

Grayver AV, Kuvshinov AV. Exploring equivalence domain in nonlinear inverse problems using Covariance Matrix Adaption Evolution Strategy (CMAES) and random sampling. *Geophysical Journal International* 2016;**205**:971–87.

Green PJ. Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination. *Biometrika* 1995;**82**:711.

Hajizadeh Y, Christie MA, Demyanov V. History matching with differential evolution approach; a look at new search strategies. In. *SPE Europec/Eage Annual Conference and Exhibition*. Society of Petroleum Engineers, 2010.

Han D-x, Wang G-y. Application of Particle Swarm Optimization to Seismic Location. *2009 Third International Conference on Genetic and Evolutionary Computing* 2009:641–4.

Hansen N, Müller SD, Koumoutsakos P. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary computation* 2003;**11**:1–18.

Hansen N, Ostermeier A. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In. *Proceedings of Ieee International Conference on Evolutionary Computation*. IEEE, 1996, 312–7.

Hansen N. Errata / Addenda for A Method for Handling Uncertainty in Evolutionary Optimization With an Application to Feedback Control of Combustion. *IEEE Transactions on Evolutionary Computation* 2010;**13**:180–97.

Hansen N. The CMA evolution strategy: A tutorial. 2011;**102**:1–34.

Hansen PC, O'Leary DP. The Use of the L-Curve in the Regularization of Discrete Ill-Posed Problems. *SIAM Journal on Scientific Computing* 1993;**14**:1487–503.

Hart DI. Automated Picking of Seismic First-Arrivals with Neural Networks. In. 2003, 13–30.

Haskell NA. The dispersion of surface waves on multilayered media. *Bulletin of the seismological Society of America* 1953;**43**:17–34.

Hewlett PS, Mendel G. Experiments in Plant Hybridisation. *Biometrics* 1966;**22**:636.

Holland JH. Genetic Algorithms and the Optimal Allocation of Trials. *SIAM Journal on Computing* 1973;**2**:88–105.

Hooke R, Jeeves TA. "Direct Search" Solution of Numerical and Statistical Problems. *Journal of*

the ACM 1961;**8**:212–29.

Hunter JD. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering* 2007;**9**:90–5.

Improta L, Zollo A, Herrero A *et al.* Seismic imaging of complex structures by non-linear traveltime inversion of dense wide-angle data: application to a thrust belt. *Geophysical Journal International* 2002;**151**:264–78.

Iwan M, Akmeliawati R, Faisal T *et al.* Performance Comparison of Differential Evolution and Particle Swarm Optimization in Constrained Optimization. *Procedia Engineering* 2012;**41**:1323–8.

Jones AG, Hutton R. A multi-station magnetotelluric study in southern Scotland – II. Monte-Carlo inversion of the data and its geophysical and tectonic implications. *Geophysical Journal International* 1979;**56**:351–68.

Jones AG, Olafsdottir B, Tiikkainen J. Geomagnetic induction studies in Scandinavia. III Magnetotelluric observations. *Journal of Geophysics Zeitschrift Geophysik* 1983;**54**:35–50.

Jones E, Oliphant T, Peterson P. SciPy: Open Source Scientific Tools for Python. 2001.

Julian B, Gubbins D. Three-dimensional seismic ray tracing. *Journal of Geophysics* 1977;**43**:95–114.

Kassahun Y, Sommer G. Efficient reinforcement learning through Evolutionary Acquisition of Neural Topologies. In. *ESANN*. 2005, 259–66.

Keilis-Borok VI, Yanovskaya TB. Inverse seismic problems (structural review). *Geophys J* 1967;**13**:223–33.

Kennedy J, Eberhart R. Particle swarm optimization. In. *Proceedings of Icnn'95 - International Conference on Neural Networks*. Vol 4. IEEE, 1995, 1942–8.

Kennedy J. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In. *Proceedings of the 1999 Congress on Evolutionary Computation-Cec99 (Cat. No. 99th8406)*. Vol 3. IEEE, 1999, 1931–8.

Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by Simulated Annealing. *Science* 1983;**220**:671–80.

Koh B-I, George AD, Haftka RT *et al.* Parallel asynchronous particle swarm optimization. *International Journal for Numerical Methods in Engineering* 2006;**67**:578–95.

Koren Z, Mosegaard K, Landa E *et al.* Monte Carlo estimation and resolution analysis of seismic background velocities. *Journal of Geophysical Research: Solid Earth* 1991;**96**:20289–99.

Koza JR. *Genetic Programming II, Automatic Discovery of Reusable Subprograms*. MIT Press, Cambridge, MA, 1992.

Koza JR. Genetic programming as a means for programming computers by natural selection. *Statistics and computing* 1994;**4**:87–112.

Krishnanand KN, Ghose D. Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications. *Multiagent and Grid Systems* 2006;**2**:209–22.

Küperkoch L, Meier T, Lee J *et al.* Automated determination of P -phase arrival times at regional and local distances using higher order statistics. *Geophysical Journal International* 2010,

DOI: 10.1111/j.1365-246X.2010.04570.x.

Lagos SR, Sabbione JI, Velis DR. Very fast simulated annealing and particle swarm optimization for microseismic event location. In. *SEG Technical Program Expanded Abstracts 2014*. Society of Exploration Geophysicists, 2014, 2188–92.

LeCun Y, Bottou L, Orr GB *et al.* Efficient BackProp. In. Vol 75. 1998, 9–50.

Lehman J, Chen J, Clune J *et al.* Safe Mutations for Deep and Recurrent Neural Networks through Output Gradients. 2017, DOI: 10.1145/3205455.

Leonard M. Comparison of Manual and Automatic Onset Time Picking. *Bulletin of the Seismological Society of America* 2000;**90**:1384–90.

Liu DC, Nocedal J. On the limited memory BFGS method for large scale optimization. *Mathematical Programming* 1989;**45**:503–28.

Lomax A, Michelini A, Curtis A. Earthquake location, direct, global-search methods. 2009:2–449.

Louie JN. Faster, better: shear-wave velocity to 100 meters depth from refraction microtremor arrays. *Bulletin of the Seismological Society of America* 2001;**91**:347–64.

Luo Y, Schuster GT. Wave-equation traveltime inversion. *Geophysics* 1991;**56**:645–53.

Luu K, Noble M, Gesret A *et al.* A parallel competitive Particle Swarm Optimization for non-linear first arrival traveltime tomography and uncertainty quantification. *Computers & Geosciences* 2018;**113**:81–93.

Luu K, Noble M, Gesret A. A competitive particle swarm optimization for nonlinear first arrival traveltime tomography. In. *SEG Technical Program Expanded Abstracts 2016*. Society of Exploration Geophysicists, 2016, 2740–4.

Maeda N. A Method for Reading and Checking Phase Time in Auto-Processing System of Seismic Wave Data. *Zisin (Journal of the Seismological Society of Japan 2nd ser)* 1985;**38**:365–79.

Maity D, Aminzadeh F, Karrenbach M. Novel hybrid artificial neural network based autopicking workflow for passive seismic data. *Geophysical Prospecting* 2014;**62**:834–47.

Malinverno A, Briggs VA. Expanded uncertainty quantification in inverse problems: Hierarchical Bayes and empirical Bayes. *Geophysics* 2004;**69**:1005–10016.

Malinverno A, Leaney S. A Monte Carlo method to quantify uncertainty in the inversion of zero-offset VSP data. In. *SEG Technical Program Expanded Abstracts 2000*. Society of Exploration Geophysicists, 2000, 2393–6.

Malinverno A, Torres-Verdín C. Bayesian inversion of DC electrical measurements with uncertainties for reservoir monitoring. *Inverse Problems* 2000;**16**:1343–56.

Maxwell S, Urbancic T, Steinsberger N *et al.* Microseismic Imaging of Hydraulic Fracture Complexity in the Barnett Shale. In. *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, 2002.

Maxwell SC, Rutledge J, Jones R *et al.* Petroleum reservoir characterization using downhole microseismic monitoring. *Geophysics* 2010;**75**:75A129–37.

Maxwell SC. Microseismic Location Uncertainty. *CSEG Recorder* 2009;**34**:41–6.

McCormack MD, Zaucha DE, Dushek DW. First-break refraction event picking and seismic data

trace editing using neural networks. *Geophysics* 1993;**58**:67–78.

McKinney W. Data Structures for Statistical Computing in Python. In. *Proceedings of the 9th Python in Science Conference*. 2010, 51–6.

McKinnon KIM. Convergence of the Nelder–Mead Simplex Method to a Nonstationary Point. *SIAM Journal on Optimization* 1998;**9**:148–58.

Mendel G. Versuche über Pflanzenhybriden. *Verhandlungen des naturforschenden Vereines in Brunn 4: 3* 1866;**44**.

Menke W. *Geophysical data analysis: Discrete inverse theory*. Elsevier/Academic Press, 2012:293.

Mirjalili S, Mirjalili SM, Lewis A. Grey Wolf Optimizer. *Advances in Engineering Software* 2014;**69**:46–61.

Mohamed L, Calderhead B, Filippone M *et al.* Population MCMC methods for history matching and uncertainty quantification. *Computational Geosciences* 2012;**16**:423–36.

Mohamed L, Christie M, Demyanov V. Comparison of Stochastic Sampling Algorithms for Uncertainty Quantification. *SPE Journal* 2010;**15**:31–8.

Mohamed L, Christie MA, Demyanov V *et al.* Application of Particle Swarms for History Matching in the Brugge Reservoir. In. *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, 2010.

Molyneux JB, Schmitt DR. First-break timing: Arrival onset times by direct correlation. *Geophysics* 1999;**64**:1492–501.

Mosegaard K, Tarantola A. Monte Carlo sampling of solutions to inverse problems. *Journal of Geophysical Research: Solid Earth* 1995;**100**:12431–47.

Murat ME, Rudman AJ. Automated first arrival picking: a neural network approach. *Geophysical Prospecting* 1992;**40**:587–604.

Mussi L, Nashed YS, Cagnoni S. GPU-based asynchronous particle swarm optimization. In. *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation - Gecco '11*. New York, New York, USA: ACM Press, 2011, 1555.

Nash SG, Nocedal J. A Numerical Study of the Limited Memory BFGS Method and the Truncated-Newton Method for Large Scale Optimization. *SIAM Journal on Optimization* 1991;**1**:358–72.

Nazarian S, Stokoe II, Kenneth H *et al. Use of spectral analysis of surface waves method for determination of moduli and thicknesses of pavement systems*., 1983.

Neal RM. Bayesian training of backpropagation networks by the hybrid Monte Carlo method. *Dept of Computer Science, University of Toronto, Tech . . .* 1992:1–21.

Neal RM. *Bayesian Learning for Neural Networks*. New York, NY: Springer New York, 1996:341.

Neal RM. *Handbook of Markov Chain Monte Carlo*. Brooks S, Gelman A, Jones G, et al. (eds.). Chapman; Hall/CRC, 2011:113–62.

Neiswanger W, Wang C, Xing E. Asymptotically Exact, Embarrassingly Parallel MCMC. 2013:1–16.

Nelder JA, Mead R. A Simplex Method for Function Minimization. *The Computer Journal*

1965;**7**:308–13.

Nemeth T, Normark E, Qin F. Dynamic smoothing in crosswell traveltime tomography. *Geophysics* 1997;**62**:168–76.

Nicolas A, Fortin J, Regnet J-B *et al.* Brittle and semi-brittle behaviours of a carbonate rock: Influence of water and temperature. *Geophysical Journal International* 2016;**206**:438–56.

Nielsen MA. *Neural networks and deep learning*. Determination press USA, 2015.

Noble M, Gesret A, Belayouni N. Accurate 3-D finite difference computation of traveltimes in strongly heterogeneous media. *Geophysical Journal International* 2014;**199**:1572–85.

Noble M, Thierry P, Taillandier C *et al.* High-performance 3D first-arrival traveltime tomography. *The Leading Edge* 2010;**29**:86–93.

Oliphant T. *A guide to NumPy*. USA: Trelgol Publishing, 2006.

Padhye N, Mittal P, Deb K. Feasibility Preserving Constraint-Handling Strategies for Real Parameter Evolutionary Optimization. 2015.

Pallero J, Fernández-Martínez J, Bonvalot S *et al.* Gravity inversion and uncertainty assessment of basement relief via Particle Swarm Optimization. *Journal of Applied Geophysics* 2015;**116**:180–91.

Park CB, Miller RD, Xia J. Multichannel analysis of surface waves. *Geophysics* 1999;**64**:800–8.

Pedregosa F, Varoquaux G, Gramfort A *et al.* Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 2012;**12**:2825–30.

Pham DT, Ghanbarzadeh A, Koç E *et al.* -The Bees Algorithm—A Novel Tool for Complex Optimisation Problems. In. *Intelligent Production Machines and Systems*. Elsevier, 2006, 454–9.

Piana Agostinetti N, Giacomuzzi G, Malinverno A. Local three-dimensional earthquake tomography by trans-dimensional Monte Carlo sampling. *Geophysical Journal International* 2015;**201**:1598–617.

Piccand S, O'Neill M, Walker J. On the scalability of particle swarm optimisation. In. *2008 Ieee Congress on Evolutionary Computation (Ieee World Congress on Computational Intelligence)*. IEEE, 2008, 2505–12.

Plessix R-E. A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. *Geophysical Journal International* 2006;**167**:495–503.

Podvin P, Lecomte I. Finite difference computation of traveltimes in very contrasted velocity models: a massively parallel approach and its associated tools. *Geophysical Journal International* 1991;**105**:271–84.

Poormirzaee R, Moghadam RH, Zarean A. Inversion seismic refraction data using particle swarm optimization: a case study of Tabriz, Iran. *Arabian Journal of Geosciences* 2015;**8**:5981–9.

Poormirzaee R. S-wave velocity profiling from refraction microtremor Rayleigh wave dispersion curves via PSO inversion algorithm. *Arabian Journal of Geosciences* 2016;**9**:673.

Powell MJD. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal* 1964;**7**:155–62.

Press F. Earth models obtained by Monte Carlo Inversion. *Journal of Geophysical Research*

1968;**73**:5223–34.

Press F. Earth models consistent with geophysical data. *Physics of the Earth and Planetary Interiors* 1970;**3**:3–22.

Price K, Storn RM, Lampinen JA. *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

Price K. Differential evolution: a fast and simple numerical optimizer. In. *Proceedings of North American Fuzzy Information Processing*. IEEE, 1996, 524–7.

Rabenseifner R, Hager G, Jost G. Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-Core SMP Nodes. In. *2009 17th Euromicro International Conference on Parallel, Distributed and Network-Based Processing*. IEEE, 2009, 427–36.

Ramirez AL, Nitao JJ, Hanley WG *et al.* Stochastic inversion of electrical resistivity changes using a Markov Chain Monte Carlo approach. *Journal of Geophysical Research: Solid Earth* 2005;**110**:1–18.

Rawlinson N, Pozgay S, Fishwick S. Seismic tomography: A window into deep Earth. *Physics of the Earth and Planetary Interiors* 2010;**178**:101–35.

Rechenberg I. Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. 1973.

Romary T. Bayesian inversion by parallel interacting Markov chains. *Inverse Problems in Science and Engineering* 2010;**18**:111–30.

Ronald E, Schoenauer M. Genetic Lander: An experiment in accurate neuro-genetic control. In. *International Conference on Parallel Problem Solving from Nature*. Springer, 1994, 452–61.

Rostami S, Neri F. A fast hypervolume driven selection mechanism for many-objective optimisation problems. *Swarm and Evolutionary Computation* 2017;**34**:50–67.

Rothert E, Shapiro SA. Microseismic monitoring of borehole fluid injections: Data modeling and inversion for hydraulic properties of rocks. *Geophysics* 2003;**68**:685–9.

Rumpf M, Tronicke J. Assessing uncertainty in refraction seismic traveltime inversion using a global inversion strategy. *Geophysical Prospecting* 2015;**63**:1188–97.

Růžek B, Kvasnička M. Differential Evolution Algorithm in the Earthquake Hypocenter Location. *Pure and Applied Geophysics* 2001;**158**:667–93.

Ryberg T, Haberland C. Bayesian inversion of refraction seismic traveltime data. *Geophysical Journal International* 2018;**212**:1645–56.

Sabbione JI, Velis D. Automatic first-breaks picking: New strategies and algorithms. *Geophysics* 2010;**75**:V67–76.

Saka M, Hasançebi O, Geem Z. Metaheuristics in structural optimization and discussions on harmony search algorithm. *Swarm and Evolutionary Computation* 2016;**28**:88–97.

Sambridge M, Drijkoningen G. Genetic algorithms in seismic waveform inversion. *Geophysical Journal International* 1992;**109**:323–42.

Sambridge M, Gallagher K. Earthquake hypocenter location using genetic algorithms. *Bulletin of the Seismological Society of America* 1993;**83**:1467–91.

Sambridge M, Mosegaard K. Monte Carlo Methods in Geophysical Inverse Problems. *Reviews*

of Geophysics 2002;**40**:1009.

Sambridge M. Geophysical inversion with a neighbourhood algorithm-I. Searching a parameter space. *Geophysical Journal International* 1999;**138**:479–94.

Sambridge M. A Parallel Tempering algorithm for probabilistic sampling and multimodal optimization. *Geophysical Journal International* 2014;**196**:357–74.

Saragiotis CD, Hadjileontiadis LJ, Panas SM. PAI-S/K: A robust automatic seismic P phase arrival identification scheme. *IEEE Transactions on Geoscience and Remote Sensing* 2002;**40**:1395–404.

Sasaki S. Characteristics of microseismic events induced during hydraulic fracturing experiments at the Hijiori hot dry rock geothermal energy site, Yamagata, Japan. *Tectonophysics* 1998;**289**:171–88.

Scales JA, Snieder R. To Bayes or not to Bayes? *Geophysics* 1997;**62**:1045–6.

Scales JA, Tenorio L. Prior information and uncertainty in inverse problems. *Geophysics* 2001;**66**:389–97.

Schmidt R. Multiple emitter location and signal parameter estimation. *IEEE transactions on antennas and propagation* 1986;**34**:276–80.

Schott J-J, Roussignol M, Menvielle M *et al.* Bayesian inversion with Markov chains-II. The one-dimensional DC multilayer case. *Geophysical Journal International* 1999;**138**:769–83.

Schulze-Riegert R, Axmann J, Haase O *et al.* Optimization Methods for History Matching of Complex Reservoirs. In. *SPE Reservoir Simulation Symposium*. Society of Petroleum Engineers, 2001.

Schutte JF, Reinbolt JA, Fregly BJ *et al.* Parallel global optimization with the particle swarm algorithm. *International Journal for Numerical Methods in Engineering* 2004;**61**:2296–315.

Schwefel H-P. Evolution strategies: A family of non-linear optimization techniques based on imitating some principles of organic evolution. *Annals of Operations Research* 1984;**1**:165–7.

Sculley D. Web-scale k-means clustering. *Proceedings of the 19th international conference on World wide web WWW 10* 2010:1177.

Sedlak P, Hirose Y, Enoki M *et al.* Arrival Time Detection in Thin Multilayer Plates on the Basis of Akaike Information Criterion. *Journal of Acoustic Emission* 2008;**26**:182–8.

Sen MK, Stoffa PL. Bayesian inference, Gibbs' sampler and uncertainty estimation in geophysical inversion. *Geophysical Prospecting* 1996;**44**:313–50.

Shapiro NM, Campillo M. Emergence of broadband Rayleigh waves from correlations of the ambient seismic noise. *Geophysical Research Letters* 2004;**31**.

Shaw R, Srivastava S. Particle swarm optimization: A new tool to invert geophysical data. *Geophysics* 2007;**72**:F75–83.

Shenfield A, Rostami S. Multi-objective evolution of artificial neural networks in multi-class medical diagnosis problems with class imbalance. In. *2017 Ieee Conference on Computational Intelligence in Bioinformatics and Computational Biology (Cibcb)*. IEEE, 2017, 1–8.

Sheriff RE, Geldart LP. *Exploration Seismology*. Cambridge University Press (ed.). Cambridge:

Cambridge University Press, 1995.

Shi Y, Eberhart RC. A modified particle swarm optimizer. In. *1998 Ieee International Conference on Evolutionary Computation Proceedings. Ieee World Congress on Computational Intelligence (Cat. No.98TH8360)*. IEEE, 1998, 69–73.

Siebel NT, Sommer G. Evolutionary reinforcement learning of artificial neural networks. *International Journal of Hybrid Intelligent Systems* 2007;**4**:171–83.

Sleeman R, Eck T van. Robust automatic P-phase picking: an on-line implementation in the analysis of broadband seismogram recordings. *Physics of the Earth and Planetary Interiors* 1999;**113**:265–75.

Socco L, Strobbia C. Surface-wave method for near-surface characterization: a tutorial. *Near Surface Geophysics* 2004;**2**:165–85.

Socco LV, Boiero D. Improved Monte Carlo inversion of surface wave data. *Geophysical Prospecting* 2008;**56**:357–71.

Song X, Tang L, Lv X *et al.* Application of particle swarm optimization to interpret Rayleigh wave dispersion curves. *Journal of Applied Geophysics* 2012;**84**:1–13.

Sörensen K. Metaheuristics-the metaphor exposed. *International Transactions in Operational Research* 2015;**22**:3–18.

Stanley KO, Miikkulainen R. Evolving neural networks through augmenting topologies. *Evolutionary computation* 2002;**10**:99–127.

Storn R, Price K. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization* 1997;**11**:341–59.

Storn R. Real-world applications in the communications industry - when do we resort to Differential Evolution? In. *2017 Ieee Congress on Evolutionary Computation (Cec)*. IEEE, 2017, 765–72.

Such FP, Madhavan V, Conti E *et al.* Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. 2017.

Taillandier C, Noble M, Chauris H *et al.* First-arrival traveltime tomography based on the adjoint-state method. *Geophysics* 2009, DOI: 10.1190/1.3250266.

Tarantola A, Valette B. Inverse Problems = Quest for Information. *Journal of Geophysics* 1982;**50**:159–70.

Tarantola A. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial; Applied Mathematics, 2005:1816–24.

Thomson WT. Transmission of elastic waves through a stratified solid medium. *Journal of applied Physics* 1950;**21**:89–93.

Tikhonov AN, Goncharsky AV, Stepanov VV *et al. Numerical methods for the solution of ill-posed problems*. Springer Science & Business Media, 2013.

Torczon V. On the Convergence of Pattern Search Algorithms. *SIAM Journal on Optimization* 1997;**7**:1–25.

Toushmalani R. Gravity inversion of a fault by Particle swarm optimization (PSO). *SpringerPlus*

2013;**2**:315.

Trelea IC. The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Information Processing Letters* 2003;**85**:317–25.

Trier J van, Symes WW. Upwind finite-difference calculation of traveltimes. *Geophysics* 1991;**56**:812–21.

Tronicke J, Paasche H, Böniger U. Crosshole traveltime tomography using particle swarm optimization: A near-surface field example. *Geophysics* 2012;**77**:R19–32.

Ulrych TJ, Sacchi MD, Woodbury A. A Bayes tour of inversion: A tutorial. *Geophysics* 2001;**66**:55–69.

Um J, Thurber C. A fast algorithm for two-point seismic ray tracing. *Bulletin of the Seismological Society of America* 1987;**77**:972–86.

Van Den Bergh F, Engelbrecht AP. A study of particle swarm optimization particle trajectories. *Information Sciences* 2006;**176**:937–71.

Van Den Bergh F. An analysis of particle swarm optimizers. 2001.

Veezhinathan J, Wagner D. A neural network approach to first break picking. In. *1990 Ijcnn International Joint Conference on Neural Networks*. IEEE, 1990, 235–40 vol.1.

Venter G, Sobieszczanski-Sobieski J. Parallel Particle Swarm Optimization Algorithm Accelerated by Asynchronous Evaluations. *Journal of Aerospace Computing, Information, and Communication* 2006;**3**:123–37.

Versteeg R. The Marmousi experience: Velocity model determination on a synthetic complex data set. *The Leading Edge* 1994;**13**:927–36.

Vidale J. Finite-difference calculation of travel times. *Bulletin of the Seismological Society of America* 1988;**78**:2062–76.

Vidale J. Finite-difference calculation of traveltimes in three dimensions. *Geophysics* 1990;**55**:521–6.

Warpinski N, Wolhart S, Wright C. Analysis and Prediction of Microseismicity Induced by Hydraulic Fracturing. *SPE Journal* 2004;**9**:24–33.

Warpinski N. Microseismic Monitoring: Inside and Out. *Journal of Petroleum Technology* 2009;**61**:80–5.

Weyland D. A Rigorous Analysis of the Harmony Search Algorithm. *International Journal of Applied Metaheuristic Computing* 2010;**1**:50–60.

White DJ. Two-Dimensional Seismic Refraction Tomography. *Geophysical Journal International* 1989;**97**:223–45.

Whitley D, Rana S, Heckendorn RB. The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology* 1999;**7**:33–47.

Whitley D. A genetic algorithm tutorial. *Statistics and Computing* 1994;**4**:65–85.

Wiggins RA. Monte Carlo inversion of body-wave observations. *Journal of Geophysical Research* 1969;**74**:3171–81.

Wilken D, Rabbel W. On the application of Particle Swarm Optimization strategies on Scholte-

wave inversion. *Geophysical Journal International* 2012;**190**:580–94.

Xiong J, Liu C, Chen Y *et al.* A Non-linear Geophysical Inversion Algorithm for the MT Data Based on Improved Differential Evolution. *Engineering Letters* 2018;**26**.

Yang X-S, Deb S. Cuckoo search via Lévy flights. In. *Nature & Biologically Inspired Computing, 2009. Nabic 2009. World Congress on*. IEEE, 2009, 210–4.

Yang X-S. A new metaheuristic bat-inspired algorithm. In. *Nature Inspired Cooperative Strategies for Optimization (Nicso 2010)*. Springer, 2010, 65–74.

Yang X-S. Flower pollination algorithm for global optimization. In. *International Conference on Unconventional Computing and Natural Computation*. Springer, 2012, 240–9.

Zelt CA, Barton PJ. Three-dimensional seismic refraction tomography: A comparison of two methods applied to data from the Faeroe Basin. *Journal of Geophysical Research: Solid Earth* 1998;**103**:7187–210.

Zhang H, Thurber C, Rowe C. Automatic P-wave arrival detection and picking with multiscale wavelet analysis for single-component recordings. *Bulletin of the Seismological Society of America* 2003;**93**:1904–12.

Zhang J, Toksöz MN. Nonlinear refraction traveltime tomography. *Geophysics* 1998;**63**:1726–37.

Zhang J, Wang C, Shi Y *et al.* Three-dimensional crustal structure in central Taiwan from gravity inversion with a parallel genetic algorithm. *Geophysics* 2004;**69**:917–24.

Zhou C, Cai W, Luo Y *et al.* Acoustic wave-equation traveltime and waveform inversion of crosshole seismic data. *Geophysics* 1995;**60**:765–73.

Zitzler E. Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. *TIK-Schriftenreihe* 1999;**30**:1–122.

Zong Woo Geem, Joong Hoon Kim, Loganathan G. A New Heuristic Optimization Algorithm: Harmony Search. *SIMULATION* 2001;**76**:60–8.

List of Figures

List of Tables

RÉSUMÉ

La tomographie sismique des temps de trajet est un problème d'optimisation mal-posé du fait de la non-linéarité entre les temps et le modèle de vitesse. Par ailleurs, l'unicité de la solution n'est pas garantie car les données peuvent être expliquées par de nombreux modèles. Les méthodes de Monte-Carlo par Chaînes de Markov qui échantillonnent l'espace des paramètres sont généralement appréciées pour répondre à cette problématique. Cependant, ces approches ne peuvent pleinement tirer parti des ressources computationnelles fournies par les super-calculateurs modernes. Dans cette thèse, je me propose de résoudre le problème de tomographie sismique à l'aide d'algorithmes évolutionnistes. Ce sont des méthodes d'optimisation stochastiques inspirées de l'évolution naturelle des espèces. Elles opèrent sur une population de modèles représentés par un ensemble d'individus qui évoluent suivant des processus stochastiques caractéristiques de l'évolution naturelle. Dès lors, la population de modèles peut être intrinsèquement évaluée en parallèle ce qui rend ces algorithmes particulièrement adaptés aux architectures des super-calculateurs. Je m'intéresse plus précisément aux trois algorithmes évolutionnistes les plus populaires, à savoir l'évolution différentielle, l'optimisation par essaim particulaire, et la stratégie d'évolution par adaptation de la matrice de covariance. Leur faisabilité est étudiée sur deux jeux de données différents: un jeu réel acquis dans le contexte de la fracturation hydraulique et un jeu synthétique de réfraction généré à partir du modèle de vitesse Marmousi réputé pour sa géologie structurale complexe.

MOTS CLÉS

algorithme évolutionniste, tomographie sismique, problème inverse, calcul haute performance, intelligence artificielle

ABSTRACT

Seismic traveltime tomography is an ill-posed optimization problem due to the non-linear relationship between traveltime and velocity model. Besides, the solution is not unique as many models are able to explain the observed data. The non-linearity and non-uniqueness issues are typically addressed by using methods relying on Monte Carlo Markov Chain that thoroughly sample the model parameter space. However, these approaches cannot fully handle the computer resources provided by modern supercomputers. In this thesis, I propose to solve seismic traveltime tomography problems using evolutionary algorithms which are population-based stochastic optimization methods inspired by the natural evolution of species. They operate on concurrent individuals within a population that represent independent models, and evolve through stochastic processes characterizing the different mechanisms involved in natural evolution. Therefore, the models within a population can be intrinsically evaluated in parallel which makes evolutionary algorithms particularly adapted to the parallel architecture of supercomputers. More specifically, the works presented in this manuscript emphasize on the three most popular evolutionary algorithms, namely Differential Evolution, Particle Swarm Optimization and Covariance Matrix Adaptation - Evolution Strategy. The feasibility of evolutionary algorithms to solve seismic tomography problems is assessed using two different data sets: a real data set acquired in the context of hydraulic fracturing and a synthetic refraction data set generated using the Marmousi velocity model that presents a complex geology structure.

KEYWORDS

evolutionary algorithm, seismic tomography, inverse problem, high performance computing, artificial intelligence