Multigrid methods for zero-sum two player stochastic games

Sylvie Detournay

THÈSE DE DOCTORAT DE L'ÉCOLE POLYTECHNIQUE
MATHÉMATIQUES APPLIQUÉES

présentée par

Sylvie Detournay

Multigrid methods for zero-sum two player stochastic games

Thèse soutenue publiquement le 25 septembre 2012
devant le jury composé de :

Marianne Akian	Directeur
Jean-Philippe Chancelier	Rapporteur
Maurizio Falcone	Rapporteur
Stéphane Gaubert	Examinateur
Yvan Notay	Examinateur
Xavier Vasseur	Examinateur
Hasnaa Zidani	Examinateur
Wieslaw Zielonka	Examinateur

Remerciements

J'adresse tout d'abord mes plus sincères remerciements à ma directrice de thèse Marianne Akian pour m'avoir encadrée dans la réalisation de cette thèse. Je lui sais gré du temps qu'elle m'a consacré, de ses conseils et de sa gentillesse. Je souhaite également remercier Stéphane Gaubert pour l'intérêt qu'il a porté à mes travaux.

Je remercie Jean-Philippe Chancelier et Maurizio Falcone qui m'ont fait l'honneur d'accepter d'être rapporteurs de cette thèse. Mes plus sincères remerciements vont également à Hasnaa Zidani, Yvan Notay, Xavier Vasseur et Wieslaw Zielonka qui ont accepté de faire partie de mon jury.

Je tiens à remercier Jean Cochet-Terrasson, pour l'intérêt qu'il a porté à mes travaux et sa relecture de ma thèse. Mes pensées vont également à Artem Napov pour ses conseils ainsi que pour les moments de détentes passés en conférence. Merci à Audrey Minten pour sa relecture et ses corrections orthographiques.

Je remercie sincèrement Wallis Filippi pour sa disponibilité et son aide précieuse ainsi que toute l'équipe administrative du CMAP. Je n'oublie pas aussi tous les membres du CMAP pour l'excellent accueil durant mes années de thèse.

Merci à aux doctorants et anciens doctorants du CMAP que j'ai eu la chance de rencontrer. Et plus particulièrement, Camille, Émilie, Irina, Isabelle, Soledad, Francisco, Jean-Baptiste, Khaled, Khalil, Zhihao, Florent, Zixian, Laurent D., Assalé, Pascal, Gilles, Olivier, Michael, Laurent P., Xavier, Maxime et tant d'autres. Je remercie en particulier tous mes co-bureaux pour tous les bons moments passés dans notre bureau à douze et en dehors.

Je remercie également tous les doctorants et chercheurs que j'ai rencontrés lors de mes conférences. En particulier, tous ceux avec qui j'ai eu l'occasion de partager pauses café, dîners, visites touristiques et culturelles.

Mes pensées vont également à mes parents pour les visites à Paris, l'aide au déménagement et la préparation de ma soutenance. Je pense aussi à mon petit frère Jérôme pour son aide et ses conseils pour mon pot de thèse.

Finalement, un grand merci à toutes les personnes, famille et amis, qui ont contribué au bon déroulement de mes études.

Résumé

Dans cette thèse, nous proposons des algorithmes et présentons des résultats numériques pour la résolution de jeux répétés stochastiques, à deux joueurs et somme nulle dont l'espace d'état est de grande taille. En particulier, nous considérons la classe de jeux en information complète et en horizon infini. Dans cette classe, nous distinguons d'une part le cas des jeux avec gain actualisé et d'autre part le cas des jeux avec gain moyen. Nos algorithmes, implémentés en C, sont principalement basés sur des algorithmes de type itérations sur les politiques et des méthodes multigrilles. Ces algorithmes sont appliqués soit à des équations de la programmation dynamique provenant de problèmes de jeux à deux joueurs à espace d'états fini, soit à des discrétisations d'équations de type Isaacs associées à des jeux stochastiques différentiels.

Dans la première partie de cette thèse, nous proposons un algorithme qui combine l'algorithme des itérations sur les politiques pour les jeux avec gain actualisé à des méthodes de multigrilles algébriques utilisées pour la résolution des systèmes linéaires. Nous présentons des résultats numériques pour des équations d'Isaacs et des inéquations variationnelles. Nous présentons également un algorithme d'itérations sur les politiques avec raffinement de grilles dans le style de la méthode FMG. Des exemples sur des inéquations variationnelles montrent que cet algorithme améliore de façon non négligeable le temps de résolution de ces inéquations.

Pour le cas des jeux avec gain moyen, nous proposons un algorithme d'itération sur les politiques pour les jeux à deux joueurs avec espaces d'états et d'actions finis, dans le cas général multichaine (c'est-à-dire sans hypothèse d'irréductibilité sur les chaînes de Markov associées aux stratégies des deux joueurs). Cet algorithme utilise une idée développée dans Cochet-Terrasson et Gaubert (2006). Cet algorithme est basé sur la notion de projecteur spectral non-linéaire d'opérateurs de la programmation dynamique de jeux à un joueur (lequel est monotone et convexe). Nous montrons que la suite des valeurs et valeurs relatives satisfont une propriété de monotonie lexicographique qui implique que l'algorithme termine en temps fini. Nous présentons des résultats numériques pour des jeux discrets provenant d'une variante des jeux de Richman et sur des problèmes de jeux de poursuite.

Ensuite, nous présentons de nouveaux algorithmes de multigrilles algébriques pour la résolution de systèmes linéaires singuliers particuliers. Ceux-ci apparaissent, par exemple, dans l'algorithme d'itérations sur les politiques pour les jeux stochastiques à deux joueurs et somme nulle avec gain moyen, décrit ci-dessus. Nous introduisons également une nouvelle

méthode pour la recherche de mesures invariantes de chaînes de Markov irréductibles basée sur une approche de contrôle stochastique. Nous présentons un algorithme qui combine les itérations sur les politiques d'Howard et des itérations de multigrilles algébriques pour les systèmes linéaires singuliers.

Finalement, nous décrivons la bibliothèque PIGAMES en C qui a été développée par l'auteur de cette thèse.

Abstract

In this thesis, we present some algorithms and numerical results for the solution of large scale zero-sum two player repeated stochastic games. In particular, we consider the class of games with perfect information and infinite horizon. In this class, we consider the games with discounted payoff and the games with mean payoff. Our algorithms are mainly based on policy iteration type algorithms and multigrid methods, and are implemented in C. These algorithms are applied either to the dynamic programming equation of a true finite state space zero-sum two player game or to the discretization of an Isaacs PDE of a zero-sum stochastic differential game.

In a first part, we propose an algorithm which combines policy iterations for discounted games and algebraic multigrid methods to solve the linear systems involved in the policy iterations. We present numerical tests on discretizations of Isaacs equations or variational inequalities. We also present a full multilevel policy iteration, similar to FMG, which allows one to improve substantially the computation time for solving some variational inequalities.

For the games with mean payoff, we develop a policy iteration algorithm to solve zero-sum stochastic games with finite state and action spaces, perfect information and in the general multichain case (i.e. without irreducibility assumption on the Markov chains determined by the strategies of the players), following an idea of Cochet-Terrasson and Gaubert (2006). This algorithm relies on a notion of nonlinear spectral projection of dynamic programming operators of one player games (which are monotone and convex). We show that the sequence of values and relative values satisfies a lexicographical monotonicity property which implies that the algorithm does terminate. We present numerical results on a variant of Richman games and on pursuit-evasion games.

We propose new algebraic multigrid algorithms to solve particular singular linear systems that arise for instance in the above policy iteration algorithm for zero-sum two player stochastic games with mean payoff. Furthermore, we introduce a new method to find the stationary probability of an irreducible Markov chain using a stochastic control approach and present an algorithm which combines the Howard policy iterations and algebraic multigrid iterations for singular systems.

At the end, we describe the C library PIGAMES which has been implemented by the author of this thesis.

Contents

Notations

- $\mathbb{N}^* := \{k > 0 \mid k \in \mathbb{N}\}$.
- $\mathbb{R}^n_+ := \{x \in \mathbb{R}^n \mid x_i \geq 0, \text{ for } 1 \leq i \leq n\}$.
- $\mathbb{R}^{n\times n}_+ := \{A \in \mathbb{R}^{n\times n} \mid a_{ij} \geq 0, \text{ for } 1 \leq i,j \leq n\}$.
- $(\mathbb{R}^*_+)^n := \{x \in \mathbb{R}^n \mid x_i > 0, \text{ for } 1 \leq i \leq n\}$.
- $(\mathbb{R}^*_+)^{n\times n} := \{A \in \mathbb{R}^{n\times n} \mid a_{ij} > 0, \text{ for } 1 \leq i,j \leq n\}$.
- $\mathbf{1} = [1 \cdots 1]^T$ is the vector of ones in $\mathbb{R}^n$.
- I is the identity matrix of $\mathbb{R}^{n\times n}$.
- For $A, B \in \mathbb{R}^{n\times n}$, we write $A \geq B$ if $a_{ij} \geq b_{ij}$ for $1 \leq i,j \leq n$.
- For $A, B \in \mathbb{R}^{n\times n}$, we write $A \leq B$ if $a_{ij} \leq b_{ij}$ for $1 \leq i,j \leq n$.
- $x^{|l}$ is a vector of $\mathbb{R}^{n_l}$.
- $A^{|l}$ is a matrix of $\mathbb{R}^{n_l\times n_l}$.
- $\mathbf{1}^{|l}$ is the vector of ones in $\mathbb{R}^{n_l}$.
- $I^{|l}$ is the identity matrix of $\mathbb{R}^{n_l\times n_l}$.
- $^{(k)}$ is an index of iteration.
- $[n] := \{1, \cdots, n\}$.
- $\rho(A) = \max_{i\in[n]} |\lambda_i|$ is the *spectral radius* of the matrix $A \in \mathbb{R}^{n\times n}$ with λ_i are its eigenvalues.

Introduction

0.1 Stochastic Games with perfect information

The (deterministic) game theory finds its origins in the field of economics and was introduced by von Neumann and Morgenstern in[VNM44]. In 1953, Shapley defined the class of stochastic games in [Sha53]. We refer to the books of Filar and Vrieze [FV97] and Sorin and Neyman [NS03] for further descriptions on the topic and applications.

An *infinitely repeated game*, or *discrete time dynamic game*, or *infinite horizon multi-stage game*, consists in an infinite sequence of state transitions, where at each step, the transition depends on the actions of the players, and each player receives a reward which depends on the state of the game and the actions of all players at this step. The aim of each player is to maximize his own objective function, for instance his payoff which is the sum of the rewards he received at all steps. The game is stochastic when the state sequence is a random process with a Markov property, then the objective function is the expected payoff. In particular, the game is a *two player zero-sum game* when there are two players with opposite rewards, i.e. the rewards sum to zero at each step. Hence, player 2 aims to minimize player 1 objective function. When the game does not stop in finite time (almost surely), it generates an infinite stream of rewards. In this case, one can for instance consider a *discounted payoff* where the reward at each step k is discounted by some multiplicative factor μ^k, with $0 < \mu < 1$. Alternatively, one can consider a *mean payoff*, that is the Cesaro limit of the expectation of the successive rewards, which represents the mean reward per step. In this thesis, we shall consider zero-sum two player stochastic games with the discounted and the mean payoff criteria.

We consider in particular a two player zero-sum stochastic game with finite state space $\mathcal{X} := \{1, \dots, n\}$ and finite action spaces $\mathcal{A} := \{1, \dots, m_1\}$ and $\mathcal{B} := \{1, \dots, m_2\}$ for player 1 and player 2 respectively. We denote by $r(x, a, b) \in \mathbb{R}$ the reward of player 1 when the state (at the current step) is $x \in \mathcal{X}$ and the actions of player 1 and 2 are $a \in \mathcal{A}$, $b \in \mathcal{B}$ respectively. Since the game is zero-sum the reward for player 2 is $-r(x, a, b)$ and we say that player 2 makes a payment of $r(x, a, b)$ to player 1. We denote by $p(y|x, a, b) \in \mathbb{R}_+$ the transition probability from state x to state y when the actions of player 1 and 2 are $a \in \mathcal{A}$ and $b \in \mathcal{B}$ respectively (and we have $\sum_{y \in \mathcal{X}} p(y|x, a, b) = 1$). In this thesis, we shall consider the class of zero-sum two player stochastic games with perfect information. Before describing these games, we introduce stochastic games with imperfect information

without entering into the details.

In a zero-sum two player stochastic game with imperfect information (as initially introduced in [Sha53]), the players play simultaneously and each player is trying to maximize his own objective function. For this purpose they choose a *strategy* which is a decision rule that tells them which action to choose in any situation. A (behavior) strategy for a player is a function which to all possible histories associates a probability distribution over the player's action space. We denote this function by α for player 1 and by β for player 2. Then, for a fixed pair of strategies (α, β), when the game starts in $x \in \mathcal{X}$, we denote the objective function (or payoff or expected payoff) of the first player by $J(\alpha, \beta, x)$. A couple of optimal strategies (α^*, β^*) for player 1 and player 2 respectively, verifies

$$J(\alpha, \beta^*, x) \leq J(\alpha^*, \beta^*, x) \leq J(\alpha^*, \beta, x), \qquad x \in \mathcal{X}$$

for all possible strategies α for player 1 and β for player 2. The strategies α^* and β^* are called *optimal strategies* for player 1 and player 2 respectively and the value $J(\alpha^*, \beta^*, x)$ is called the value of the game starting in $x \in \mathcal{X}$. For the discounted setting (or terminating games), Shapley showed in [Sha53] that there always exist optimal *Markov stationary strategies* for both players, that are strategies that only depend on the current state. Moreover, he showed that the value of the game, defined for each starting state $x \in \mathcal{X}$, is unique and verifies the Shapley equation, that has the same form as the one of perfect information games given in Equation (1), but with $\mathcal{A}$ and $\mathcal{B}$ replaced by the sets of probability distributions over $\mathcal{A}$ and $\mathcal{B}$ respectively and p and r modified accordingly.

A two player stochastic game is of *perfect information* when player 1 plays before player 2 and each player is informed at every move of the exact sequence of choices preceding that move. Equivalently, a perfect information two player stochastic game is a stochastic game where in each state one of the two players has no more than one possible action. Moreover, when the same player has exactly one possible action in each state, the game can be seen as a game with one player. The particular class of *Markov Decision Processes (MDP)* or *stochastic control problems*, which can be viewed as stochastic games with one player, has been introduced and developed by Bellman [Bel57] and by Howard [How60]. Albeit the class of MDPs and the class of stochastic games are closely related, they have been discovered independently, stochastic games preceding MDP and the two theories evolved independently. See in particular the book of Puterman [Put94] for a wide description of MDPs and applications. For the relation between MDPs and stochastic games, we refer to the books of [FV97, NS03].

0.1.1 Dynamic programming equation

Now, we consider a zero-sum two player stochastic game with perfect information and discounted payoff as previously defined. Hence, we assume that player 1 plays before player 2, and that at each step, player 1 is choosing his action $a \in \mathcal{A}$ as a function of the current state $x \in \mathcal{X}$, and player 2 is choosing his action $b \in \mathcal{B}$ as a function of the current state $x \in \mathcal{X}$ and the action $a \in \mathcal{A}$ of player 1.

Under the finiteness conditions on the sets $\mathcal{X}$, $\mathcal{A}$ and $\mathcal{B}$, there exists a function $v : \mathcal{X} \to \mathbb{R}$ which associates to each $x \in \mathcal{X}$ the discounted value of the game $v(x) = J(\alpha^*, \beta^*, x)$ where $J(\alpha, \beta, x)$ is the discounted payoff for player 1 when he applies strategy α, player 2 applies strategy β and the initial state of the game is x. This function is called the *value or the value function of the game* and is the unique solution [Sha53] of:

$$v(x) = \underbrace{\max_{a\in\mathcal{A}} \left(\min_{b\in\mathcal{B}} \left(\sum_{y\in\mathcal{X}} p(y \,|\, x, a, b)\, \mu\, v(y) \;+\; r(x, a, b) \right) \right)}_{=:F(\mu v; x)} \qquad \forall x \in \mathcal{X} \qquad (1)$$

where $0 < \mu < 1$. This Equation (1), called the *dynamic programming equation* or *Shapley equation* of the game, also gives the optimal strategies for both players which are pure Markov stationary strategies (see [Sha53]). A *pure stationary Markov strategy* or *feedback policy* for a player, is a decision rule that prescribes him the same action each time the same state is reached independently of the past plays. Hence, it is a function which maps any state $x \in \mathcal{X}$ to an action $a \in \mathcal{A}$ for player 1, respectively $b \in \mathcal{B}$ for player 2. (Now, we shall denote by $\alpha : \mathcal{X} \to \mathcal{A}$ and $\beta : \mathcal{X} \to \mathcal{B}$ pure stationary Markov strategies for player 1 and for player 2 respectively.)

Moreover, the operator $F(\mu\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ which maps $v \in \mathbb{R}^n$ to the function

$$\begin{array}{rccc} F(\mu v): & \mathcal{X} & \to & \mathbb{R} \\ & x & \mapsto & F(\mu v; x) \end{array}$$

where $F(\mu v; x)$ is defined in (1) and $0 < \mu \leq 1$, is called the *dynamic programming operator* or *Shapley operator* [Sha53]. From [Sha53, Sor03], this operator is *monotone* or *order-preserving*, meaning that $v \leq v' \implies F(\mu v) \leq F(\mu v')$ where $\leq$ denotes the partial order of $\mathbb{R}^n$, and when $\mu = 1$, it is *additively homogeneous*, meaning that it commutes with the addition of a constant vector (see for instance [CT80, GG04] for more background on this class of nonlinear maps). Hence, the Shapley operator is contracting with constant μ in the sup-norm, that is $\|F(\mu v) - F(\mu v')\|_\infty \leq \mu \|v - v'\|_\infty$ for all $v, v' \in \mathbb{R}^n$ when $0 < \mu < 1$ and nonexpansive in the sup-norm when μ equals one.

In the undiscounted case, the discount factor μ equals one and we denote by F the operator $F(\mu\cdot)$ defined above. In this case, we look at the mean reward per turn for player 1 and the value of the game is his mean payoff which we denote by $\rho(x)$ for each starting point $x \in \mathcal{X}$. The Shapley operator F is *polyhedral*, meaning that there is a covering of $\mathbb{R}^n$ by finitely many polyhedra such that the restriction of F to any of these polyhedra is affine. Kohlberg [Koh80] showed that if a map F is a polyhedral self-map of $\mathbb{R}^n$ that is nonexpansive in some norm, then, there exist two vectors η and v in $\mathbb{R}^n$ such that $F(t\eta + v) = (t+1)\eta + v$, for all $t \in \mathbb{R}$ large enough. A map of the form $t \mapsto t\eta + v$ is called a *half-line*, and η is its *slope*. It is *invariant* if it satisfies the latter property. When F has an invariant half-line with slope η, the growth rate of its orbits $\chi(F) := \lim_{k\to\infty} F^k(v)/k$ exists and is equal to η (where F^k denotes the k-th iterate of F and $v \in \mathbb{R}^n$), and $\rho(x) = \eta_x$

for all $x \in \mathcal{X}$. Moreover, we have that the couple (η, v) is an invariant half-line of F if, and only if, it satisfies:

$$\begin{cases} \eta &= \hat{F}(\eta) \ , \\ \eta + v &= \acute{F}_\eta(v), \end{cases} \tag{2}$$

where the maps $\hat{F}$ and $\acute{F}_\eta$ are constructed from F (see [ADCTG12] or Chapter 4). For the Shapley operator F, this couple system of equations (2) is what is solved in practice to find the value function $\rho = \eta$ of the mean payoff game and v is called the relative value of this game.

0.1.2 Example of a game and applications

Typical examples of zero-sum two player stochastic games with perfect information are, for instance, parlor games such as chess games or draughts (see [FV97]), where the players play in turn and the rewards are of zero-sum type. Another typical class of perfect information games are the pursuit evasion games, that appear for instance in the context of warfare (see [FV97] and the work of McEneaney, Fitzpatrick and Lauko in [MFL04]). In these games, the two players, the pursuer and the evader, play in turn. The evader wants to escape from the pursuer, and the pursuer has the opposite objective. The reward can be chosen as the distance between the two players that the pursuer wants to minimize and the evader wants to maximize, the actions are for instance the possible directions and the transition probabilities from a state to another are determined by the directions of both players with a possible stochastic perturbation. We shall present numerical results on pursuit evasion games in Chapter 4 and Chapter 5 with a mean payoff criterion. The class of pursuit evasion game is more commonly studied in the case of deterministic games or in the field of differential stochastic games, see the works of Bardi, Falcone and Soravia in [BFS94, BFS99], and of Li, Cruz and Schumacher in [LCS08]. See also the works of Bardi, Koike and Soravia in [BKS00], of Falcone in [Fal06], and of Cristiani and Falcone in [CF09] for stochastic games with state constraints. More recent applications of zero-sum two player stochastic games with perfect information appeared in computer science. In verification of programs, see the works of Costan, Gaubert, Goubault, Martel, and Putot in [CGG$^+$05], Gaubert, Goubault, Taly and Zennou in [GGTZ07], Gawlitza and Seidl in [GS07b, GS07c], Adjé, Gaubert, and Goubault in [AGG08, AGG10, AGG12a], and of Gawlitza, Seidl, Adjé, Gaubert and Goubault in [GSA$^+$12]. In problems of the internet and network flow control, see the work of Altman [Alt94]. See also the book of [NS03] for other applications references.

Example 0.1 (Tug-of-war)**.** *We shall now describe an example of a game, called tug-of-war or quasi-combinatorial game, which belongs to the class of random turn games (see [LLPU96, LLP$^+$99, PSSW09]). We shall present some numerical tests on a variant of this game in Chapter 4.*

In this game, we consider two players, called Mr. Blue and Ms. Red, and a directed graph $G = (V, E)$ where V is a finite set of vertices and $E := \{(i, j) \mid i, j \in V\}$ is a finite

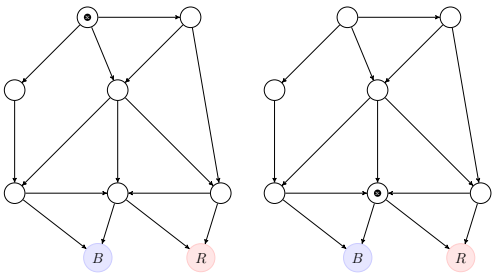

Figure 1: Example of a tug-of-war game, on the left the graph when the token is in the initial state and on the right the graph on a later step of the game.

set of edges (or arcs). The game is played on the graph in the following way. A token is placed on an initial vertex $x_0 \in V$ and represents the current state of the game. Then, a fair coin is tossed, the winner of the toss can choose to move the token to any vertex $x_1 \in V$ such that there exists an edge, $(x_0, x_1) \in E$, between the two vertices. At each turn k, the coin is tossed again and the winner controls the next move of the token in the graph. When a player cannot move, the game terminates. Moreover, there are two special vertices, $b \in V$ and $r \in V$, where there are no possible moves, that are called terminal states. Mr Blue wins when the game stops at b and Mrs Red wins when the game stops at r. Hence, Mr Blue's goal is to bring the token in b and Mrs. Red's goal is to bring the token in r. The game is a draw if neither vertex b or r is ever reached. This game is a zero-sum two player stochastic game.

For instance, we represent in Figure 1, a directed graph that represents a tug of war game. There are two terminal vertices: the vertex labeled B and the vertex labeled R for Mr. Blue and Mrs. Red respectively. The black token is placed in a vertex of the graph. We show in this Figure the game at two different steps of the game.

This kind of game was introduced by Richman in a context of auction game (see for instance [LLPU96, LLP+99]). Indeed, it is closely related to the following auction game. Again, we consider the same graph and the same players, Mr. Blue and Mrs. Red. We assume now that both of them has an initial amount of money. Then rather than tossing a coin, the two players bid at each stage of the game for the right to make the next move. The player with the highest bid wins the choice of the next move and pays the bid to the opponent. If the bids are equal, they toss a coin. As previously, Mr Blue wins when the game stops at b and Mrs Red wins when the game stops at r regardless of the final distribution of money. The game is a draw if neither vertex b or r is ever reached.

From [LLPU96, LLP+99], both players admit optimal strategies which are pure stationary strategies. Moreover, when the graph G is finite, the game has a value $v : V \to \mathbb{R}$, called the Richman cost function. It is the unique solution of

$$v(x) = \frac{1}{2}\left(\max_{y \in S(x)} v(y) + \min_{z \in S(x)} v(z)\right) \qquad x \in V \setminus \{r, b\}$$

with $v(b) = 0$ and $v(r) = 1$, and where $S(x) := \{y \in V \mid (x, y) \in V\}$ is the set of successors of $x \in V$.

There exists many variants of this game, see for instance [LLP+99] and also applications, such as the infinite laplacian, see [PSSW09, Obe05].

0.1.3 Stochastic Differential Games

Another class of games which we shall consider is the class of zero-sum two player differential stochastic games in continuous time. In these games, the state space is a regular open subset $\mathcal{X}$ of $\mathbb{R}^d$ and the dynamics of the game is governed by a stochastic differential equation which is jointly controlled by two players, see the works of Fleming and Souganidis in [FS89] and of Świech in [Świ96].

Let us consider here the second order Hamiltonian:

$$H(x,p,K) = \min_{a\in\mathcal{A}} \max_{b\in\mathcal{B}} \left[p\cdot g(x,a,b) + \frac{1}{2}\mathrm{tr}(\sigma\sigma^T(x,a,b)\,K) + r(x,a,b) \right] \tag{3}$$

where $x \in X, p \in \mathbb{R}^n, K \in \mathbb{R}^{n\times n}$, $\mathcal{A}$, $\mathcal{B}$ are either finite sets or subsets of some $\mathbb{R}^p$ spaces, $(x,a,b) \in \mathcal{X}\times\mathcal{A}\times\mathcal{B} \mapsto g(x,a,b) \in \mathbb{R}^d$ and $(x,a,b) \in \mathcal{X}\times\mathcal{A}\times\mathcal{B} \mapsto \sigma(x,a,b) \in \mathbb{R}_+^{d\times d'}$ are given functions.

In the discounted case [FS89, Świ96], the value v of a zero-sum two player stochastic differential game is solution of the following dynamic programming equation[1]:

$$-\lambda v(x) + H\left(x, \frac{\partial v}{\partial x_i}, \frac{\partial^2 v}{\partial x_i \partial x_j}\right) = 0, \quad x \in X \tag{4}$$

where $\lambda \geq 0$ is a scalar and H is of the form . This nonlinear elliptic partial differential equation is called *Isaacs* or *Bellman-Isaacs equation.*

For a zero-sum two player *differential stochastic games with mean payoff* or *ergodic differential stochastic games*, when the mean payoff ρ $(= \eta)$ is independent of the initial state, it is the unique constant such that there exists v is the unique solution of the following dynamic programming equation[1]:

$$-\eta + H\left(x, \frac{\partial v}{\partial x_i}, \frac{\partial^2 v}{\partial x_i \partial x_j}\right) = 0, \quad x \in X. \tag{5}$$

This partial differential equation is a stationary *Isaacs* or *Bellman-Isaacs equation.* See the works of Alvarez and Bardi in [AB07] and of Bardi in [Bar09].

The discretization of Equation (4) (or Equation (5)) with a monotone scheme in the sense of [BS91] yields the dynamic programming Equation (1) (or Equation (2)) of a stochastic game with discrete state space. Hence, it allows one to solve a differential game in the same way as (1) (or (2)) solves a discrete time dynamic game. Suitable possible discretizations schemes are for instance: Markov chain discretizations [Kus77, KD92], monotone discretizations [BS91], full discretizations of semi-Lagrangian type [BFS94], and max-plus finite element method [FM00, AGL08] for deterministic games or control problems. Hence, we are interested in solving discretizations of Equation (4) (or Equation (5))

1. We denote by $\partial v/\partial x_i$ (resp. $\partial v/\partial x_i \partial x_j$) the vector (resp. matrix) of the first (resp. second) partial derivatives of v.

which have the form of Equation (1) (or Equation (2)), in order to find an approximation of the value of the corresponding differential stochastic game.

Such equations may be applied in particular to pursuit-evasion games (see for instance [BFS94]), but they also appear in solving H^∞ optimal control problems (see for instance [BB95]), or risk-sensitive optimal control problems [Fle06], in particular for finance applications [ES11, BCPS04].

A formal description of all the above definitions will be given in Chapter 1 for discounted games and in Chapter 4 for games with mean payoff.

0.2 Algorithms for stochastic games

The classical method to solve the dynamic programming Equation (1) of a two player zero-sum stochastic game with discounted payoff is the *value iterations*, also called *successive approximations* or the *value iteration algorithm*. It was first introduced by Shapley [Sha53] for zero-sum two player discounted games with imperfect information and after by Bellman [Bel57] for stochastic control problems (or MDPs) and zero-sum two player stochastic games with perfect information. The iterations of this method are cheap but their convergence slows considerably as the discount factor μ approaches one. However, when we discretize Equation (4) with a finite difference or finite element method with a discretization step h, we obtain an equation of the form (1) with a discount factor $\mu = 1 - O(\lambda h^2)$, then when h is small μ is close to one and the value iteration method is as slow as the Jacobi or Gauss-Seidel iterations for a discretized linear elliptic equation. We refer also to [KK71, vdW77] for variants of this method.

Another approach consists in the so-called *policy iterations* or *policy iteration algorithm*, initially introduced by Howard [How60] to solve stochastic control problems (i.e. one player games), with proof of convergence for the discounted payoff case and without proof for the mean payoff case. The proof in the mean payoff case was given by Denardo and Fox in [DF68]. The first generalization of the policy iteration algorithm to zero-sum two player stochastic games was proposed by Hoffman and Karp [HK66a] for the mean payoff, imperfect information, when all the Markov chains associated to the strategies are irreducible. In his study of contracting maps [Den67], Denardo also proposed a convergent generalization of Howard's algorithm with proof of convergence and with possibly approximate solutions. His algorithm applies to zero-sum two player stochastic games with imperfect information and discounted payoff.

The policy iteration algorithm for discounted games appeared also, as an adaptation of the Hoffman-Karp algorithm, in the work of Rao, Chandrasekaran, and Nair [RCN73, Algorithm 1] for the imperfect information case and the work of Puri [Pur95] for deterministic games with perfect information. More recently, Raghavan and Syed [RS03] developed a related algorithm in which strategy improvements involve only one state at each iteration. Some other attempts of generalization were proposed for discounted games, such as the most natural extension of Howard's algorithm given in [PAI69] or in [RCN73, Algo-

rithm 2] which is sometimes called Newton-Raphson technique (because it is a Newton like algorithm). However the Newton-Raphson policy iteration algorithm does not converge in general [vdW78, Con93] except with very restrictive assumptions [PAI69].

Inspired by Hoffman and Karp algorithm, Cochet-Terrasson, Gaubert and Gunawardena [GG98, CTGG99] developed a policy iteration algorithm for zero-sum two player deterministic games with mean payoff, which has been intensively tested numerically in [DG06]. Then, from this algorithm and the policy iterations for multichain MDPs of [How60, DF68], Cochet-Terrasson and Gaubert proposed in [CTG06] a policy iteration algorithm for zero-sum two player stochastic games with mean payoff in the multichain case, i.e. without any irreducibility assumptions on the Markov chains associated to the strategies. Another algorithm has also been proposed in [BEGM10] which is pseudo-polynomial in time. However, it solves zero-sum two player stochastic mean payoff games (which they call BWR) in the ergodic case, i.e. when the value of the game does not depend on the initial state. The latter condition is more general than unichain games but does not include multichain games.

Moreover, in [BCPS04], Bielecki, Chancelier, Pliska, and Sulem used a policy iteration algorithm to solve a semi-Markov mean-payoff ergodic game problem with infinite action spaces obtained from the discretization of a quasi-variational inequality. Their algorithm is based on the approach of [CTGG99, GG98] and proceeds in a Hoffman and Karp fashion.

0.2.1 Policy iterations for zero-sum stochastic games

We shall now briefly describe the policy iteration algorithm of [Den67, Pur95] for solving Equation (1). Starting with an initial policy $\alpha^{(0)} : \mathcal{X} \to \mathcal{A}$ for player 1, the policy iteration algorithm for a zero-sum two player discounted stochastic game consists in applying successively a policy evaluation step followed by a policy improvement step, that are given by:

1. *Policy Evaluation:* Compute $v^{(k+1)}$ solution of the stochastic control problem obtained by fixing in Equation (1), the actions $a = \alpha^{(k)}(x)$ for all $x \in \mathcal{X}$.
2. *Policy Improvement:* Choose the policy $\alpha^{(k+1)} : \mathcal{X} \to \mathcal{A}$ such that $a = \alpha^{(k+1)}(x)$ attains the maximum in the right hand side of Equation (1) when the value v is replaced by $v^{(k+1)}$.

The iterations stop when $\alpha^{(k+1)} = \alpha^{(k)}$, i.e. when we cannot improve the strategy. The policy iterations stop after a finite number of steps when the sets of actions are finite. Computing the value functions in the policy evaluation step (first step) is performed using the policy iteration algorithm for a one-player game. For one player games, the policy iterations stop after a finite number of steps when the sets of actions are finite, see the works of Lions and Mercier [LM80], Bertsekas [Ber87] and Puterman [Put94]. In addition, under regularity assumptions on the maps r and p, the policy iteration algorithm for a one player game with infinite action spaces is equivalent to Newton's method, thus can have a super-linear convergence in the neighborhood of the solution, see the works

of Puterman and Brumelle [PB79], and Bokanowski, Maroso, and Zidani [BMZ09], for superlinear convergence under general regularity assumptions; and the works of [PB79], Akian [Aki90b], and Bank and Rose [BR82], for order $p > 0$ superlinear convergence under additional regularity and strong convexity assumptions.

For stochastic games with mean payoff and irreducibility condition, one can use similar policy iteration algorithm as stated above. But for the multichain stochastic games, these policy iterations may cycle. Recall that for zero-sum two player stochastic games with mean payoff, we have to solve Equation (2) for the dynamic programming operator F to find the value ρ $(= \eta)$ and relative value v functions of the game. Each policy evaluation step for a one player game (or each policy evaluation iteration in the inner loop of the two player algorithm) consists in solving a couple of linear singular systems, that is Equation (2) with fixed strategies α and β for player 1 and player 2 respectively, for which there exists an infinite number of relative values solutions. Indeed, if we denote by $P^{\alpha\beta}$ the probabilities matrix given by $P^{\alpha\beta}_{xy} = p(x|y, \alpha(x), \beta(x))$ for $x, y \in \mathcal{X}$, the relative value which is solution of Equation (2) with fixed α and β, is defined up to an element of the kernel of the matrix $(I - P^{\alpha\beta})$ whose dimension equals the number of final classes of $P^{\alpha\beta}$. Hence, for the one player game, when the value of the game ρ does not change from one iteration to another, the policy iterations may return different relative values v and the algorithm may cycle. Howard [How60] and Denardo and Fox [DF68], proposed a policy iteration algorithm to solve MDPs which overcome cycling by using conservative improvement of the strategies and a way to solve the linear system that selects a unique relative value for each couple linear system of the form Equation (2) with fixed strategies α and β. For zero-sum two player stochastic games with mean payoff, similar behavior may also appear in the sequence of relative values of the outer loop, i.e. iterations on the first player's policies. In this case, Cochet-Terrasson and Gaubert proposed a policy iteration algorithm [CTG06] where the relative values for the first player are constructed using the nonlinear analogues of spectral projectors. These nonlinear projectors where introduced by Akian and Gaubert in [AG03].

The policy iterations for discounted games will be given in details in Chapter 1 and for mean payoff games in Chapter 4.

0.2.2 Approximation and complexity

In the policy iteration algorithm for solving MDPs (and hence in the intern policy iteration of two player games), the costly part of the algorithm lies in the policy evaluation step which requires the solution of a linear system that has the size of the discrete state space. However, the exact solution of this system may not be necessary and an approximation of this solution may be used, for instance by using an iterative solver. This leads to a class of modified policy iterations which gives ϵ-optimal solution and ϵ-optimal strategies. See for instance [Por72, Por75, vN76, Por80, Put94, Aki90b] for MDPs, [vdW78, BMZ09] for discounted stochastic games and [vdW80] for unichain stochastic games with mean payoff.

One can easily see that a trivial bound for the number of policy iterations for an MDP is the total number of possible policies which is exponential in the number of states. This

is also the case for two player games. The complexity of the policy iteration algorithm has been of large interest in the computer science community in the last decades. Condon has shown [Con92] that the decision problem corresponding to a simple stochastic game belongs to NP∩coNP. Friedmann has shown [Fri09] that a strategy improvement algorithm requires an exponential number of iterations for a "worst"-case family of games called parity games. Moreover, we have that parity games can be reduced to mean payoff deterministic games (Puri [Pur95]), the latter ones to discounted deterministic games which in turn can be reduced to simple stochastic games (Zwick and Paterson [ZP96]). In [AM09], Andersson and Miltersen generalized this result showing that stochastic mean payoff games with perfect information, stochastic parity games and simple stochastic games are polynomial time equivalent. Hence, the decision problem corresponding to a game of one of these classes lies in the complexity class of NP∩coNP. Moreover, Jurdzinski [Jur98] has shown that solving parity games as well as mean payoff games belongs UP∩coUP. Note that the result of Friedmann has also been extended to total reward and undiscounted MDP by Fearnley [Fea10a, Fea10b] and to simple stochastic games and weighted discounted stochastic games by Andersson [And09].

Moreover, for Markov decision processes with a fixed discount factor, some upper bounds on the number of policy iterations have been given in [MH86]. More recently, Ye presented the first strongly polynomial bound in [Ye05, Ye11]. The latter bound has been improved and generalized to zero-sum two player stochastic games with perfect information and fixed discount factor by Hansen, Miltersen and Zwick in [HMZ11], giving the first strongly polynomial bound for these games.

0.2.3 Multigrid methods

Standard multigrid method was originally created in the seventies to solve efficiently linear elliptic partial differential equations (see for instance [McC87]). It works as follows. Multigrid methods require discretizations of the given continuous equation on a sequence of grids, each of them, starting from a coarse grid, being a refinement of the previous one until a given accuracy is attained. The size of the coarsest grid is chosen such that the cost of solving the problem on it is cheap. Assume also that transfer operators between these grids are given: interpolation and restriction. Then, a multigrid cycle on the finest grid consists in: first, the application of a smoother on the finest grid; then a restriction of the residual on the next coarse grid; then solving the residual problem on this coarse grid using the same multigrid scheme; then, interpolate this solution (which is an approximation of the error) and correct the error on the fine grid; finally, the application of a smoother on the finest grid. If the multigrid components are properly chosen, this process is efficient to find the solution on the finest grid. Indeed, in general the relaxation process is smoothing the error which then can be well approximated by elements in the range of the interpolation. This implies, in good cases, that the contraction factor of the multigrid method is independent of the discretization step and also the complexity is in the order of the number of discretization points. We shall refer to this standard method as geometric

multigrid.

Algebraic multigrid method, called AMG, has been initially developed in the early eighties (see [Bra86, BMR85, RS87]) for solving large sparse linear systems arising from the discretization of partial differential equations with unstructured grids or PDE's not suitable for the application of the geometric multigrid solver or large discrete problems not derived from any continuous problem. The AMG method includes a "setup phase" in which the coarse levels (coarse "grids") are constructed and which is based only on the algebraic equations (in contrast to geometric multigrids). The points of the fine grids are represented by the variables and the points of the coarse grids by subsets of these variables. The selection of those coarse variables and the construction of the transfer operators between levels are done in such a way that the range of the interpolation approximates the errors not reduced by a given relaxation scheme.

In the recent years, multigrid methods have also been introduced [HL94, DSMM$^+$08, BBB$^+$10, TY10, TY11, Vir07] for the computation of the stationary probability of irreducible Markov Chains. Also [DSMM$^+$10a, DSMM$^+$10b, DSMMS11, DSMSW10] proposed variants or accelerations of [DSMM$^+$08]. The method of the latter is based on the Iterative Aggregation/Disaggregation (IAD) scheme which is suitable for Nearly Completely Decomposable (NCD) Markov Chains and are working on two levels. See [DS00] and references therein, for a broad overview of IAD partitioning techniques. A local convergence result for this two level scheme is available in [MM03] and a global convergence result is available in [MM98] in some specific setting.

Algebraic multigrid methods and some basic iterative algorithms to solve linear systems will be explained in more details in Chapter 2.

0.3 Contributions

0.3.1 Handling stochastic discounted games with large state space

Each policy iteration for a one player game (or each iteration in the inner loop of the two player algorithm) requires the solution of a linear system. Indeed, when we fix feedback policies $\alpha : \mathcal{X} \to \mathcal{A}$ and $\beta : \mathcal{X} \to \mathcal{B}$ for player 1 and 2 respectively, the system of equations (1) yields a linear system of the form: $v = \mu P^{\alpha\beta} v + r$ where $v, r \in \mathbb{R}^n$ are respectively the value function of the game and the vector of rewards for the fixed policies α and β, $0 < \mu < 1$ is the discount factor and $P^{\alpha\beta} \in \mathbb{R}^{n\times n}$ is a Markov matrix whose elements are the transition probabilities $P^{\alpha\beta}_{xy} = p(x|y, \alpha(x), \beta(x)) \in \mathbb{R}_+$ for $x, y \in \mathcal{X}$ (and each rowsum of $P^{\alpha\beta}$ equals one). When the dynamic programming Equation (1) is coming from the discretization of an Isaacs partial differential Equation (4), this linear system corresponds to the discretization of a linear elliptic partial differential equation, hence it may be solved in the best case in a time that is linear in the number of discretization points by using multigrid methods, that is the cardinality $|\mathcal{X}|$ of the discretized state space $\mathcal{X}$, or the size of the matrix $P^{\alpha\beta}$. For general stochastic games on a finite state space $\mathcal{X}$, since $P^{\alpha\beta}$ is a Markov matrix, the matrix $(I - \mu P^{\alpha\beta})$ of the linear system is an invertible

M-matrix [BP94], and one may expect the same complexity when solving it by using an algebraic multigrid method.

In Chapter 3, we propose an algorithm which combines the policy iteration algorithm for solving zero-sum two player discounted stochastic games with the algebraic multigrid method (AMG). We shall call AMGπ the resulting algorithm. This algorithm can be applied either to a true finite state space zero-sum two player game or to the discretization of an Isaacs equation, although in the thesis we restrict ourselves to numerical tests for the discretization of stochastic differential games. Such an association of multigrid methods with policy iteration has already been used and studied in the case of one player games, that is discounted stochastic control problems (see Hoppe [Hop86, Hop87] and Akian [Aki90a, Aki90b] for Hamilton-Jacobi-Bellman equations or variational inequalities, Ziv and Shimkin [OZ05] for AMG with learning methods). A two-level partitioning technique combined with policy iteration was also used by [BC89] in the context of solving Markov Decision process. However, the approach is new in the case of two player games. The AMGπ algorithm has been implemented in the C library PIGAMES for this thesis. We present in Chapter 3, numerical tests on discretizations of Isaacs or Hamilton-Jacobi-Bellman equations or variational inequalities. In our numerical tests, we observe that the computation time is smaller when using AMGπ instead of the policy iteration with LU and that the computation time needed by AMGπ increases linearly with the size of the problem, what is expected from a multigrid method.

Furthermore, we notice in the numerical tests that for some variational inequalities the number of policy iterations to solve the problem may be large. However, as for Newton's algorithm, convergence can be improved by starting the policy iteration with a good initial guess, close to the solution. With this in mind, we present a full multi-level policy iteration, similar to FMG. It consists in solving the problem at each grid level by performing policy iterations until a convergence criterion is verified, then to interpolate the strategies and value to the next level, in order to initialize the policy iterations of the next level, until the finest level is attained. When at each level policy iterations are combined with the algebraic multigrid method, we shall call FAMGπ the resulting full multi-level policy iteration algorithm. For one-player discounted games with infinite number of actions and under regularity assumptions, one can show [Aki90b, Aki90a] that this kind of full multi-level policy iteration has a computing time in the order of the cardinality $|\mathcal{X}|$ of the discretized state space $\mathcal{X}$ at the finest level. In Chapter 3, we give numerical examples on variational inequalities for two player games, the computation time of which is improved substantially using FAMGπ instead of AMGπ. However, the FAMGπ algorithm uses coarse grids discretizations of the partial differential equation and so cannot be applied directly to the dynamic programming equation of a two player zero-sum stochastic game with finite state space.

The aforementioned algorithms and numerical results of Chapter 3 have been published in the paper [AD12] co-authored with Marianne Akian.

0.3.2 Policy iterations for stochastic mutlichain games with mean payoff

In Chapter 4, we shall present the policy iteration algorithm for zero-sum two player multichain stochastic games with mean-payoff of which an initial version was given in [CTG06]. However, no implementation details were given in the short note [CTG06], in which the algorithm was stated abstractly, in terms of invariant half-lines. An iteration k on the policies of player 1 (outer loop) is called degenerate when the mean value $\eta^{(k)}$ equals the mean value $\eta^{(k-1)}$ of the previous iteration. In this case, as for the one player case, the general policy iterations may cycle. When a degenerate iteration occur, the policy iteration algorithm of [CTG06] avoids cycling by constructing the relative values using the nonlinear analogue of spectral projectors. It allows one to determine a unique relative value for Equation (2) when the strategy α of player 1 is fixed. This spectral projection or reduced super-harmonic function for a non linear convex order-preserving additively homogeneous map, was introduced in the work of Akian and Gaubert on spectral theory of such maps [AG03].

In Chapter 4, we develop fully the idea of [CTG06], and describe a policy iteration algorithm for multichain stochastic games with mean-payoff (see Section 4.4.2). We explain how nonlinear systems of the form (2) are solved at each iteration. We show in particular how non-linear spectral projections can be computed, by solving an auxiliary (one player) optimal stopping problem. This relies on the determination of the so called critical graph, the nodes of which (critical nodes) are visited infinitely often (almost surely) by an optimal strategy of a one player mean payoff stochastic game. An algorithm to compute the critical graph, based on results on [AG03], is given in Section 4.5.3.

We shall also give a proof of the convergence result which was only stated in [CTG06]. In particular, we show that the sequence $(\eta^{(k)}, v^{(k)}, C^{(k)})$ consisting of the mean payoff vector, relative value vector, and set of critical nodes, constructed by the algorithm satisfies a kind of lexicographical monotonicity property so that it converges in finite time (see Section 4.4.3). The proof of convergence exploits some results of spectral theory of convex order-preserving additively homogeneous maps, by Akian and Gaubert [AG03]. Then, we discuss an example (see Section 4.6) involving a variant of Richman games [LLP$^+$99] (also called stochastic tug-of-war [PSSW09], related with discretizations of the infinity Laplacian [Obe05]), showing that degenerate iterations do occur and that cycling may occur with naive policy iteration rules. Hence, the handling of degenerate iterations, that we do here by nonlinear spectral projectors, cannot be dispensed with.

The present algorithm has been implemented in the C library PIGAMES for this thesis. We finally report numerical experiments (see Section 4.7) carried out using this library, both on random instances of Richman type games with various numbers of states and on a class of discrete games arising from the monotone discretization of a pursuit-evasion differential game. These examples indicate that degenerate iterations are frequent, so that their treatment cannot be dispensed with. They also show that the algorithm scales well, allowing one to solve structured instances with 10^6 nodes and 10^7 actions in a few hours of CPU time on a single core processor (the bottleneck being the resolution of linear

systems).

Chapter 4 consists in the paper [ADCTG12] which is a joint work with Marianne Akian, Jean Cochet-Terrasson and Stéphane Gaubert.

0.3.3 Multigrids methods for particular linear systems with applications to Markov chains and to zero-sum two player stochastic games with mean payoff

In Chapter 5, we present some new algebraic multigrid algorithms to solve particular singular linear systems that arise for instance in the policy iteration algorithm for zero-sum two player stochastic games with mean payoff. In particular, we present our algorithm AMGsingular2. Furthermore, we introduce a new method to find the stationary probability of an irreducible Markov chain using a stochastic control approach and present our algorithm MGPIMG which combines the policy iterations of [How60, DF68] and AMGsingular2. We shall present some numerical results on random matrices that represent random walks on an uniform square grid and compare our our algorithm MCPIMG with the algorithm MAA of [DSMM$^+$08]. In the last section of this chapter, we apply the policy iterations combined with AMGsingular2 to solve some ergodic differential games with mean payoff. We shall compare the policy iteration algorithm with a direct solver LU versus the policy iteration algorithm with AMGsingular2 on pursuit evasion games.

0.3.4 Modeling and implementation

In Chapter 6, we describe the modeling that we use to represent a zero-sum two player stochastic game with discounted payoff or mean payoff. Then, we give the main ideas in term of structures (objects) and algorithms used in the C library PIGAMES which has been implemented by the author of this thesis. In section 6.4, we give the readme file of the library PIGAMES which describe the contents, the instructions and the options of the PIGAMES package.

Chapter 1

Stochastic Games with perfect information

1.1 Two player zero-sum stochastic games: the discrete case

The class of *two player zero-sum stochastic games* was first introduced by Shapley in the early fifties [Sha53]. We recall in this section the definition of a game with perfect information in the case of finite state space, discrete time and perfect information (for more details see [Sha53, FV97, Sor03]). A game is *of perfect information* means that a player is informed at every move of the exact sequence of choices preceding that move.

1.1.1 The model

We consider a finite state space $\mathcal{X} = \{1, \dots, n\}$. A stochastic process $(\xi_k)_{k\geq 0}$ on $\mathcal{X}$ gives the state of the game at each point time k, called stage. At each of these stages, both players have the possibility to influence the course of the game.

The stochastic game $\Gamma(x_0)$ starting from $x_0 \in \mathcal{X}$ is played in stages as follows. The initial state ξ_0 is equal to x_0 and known by the players. The player who plays first, say MAX, chooses an action ζ_0 in a set of possible actions $\mathcal{A}(\xi_0)$. Then the second player, called MIN chooses an action η_0 in a set of possible actions $\mathcal{B}(\xi_0, \zeta_0)$. The actions of both players and the current state determine the payment $r(\xi_0, \zeta_0, \eta_0)$ made by MIN to MAX and the probability distribution $p(\cdot|\xi_0, \zeta_0, \eta_0)$ of the new state ξ_1. Then the game continues in the same way with state ξ_1 and so on.

At a stage k, each player chooses an action knowing the history defined by $\iota_k = (\xi_0, \zeta_0, \eta_0, \cdots, \xi_{k-1}, \zeta_{k-1}, \eta_{k-1}, \xi_k)$ for MAX and (ι_k, ξ_k) for MIN. We call a *strategy or policy* for a player, a rule which tells him the action to choose at any stage and in any situation. There are several classes of strategies. Assume $\mathcal{A}(x) \subset \mathcal{A}$ and $\mathcal{B}(x, a) \subset \mathcal{B}$ for some sets $\mathcal{A}$ and $\mathcal{B}$. A behavior or randomized strategy for MAX (resp. MIN) is a sequence $\bar{\alpha} := (\alpha_0, \alpha_1, \cdots)$ (resp. $\bar{\beta} := (\beta_0, \beta_1, \cdots)$) where α_k (resp. β_k) is a map which to a history $h_k = (x_0, a_0, b_0, \dots, x_{k-1}, a_{k-1}, b_{k-1}, x_k)$ with $x_i \in \mathcal{X}$, $a_i \in \mathcal{A}(x_i)$, $b_i \in \mathcal{B}(x_i, a_i)$ for $0 \leq i \leq k$ (resp. (h_k, a_k)) at stage k associates a probability distribution on a probability

space over $\mathcal{A}$ (resp. $\mathcal{B}$) whose support is included in the possible actions space $\mathcal{A}(x_k)$ (resp. $\mathcal{B}(x_k, a_k)$). A Markovian (or feedback) strategy is a strategy which only depends on the information of the current stage k: α_k (resp. β_k) depends only on x_k (resp. (x_k, a_k)), then $\alpha_k(h_k)$ (resp. $\beta_k(h_k, a_k)$) will be denoted $\alpha_k(x_k)$ (resp. $\beta_k(x_k, a_k)$). It is called stationary if it is independent of k, then α_k is also denoted by α and β_k by β. A strategy of any type is called pure if for any stage k, the values of α_k (resp. β_k) are Dirac probability measures at certain actions in $\mathcal{A}(x_k)$ (resp. $\mathcal{B}(x_k, a_k)$) then we denote also by α_k (resp. β_k) the map which to the history assigns the only possible action in $\mathcal{A}(x_k)$ (resp. $\mathcal{B}(x_k, a_k)$).

In particular, if $\bar{\alpha}$ is a pure Markovian stationary strategy, then $\bar{\alpha} = (\alpha_k)_{k\geq 0}$ with $\alpha_k = \alpha$ for all k and α is a map $\mathcal{X} \to \mathcal{A}$ such that $\alpha(x) \in \mathcal{A}(x)$ for all $x \in \mathcal{X}$. In this case, we also speak about pure Markovian stationary strategy for α and we denote by $\mathcal{A}_M$ the set of such maps. We adopt a similar convention for player MIN: $\mathcal{B}_M := \{\beta : \mathcal{X} \times \mathcal{A} \to \mathcal{B} \mid \beta(x, a) \in \mathcal{B}(x, a)\, \forall x \in \mathcal{X},\, a \in \mathcal{A}(x)\}$.

A strategy $\bar{\alpha} = (\alpha_k)_{k\geq 0}$ (resp. $\bar{\beta} = (\beta_k)_{k\geq 0}$) together with an initial state determines stochastic processes $(\zeta_k)_{k\geq 0}$ for the actions of MAX, $(\eta_k)_{k\geq 0}$ for the actions of MIN and $(\xi_k)_{k\geq 0}$ for the states of the game such that

$$P(\xi_{k+1} = y \,|\, \iota_k = h_k, \zeta_k = a, \eta_k = b) = p(y \,|\, x, a, b) \tag{1.1a}$$

$$P(\zeta_k \in A \,|\, \iota_k = h_k) = \alpha_k(h_k)(A) \tag{1.1b}$$

$$P(\eta_k \in B \,|\, \iota_k = h_k, \zeta_k = a) = \beta_k(h_k, a)(B) \tag{1.1c}$$

where $\iota_k := (\xi_0, \zeta_0, \eta_0, \ldots, \xi_{k-1}, \zeta_{k-1}, \eta_{k-1}\xi_k)$ is the history process, h_k is a history vector at time k: $h_k = (x_0, a_0, b_0, \ldots, x_{k-1}, a_{k-1}, b_{k-1}, x)$ and A (resp. B) are measurable sets in $\mathcal{A}(x)$ ($\mathcal{B}(x, a)$ resp.). For instance, for each pair of pure Markovian stationary strategies $(\bar{\alpha}, \bar{\beta})$ of the two players, that is such that for $k \geq 0$: $\alpha_k = \alpha$ with $\alpha \in \mathcal{A}_M$ and $\beta_k = \beta$ with $\beta \in \mathcal{B}_M$, the state process $(\xi_k)_{k\geq 0}$ is a Markov chain on $\mathcal{X}$ with transition probability

$$P(\xi_{k+1} = y \,|\, \xi_k = x) \;=\; p(y|x, \alpha(x), \beta(x, \alpha(x))) \quad \text{ for } x, y \in \mathcal{X}$$

and $\zeta_k = \alpha(\xi_k)$ and $\eta_k = \beta(\xi_k, \zeta_k)$.

1.1.2 Payoff and dynamic programming equation

We consider here the *payoff* of the game $\Gamma(x_0)$ starting from $x_0 \in \mathcal{X}$ as the expected sum of the rewards at all steps of the game that MAX wants to maximize and MIN to minimize. When the strategies $\bar{\alpha}$ for MAX and $\bar{\beta}$ for MIN are fixed, the payoff in finite horizon τ of the game $\Gamma(x_0, \bar{\alpha}, \bar{\beta})$ starting from x_0 is given by

$$J^\tau(x_0, \bar{\alpha}, \bar{\beta}) \;=\; \mathbb{E}_{x_0}^{\bar{\alpha}\bar{\beta}} \left[\sum_{k=0}^{\tau-1} r(\xi_k, \zeta_k, \eta_k) \right],$$

where $\mathbb{E}_{x_0}^{\bar{\alpha},\bar{\beta}}$ denotes the expectation for the probability law determined by (1.1).

When the action spaces $\mathcal{A}(x)$ and $\mathcal{B}(x)$ are finite sets for all $x \in \mathcal{X}$, the finite horizon game has a value $v : \mathcal{X} \to \mathbb{R}$ which is given by:

$$v^\tau(x) \;=\; \sup_{\bar\alpha} \inf_{\bar\beta}\; J^\tau(x, \bar\alpha, \bar\beta), \quad \forall x \in \mathcal{X} \tag{1.2}$$

where the supremum is taken among all strategies $\bar\alpha$ for MAX and the infimum is taken over all strategies $\bar\beta$ for MIN, and the value v^T satisfies the *dynamic programming equation* [Sha53]:

$$v^{\tau+1}(x) \;=\; \max_{a\in\mathcal{A}(x)} \left(\min_{b\in\mathcal{B}(x,a)} \left(\sum_{y\in\mathcal{X}} p(y \,|\, x, a, b)\, v^\tau(y) \;+\; r(x, a, b) \right) \right) \qquad \forall x \in \mathcal{X}. \tag{1.3}$$

Moreover [Sha53], optimal strategies are obtained for both players by taking in (1.2) pure Markovian stationary strategies $\alpha \in \mathcal{A}_M$ for MAX and $\beta \in \mathcal{B}_M$ for MIN such that for all x in $\mathcal{X}$, $\alpha(x)$ attains the maximum in (1.3):

$$\alpha(x) \;\in\; \operatorname*{Argmax}_{a\in\mathcal{A}(x)} F(v; x, a)$$

where

$$F(v; x, a) \;:=\; \min_{b\in\mathcal{B}(x,a)} \underbrace{\left(\sum_{y\in\mathcal{X}} p(y|x, a, b)\; v(y) \;+\; r(x, a, b) \right)}_{:= F(v;x,a,b)}, \tag{1.4}$$

and for all x in $\mathcal{X}$ and a in $\mathcal{A}(x)$, $\beta(x, a)$ attains the minimum in (1.4):

$$\beta(x, a) \;\in\; \operatorname*{Argmin}_{b\in\mathcal{B}(x,a)} F(v; x, a, b) \;\; .$$

Here we use the notation $\operatorname*{Argmax}_{c\in C} f(c) := \{c \in C \;\mid\; f(c) = \max_{c'\in C} f(c')\}$ and similarly for Argmin.

We denote by F the *dynamic programming operator* or *Shapley operator* from $\mathbb{R}^{\mathcal{X}}$ (that is here $\mathbb{R}^n$) to itself given by:

$$F(v; x) \;:=\; \min_{a\in\mathcal{A}(x)} \max_{b\in\mathcal{B}(x,a)} \left(\sum_{y\in\mathcal{X}} p(y \,|\, x, a, b)\, v(y) \;+\; r(x, a, b) \right), \qquad \forall x \in \mathcal{X},\; v \in \mathbb{R}^{\mathcal{X}}. \tag{1.5}$$

Then, the dynamic programming equation of the finite horizon game writes:

$$v^{\tau+1} \;=\; F(v^\tau).$$

The operator F is *order-preserving*, i.e. $v \le w \implies F(v) \le F(w)$ where $\le$ denotes the partial ordering of $\mathbb{R}^n$, and *additively homogeneous*, i.e. it commutes with the addition of a constant vector; it implies that F is nonexpansive in the sup-norm.

Note that in [Sha53], the theorems are given for games which are not necessarily with perfect information, the application to perfect information games is given in section 4 of [Sha53].

When we consider an infinite horizon, the game may not stop in finite time (almost surely), hence the expected sum of the rewards may not be bounded and one consider other payoffs for instance a discounted payoff described below (or a mean payoff which will be the subject of Chapter 4).

Discounted payoff

Now, consider that the reward at time k is the payment made by MIN to MAX times μ^k where $0 < \mu < 1$ is called the *discount factor*. We denote by Γ_μ a game with a *discounted payoff*. In this case, for a couple of fixed strategies $\bar{\alpha}$ for MAX and $\bar{\beta}$ for MIN, the payoff of the game $\Gamma_\mu(x_0, \bar{\alpha}, \bar{\beta})$ starting from x_0 is defined by

$$J(x_0, \bar{\alpha}, \bar{\beta}) \;=\; \mathbb{E}_{x_0}^{\bar{\alpha},\bar{\beta}} \left[\sum_{k=0}^{\infty} \mu^k r(\xi_k, \zeta_k, \eta_k) \right],$$

where $\mathbb{E}_{x_0}^{\bar{\alpha},\bar{\beta}}$ denotes the expectation for the probability law determined by (1.1). The value of the game starting from $x_0 \in \mathcal{X}$, denoted by $\Gamma_\mu(x_0)$, is then given by

$$v(x_0) \;=\; \sup_{\bar{\alpha}} \inf_{\bar{\beta}} J(x_0, \bar{\alpha}, \bar{\beta}), \tag{1.6}$$

where the supremum is taken over all strategies $\bar{\alpha}$ for MAX and the infimum is taken over all strategies $\bar{\beta}$ for MIN. Note that a non terminating game without any discount factor (or $\mu = 1$) is called ergodic.

When the actions sets $\mathcal{A}(x)$ and $\mathcal{B}(x, a)$ are finite sets for all $x \in \mathcal{X}$, $a \in \mathcal{A}(x)$, the value $v : \mathcal{X} \to \mathbb{R}$ of the stochastic game Γ_μ satisfies the dynamic programming equation [Sha53]:

$$v \;=\; F(\mu\, v). \tag{1.7}$$

In this case, the dynamic operator $F(\mu\, \cdot)$ is monotone and contracting with constant $0 < \mu < 1$ in the sup-norm, i.e. $\left\| F(\mu\, v) - F(\mu\, v') \right\|_\infty \;\leq\; \mu \left\| v - v' \right\|_\infty$ for all $v, v' \in \mathbb{R}^{\mathcal{X}}$.

Moreover [Sha53], optimal strategies are obtained for both players by taking in (1.6) pure Markovian stationary strategies $\alpha \in \mathcal{A}_M$ for MAX and $\beta \in \mathcal{B}_M$ for MIN such that for all x in $\mathcal{X}$:

$$\alpha(x) \;\in\; \operatorname*{Argmax}_{a \in \mathcal{A}(x)} F(\mu\, v; x, a)$$

and for all x in $\mathcal{X}$ and a in $\mathcal{A}(x)$:

$$\beta(x, a) \;\in\; \operatorname*{Argmin}_{b \in \mathcal{B}(x,a)} F(\mu\, v; x, a, b) \;\; .$$

Note that the above results hold also when we consider total payoff ($\mu = 1$) for games that stop almost surely in finite time. Those games are called *terminating game* (or sometimes *stopping game*). That is for any pair of strategies $(\bar{\alpha}, \bar{\beta})$ the limit:

$$\lim_{\tau \to \infty} J^\tau(x, \bar{\alpha}, \bar{\beta}) \;<\; \infty$$

is finite, then the dynamic programming operator F defined in (1.5) is contracting in the supnorm and $v = F(v)$ has a unique solution (see for instance [FV97, Sor03]). This is the case, for instance, when the game has an absorbing state, that is a state $x \in \mathcal{X}$ such that $p(x|x,a,b) = 1$ and $r(x,a,b) = 0$ for all possible actions $a \in \mathcal{A}(x)$ and $b \in \mathcal{B}(x,a)$. Hence, a discounted stochastic game can be seen as a game with an absorbing state y such that for all $x \neq y \in \mathcal{X}$, $p(y|x,a,b) = 1-\mu$ for all $a \in \mathcal{A}(x)$ and $b \in \mathcal{B}(x,a)$. Note that in his pioneer paper [Sha53], Shapley considered stochastic games with a positive probability $s(x,a,b)$ that the game stops in each state $x \in \mathcal{X}$ for all $a \in \mathcal{A}(x)$ and $b \in \mathcal{B}(x,a)$, that is $p(y|x,a,b) = 1-s(x,a,b) < 1$, his games are called terminating games. When $s(x,a,b) = s$ for all $x \in \mathcal{X}$ for all $a \in \mathcal{A}(x)$ and $b \in \mathcal{B}(x,a)$, it is equivalent to our discounted payoff case explained above (see also remark in [Sha53] section 4). It is this last setting (fixed discount factor) that appears mostly in the literature.

A related class of perfect information zero-sum stochastic games, is the class of Additive Reward and Additive Transition(AR-AT) zero-sum stochastic games. See for instance [RTV85, TR97, Vri03] for a complete description and applications. In these class of AR-AT games, at each state of the game, a part of the reward depends only on the action of MAX and the other part depends only on the action of MIN. The reward is then the addition of both parts, that is $r(x,a,b) = r(x,a)+r(x,b)$ for each state x in $\mathcal{X}$ and actions a in $\mathcal{A}$ and b in $\mathcal{B}$ for MAX and MIN respectively. Also the transition probability is the addition of two independents parts of both players, that is $p(y|x,a,b) = p(y|x,a)+p(y|x,b)$ for $x, y \in \mathcal{X}$. This class of games shares the same properties as the perfect information games see [RTV85, TR97, Vri03]. Indeed [RTV85, TR97, Vri03], two person zero-sum AR-AT stochastic games with discounted or mean payoff admit pure stationary Markov strategies for both players. Hence, the dynamic programming equation for these games can be solved using the same algorithms.

1.2 Numerical solution of discrete dynamic programming equations of discounted games

One can classically solve the dynamic programming Equation (1.7) of a two player zero-sum stochastic game with discounted payoff and perfect information by using the value iteration algorithm that Shapley [Sha53] first introduced for general discounted games. This method is sometimes also called value iterations or successive approximations. The iterations of this method are cheap but their convergence slows considerably as the discount factor μ approaches one. Another approach consists in the so called policy iteration algorithm, initially introduced by Howard [How60] for one player stochastic games (i.e. stochastic control problems). Further adaptations of this algorithm were proposed for the two player games, the first by Hoffman and Karp [HK66a] for a special mean-payoff case. And later, the policy iteration algorithm was given by Denardo [Den67] for zero-sum two player discounted stochastic games and by Puri [Pur95] in the case deterministic games of perfect information. In all cases, the policy iteration algorithm converges faster than the

value iteration algorithm (see for instance [Aki90a, HMZ11]) and in practice it ends in few steps (see for instance [DG06] for numerical examples in the case of deterministic games).

In this section, we shall present the policy iteration algorithm to solve the dynamic programming Equation (1.7) of a two player zero-sum discounted stochastic game with finite state space. Before, we provide the value iteration algorithm for those games and also the policy iteration algorithm for a one player game.

1.2.1 Value iteration algorithm for two player stochastic games

We consider here the *value iterations* or *successive approximations* or the *value iteration algorithm* that were first introduced by Shapley [Sha53]. (see for instance [Bel57]). Given an initial approximation $v^{(0)} \in \mathbb{R}^n$ and setting $k = 0$, the algorithm iterates on the values $v^{(k)}$ that are computed at each iteration by:

$$v^{(k+1)} = F(\mu\, v^{(k)}),$$

until the desired convergence is obtained, e.g. $\|v^{(k+1)} - F(\mu v^{(k+1)})\| < \epsilon$ for some positive ϵ. Using the fact that the dynamic operator $F(\mu\,\cdot)$ is monotone and contracting with constant $0 < \mu < 1$ in the sup-norm, the iterates $(v^{(k)})_{k\geq 1}$ converge to the discounted value of the game (see [Bel57]). However, the convergence is slow when the discount factor μ approaches one. Acceleration of this method have been proposed in the literature, see for instance [vdW77] for a successive approximation method in a "Gauss Seidel" fashion or [KK71] for the one player case.

1.2.2 Policy iteration algorithm for Markov Decision Process with discounted payoff

Now, let us consider a one player stochastic game with only a MIN player and finite state space $\mathcal{X}$. The discounted value of this game starting in $x \in \mathcal{X}$ is given by:

$$v(x) \;=\; \inf_{\bar\beta} \mathbb{E}_x^{\bar\beta}\left[\sum_{k=0}^{\infty} \mu^k r(\xi_k, \eta_k)\right],$$

where the processes ξ_k, η_k and strategies $\bar\beta$ are defined such as in the Section 4.2 without the MAX player. As before, we define the set of feedback strategies for player MIN by $\mathcal{B}_M := \{\beta : \mathcal{X} \to \mathcal{B} \mid \beta(x) \in \mathcal{B}(x)\, \forall x \in \mathcal{X}\}$, where $\mathcal{B}$ contains all the sets $\mathcal{B}(x)$. In this case, the dynamic programming operator F, mapping $\mathbb{R}^n$ to itself, is defined by:

$$F(v;x) \;:=\; \min_{b\in\mathcal{B}(x)} \left(\sum_{y\in\mathcal{X}} p(y|x,b)\; v(y) \;+\; r(x,b)\right), \qquad \forall x \in \mathcal{X}, \forall v \in \mathbb{R}^n \;.$$

Then, the discounted value $v \in \mathbb{R}^n$ of the game is the solution of the dynamic programming equation: $v = F(\mu\, v)$, which also gives the optimal strategy $\beta \in \mathcal{B}_M$ for MIN. This game is also called *stochastic control problem* or *Markov Decision Process* with finite state

space $\mathcal{X}$. We shall use the term Markov Decision Process or *MDP* for short. We refer to [How60, DF68, Put94] for a deeper description on this topic.

The policy iteration algorithm that solves the dynamic programming equation of a Markov Decision Processes with discounted payoff $0 < \mu < 1$ (or $\mu = 1$ for a MDP that stops almost surely in finite time), was first introduced by Howard [How60] and is given in Algorithm 1.1. Given an initial policy $\beta^{(0)} \in \mathcal{B}_M$, this algorithm returns the value of the game $v : \mathcal{X} \to \mathbb{R}$ and the optimal policy $\beta \in \mathcal{B}_M$.

Algorithm 1.1 Policy iteration for Markov Decision Processes

Input: An initial policy $\beta^{(0)} \in \mathcal{B}_M$. *Ouput:* The value $v \in \mathbb{R}^n$ of the MDP and the optimal policy $\beta \in \mathcal{B}_M$.

1. Compute the value $v^{(k+1)}$ of the game with fixed feedback policy $\beta^{(k)}$, that is the solution of

$$v^{(k+1)}(x) = \sum_{y \in \mathcal{X}} \mu\, p(y \,|\, x, \beta^{(k)}(x))\, v^{(k+1)}(y) \;+\; r(x, \beta^{(k)}(x)) \qquad x \in \mathcal{X},\ v \in \mathbb{R}^n.$$

2. Improve the policy. Find the optimal feedback policy $\beta^{(k+1)}$ for the value $v^{(k+1)}$, i.e. for each x in $\mathcal{X}$, choose $\beta^{(k+1)}(x)$ such that:

$$\beta^{(k+1)}(x) \;\in\; \operatorname*{Argmin}_{b \in \mathcal{B}(x)} \left\{ \sum_{y \in \mathcal{X}} \mu\, p(y \,|\, x, b)\, v^{(k+1)}(y) \;+\; r(x, b) \right\}.$$

3. If $\beta^{(k+1)}(x) = \beta^{(k)}(x)$ for $x \in \mathcal{X}$ then **STOP** and return $v^{(k+1)}$ and $\beta^{(k+1)}$.
4. Increment k by one and go to Step 1.

Each policy iteration of Howard's Algorithm 1.1 strictly improves the current policy and produces a non increasing sequence of values $(v^{(k)})_{k \geq 1}$. It implies that the algorithm never visits twice the same policy. Hence if the action sets are finite in each point of $\mathcal{X}$, the policy iterations stop after a finite time (see for instance [PB79, LM80, Ber87]). Moreover, under regularity assumptions, the policy iteration algorithm for a one player game with infinite action spaces is equivalent to Newton's method [PB79, BR82, Aki90b, BMZ09]. Indeed, define $G(v) = F(v) - v$, then the problem is to find the solution of $G(v) = 0$ where all entries of G are concave functions. The policy improvement step can be seen as the computation of an element of the sup-differential of G in the current approximation $v^{(k+1)}$ and the value improvement step computes the zero of the previous sup-differential. When G is regular, the sequence of value functions $(v^{(k)})_{k \geq 1}$ is exactly the sequence of the Newton's algorithm.

Moreover, Chancelier, Messaoud, and Sulem presented in [CMS07] a policy iteration algorithm for quasi-variational inequalities, partially undiscounted infinite horizon problems. They proved the contraction of their policy iteration algorithm.

1.2.3 Policy iteration algorithm for two player games

Some algorithms for solving zero-sum two player stochastic games that do not converge have been proposed in the literature, see for instance [vdW78, Con93] for proofs of the corresponding "non convergence". The most known of these algorithms is the most natural extension of Howard's algorithm, sometimes called Newton-Raphson technique (because it is a Newton like algorithm). Given an initial value $v^{(0)}$, it consists in successively applying the two following steps:

1. Determine $\alpha^{(k+1)} \in \mathcal{A}_M$ such that $\alpha^{(k+1)}(x) \in \operatorname*{Argmax}_{a\in\mathcal{A}(x)} F(\mu\, v^{(k)}; x, a)$, $x \in \mathcal{X}$, and $\beta^{(k+1)} \in \mathcal{B}_M$ such that $\beta^{(k+1)}(x,a) \in \operatorname*{Argmin}_{b\in\mathcal{B}(x,a)} F(\mu\, v^{(k)}; x, a, b)$, $a \in \mathcal{A}(x)$, $x \in \mathcal{X}$.
2. Compute $v^{(k+1)}$ solution of $v(x) = F(\mu\, v; x, \alpha^{(k+1)}(x), \beta^{(k+1)}(x, \alpha^{(k+1)}(x)))$, $x \in \mathcal{X}$.

until convergence. In [PAI69], it has been shown that this algorithm converges under very restrictive assumptions and in [RCN73, Algorithm 2] that it converges in general. However, this last proof is false, see [vdW78] for a counter-example and [Con93]. Indeed, we notice that the dynamic programming operator F for the two player game is neither convex nor concave.

A convergent extension of Howard's algorithm for two player games was first proposed by Hoffman and Karp [HK66a] for zero-sum two player mean-payoff games with imperfect information and with assumption on the transition probabilities of the game (see chapter 4). In his study of contracting maps [Den67], Denardo proposed a generalization of Howard's algorithm with an approximate solution and convergence results. He applied his method to Shapley's zero-sum two player stochastic terminating games (or with discount payoff) with imperfect information. The policy iteration algorithm for discounted games appeared also, as an adaptation of the Hoffman-Karp algorithm, in the work of Rao, Chandrasekaran, and Nair [RCN73, Algorithm 1] for the imperfect information case and the work of Puri [Pur95] for deterministic games with perfect information. More recently, Raghavan and Syed [RS03] developed a related algorithm in which strategy improvements involve only one state at each iteration.

We give here the policy iteration algorithm for solving a two player zero-sum stochastic games with finite state space $\mathcal{X}$ and perfect information, as defined in [Den67, Pur95] (and adapted to our case and notations). Based on the definitions of Section 1.1, we need to solve the dynamic programming Equation (1.7) which give us the value of the game (Equation (1.6)) and the optimal strategies for both players. For a fixed pure feedback policy for MAX $\alpha \in \mathcal{A}_M$, the value v of the game is the solution of the equation $v = F^\alpha(\mu\, v)$ where F^α is an operator mapping $\mathbb{R}^n$ to itself whose x-coordinate is given by:

$$F^\alpha(v; x) := F(v; x, \alpha(x)),$$

for each $x \in \mathcal{X}$, $v \in \mathbb{R}^n$ and $F(v; x, a)$ is defined by (1.4). Note that F^α is the dynamic programming operator of a one player game with only the MIN player. Then the policy iteration algorithm is given in Algorithm 1.2.

Algorithm 1.2 Policy Iteration for two player games

Input: An initial policy $\alpha^{(0)} \in \mathcal{A}_M$ for MAX. *Ouput:* The value $v \in \mathbb{R}^n$ of the game and the optimal strategies $\alpha \in \mathcal{A}_M$ for MAX and $\beta \in \mathcal{B}_M$ for MIN.

1. Compute the value $v^{(s+1)}$ of the game with fixed feedback policy $\alpha^{(s)}$, that is the solution of
$$v^{(s+1)} \;=\; F^{\alpha^{(s)}}(\mu\, v^{(s+1)})$$
by using Algorithm 1.1 that returns $\beta^{(k)}$.
2. Improve the policy: Find the optimal feedback policy $\alpha^{(s+1)}$ of MAX for the value $v^{(s+1)}$, i.e. for each x in $\mathcal{X}$, choose $\alpha^{(s+1)}(x)$ such that:
$$\alpha^{(s+1)}(x) \;\in\; \operatorname*{Argmax}_{a\in\mathcal{A}(x)} F(\mu\, v^{(s+1)}; x, a)$$
where $F(v; x, a)$ is defined by (1.4).
3. If $\alpha^{(s+1)}(x) = \alpha^{(s)}(x)$ for $x \in \mathcal{X}$ then **STOP** and return $v^{(s+1)}$, $\alpha^{(s+1)}$ and $\beta^{(k)}$.
4. Increment s by one and go to Step 1.

In Algorithm 1.2, Step 1 is performed by using the policy iteration algorithm for a one player game. That is, given an initial feedback policy for MIN $\beta^{(s,0)} \in \mathcal{B}_M$, we iterate on MIN policies $\beta^{(s,k)} \in \mathcal{B}_M$ and value functions $v^{(s,k)}$. Then at each step k of the interior policy iteration (Algorithm 1.1 Step 1), one computes $v^{(s,k+1)}$, the value of the game with fixed strategies $\alpha^{(s)} \in \mathcal{A}_M$ for MAX and $\beta^{(s,k)} \in \mathcal{B}_M$ for MIN. This is done by solving the linear system:
$$v^{(s,k+1)} \;=\; \mu\, P^{\alpha^{(s)}\beta^{(s,k)}}\, v^{(s,k+1)} \;+\; r^{\alpha^{(s)}\beta^{(s,k)}} \quad , \tag{1.8}$$
where for all $\alpha \in \mathcal{A}_M$, $\beta \in \mathcal{B}_M$: $P^{\alpha\beta} \in \mathbb{R}^{n\times n}$ is a stochastic matrix whose elements are defined by $(P^{\alpha\beta})_{xy} = p(y|x, \alpha(x), \beta(x))$ for all $x, y \in \mathcal{X}$ and $r^{\alpha\beta} \in \mathbb{R}^n$ is the vector whose elements are defined by $(r^{\alpha\beta})_x = r(x, \alpha(x)\beta(x))$ for $x \in \mathcal{X}$.

As for the one player case, each iteration of the policy iteration algorithm strictly improves the current policy, hence it can never visit twice the same policy. Moreover, the algorithm produces a non decreasing (resp. non increasing) sequence of values $(v^{(s)})_{s\geq 1}$ (resp. $(v^{(s,k)})_{k\geq 1}$) of the external loop (resp. internal loop), see [Pur95, CTG06]. It follows that if the action sets for both players are finite in each point of $\mathcal{X}$, the policy iterations stop after a finite time [Pur95].

Note that both Algorithm 1.1 and Algorithm 1.2 also apply to stochastic terminating games, that is $\mu = 1$, see remark at the end of Section 1.1.2.

Another extension of the policy iteration algorithm has been proposed by Raghavan and Syed [RS03] with proved convergence for discounted stochastic games with perfect information. It is based on graph theory interpretation. The algorithm is based on so called "policy adjacent improvement" following a specific lexicographical order (see [RS03, Rag03] for more details). This algorithm is closely related to the policy iteration algorithm

of [Den67, Pur95] but it improves only one action per improvement step.

1.2.4 About the complexity of the policy iteration algorithm

One can easily see that a trivial bound for the number of policy iteration for an MDP is the total number of possible policies which is exponential in the number of states. The complexity of the policy iterations have been of large interest in the computer science community in the last decades.

First, we recall here some "informal" definitions of the complexity classes P, NP and co-NP from [CLRS01] (see also [CLRS01] for formal definitions). The class P consists of all the problems which are solvable in polynomial time. These problems can be solved in time $O(L^d)$ where d is an integer and L the size of the input of the problem (generally the length of its binary-encoded inputs). The class NP consist of the problems that are "verifiable" in polynomial time. That is, if we were somehow given a "certificate" of a solution for the problem, then we could verify that the certificate is correct in polynomial time in the size of the problem. We have that P$\subseteq$NP. The class of problems with such verification of certificates for the "no-solution-answers" is called co-NP, the complementarity class of NP. A nondeterministic machine is unambiguous if, for any input, there is at most one accepting computation of the machine. UP is the class of languages accepted by unambiguous machines in polynomial time.

Condon has shown in [Con92] that the decision problem corresponding to a simple stochastic game (which is restriction of the general stochastic games) belongs to NP$\cap$coNP. Moreover, no polynomial time algorithm is known to solve this problem. Friedmann has shown [Fri09] that a strategy improvement algorithm requires an exponential number of iterations for a "worst" case family of games called parity games. From Puri thesis [Pur95], we know that parity games can be reduced to mean payoff (deterministic) games. Also, Zwick and Paterson have shown in [ZP96] a polynomial reduction from mean payoff (deterministic) games to discounted (deterministic) payoff games, and from the latter ones to simple stochastic games. In [AM09], Andersson and Miltersen generalized this result showing that stochastic mean payoff games with perfect information, stochastic parity games and simple stochastic games are also polynomial time equivalent. Hence, the decision problem corresponding to a game of one of these classes lies in the complexity class of NP$\cap$coNP. Moreover in [Jur98], Jurdzinski has shown that solving parity games as well as mean payoff games belongs to UP$\cap$coUP. Note that the result of Friedmann has also been extended to total reward and undiscounted MDP by Fearnley [Fea10a, Fea10b] and to simple stochastic games and weighted discounted stochastic games by Andersson [And09].

In computer science, an algorithm is strongly polynomial in time when its running time is bounded polynomially in the inherent dimensions of the problem (that is here the number n of states and the total number of actions m) and independent of the sizes of the numerical data (the bitzise of the encoded problem).

However, some upper bounds on the number of policy iterations have been studied for the solution of MDP and more recently for zero-sum two player stochastic games with

perfect information, when the discount factor $\mu < 1$ is fixed. In 1986, Holzbaur and Meister gave in [MH86] a polynomial bound for Howard algorithm to solve MDP with discount factor $0 < \mu < 1$, that is the number of policy iterations is bounded by $O(\frac{nL}{1-\mu} \log \frac{1}{1-\mu})$ where n is the size of the problem and L is the largest bits size of the rewards. A strongly polynomial time algorithm for solving those games is given by Ye in [Ye05] using an interior point algorithm. Recently, Ye presented in [Ye11] a new strongly polynomial bound for the number of iterations of Howard's algorithm for solving MDP with discount factor $0 < \mu < 1$, that is given by $O(\frac{mn}{1-\mu} \log \frac{n}{1-\mu})$ where m is the total number of actions. His proof is based on a linear programming formulation of the problem and linear programming duality. Based on the work of Ye, Hansen, Miltersen and Zwick improved the bound to $O(\frac{m}{1-\mu} \log \frac{n}{1-\mu})$ in [HMZ11]. They have shown that this bound also applies to policy iterations for solving zero-sum two player stochastic games with perfect information and fixed discount factor $\mu < 1$, giving the first strongly polynomial bound for this problem.

1.2.5 Approximation in Policy iterations

In the policy iteration algorithm of Howard for solving Markov Decision Processes, the costly part of the algorithm lies in the policy evaluation step (Step 1 of Algorithm 1.1). Indeed, this step requires the solution of a linear system which has the size of the discrete state space $\mathcal{X}$. If n is the size of state space, a direct solver may require up to $\frac{2}{3}n^3$ arithmetic operations (see Section 2.1). However, the exact solution of this system may not be necessary and an approximation of this solution may be used by using for instance an iterative solver (see Section 2.2). This leads to a class of modified Policy Iteration which gives an ϵ-optimal solution and an ϵ-optimal strategy of the MDP. The first was proposed and studied in [Por72], see for instance [vN76, Put94, Por80]. See also [Por75] and references therein for related bounds on the solution.

For two player zero-sum stochastic games, the evaluation step (Step 1 of Algorithm 1.2) is also the costly part of the policy iteration since it involves the solution of a Markov Decision Process. Then, it was proposed and studied in [vdW78] and [BMZ09], to replace this solution by an approximation using for instance one of the above cited methods for MDP.

1.3 Two player zero-sum stochastic differential games: the continuous case

Another class of games which we consider is the class of two player differential stochastic games in continuous time. In these games, the state space is a regular open subset $\mathcal{X}$ of $\mathbb{R}^d$ and the dynamics of the game is governed by a stochastic differential equation which is jointly controlled by two players (see [FS89, Świ96] and below). In this case, the value of the game (defined below) is solution of a non linear elliptic partial differential equation of type, called Isaacs equation (see also [FS89, Świ96]). The discretization of this equation with a

monotone scheme in the sense of [BS91] yields the dynamic programming Equation (1.7) of a stochastic game with discrete state space which was described in the Section 1.1.

In the first following subsection, we give the definitions of differential stochastic games with a bounded state space and a discounted payoff. Then, in the next subsection, we present a subclass of these differential games called optimal stopping time games. Finally, in the last subsection, we introduce the finite difference discretization scheme that we use to discretize the Isaacs Equation (1.13) and (1.14) respectively. Numerical examples of such kind of games will be presented in Section 3.2.

1.3.1 Differential games with regular controls.

Assume now that the state space is a regular open subset $\mathcal{X}$ of $\mathbb{R}^d$. Suppose a probability space Ω is given, as well as a filtration $(\mathcal{F}_t)_{t\geq 0}$ over it (that is a non decreasing sequence of σ-algebras over Ω). We consider games which dynamics is governed by the following stochastic differential equation:

$$d\xi_t \;=\; g(\xi_t,\zeta_t,\eta_t)\,dt + \sigma(\xi_t,\zeta_t,\eta_t)\,dW_t, \tag{1.9}$$

with initial state $\xi_0 = x \in \mathcal{X}$. Here W_t is a d'-dimensional Wiener process on $(\Omega,(\mathcal{F}_t)_{t\geq 0})$; ζ_t and η_t are stochastic processes taking values in closed subsets $\mathcal{A}$ and $\mathcal{B}$ of $\mathbb{R}^p$ and $\mathbb{R}^q$ respectively; $(x,a,b)\in\mathcal{X}\times\mathcal{A}\times\mathcal{B}\mapsto g(x,a,b)\in\mathbb{R}^d$ and $(x,a,b)\in\mathcal{X}\times\mathcal{A}\times\mathcal{B}\mapsto\sigma(x,a,b)\in\mathbb{R}^{d\times d'}$ are given functions. The dimension d' of the Wiener process may be different from d and is given by the modeling of the problem. Assuming that ζ_t and η_t are adapted to the filtration $(\mathcal{F}_t)_{t\geq 0}$ (that is for all $t\geq 0$, ζ_t and η_t are $\mathcal{F}_t$-measurable), allows one to define the stochastic process ξ_t satisfying Equation (1.9) and it is a necessary condition to the assumption that the actions of the two players depend only on the past states and actions. We also consider strategies $\bar\alpha=(\alpha_t)_{t\geq 0}$ (resp. $\bar\beta=(\beta_t)_{t\geq 0}$) of player MAX (resp. MIN) determining the process $(\zeta_t)_{t\geq 0}$ (resp. $(\eta_t)_{t\geq 0}$). In particular, for pure Markovian stationary strategies, one has $\zeta_t=\alpha(\xi_t)$ and $\zeta_t=\beta(\xi_t,\zeta_t)$.

When $\mathcal{X}=\mathbb{R}^d$, the discounted payoff of the game with discount rate $\lambda>0$ is given by:

$$J(x;\bar\alpha,\bar\beta) \;=\; \mathbb{E}_x^{\bar\alpha,\bar\beta}\left[\int_0^\infty e^{-\lambda t} r(\xi_t,\zeta_t,\eta_t)\,dt \,|\, \xi_0=x\right] \tag{1.10}$$

where $(x,a,b)\in\mathcal{X}\times\mathcal{A}\times\mathcal{B}\mapsto r(x,a,b)\in\mathbb{R}$ is the (instantaneous, or running) reward function. Now, we consider that $\mathcal{X}$ is a regular open subset $\mathcal{X}$ of $\mathbb{R}^d$. In this case, we denote by τ the first exit time of the process $(\xi_t)_{t\geq 0}$ from $\mathcal{X}$, i.e. $\tau=\inf\{t\geq 0 \mid \xi_t\notin\mathcal{X}\}$. Then, the discounted payoff of the game stopped at the boundary is:

$$J(x;\bar\alpha,\bar\beta) \;=\; \mathbb{E}_x^{\bar\alpha,\bar\beta}\left[\int_0^\tau e^{-\lambda t} r(\xi_t,\zeta_t,\eta_t)\,dt + e^{-\lambda\tau}\psi_1(\xi_\tau) \,|\, \xi_0=x\right] \tag{1.11}$$

where the function $x\in\partial\mathcal{X}\to\psi_1(x)\in\mathbb{R}$ is called the terminal reward. The value function of the differential stochastic game starting from x is defined as in Section 4.2 by

$$v(x) \;=\; \sup_{\bar\alpha}\,\inf_{\bar\beta}\; J(x;\bar\alpha,\bar\beta) \tag{1.12}$$

where the supremum is taken over all strategies $\bar{\alpha}$ for MAX and the infimum is taken over all strategies $\bar{\beta}$ for MIN.

As previously, we are interested in finding the value function of the game and the corresponding optimal strategies. We denote by $L(v; x, a, b)$ the following second order partial differential operator:

$$L(v; x, a, b) := \sum_{i,j=1}^{d} q_{ij}(x, a, b) \frac{\partial^2 v(x)}{\partial x_i \partial x_j} + \sum_{j=1}^{d} g_j(x, a, b) \frac{\partial v(x)}{\partial x_j} - \lambda v(x),$$

with $(q_{ij})_{i,j=1,..,d} = \frac{1}{2}\sigma\sigma^T$. When $d' \geq d$ and $\sigma(x, a, b)$ is onto for all $x \in \mathcal{X},\ a \in \mathcal{A},\ b \in \mathcal{B}$, the matrix $q(x, a, b)$ is of full rank and the operator L is elliptic. The value of the game v is solution, under some regularity assumptions on Ω and on the functions g, σ, r and ψ (for instance boundedness and uniform Lipschitz continuity), of the dynamic programming equation, called Isaacs partial differential equation:

$$\begin{cases} \max\limits_{a \in \mathcal{A}} \left(\min\limits_{b \in \mathcal{B}} \left(L(v; x, a, b) + r(x, a, b) \right) \right) = 0 & \text{for } x \in \mathcal{X} \\ v(x) = \psi_1(x) & \text{for } x \in \partial\mathcal{X}. \end{cases} \tag{1.13}$$

This has been shown in the viscosity sense in [FS89]. See also [CIL92] and references therein for uniqueness of the solution of (1.13). If the value v of the game is a classical solution of (1.13), α and β are strategies such that for all x in $\mathcal{X}$ and a in $\mathcal{A}(x)$, $\alpha(x)$ and $\beta(x, a)$ are the unique actions that realize the maximum and the minimum in Equation (1.13) for MAX and MIN respectively, then α and β are pure Markovian stationary strategies, that are optimal for (1.12) (with ξ, ζ, η satisfying (1.9), (1.11), with $\zeta_t = \alpha(\xi_t)$ and $\eta_t = \beta(\xi_t, \zeta_t)$).

Note that for a game with one player, i.e. for a stochastic control problem, Equation (1.13) is the so-called Hamilton-Jacobi-Bellman equation. Also when $\mathcal{X}$ is bounded, and L is strongly uniformly elliptic (if for some $c > 0$, $q(x, a, b) \geq cI$ for all $x \in \mathcal{X},\ a \in \mathcal{A},\ b \in \mathcal{B}$), then the case $\lambda = 0$ can also be considered.

1.3.2 Differential games with optimal stopping control

When the action (ζ_t, η_t) of the players are not continuous or not bounded, the dynamic programming equation of the game is no more of the form of Equation (1.13), but may be a variational inequality or a quasi-variational inequality, see for instance [Fri73, BL78] for the case of optimal stopping games with one or two players and [FS06, BL82] for impulse or singular control.

We consider here an optimal stopping game, that is a game in which one of the players has the choice of stopping the game at any moment (see [Fri73] for a more general case). We assume here that MAX has this ability. Then at each time t, he chooses to stop or not the game, that is he is choosing an element of the action space $\{0, 1\}$ where 1 means that the game is continuing, 0 that the game stops, with $\zeta_s = 0$ and $\xi_s = \xi_t$ for $s \geq t$ when $\zeta_t = 0$ (i.e. $g(x, 0, b) = 0$, $\sigma(x, 0, b) = 0\ \forall b \in \mathcal{B}, x \in \mathcal{X}$ in (1.9)). The second player MIN

plays as previously and we consider the same model as in previous subsection. The value of a strategy $\bar{\alpha}$ for MAX determines a process $(\zeta_t)_{t\geq 0}$ adapted to the filtration of $(\xi_t)_{t\geq 0}$ (that is $(\sigma(\xi_t))_{t\geq 0}$), then a stopping time $\kappa = \inf\{t \geq 0 \mid \zeta_t = 0\}$ adapted to the process $(\xi_t)_{t\geq 0}$ and vice versa.

So if $r(x,0,b) = \lambda\psi_2(x)\,\forall b \in \mathcal{B}$, the discounted payoff (1.11) can be written as a function of the stopping time κ instead of $\bar{\alpha}$:

$$J(x;\kappa,\beta) = \mathbb{E}_x^{\kappa,\beta}\left[\int_0^\kappa e^{-\lambda t} r(\xi_t,1,\eta_t)\,dt + e^{-\lambda\kappa}\psi_2(\xi_\kappa)\,\mathbb{1}_{\kappa<\tau} + e^{-\lambda\tau}\psi_1(\xi_\tau)\,\mathbb{1}_{\kappa=\tau}\,\middle|\,\xi_0 = x\right] .$$

Indeed, if $\kappa < \tau$, then $\xi_s = \xi_\kappa \in \mathcal{X}$, $s \geq \kappa$, so $\tau = +\infty$, and $\int_\kappa^\tau e^{-\lambda t} r(\xi_t,\zeta_t,\eta_t)\,dt = e^{-\lambda\kappa}\psi_2(\xi_\kappa)$. The value function (1.12) of the game starting from x is then given by:

$$v(x) = \sup_\kappa \inf_\beta J(x;\kappa,\beta)$$

where the supremum is taken over all stopping times $\kappa \leq \tau$ and the infimum is taken over all strategies β for MIN.

Since the variable "a" appears only when equal to 1, one can omit it in equations, hence Equation (1.13) becomes:

$$\begin{cases} \max\left\{ \underbrace{\min_{b\in\mathcal{B}}\,(\,L(v;x,b) + r(x,b)\,)}_{①}\,,\ \underbrace{\lambda(\psi_2(x) - v(x))}_{②} \right\} = 0 \qquad \text{for } x \text{ in } \mathcal{X}, \\ v(x) = \psi_1(x) \qquad \text{for } x \in \partial\mathcal{X}, \end{cases} \tag{1.14}$$

since $\lambda > 0$, one can divide the term ② by λ, and get the variational inequality in the usual form used in viscosity solutions literature. In another usual way, Equation (1.14) can be written as:

$$\text{for } x \in \mathcal{X} \quad \begin{cases} \min_{b\in\mathcal{B}}\,(\,L(v;x,b) + r(x,b)\,) \leq 0 \\ \psi_2(x) - v(x) \leq 0 \\ \left(\min_{b\in\mathcal{B}}\,(\,L(v;x,b) + r(x,b)\,)\right)\,(\psi_2(x) - v(x)) = 0 \end{cases} \tag{1.15}$$

with $v(x) = \psi_1(x)$ for $x \in \partial\mathcal{X}$. Both Equation (1.14) and Equation (1.15) are called variational inequalities. Note however, that Equation (1.14), or the resulting equation obtained by simplifying by λ in ②, reveals more the control nature and can be used to define viscosity solutions (where one needs to write equations in the form $F(x,v(x),Dv(x),D^2v(x)) = 0$ on $\mathcal{X}$), whereas Equation (1.15) is more adapted to a variational approach.

As for (1.13), if v is a classical solution of (1.14) or (1.15), if for all x in $\mathcal{X}$: $\alpha(x)$ is equal to 1 or 0 if resp. ① or ② is maximum in (1.14) and if for all x in $\mathcal{X}$: $\beta(x,1)$ is the action $b \in \mathcal{B}$ which realizes the minimum in ①, then an optimal pure Markovian stationary strategy is obtained by taking $\eta_t = \beta(\xi_t,1)$ and κ equal to the first time when $\alpha(\xi_t) = 0$. So this equation behaves as Equation (1.13) but where the first player has a discrete action space equal to $\{0,1\}$, 1 meaning continue to play and 0 meaning stop the game. This variational inequality can be treated with the same methods as (1.13).

1.3.3 Discretization

Several discretization methods may transform equations (1.13) or (1.14) into a dynamic programming equation of the form (1.7). This is the case when using Markov discretization of the diffusion (1.9) as in [Kus77, KD92] and in general when using discretization schemes for (1.13) or (1.14) that are monotone in the sense of [BS91]. One can obtain such discretizations by using the simple finite difference scheme below when there are no mixed derivative (that is $\sigma\sigma^T$ is a diagonal matrix). Under less restrictive assumptions on the coefficients, finite difference schemes with larger stencil also lead to monotone schemes [BZ03, MZ05]. In the deterministic case (when $\sigma \equiv 0$), one can also use a semi-Lagrangian scheme [BFS94, BFS99] or a max-plus finite element method [AGL08], both of them having the property of leading to a discrete equation of the form (1.7).

We suppose that $\mathcal{X}$ is the d-dimensional open unit cube and that there are no mixed derivatives ($q_{i,j}(x,a,b) = 0$ if $i \neq j$, $i,j \in \{1,\dots,d\}$). Let $h = \frac{1}{m}$ ($m \in \mathbb{N}^*$) denote the finite difference step in each coordinate direction, e_i the unit vector in the i^{th}-coordinate direction, and $x = (x_1,\dots,x_d)$ a point of the uniform grid $\mathcal{X}_h = \mathcal{X} \cap (h\mathbb{Z})^d$. Equation (1.13) is discretized by replacing the first and second order derivatives of v by the following approximation, for $i = 1,\dots,d$:

$$\frac{\partial v(x)}{\partial x_i} \sim \frac{v(x+he_i) - v(x-he_i)}{2h} \tag{1.16}$$

or

$$\frac{\partial v(x)}{\partial x_i} \sim \begin{cases} \dfrac{v(x+he_i) - v(x)}{h} & \text{when} \quad g_i(x,a,b) \geq 0 \\ \dfrac{v(x) - v(x-he_i)}{h} & \text{when} \quad g_i(x,a,b) < 0. \end{cases} \tag{1.17}$$

$$\frac{\partial^2 v}{\partial x_i^2}(x) \sim \frac{v(x+he_i) - 2v(x) + v(x-he_i)}{h^2}, \tag{1.18}$$

Approximation (1.16) may be used when L is uniformly elliptic and h is small, whereas (1.17) has to be used when L is degenerate (see [Kus77, KD92]). For equations (1.13) and (1.14), these differences are computed in the entire grid $\mathcal{X}_h$, by prolonging v on the "boundary" $\partial\mathcal{X}_h := \partial\mathcal{X} \cap (h\mathbb{Z})^d$ using Dirichlet boundary condition:

$$v(x) \;=\; \psi_1(x) \qquad \forall\, x \in \partial\mathcal{X} \cap (h\mathbb{Z})^d.$$

We obtain a system of N_h non linear equations of N_h unknowns, the values of the function $v_h : x \in \mathcal{X}_h \mapsto v_h(x) \in \mathbb{R}$:

$$\max_{a\in\mathcal{A}} \big(\min_{b\in\mathcal{B}} \big(L_h(v_h;(x,a,b)) + r(x,a,b) \big) \big) \;=\; 0 \quad \forall\, x \in \mathcal{X}_h \;, \tag{1.19}$$

where $N_h = \sharp\mathcal{X}_h \sim 1/h^d$ and L_h is a function which to $v \in \mathbb{R}^{\mathcal{X}_h}$, $x \in \mathcal{X}_h$, $a \in \mathcal{A}$, $b \in \mathcal{B}$ associates the approximation of $L(v;x,a,b)$.

The discretization is monotone in the sense of [BS91], then if (1.13) has a unique viscosity solution, the solution v_h of (1.19) converges uniformly to the solution v of (1.13) [BS91].

Moreover, multiplying Equation (1.19) by ch^2 with c small enough, it can be rewritten in the form (1.7), with a discount factor $\mu = 1 - O(\lambda ch^2)$. A similar result holds for the discretization of (1.14) (by multiplying only the diffusion part by ch^2).

We refer to Section 3.2.1 for an example of an Isaacs Equation (3.3) whose discretization (using scheme (1.17)–(1.18)) yields an Equation (3.4) which has the form of (1.7).

Chapter 2

Methods for solving linear systems

In this chapter, we give an overview of methods for the solution of linear systems. In particular, we are interested in multigrids methods. In Section 2.3, we explain multigrids methods for non singular linear systems and in Section 2.4, we give some specific multigrid methods to find the stationary probability of an irreducible Markov Chain. Before entering in the multigrid methods sections, we give a short survey of direct and basic iterative methods to solve non singular linear systems.

2.1 Direct solvers for linear systems

We consider here the following system of n linear equations:

$$A\,v \;=\; b \tag{2.1}$$

where $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$ are given and we are looking for the solution $v \in \mathbb{R}^n$ of the linear system. The most known direct solver is the *Gauss elimination* method (see for instance [GVL96] and references therein). It is the most used algorithm when the matrix A is square, dense and unstructured. This algorithm is based on the idea that triangular systems are easy to solve (they require n^2 flops or arithmetic operations) and that the matrix of the system A can be decomposed into the product of two triangular matrices. The Gauss elimination scheme to solve Equation (2.1) works as follows.

1. Factorize the matrix A into two triangular matrices, that is :

   $$A = L\,U \;\;,$$

   where $L \in \mathbb{R}^{n\times n}$ and $U \in \mathbb{R}^{n\times n}$ are respectively lower unitary triangular and upper triangular matrices. This LU decomposition is sometimes called Doolittle.
2. Compute the solution $w \in \mathbb{R}^n$ of the lower linear system $L\,w = b$.
3. Solve the upper linear system : $U\,v = w$.

The LU factorization is the most costly part of the Gauss elimination algorithm, it requires $\frac{2}{3}n^3$ flops. The following theorem of [GVL96], states the existence and unicity of the LU decomposition.

Theorem 2.1 ([GVL96]). *The LU factorization of a square matrix $A \in \mathbb{R}^{n\times n}$ exists if the determinants* $\det(A_{1\dots k,1\dots k}) \neq 0$ *for* $k = 1,\dots,n-1$*, where $A_{1\dots k,1\dots k}$ is the submatrix composed of the k first lines and columns of A. When the diagonal of L is fixed, if the LU factorization exists and the matrix A is non-singular, the LU factorization is unique and* $\det(A) = u_{11}\dots u_{nn}$ *where u_{ii} are the diagonal elements of U.*

However, the LU factorization may be unstable when it involves small pivots (see [GVL96]). To avoid this poor behavior of the algorithm, one can permute the rows and columns of the matrix. These methods are known as partial or complete pivoting and are commonly used to stabilize Gaussian elimination, see for instance [GVL96] for more details. Other factorizations also exist that are more adapted to some special classes of matrices.

The above Gauss elimination algorithm is efficient to solve dense and unstructured systems, however it is quite expensive for solving sparse systems. Indeed, the LU factorization may lead to dense triangular matrices L and U. This phenomenon, usually called fill-in, generates costs in both storage and computation. However the cost of the Gauss elimination method can be reduced by exploiting the sparsity of the linear system. This can be done by finding a good ordering (permutation) of the matrix of the system, see for instance [GL81] for symmetric matrices and [Dav06] for non symmetric matrices. For instance, the Curthill-McKee [CM69] algorithm is an algorithm to permute sparse symmetric matrices which is based on the class of breadth-first search algorithms from graph theory, see [CLRS01]. Note that finding the best ordering is an NP-complete problem [Yan81]. However good heuristic algorithms are known, see for instance [Dav04, DEG$^+$99, Dav06] and references therein. In our numerical tests, we use both libraries UMFPACK and SuperLU which include suitable Gauss elimination algorithms or LU solvers for non symmetric and unstructured sparse matrices.

Beside the class of direct solvers, a large amount of iterative algorithms exist to solve sparse linear systems. Starting from an initial approximation of the solution of Equation (2.1), an iterative method for solving a linear system consists in generating a sequence of successive approximations which converges to the solution of the linear system.

2.2 Basic iterative methods

Consider the linear System (2.1), a *smoothing process* or *relaxation scheme* or *splitting method* is an iterative method to solve a linear system having the following specificity. If we split the matrix $A = M - N$ with $M \in \mathbb{R}^{n\times n}$ a non singular matrix, starting from an initial approximation $v^{(0)} \in \mathbb{R}^n$, one smoothing step consists in the following approximation :

$$v^{(k+1)} \;=\; S\,v^{(k)} \;+\; M^{-1}\,b \quad \text{with} \quad S \;=\; M^{-1}\,N$$

where $S \in \mathbb{R}^{n\times n}$ is called the smoothing operator or smoother and M^{-1} is an approximation of the inverse of A. Consequently, the error $e^{(k)} = v^{(k)} - v$ at iteration k propagates as :

$$e^{(k+1)} \;=\; S\,e^{(k)} \;\;.$$

The sequence of iterates $(v^{(k)})_{k\geq 0}$ converges to the solution v of the System (2.1) if $\rho(S) < 1$ where $\rho(S) = \max_i |\lambda_i|$ is the *spectral radius* of S with λ_i its eigenvalues, see [Var62, GVL96, Saa03, BP94].

A simple iterative method, called *Richardson iteration*, consists in the following iterations :

$$v^{(k+1)} = v^{(k)} + (b - A\,v^{(k)}) \ ,$$

then $S = (I - A)$ and $M = I$. When $A = I - P$ where $P \in \mathbb{R}_+^{n\times n}$ is a substochastic matrix, it is equivalent to the successive approximations :

$$v^{(k+1)} = Pv^{(k)} + b \ ,$$

hence the convergence of the method depends on the spectral radius of P.

Let us now denote by D, L, U, respectively the diagonal, the lower triangular part and the upper triangular part of the matrix A such that $A = D - L - U$. An example of smoothing process is the Jacobi relaxation. First, we write the *Jacobi method* in equation form at step $k + 1$:

$$\begin{aligned} v_i^{(k+1)} &= \frac{1}{a_{ii}} \left(b_i - \sum_{j\neq i} a_{ij}\, v_j^{(k)} \right) && i \in \{1, \ldots, n\} \\ v_i^{(k+1)} &= \frac{-1}{a_{ii}} \left(\sum_{j\neq i} a_{ij}\, v_j^{(k)} \right) + \frac{1}{a_{ii}}\, b_i && i \in \{1, \ldots, n\}. \end{aligned}$$

The smoother operator of the Jacobi method is then given by

$$S = D^{-1}\,(L + U)$$

with $M = D$ and $N = (L + U)$. Another example is the *Gauss-Seidel method* whose component form for the $(k + 1)$th-iteration is given by :

$$v_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}\, v_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}\, v_j^{(k)} \right) \quad i \in \{1, \ldots, n\} \tag{2.2}$$

$$\left(\sum_{j=1}^{i-1} a_{ij}\, v_j^{(k+1)} \right) + a_{ii}\, v_i^{(k+1)} = \left(- \sum_{j=i+1}^{n} a_{ij}\, v_j^{(k)} \right) + b_i \qquad i \in \{1, \ldots, n\} \tag{2.3}$$

and its smoothing operator is $S = (D - L)^{-1}U$ with $M = (D - L)$ and $N = U$. Unlike the Jacobi method, the Gauss-Seidel uses the most updated information when updating each component of the approximation. Several variants of the Gauss-Seidel smoother exist, one can for instance change the order of the updating of the components of the approximation.

Another method is the *Successive Over-Relaxation (SOR)* iterative method. It is derived from the Gauss-Seidel relaxation scheme. Denote by $\tilde{v}^{(k+1)}$ the iterate defined by

the Gauss-Seidel iteration in Equation (2.2), then the iteration $k+1$ of the SOR method is given by :

$$v_i^{(k+1)} = (1-w)v_i^{(k)} + w\,\tilde{v}_i^{(k+1)} \qquad i \in \{1,\ldots,n\}$$

where $0 < w < 2$ is the relaxation factor. Note that when $w = 1$, this is exactly the Gauss-Seidel scheme. Writing the equation without the intermediate iteration and in matrix form, we obtain :

$$v^{(k+1)} = S_w\, v^{(k)} + w\,(D - wL)^{-1}\, b\,, \tag{2.4}$$

where $S_w = (D - wL)^{-1}\,[(1-w)D + wU]$ is the SOR relaxation operator.

See for instance the books of [Var62, GVL96, Saa03, BP94] for more details about these methods and other non basic iterative algorithms. In the next section, we describe another class of iterative methods called multigrids methods.

2.3 Multigrids methods for non singular linear systems

In this section, we recall the *multigrids methods* which are used to solve linear systems. In the first subsection, we recall briefly what are the Geometric multigrid methods and the rest of the chapter will be about Algebraic multigrids methods. The main references which were used to describe here the general framework of the multigrids methods are the following [RS87, Stü01, BHM00].

2.3.1 Geometric multigrid methods

Standard multigrid was originally created in the seventies to efficiently solve linear systems of the form of Equation (2.1) that arise from the discretization of elliptic partial differential equations (see for instance [McC87]). Assume that a discretization scheme is chosen and the discretization of the partial differential equation on a grid is given. To get a good approximation of the solution, this grid, called "fine grid" has a small step size in each direction.

The multigrid method requires a sequence of grids, each of them is a discretization of the domain of the continuous equation with a different mesh (or step) size. Starting from the finest grid, each of them are constructed by enlarging the step size of the grid, i.e by reducing the number of nodes. The coarsest grid which is the grid with the fewest nodes, is chosen such that the cost of solving the linear system on it is cheap. When those grids are given and also the transfer operators between these grids: interpolation and restriction, a multigrid cycle on the finest grid consists in : first, the application of a smoother on the finest grid; then a restriction of the residual on the next coarse grid; then solving the residual problem on this coarse grid using the same multigrid scheme; then, interpolate this solution (which is an approximation of the error) and correct the error on the fine grid; finally, the application of a smoother on the finest grid. This algorithm is explained in detail in the next section for the algebraic multigrid case.

If the multigrid components are properly chosen, this process is efficient to find the solution on the finest grid. Indeed, in general the relaxation process is smoothing the error which then can be well approximated by elements in the range of the interpolation. It implies, in good cases, that the contraction factor of the multigrid method is independent of the discretization step and also the complexity is in the order of the number of discretization nodes. We shall refer to this standard method as *geometric multigrid*.

2.3.2 Algebraic multigrid methods

Let us now introduce the *algebraic multigrid* method, called AMG. The algebraic multigrid method has been initially developed in the early eighties (see for example [Bra86, BMR85, RS87]) for solving large sparse linear systems arising from the discretization of partial differential equations with unstructured grids or PDE's not suitable for the application of the geometric multigrid solver or large discrete problems not derived from any continuous problem.

Recall that we have to solve a system of n linear equations :

$$A\,v \;=\; b \tag{2.5}$$

where $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$ are given and we denote by $v \in \mathbb{R}^n$ the exact solution of the linear system. We denote by $u \in \mathbb{R}^n$ an approximation of the exact solution v. For theoritical studies, the matrix A is usualy assumed to be sparse symmetric and semi positive definite. In the context of algebraic multigrid, the finest grid, denoted by $\Omega^{|h}$ is defined as the set of all variables of the System (2.5), i.e. $\Omega^{|h} = \{1, \ldots, n\}$.

The AMG method consists of two main phases, called "setup phase" and "solving phase". In the "setup phase", one constructs coarse grids by selecting subsets of the fine grid $\Omega^{|h}$ only based on the linear system to be solved and not on the underlying continuous equation. The selection of those coarse variables and the construction of the transfer operators between levels are done in such a way that the range of the interpolation approximates well the errors not reduced by a given relaxation scheme. Then the "solving phase" is performed in the same way as a geometric multigrid method and consists of the application of a smoother and a correction of the error by a coarse grid solution.

Let us define the following inner product which is used in the algebraic multigrid literature :

$$\langle u, v\rangle_K = \langle Ku, v\rangle\,, \tag{2.6}$$

where $u, v \in \mathbb{R}^n$, and $K \in \mathbb{R}^{n\times n}$ is a symmetric definite positive matrix, $\langle\cdot,\cdot\rangle$ is the euclidean inner product of $\mathbb{R}^n$ and $\|u\|_K = (\langle u, u\rangle_K)^{1/2}$ is the associate norm. We shall used this norm for $K = D, A, D^{-1}$, where $D = \mathrm{diag}(A)$ is the diagonal part of A. The norm $\|\cdot\|_A$ is commonly called energy norm.

Before explaining the whole process of algebraic multigrid methods, we recall some properties of the relaxation schemes.

2.3.3 Smoothing property

A important concept in algebraic multigrid methods is the *algebraic smoothness*, see [RS87, Stü01]. An error vector e is said to be *algebraically smooth* if it is not efficiently reduced by the smoother S (i.e. $Se \approx e$), that is the error e contributes to the slowness of the smoothing process. The term *algebraic smoothness* is to be understood as *slow to converge*, the use of smooth came from the geometric multigrid where the algebraic smooth errors are also geometrically smooth.

The smoothing operator S satisfies the *smoothing property* if

$$\|S\,e\|_A^2 \;\leq\; \|e\|_A^2 \;-\; \sigma\|A\,e\|_{D^{-1}}^2 \qquad (\sigma > 0) \tag{2.7}$$

holds for all vector e independently of σ. It means that the smoother S reduces efficiently the errors e which are such that $\|A\,e\|_{D^{-1}}$ is significantly larger than $\|e\|_A$. However, the smoother is not efficient in reducing the errors such that $\|A\,e\|_{D^{-1}} << \|e\|_A$ which are the so-called algebraically smooth errors. The eigenvectors of $D^{-1}A$ corresponding to its small eigenvalues, are those which characterize the slowness of the smoothing process (see [Stü01]). Indeed, if we consider the Jacobi smoother, the eigenvalues of $S = D^{-1}(L + U) = (I - D^{-1}A)$ which are close to one are responsible for the slow convergence, these ones are the small eigenvalues of $D^{-1}A$. This is also the case for the Gauss-Seidel smoothing process [Stü01]. For instance, for systems arising from the discretization of standard elliptic partial differential equation on a regular grid with mesh size h, the smallest eigenvalues λ of $D^{-1}A$ are in the order of $O(h^2)$. If h is small, they are close to zero and thus the corresponding eigenvalues of S are close to one.

It is said that S satisfies the smoothing property w.r.t. a class $\mathcal{A}$ of matrices if the property (2.7) is satisfied uniformly for all matrices in $\mathcal{A}$ with the same σ.

Note that $\sigma\|A\,e\|_{D^{-1}}^2 \leq \|e\|_A^2$ is necessary for inequality (2.7) to hold. Because $\rho(D^{-1}A) = \sup \frac{\|A\,e\|_{D^{-1}}}{\|e\|_A}$, this condition is equivalent to the uniform boundness of $\rho(D^{-1}A)$ which is verified for all important class of matrices in consideration.

Before using this concept of algebraic smoothness for the construction of the grids (which will be the subject of Section 2.3.5), we first explain the solution phase in the next section and its convergence in the following section.

2.3.4 Solution phase

In this section, we explain the different schemes that can be used for the solution phase of the algebraic multigrid methods. We assume in this section that the coarse grids are given with the corresponding transfer operators : restriction and interpolation. Those operators are supposed to be properly chosen such that smooth errors are well approximated in the range of the interpolation operators. First, we shall consider a simple scheme with two grids and then we extend this scheme to more grids.

A two-grid scheme

We assume that during the construction phase of AMG, the n variables from the fine level $\Omega^{|h}$ have been split into two distinct subsets, namely C which contains the variables belonging to both fine and coarse grid, and F the variables belonging to the fine grid only, such that $\Omega^{|h} = C \cup F$. The coarse grid is denoted by $\Omega^{|H}$ and contains the n_C variables of C. We denote by $\mathcal{P}_H^h : \mathbb{R}^{n_C} \to \mathbb{R}^n$, the prolongation or interpolation operator which maps vectors from the coarse grid to the fine grid. The restriction operator is denoted by $\mathcal{R}_h^H : \mathbb{R}^n \to \mathbb{R}^{n_C}$ and maps vectors from the fine grid to the coarse grid. Often the restriction $\mathcal{R}_h^H$ is chosen to be the transpose of the interpolation $\mathcal{P}_H^h$, i.e. $\mathcal{R}_h^H = (\mathcal{P}_H^h)^T$. Then, we can define the coarse grid operator by $A^{|H} = \mathcal{R}_h^H A^{|h} \mathcal{P}_H^h$ where $A^{|h} = A$ and $A^{|H}$ is the approximation of A on $\Omega^{|H}$, which maps $\mathbb{R}^{n_C}$ to itself. For any vector $z^{|h} \in \mathbb{R}^n$, we denote $z^{|H} = \mathcal{R}_h^H z^{|h}$ its restriction on the coarse grid $\Omega^{|H}$ and similarly, for any vector $z^{|H} \in \mathbb{R}^{n_C}$, we denote by $z^{|h} = \mathcal{P}_H^h z^{|H}$ its interpolation on the fine grid $\Omega^{|h}$. In like manner, we represent all the components defined on the fine grid with an exponent h and similarly with an exponent H if the component is defined on the coarse grid.

Now, we have all the ingredients to define a two-grid algebraic method which combines the action of a smoother on the fine grid and a coarse grid correction. Given a chosen smoother S and an initial approximation $u^{|h}$ on the fine grid, it consists in applying successively the two grid correction scheme $u^{|h} \leftarrow TG(u^{|h}, b^{|h})$ given in Algorithm 2.1 until the iterates converge where ν_1 and ν_2 are respectively the fixed number of pre and post relaxation to be performed on the fine grid.

In more details, one iteration of the two-grid scheme $u^{|h} \leftarrow TG(u^{|h}, b^{|h})$ consists in : first, applying ν_1 iterations of the chosen relaxation scheme on the fine level, then restrict the residual on the coarse level $r^{|H} = \mathcal{R}_h^H(b^{|h} - A^{|h}u^{|h})$, then solve the residual system defined on the coarse level by $A^{|H}e^{|H} = r^{|H}$ to find an approximation of the error (the error $e^{|h}$ on the fine level is assumed to be algebraically smooth according to the smoother S), next the approximation of the error is interpolate on the fine level, this approximation of the error is used to correct the approximation $u^{|h}$ on the fine grid, and finally, ν_2 smoothing iterations are applied to the new approximation $u^{|h}$.

Algorithm 2.1 Two grid correction scheme $u^{|h} \leftarrow TG(u^{|h}, b^{|h})$

relaxation phase : $u^{|h} \leftarrow Su^{|h} + M^{-1}b^{|h}$ (on $\Omega^{|h}$) ν_1 times

coarse grid correction : $u^{|h} \leftarrow u^{|h} + \mathcal{P}_H^h z^{|H}$ where $z^{|H}$ is solution of

$$A^{|H}z^{|H} = \mathcal{R}_h^H(b^{|h} - A^{|h}u^{|h}) \quad \text{(on the coarse grid } \Omega^{|H})$$

relaxation phase : $u^{|h} \leftarrow Su^{|h} + M^{-1}b^{|h}$ (on $\Omega^{|h}$) ν_2 times

The error propagator operator $\mathcal{T}^{|h}$ associated to the two grid correction scheme is given by :

$$\mathcal{T}^{|h} = S^{\nu_2}\left(I^{|h} - \mathcal{P}_H^h \left(A^{|H}\right)^{-1} \mathcal{R}_h^H A^{|h}\right) S^{\nu_1} \ ,$$

such that the error $e^{|h}$ propagates as follow :

$$e^{|h} \leftarrow \mathcal{T}^{|h} e^{|h} \ ,$$

where $I^{|h}$ is the identity operator on the fine grid which maps $\mathbb{R}^n$ to itself. This operator $\mathcal{T}^{|h}$ is called *coarse-grid correction operator*.

This scheme can be defined recursively with more grids as explained in the next subsection.

The multigrid schemes

Now, we assume that L grids are constructed during the setup phase with the corresponding transfer operators. We denote with an exponent l, the components define on level l and by n_l the number of nodes on the grid $\Omega^{|l}$. Then, going from the finest level $l = 0$ to the coarsest one $l = L - 1$, each grid $\Omega^{|l+1}$ is a subset of the grid $\Omega^{|l}$, that is

$$\Omega^{|L-1} \subset \Omega^{|L-2} \subset \cdots \subset \Omega^{|1} \subset \Omega^{|0} = \Omega^{|h} \ .$$

If we consider a fine level $l-1$ and a coarse level l for $l = 0, \ldots, L-1$, we denote $\mathcal{R}_{l-1}^{l}$ the restriction operator which maps a vector from $\mathbb{R}^{n_{l-1}}$ to $\mathbb{R}^{n_l}$, $\mathcal{P}_{l}^{l-1}$ the interpolation operator which maps a vector from $\mathbb{R}^{n_l}$ to $\mathbb{R}^{n_{l-1}}$, and $A^{|l} = \mathcal{R}_{l-1}^{l} A^{|l-1} \mathcal{P}_{l}^{l-1}$ the coarse grid operator on level l. We also denote by $S^{|l}$ the smoother defined on level l corresponding to the linear system $A^{|l} v^{|l} = b^{|l}$. The multigrid solution phase with L grids, consists in applying recursively the two-grid scheme, as described in Algorithm 2.2. Given a chosen smoother S and an initial approximation $u^{|0}$ on the fine grid, it consists in applying successively $u^{|0} \leftarrow MG(u^{|0}, b^{|0})$ until the iterates converge to the solution of the system. It is called V(ν_1,ν_2)-cycle if $\gamma = 1$ and W(ν_1,ν_2)-cycle if $\gamma = 2$ in Algorithm 2.2. The V-cycle and W-cycle are schematically represented in Figure 2.1.

Algorithm 2.2 Multi-level scheme $u^{|l} \leftarrow MG(u^{|l}, b^{|l})$

if $l < L - 1$ **then**
  pre relaxation :
    $u^{|l} \leftarrow S^{|l} u^{|l} + M^{-1} b^{|l}$ (on $\Omega^{|l}$) ν_1 times
  coarse grid correction :
    $b^{|l+1} \leftarrow \mathcal{R}_{l}^{l+1}(b^{|l} - A^{|l} u^{|l})$
    $u^{|l+1} \leftarrow 0$
    $u^{|l+1} \leftarrow MG(u^{|l+1}, b^{|l+1})$ γ times
    $u^{|l} \leftarrow u^{|l} + \mathcal{P}_{l+1}^{l} u^{|l+1}$
  post relaxation :
    $u^{|l} \leftarrow S^{|l} u^{|l} + M^{-1} b^{|l}$ (on $\Omega^{|l}$) ν_2 times
else
  Solve $A^{|L-1} u^{|L-1} = b^{|L-1}$
end if

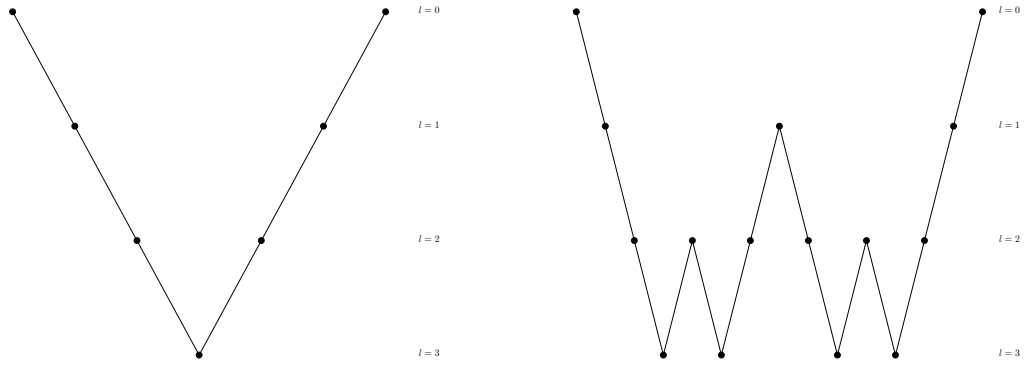

Figure 2.1: Graphical illustration of a V-cycle (on the left) and a W-cycle (on the right) with 4 grids.

Note that on each level l, with $l < L-1$, the coarse grid correction consists in solving the residual system on the next coarse level $l+1$ using the same multigrid scheme $u^{|l+1} \leftarrow MG(u^{|l+1}, b^{|l+1})$ with initial approximation $u^{|l+1} = 0$, since this is an intuitive approximation for the error. On level $l = L-1$, i.e. on the coarsest level, the system $A^{|L-1}u^{|L-1} = b^{|L-1}$ is assumed to be exactly solved.

Besides the schemes based on the two grid correction scheme, there exist other schemes such as the full multigrid scheme which is presented in the next section.

Full multigrid scheme

Another multigrid scheme called full multigrid (FMG) is based on the idea of starting, at each level, the multigrid Algorithm 2.2 with a good initial guess. The scheme starts from the coarsest level $L-1$ on which the problem is solved, this solution is then interpolate to the next level $L-2$ and is used as initial approximation to start the multigrid cycle on this level. Again the solution is interpolated on the next level $L-3$ to start the multigrid cycle, this scheme is repeated until the finest grid is attained. Therefore interpolation operators are defined and may differ from those used in the multigrid cycle. The full multigrid scheme starts from the coarsest grid and is given in Algorithm 2.3 and schematically represented in Figure 2.2.

Algorithm 2.3 Full multigrid scheme $u^{|l} \leftarrow FMG(b^{|l})$

if $l < L-1$ **then**
  $b^{|l+1} \leftarrow \mathcal{R}_l^{l+1}\, b^{|l}$
  $u^{|l+1} \leftarrow FMG(b^{|l+1})$
  $u^{|l} \leftarrow \mathcal{P}_{l+1}^{l}\, u^{|l+1}$
end if
$u^{|l} \leftarrow MG(u^{|l}, b^{|l})$ γ times

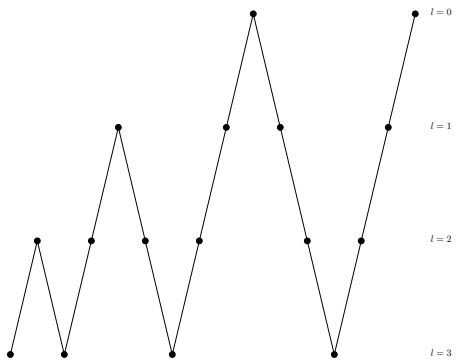

Figure 2.2: Graphical illustration of a FMG-cycle with 4 grids.

Theoretical convergence of the V-cycle of [RS87]

We recall here a theorem of convergence for the V-cycle and estimation of the convergence factor from [RS87]. First, we define the *(l,l+1) coarse grid correction operator* for all level l by :

$$\mathcal{T}^{|l} \;=\; I^{|l} \;-\; \mathcal{P}^{l}_{l+1}\,(A^{|l+1})^{-1}\,\mathcal{R}^{l+1}_{l}\,A^{|l} \;\;.$$

Assuming $A^{|l}$ to be symmetric and positive definite, the interpolation operators $\mathcal{P}^{l}_{l+1}$ having full rank and $\mathcal{R}^{l+1}_{l} = (\mathcal{P}^{l}_{l+1})^T$, then all the operators $\mathcal{T}^{|l}$ are orthogonal projections with respect to the energy product $\langle\cdot,\cdot\rangle_A$ defined on level l. In particular, we have that Range($\mathcal{T}^{|l}$) is orthogonal to Range($\mathcal{P}^{l}_{l+1}$) with respect to $\langle\cdot,\cdot\rangle_A$, $\|\mathcal{T}^{|l}\|_A = 1$ and for all $e^{|l}$: $\|\mathcal{T}^{|l}e^{|l}\|_A = \min_{e^{|l+1}} \|e^{|l} - \mathcal{P}^{l}_{l+1}e^{|l+1}\|_A$, meaning that the energy norm of the error after one $(l, l+1)$ grid correction step is minimum with respect to changes in Range($\mathcal{P}^{l}_{l+1}$). Now, we give the theorem of [RS87].

Theorem 2.2 ([RS87])**.** *Assume A to be symmetric and positive definite, that the interpolation operators $\mathcal{P}^{l}_{l+1}$ have full rank, that the restriction operators are defined as $\mathcal{R}^{l+1}_{l} = (\mathcal{P}^{l}_{l+1})^T$ and the coarse grid operators are given, as before, by $A^{|l+1} = \mathcal{R}^{l+1}_{l}A^{|l}\mathcal{P}^{l}_{l+1}$. Furthermore, suppose that for all $e^{|l}$,*

$$\|S^{|l}\,e^{|l}\|^2_A \;\leq\; \|e^{|l}\|^2_A \;-\; \delta\,\|\mathcal{T}^{|l}\,e^{|l}\|^2_A \tag{2.8}$$

holds with some $\delta > 0$ independently of $e^{|l}$ and l. Then $\delta \leq 1$, and provided that the coarsest grid equation is solved and that at least one smoothing step is performed after each coarse grid correction step, the V-cycle to solve (2.5) *has a convergence factor (with respect to the energy norm) bounded above by $\sqrt{1-\delta}$.*

The condition (2.8) gives condition on both the smoothing operator $S^{|l}$ and the coarse grid correction operator $\mathcal{T}^{|l}$. Indeed, if for an error $e^{|l}$, $\mathcal{T}^{|l}$ is inefficient (i.e. $\|\mathcal{T}^{|l}e^{|l}\|_A \approx \|e^{|l}\|_A$), then $S^{|l}$ has to properly reduce the error. On the contrary, if for an error $e^{|l}$, $\mathcal{T}^{|l}$ is efficient, i.e. $\|\mathcal{T}^{|l}e^{|l}\|_A << \|e^{|l}\|_A$ and $e^{|l}$ is in the range of $\mathcal{T}^{|l}$, the smoother $S^{|l}$ is allowed to be inefficient. In practice, one can replace the condition (2.8) by a couple of

conditions : one condition on the smoother $S^{|l}$ which is the smoothing property given in Equation (2.7) and one condition on the the coarse grid correction operator :

$$\|\mathcal{T}^{|l}\, e^{|l}\|_A^2 \;\leq\; \beta \|e^{|l}\|_A^2 \;.$$

Nevertheless, as mentioned in [RS87], a realistic coarsening strategy can hardly satisfy those conditions for the V-cycle convergence except for linear systems arising from regular elliptic PDE's. Weaker conditions exist for two-level convergence for linear systems where the matrix of the system is an M-matrix (see Definition 2.3 below), symmetric and positive definite, see for instance [Bra86, RS87, FVZ05].

Note that if we require instead of a post-smoothing at least one pre-smoothing before each coarse grid correction scheme in Theorem 2.2, then [RS87] Inequality 2.8 is replaced by

$$\|S^{|l}\, e^{|l}\|_A^2 \;\leq\; \|e^{|l}\|_A^2 \;-\; \delta\, \|\mathcal{T}^{|l}\, S^{|l}\, e^{|l}\|_A^2 \tag{2.9}$$

and the V-cycle convergence factor is bounded above by $\frac{1}{\sqrt{1+\delta}}$. Similarly, if one uses a V-cycle with at least one post-smoothing and one pre-smoothing, after and before each coarse grid correction scheme respectively, then one of the conditions, (2.8) or (2.9), need to be hold, if both are satisfied with constant δ_1 and δ_2 respectively, then [RS87] the V-cycle convergence factor is bounded above by $\sqrt{\frac{1-\delta_1}{1+\delta_2}}$.

2.3.5 Setup phase : the classical way

In this section, we explain the different algorithms to construct the coarse grids. As explained before, the construction of the grids is made in such a way that the solution on the coarsest grid is cheap and such that the range of the interpolation operator approximates well the errors not reduced by a given relaxation scheme applied on the fine grid. We consider in this section multigrid methods where the restriction operator is the transpose of the interpolation operator. We also assume that the matrix A of the system is a *M-matrix* whose definition is given below; we refer to [PB74, BP94] and reference therein for a survey about M-matrices.

Definition 2.3 (M-matrix)**.** *A matrix A in $Z^{n\times n} = \{A \in \mathbb{R}^{n\times n} \mid a_{ij} \leq 0,\ i \neq j\}$ is called a* M-matrix *if A can be split into*

$$A = sI - B$$

such that $B \in \mathbb{R}_+^{n\times n}$ is a matrix with only nonnegative elements and $s \geq \rho(B)$.

Indeed, the matrices of the linear systems that we consider are all of the form $A = I - P$ where I is the identity matrix of $\mathbb{R}^{n\times n}$ and $P \in \mathbb{R}_+^{n\times n}$ is a stochastic or sub-stochastic matrix (see Step 1 Algorithm 1.1 or see Step 2 Algorithm 4.4), hence the matrices we consider are all M-matrices.

Classical coarse grids construction and interpolation operator

Many ways to construct grids exist, we shall explain some of them that we used in our code. In this subsection, we describe the classical method such as explained in [RS87, BHM00, Stü01]. In this method, the matrix A is also considered to be symmetric and positive-definite.

Recall that for common relaxation schemes, an algebraically smooth error e satisfies $\|A\,e\|_{D^{-1}} << \|e\|_A$, by definition of the norms it means that $\sum_i \frac{r_i}{a_{ii}} << \sum_i e_i$. On the average, we can expect to have $|r_i| << a_{ii}\,|e_i|$ for all i, it means that the residual is small compared to the error and that

$$a_{ii}\,e_i \,+\, \sum_{j\in N_i} a_{ij}\,e_j \,=:\, r_i \,\approx\, 0$$

where $N_i := \{j \neq i : a_{ij} \neq 0\}$ is the neighborhood of i. A good approximation of the algebraically smooth error is then given by :

$$e_i \,=\, \frac{-1}{a_{ii}} \sum_{j\in N_i} a_{ij}\,e_j \;\;. \tag{2.10}$$

Furthermore, we also know by the discussion of Section 2.3.3 that for an algebraically smooth error e, we have $\|e\|_A << \|e\|_D$. For M-matrices, see [RS87, Stü01], it follows that on the average for each i

$$\sum_{j\neq i} \frac{|a_{ij}|}{a_{ii}} \frac{(e_i \,-\, e_j)^2}{e_i^2} \,<<\, 1 \;\;. \tag{2.11}$$

Note that if a_{ij} is close to a_{ii}, meaning that $\frac{|a_{ij}|}{a_{ii}}$ is close to one, then $e_i - e_j \approx 0$. We conclude [RS87, Stü01] that *algebraically smooth error varies slowly in the direction of large (negative) connections*, i.e. from e_i to e_j if $\frac{|a_{ij}|}{a_{ii}}$ is relatively large.

These properties lead to the notion of strong connections between the nodes. We define the set of all nodes j that *strongly influence* i by :

$$S_i := \left\{ j \,\neq\, i \;\mid\; -a_{ij} \,\geq\, \theta \max_{a_{ik}<0} \{-a_{ik}\} \right\} \;\;, \tag{2.12}$$

where $\theta \in [0, 1[$ is fixed (usually taken $\theta = 0.25$). Note that the positive off-diagonal elements (if any) are not in S_i, this condition is added in case the matrix A is only closed to be a M-matrix, for instance when A contains small off-diagonal positive elements. The set of nodes that *strongly depend* on the node i is denoted by S_i^T and defined by $S_i^T := \{j \neq i \;\mid\; i \in S_j\}$. We define also some other useful sets :

$$C_i \,=\, S_i \cap C \;\;, \qquad D_i^s \,=\, S_i \cap F \;\;, \qquad D_i^w \,=\, N_i \setminus S_i \;\;. \tag{2.13}$$

Now, we consider a fine grid $\Omega^{|h}$ and we have to construct the coarse grid $\Omega^{|H}$ and the interpolation operator $\mathcal{P}_H^h : \mathbb{R}^{n_C} \to \mathbb{R}^n$. Assuming that the splitting of the fine grid is made such that $\Omega^{|h} = F \cup C$ where $\Omega^{|H} = C$, the interpolation operator $\mathcal{P}_H^h$ applied to a

vector $e^{|H}$ of $\Omega^{|H}$ is of the form:

$$(\mathcal{P}_H^h\ e^{|H})_i\ =\ \begin{cases} e_i^{|H} & i \in C \\ \displaystyle\sum_{j \in C} w_{ij}\, e_j^{|H} & i \in F \end{cases} \tag{2.14}$$

where w_{ij} are the coefficient of $\mathcal{P}_H^h$ such that for $i \in C$: $w_{ij} = 1$ if $j = i$ else $w_{ij} = 0$. To have a good interpolation, it is usually required that the constant vectors on the coarse grid are exactly interpolated on the fine grid, i.e. $\sum_{j \in C} w_{ij} = 1$ for all $i \in \Omega^{|h}$. Many choices of weights w_{ij} for the interpolation operator exist. In general, the interpolation operator has full rank. Its construction is based on the properties of the smooth error such as property (2.11) and more generally approximation (2.10). We give here the definition of the interpolation operator from the references [RS87, BHM00], for other possible definitions of interpolation operators, see for instance [RS87, Stü01].

In [RS87, BHM00], based on the equations (2.11), (2.10) and the definition of the sets (2.13), the interpolation weights w_{ij} in Equation (2.14) are given, for $i \in F$ and $j \in C_i$, by :

$$w_{ij} = -\ \frac{a_{ij}\ +\ \displaystyle\sum_{m\, \in\, D_i^s} \left(\frac{a_{im}\, a_{mj}}{\displaystyle\sum_{k\, \in\, C_i} a_{mk}} \right)}{a_{ii}\ +\ \displaystyle\sum_{l\, \in\, D_i^w} a_{il}}\ , \tag{2.15}$$

and $w_{ij} = 0$ for $j \notin C_i$. The denominator of the weights (2.15) is not null because of the property of A and the definition of the sets D_i^w, we have that $\sum_{l\, \in\, D_i^w} a_{il} << a_{ii}$. To ensure the existence of the numerator in Equation (2.15), we need to add the following assumption : if two nodes i and m, such that $i, m \in F$, are strongly connected then there must exist at least one common node in C_i and C_m, or equivalently :

$$\forall\, m \in D_i^s \quad \exists\, k \in C_i \quad : \quad m \in S_k^T\ . \tag{2.16}$$

This interpolation formula is used in our code and in our tests see Chapter 3.

Other interpolation formulas exist, see for instance [RS87]. In particular, for matrices that may have off diagonal positive elements. In this case, the interpolation weights w_{ij} for Equation (2.14) of [RS87] are given, for $i \in F$ and $j \in C_i$, by :

$$w_{ij} = \begin{cases} -\theta_i^-\ \dfrac{a_{ij}}{a_{ii}} & \text{if } a_{ij} < 0 \\ -\theta_i^+\ \dfrac{a_{ij}}{a_{ii}} & \text{if } a_{ij} > 0 \end{cases}$$

where

$$\theta_i^- = \frac{\sum_{j \in N_i}\ a_{ij}^-}{\sum_{k \in C_i}\ a_{ik}^-} \qquad \text{and} \qquad \theta_i^+ = \frac{\sum_{j \in N_i}\ a_{ij}^+}{\sum_{k \in C_i}\ a_{ik}^+}\ ,$$

and a_{ij}^+ and a_{ij}^- are respectively the positive and negative elements of the ith-row for $j \in N_i$.

Now, we still need to select the nodes which will belong to the coarse grid $\Omega^{|H}$. The construction of the coarse grid will be made in such a way that the solution on the coarsest grid is cheap, meaning that the coarse grid must contain substantially less nodes than the fine grid. And the range of the interpolation operator from one coarse grid to a fine grid, approximates the errors not reduced by a given relaxation scheme applied on the fine grid. We focus on the methods presented in [RS87, Stü01] which are the ones we used in the implementation of our code (see Chapter 6). From [RS87, Stü01], two conditions are mentioned to try to satisfy these conditions :

1. For each $i \in F$, each node $j \in S_i$ either should be in C_i, or should be strongly connected to at least one node in C_i, i.e. :

$$\forall\, i \in F \qquad \forall\, j \in S_i \quad \begin{cases} \text{either} & j \in C_i, \\ \text{either} & \exists\, k \in C_i \;\; k \in S_j \;\; , \end{cases} \tag{H1}$$

2. C should be a maximal subset of all nodes $\Omega^{|h}$ with the property that no two nodes in C are strongly connected to each other, i.e. :

$$C \;=\; \max \left\{ K \subset \Omega^{|h} \;\; | \;\; \forall\, i \in K \;\; \nexists\, j \in K \quad j \in S_i \right\} \;. \tag{H2}$$

The first assumption (H1) ensure the existence of the weights of the interpolation operator, indeed Equation (H1) implies condition (2.16). The second assumption (H2) limits the size of the coarse. In practice, both assumptions are usually not satisfied together, (H1) has priority on (H2) to ensure the existence of the interpolation operator.

In [RS87, BHM00], the selection of the coarse nodes C is made in two passes which are called *coloring schemes*. The *first coloring scheme* consists in trying to satisfy (H2) and the *second coloring scheme* to ensure (H1). The *first coloring scheme* of [RS87] is presented in Algorithm 2.4. It consists in first assigning to each node $i \in \Omega^{|h}$, a measure of possibly being in C. This measure, denoted by λ_i and defined as the number of nodes that strongly depend on i, is then used in Algorithm 2.4 to make the first splitting $\Omega^{|h} = F \cup C$. The same algorithm is used in [BHM00] but without Step 3.5.

Now, we describe the *second coloring scheme* from [RS87, BHM00] given in Algorithm 2.5. It consists in passing over all nodes of F, then verify if condition (H1) is satisfied. It works as follow : for all nodes i in F, we check for all nodes j of D_i^s if there exists a node in C_i that strongly depends on j (i.e. $C_i \cap S_j \neq \emptyset$). If we do not find such a node, then node j becomes a candidate K for the coarse grid (C). Then we continue with the next node j in D_i^s and so on. If all the following nodes in D_i^s verify condition (H1) then the candidate K becomes a node of C; else if we find another candidate j in D_i^s that doesn't verify condition (H1), then it is the node i which becomes a node of C (indeed it means that there exist at least two nodes of F with no dependence in C which depend of node i). Then we go to the next node i in F. Note that we usually compute the interpolation operator during the *second coloring scheme*.

This construction of the coarse grid and interpolation operator can be done recursively, starting from the finest grid $\Omega^{|0} = \Omega^{|h}$ until the coarse grid is small enough, meaning it has

Algorithm 2.4 First coloring scheme $[C,\, F] \leftarrow SC1(\Omega^{|h})$

1. $C = \emptyset,\ F = \emptyset,\ K = \Omega^{|h}$,
2. **for all** $i \in \Omega^{|h}$ **do** $\lambda_i = \sharp(S_i^T)$ **end for**
3. **repeat**
   - 3.1 pick up i such that $\lambda_i = \max_{k \in \Omega^{|h}} \lambda_k$
   - 3.2 $C = C \cup \{i\},\ K = K \setminus \{i\}$
   - 3.3 **for all** $j \in S_i^T \cap K$ **do**
     $F = F \cup \{j\}$
     $K = K \setminus \{j\}$
     for all $k \in S_j \cap K$ **do**
     $\lambda_k = \lambda_k + 1$
     end for
   - 3.4 **end for**
   - 3.5 **for all** $j \in S_i \cap K$ **do** $\lambda_j = \lambda_j - 1$ **end for**
4. **until** $K = \emptyset$

Algorithm 2.5 Second coloring scheme $[C,\, F] \leftarrow SC2(C,\, F)$

```
K = ∅
for all i ∈ F do
  S_i = C ∩ S_i, D_i^s = S_i \ C_i
  for all j ∈ D_i^s do
    if C_i ∩ S_j = ∅ then
      if K = ∅ then
        C_i = C_i ∪ {j},  D_i^s = D_i^s \ {j}
        K = j
      else
        C = C ∪ {i},  F = F \ {i}
        break (for-loop)
      end if
    end if
  end for
  if K ≠ ∅ and i ∉ ℂ then
    C = C∪ K
  else
    K = ∅
  end if
end for
```

Algorithm 2.6 Setup Phase

$l = 0$
repeat
  for all i in $\Omega^{|l}$ **do**
    compute the sets S_i and S_i^T
  end for
  $[C, F] \leftarrow SC1(\Omega^{|l})$ using Algorithm 2.4
  $[C, F] \leftarrow SC2(C, F)$ using Algorithm 2.5
  construct $\mathcal{P}_l^{l-1}$ defined by equations (2.14) and (2.15)
  construct $\mathcal{R}_{l-1}^l = (\mathcal{P}_l^{l-1})^T$
  set $A^{|l} = \mathcal{R}_{l-1}^l A^{|l-1} \mathcal{P}_l^{l-1}$
  $l = l + 1$
until $\Omega^{|l}$ is small enough

few enough nodes. The setup phase associated to the methods of [RS87, BHM00] is given in Algorithm 2.6. Other algebraic multigrid methods have similar setup phase but with other construction algorithms. In the next section, we give another kind of construction of grid, which is called aggregation.

2.3.6 Setup phase : aggregation methods

In this section, we broach other common methods for the setup phase which are the *aggregation methods*. As in the previous section, we consider working on a fine grid $\Omega^{|h}$ and we have to construct the coarse grid $\Omega^{|H} = C$ with $\Omega^{|h} = F \cup C$ and the interpolation operator $\mathcal{P}_H^h : \mathbb{R}^{n_C} \to \mathbb{R}^n$. The concept is to partition the fine grid $\Omega^{|h}$ into disjoint subsets, called *aggregates* and denoted by G_k^h. In each aggregate, one node is selected to be a node of the coarse grid C and will serve to interpolate the other nodes of the aggregate which belong to the fine grid only, i.e. F. This kind of method was for instance introduced in [VMB96, VBM01].

It works as follow, the fine grid $\Omega^{|h}$ is divided in *aggregates*, denoted by G_k^h where $k \in C$ is the node of C. The other nodes $i \neq k$ in G_k^h are nodes of F. The number of aggregates is denoted by n_C. The partitioning in aggregates is based on the connections between the nodes, i.e. on the elements a_{ij} of the matrix A of the System (2.5). When those aggregates are selected, a piecewise constant interpolation operator $\mathcal{P}_H^h$ is defined by :

$$(\mathcal{P}_H^h)_{ij} = \begin{cases} 1 & \text{if } \ i \in G_j^h \\ 0 & \text{else } . \end{cases} \tag{2.17}$$

If we apply this interpolation operator to a vector e in $\Omega^{|H}$, we have that $(\mathcal{P}_H^h \ e)_i = e_k$ for all nodes i in the same aggregate G_k^h. By construction, the interpolation operator is of full rank and interpolates constant exactly. If the restriction operator is the transpose of the

interpolation operator, we have $\mathcal{R}_h^H = (\mathcal{P}_H^h)^T$ and the coarse grid operator is given by :

$$(A^{|H})_{kl} = (\mathcal{R}_h^H A^{|h} \mathcal{P}_H^h)_{kl} = \sum_{j \in G_l^h} \sum_{i \in G_k^h} (A^{|h})_{ij} \ .$$

The coarse grid operator is closely related to the choice of the aggregates. This scheme can be repeated recursively for more levels. The resulting algebraic multigrid method is not efficient due in part to the inaccuracy of the interpolation operator. In practice, this method is improved, for instance by constructing a more accurate interpolation operator [VMB96], or by using the new multigrid K-cycle [NV08]. We will explain these two methods in the next subsections.

Smooth aggregation

This method, called *smooth aggregation*, has been introduced by Vanek [VMB96]. We describe here the setup phase of the method, the solution phase is the same as in Section 2.3.4. It consists in first splitting the fine grid $\Omega^{|h}$ into disjoint aggregates, then constructing the piecewise constant interpolation operator defined by Equation (2.17), which is called the tentative prolongator and denoted by $\widetilde{\mathcal{P}}_H^h$, finally an iteration of Jacobi relaxation is applied to the tentative prolongator to get the final interpolation operator $\mathcal{P}_H^h$. The restriction operator, as before, is $\mathcal{R}_h^H = (\mathcal{P}_H^h)^T$ and the coarse grid operator $A^{|H} = \mathcal{R}_h^H A^{|h} \mathcal{P}_H^h$. This method can be repeated recursively for more levels.

This method can be applied to any System (2.5) with the assumption that the matrix A is symmetric positive definite. However, it is more adapted to systems arising from the discretization of second and fourth order elliptic problems by a finite element discretization [VMB96]. The convergence of this method is presented in [VBM01] and the application [VMB96] to second and fourth order elliptic problems discretized with finite element methods.

We only recall here the setup phase. First, we define the notion of neighborhood of [VMB96] which will be used to construct the aggregates. The neighborhood of a node i is given by :

$$N_i^h(\epsilon) \ = \ \{j \in \Omega^{|h} \ | \ |a_{ij}| \ \geq \ \epsilon \sqrt{a_{ii}\, a_{jj}}\}, \tag{2.18}$$

where $\epsilon \in [0, 1[$ is fixed. The algorithm of [VMB96] to generate the disjoint aggregates of the fine grid $\Omega^{|h}$ is described in Algorithm 2.7.

Now that we have the aggregates $\{G_k^h\}_{k=1}^{n_C}$, we can construct the interpolation operator. A first tentative piecewise constant prolongator, $\widetilde{\mathcal{P}}_H^h$, is defined using Equation (2.17). Then, the final interpolation operator is obtained by applying one smoothing step of the damped Jacobi relaxation on the operator $\widetilde{\mathcal{P}}_H^h$:

$$\mathcal{P}_H^h \ = \ (I \ - \ w\, D^{-1}\, A_F^{|h}) \, \widetilde{\mathcal{P}}_H^h, \tag{2.19}$$

where $A_F^{|h}$ is the *filtered matrix* defined as :

$$(A_F^{|h}){}_{ij} \ = \ \begin{cases} (A^{|h})_{ij} & \text{if } j \in N_i^h(\epsilon) \\ 0 & \text{else} \end{cases}$$

Algorithm 2.7 Aggregation : $\{G_k^h\}_{k=1}^{n_C} \leftarrow SAGGR(\Omega^{|h})$

Initialization :

for all $i \in \Omega^{|h}$ **do**

compute the set $N_i^h(\epsilon)$ defined by (2.18)

end for

Set $K = \{i \in \Omega^{|h} \mid N_i^h(0) \neq \{i\}\}$ (isolated nodes are not aggregate)

Set $n_C = 0$

Algorithm :

1. **Start up aggregation :** Select disjoint neighborhoods as an initial partitioning of the fine grid $\Omega^{|h}$:

   for all $i \in K$ **do**
   if $N_i^h(\epsilon) \subset K$ **then**
   $n_C = n_C + 1$
   $G_{n_C}^h \leftarrow N_i^h(\epsilon)$
   $K \leftarrow K \backslash G_{n_C}^h$.
   end if
   end for

2. **Enlarging the decomposition sets :** Add the remaining nodes $i \in K$ to one of the existing aggregates G_l^h to which node i is the (mostly) strongly connected :

   Copy all the aggregates : $\widetilde{G}_k^h \leftarrow G_k^h \qquad k = 1, \ldots, n_C$.
   for all $i \in K$ **do**
   if $\exists\, k \ : \ N_i^h(\epsilon) \cap \widetilde{G}_k^h \neq \emptyset$ **then**
   $G_k^h \leftarrow \tilde{G}_k^h \cup \{i\}$
   $K \leftarrow K \setminus \{i\}$
   end if
   end for

3. **Handling the remnants :** Make new aggregates with the remaining nodes

   for all $i \in K$ **do**
   $n_C = n_C + 1$
   $G_{n_C}^h \leftarrow K \cap N_i^h(\epsilon)$,
   $K \leftarrow K \setminus G_{n_C}^h$.
   end for

for $i \neq j$ and

$$(A_F^{|h})_{ii} = (A^{|h})_{ii} - \sum_{\substack{j \in \Omega^{|h} \\ j \neq i}} ((A^{|h})_{ij} - (A_F^{|h})_{ij}) \ .$$

The aim of using the filtered matrix in Equation (2.19) is to reduce the number of non zero elements in the interpolation operator.

From numerical experiments in [VMB96], the parameters are given by :

$$\epsilon = 0.08 \left(\frac{1}{2}\right)^{l-1}, \qquad w = \frac{2}{3},$$

where $l > 1$ is the number of level.

In the next section, we give the complete AMG algorithm independent of the choice of the Setup phase.

2.3.7 The AMG algorithm

We have now all the ingredients to define the AMG algorithm. It starts with the construction of the L grids and the corresponding transfer operators, by using for instance one of the methods of the previous sections: 2.3.5 or 2.3.6. Then, given an initial approximation $v^{(0)}$ on the fine grid, it consists in applying successively the V-cycles of Algorithm 2.2 until the iterates converge to a good approximation of solution of the system. The resulting algorithm is given in Algorithm 2.8.

Algorithm 2.8 AMG for linear system: $v \leftarrow AMG(v^{(0)}, A, b)$

1. Construct the coarse grid operators for $0 \leq l < L$:
   - build $\mathcal{P}_l^{l-1}$ and $\mathcal{R}_{l-1}^l$,
   - set $A^{|l} = \mathcal{R}_{l-1}^l A^{|l-1} \mathcal{P}_l^{l-1}$.
2. Initialize $k = 0$.
3. Compute $v^{(k+1)} = MG(v^{(k)}, b)$.
4. If $\|(b - Av^{(k+1)})\| < \epsilon$ then STOP and return $v^{(k+1)}$.
5. Else, set $k = k + 1$ and go to Step 3.

In the next section, we consider another aggregation method without smoothing the interpolation operator. Instead a better multigrid cycle is proposed.

2.3.8 AGMG

The aggregation method, called *AGMG*, has been introduced by [Not10a]. The main idea is the use of aggregation based algebraic multigrid without smoothing the interpolation operator and to improve the convergence by introducing a new multigrid cycle [NV08] in the solution phase, called *K-cycle*, providing Krylov subspaces acceleration. The AGMG method is proposed as a preconditioner that is applicable as a black box solver for linear

systems. We recall in this section the method from [Not10a]. For an introduction to preconditioner methods and Krylov subspace iterations, see references [GVL96, Saa03].

The setup phase of *AGMG* consists, as for other aggregation methods, in splitting the fine grid $\Omega^{|h}$ into n_C disjoint aggregates. In *AGMG*, the aggregation is performed in two passes. The first pass consists in applying a pairwise aggregation, recalled here in Algorithm 2.9, which constitutes a first tentative partition of the grid $\Omega^{|h}$. This aggregation algorithm is based on strongly negative connections between the nodes like in classical algebraic multigrid, see Section 2.3.5. The simple pairwise aggregation results in a relatively slow multigrid method so that a second pass is proposed. The resulting algorithm is given in Algorithm 2.10, which splits a fine grid $\Omega^{|h}$ into n_C disjoint aggregates : $\{G_k^h\}_{k=1}^{n_C}$. From numerical experiences from [Not10a], the resulting aggregates are commonly quadruplets with some aggregates of size less than four. This implies that the coarsening rate is approximately a bit less then four. The setup phase ends by building a piecewise constant interpolation operator, $\mathcal{P}_H^h$, using Equation (2.17). The restriction operator is defined by $\mathcal{R}_h^H = (\mathcal{P}_H^h)^T$ and the coarse grid operator by $A^{|H} = \mathcal{R}_h^H A^{|h} \mathcal{P}_H^h$. This method can be repeated recursively for more levels.

Note that Algorithm 2.9 takes an optional parameter check, denoted by $CKDD$, if it is true then nodes associated to row with strongly dominant diagonal element, are not aggregated. This option is based on a heuristic view [Not10a] that those nodes have fast enough corresponding error reduction without multigrids help.

The solving phase of *AGMG* consists then in applying multigrid K-cycles as a preconditioner to an adapted iterative method. It is proposed to use flexible conjugate gradient method (FCG) if the matrix is symmetric positive definite and a preconditioned variant of GCR, called GMRESR. First, we describe in Algorithm 2.11, the action of a multigrid preconditioner to a residual vector $r^{|l}$ at a level l, as given in [NV08, Not10a]. A V-cycle MG preconditioner consists in recursively calling the MG preconditioner on the next coarse level, a K-cycle instead applies one or two steps of a Krylov subspace iterative method. In practice, a K-cycle may not be called at each level but instead a mix V-cycle and K-cycle is used, see [Not10a] for details.

Note that a package in FORTRAN implementing *AGMG* is freely available for download on the website of its author [Not10a]. Interfacing functions are implemented in the code of *PIGAMES* to enable the use of this package when resolving linear systems.

2.3.9 Other methods

Also we can find in the literature two-grid convergence analysis for non-symmetric linear systems in [Not10b], also in [MN08] which is based on the analysis of AMLI, a block incomplete factorization partitioned in hierarchical form.

In the next section, we shall present an overview of some methods to find the stationary probability of an irreducible Markov Chain. We will see that multigrid methods and other closely related methods exist for such problems. These methods will be used in the last chapters, Chapter 4 and Chapter 5, for solving mean payoff stochastic games.

Algorithm 2.9 Pairwise aggregation : $\{G_k^h\}_{k=1}^{n_C} \leftarrow PairwiseAGGR(A^{|h}, CkDD)$

Initialization :
if CkDD = TRUE **then**
$K = \Omega^{|h} \setminus \left\{ i \mid a_{ii} > 5 \sum_{j \neq i} |a_{ij}| \right\}$
else
$K = \Omega^{|h}$
end if
for all $i \in K$ **do**
compute S_i as defined in (2.12)
set $\lambda_i = \sharp(S_i)$
set $n_C = 0$
end for
Algorithm :
while $K \neq \emptyset$ **do**
select $i \in K$ with minimal λ
$n_C = n_C + 1$
select $j \in K$ such that $a_{ij} = \min_{k \in K} \{a_{ik}\}$
if $j \in S_i$ **then** $G_{n_C}^h = \{i, j\}$ **else** $G_{n_C}^h = \{i\}$ **end if**
$K \leftarrow K \setminus G_{n_C}^h$
for all $k \in G_{n_C}^h$ **do**
for all $l \in S_k$ **do** $\lambda_l = \lambda_l - 1$ **end for**
end for
end while

Algorithm 2.10 Double Pairwise aggregation : $\{G_k^h\}_{k=1}^{n_C} \leftarrow 2PairwiseAGGR(A^{|h}, CkDD)$

Apply Algorithm 2.9 to $A^{|h}$:

$$\{\widetilde{G}_k^h\}_{k=1}^{\tilde{n}_C} \leftarrow PairwiseAGGR(A^{|h}, CkDD)$$

Compute the $\tilde{n}_C \times \tilde{n}_C$ auxiliary matrix $\widetilde{A}^h$ with elements :

$$\tilde{a}_{ij} = \sum_{k \in \widetilde{G}_i^h} \sum_{l \in \widetilde{G}_l^h} \tilde{a}_{ik}$$

Apply Algorithm 2.9 to $\widetilde{A}^h$:

$$\{\bar{G}_k^h\}_{k=1}^{n_C} \leftarrow PairwiseAGGR(\widetilde{A}^h, FALSE)$$

for $i = 1$ **to** n_C **do**
$G_i^h = \cup_{j \in \bar{G}_i^h} \bar{G}_j$
end for

Algorithm 2.11 MG as preconditioner at level l : $y^{|l} \leftarrow MGprec(r^{|l}, l)$

pre relaxation :

$y^{|l} \leftarrow M^{-1}\, r^{|l}$ $\quad \nu_1$ times (on $\Omega^{|l}$)

coarse grid correction :

$r^{|l} \leftarrow r^{|l} - A^{|l}\, y^{|l}$

$r^{|l+1} \leftarrow \mathcal{R}_l^{l+1}\, r^{|l}$

compute (approximate) solution $z^{|l+1}$ **of** $A^{|l+1}\, e^{|l+1} = r^{|l+1}$

if $l = L-1$ **then** $z^{|l+1} = (A^{|l+1})^{-1}\, r^{|l+1}$

else if V-cycle **then** $z^{|l+1} \leftarrow MGprec(r^{|l+1}, l+1)$

else if K-cycle **then** Perform 1 or 2 iterations with MG prec. :

$c_1^{|l+1} \leftarrow MGprec(r^{|l+1}, l+1)$

if FGC **then** $\alpha_1 = (c_1^{|l+1})^T\, r^{|l+1}$; $\beta_1 = (c_1^{|l+1})^T\, A^{|l+1}\, c_1^{|l+1}$ **end if**

if GCR **then** $\alpha_1 = (c_1^{|l+1}\, A^{|l+1})^T r^{|l+1}$; $\beta_1 = \|A^{|l+1}\, c_1^{|l+1}\|^2$ **end if**

$z^{|l+1} \leftarrow z^{|l+1} + \dfrac{\alpha_1}{\beta_1}\, c_1^{|l+1}$

if $\|r^{|l+1} - \dfrac{\alpha_1}{\beta_1}\, A^{|l+1}\, c_1^{|l+1}\| > \epsilon_{prec}\, \|r^{|l+1}\|$ **then**

$r^{|l+1} \leftarrow r^{|l+1} - \dfrac{\alpha_1}{\beta_1}\, A^{|l+1}\, c_1^{|l+1}$

$c_2^{|l+1} \leftarrow MGprec(r^{|l+1}, l+1)$

if FGC **then**

$\alpha_{12} = (c_2^{|l+1})^T\, A^{|l+1}\, c_1^{|l+1}$; $\alpha_2 = (c_2^{|l+1})^T\, r^{|l+1}$;

$\beta_2 = (c_2^{|l+1})^T\, A^{|l+1}\, c_2^{|l+1} - \dfrac{\alpha_{12}^2}{\beta_1}$;

end if

if GCR **then**

$\alpha_{12} = (A^{|l+1}\, c_2^{|l+1})^T\, A^{|l+1}\, c_1^{|l+1}$; $\alpha_2 = (A^{|l+1}\, c_2^{|l+1})^T\, r^{|l+1}$;

$\beta_2 = \|(A^{|l+1}\, c_2^{|l+1})^T\|^2 - \dfrac{\alpha_{12}^2}{\beta_1}$;

end if

$z^{|l+1} \leftarrow z^{|l+1} - \dfrac{\alpha_{12}\alpha_2}{\beta_1\beta_2}\, c_1^{|l+1} + \dfrac{\alpha_2}{\beta_2}\, c_2^{|l+1}$

end if

$y^{|l} \leftarrow y^{|l} + \mathcal{P}_{l+1}^{l}\, z^{|l+1}$

$r^{|l} \leftarrow r^{|l} - A^{|l}\, y^{|l}$

post relaxation :

$y^{|l} \leftarrow y^{|l} + M^{-1}\, r^{|l}$ $\quad \nu_2$ times (on $\Omega^{|l}$)

2.4 Find the stationary probability of an irreducible Markov Chain

Here, we consider the probability transition matrix $P \in \mathbb{R}^{n\times n}$ of a Markov Chain and we are looking for the *stationary probability vector* $\pi \in \mathbb{R}^n_+$ which satisfies the following equation :

$$\pi^T P = \pi^T, \qquad \|\pi\|_1 = 1 \tag{2.20}$$

where P is row stochastic (i.e. $\sum_{j\in[n]} P_{ij} = 1$, $i \in [n]$). Or equivalently, we have to solve the singular linear system :

$$A\,\pi = 0, \qquad \|\pi\|_1 = 1 \tag{2.21}$$

where $A = (I - P^T)$ with I the identity matrix in $\mathbb{R}^{n\times n}$. In this section, we shall consider only irreducible Markov chains, i.e. we assume that the corresponding transition probability matrices P are irreducible.

Definition 2.4 (Irreducible matrix)**.** *Let $A \in \mathbb{R}^{n\times n}_+$ be a non negative matrix. The matrix A is* irreducible *if and only if for every $i, j \in \{1, \cdots, n\}$, there exists a natural number $k \in N^*$ such that $a^k_{ij} > 0$.*

Definition 2.5 (Primitive matrix)**.** *Let $A \in \mathbb{R}^{n\times n}_+$ be a non negative matrix. The matrix A is* primitive *if and only if there exists a natural number $k \in N^*$ such that for every $i, j \in \{1, \cdots, n\}$, $a^k_{ij} > 0$ (i.e. $A^k \in (\mathbb{R}^*_+)^{n\times n}$).*

We recall the *Perron-Frobenius* theorem for non negative matrices from the book [Var62] and [BP94]. This theorem plays a key role in the analysis of Markov Chains.

Theorem 2.6 (Perron-Frobenius theorem)**.** *Let $A \in \mathbb{R}^{n\times n}_+$ be a non negative matrix. Then,*

1. *$\rho(A)$ is an eigenvalue of A, associated to a non negative eigenvector, $x \in \mathbb{R}^n_+$.*
2. *The spectral radius $\rho(A)$ is non decreasing with respect to any entry of A.*

If in addition A is irreducible, we have :

3. *To $\rho(A)$ corresponds a positive eigenvector $x \in (\mathbb{R}^*_+)^n$.*
4. *For any $B \in \mathbb{R}^{n\times n}_+$, if $B \leq A$ and $B \neq A$ then $\rho(B) < \rho(A)$.*
5. *$\rho(A)$ is a simple eigenvalue of A.*
6. *Each eigenvalue of A with modulus equal to $\rho(A)$ is also simple.*
7. *There exists $k \in N^*$ such that A has exactly k eigenvalues of modulus $\rho(A)$ which are precisely the distinct roots of $\lambda^k - (\rho(A))^k = 0$. This number k is called the cyclicity of A.*
8. *When A is primitive, the cyclicity k equals one.*

It follows from this theorem that Equation (2.20) has a unique solution, see [Var62, BP94].

Note that in the Markov chain's literature, we have the following terminology. Consider the transition stochastic matrix $P \in \mathbb{R}_+^{n\times n}$ (with $P\mathbf{1} = \mathbf{1}$) of a Markov chain. The matrix P is called *ergodic* if the matrix P has one final class, it is called *regular* if P is primitive and it is called *periodic* if P is irreducible and has cyclicity greater then one. The same adjectives are used for the corresponding Markov chains.

In the next subsections, we describe some methods from the literature to solve Equation (2.20). We will first consider direct solvers and then iterative solvers.

2.4.1 Direct Solver

A well-known direct solver for solving non singular linear systems is the Gauss elimination method (explained in section 2.1). In Equation (2.21), the linear system $A\pi = 0$ which is singular admits an infinite set of solutions and the codimension of the range of A is one. Hence, we can remove a row, $i \in \{1, \ldots, n\}$, of the matrix A without changing the set of solutions. To satisfy the condition $\|\pi\|_1 = 1$, we can for instance replace the removed row i by a row of ones and set the corresponding ith coordinate of the right hand side to one. However, this trick may slow down the Gauss elimination algorithm for sparse matrices. Hence, one can also set $\pi_i = 1$ in the linear system and at the end normalize the solution. Using one of both methods described above, the resulting non singular system can be solved with the Gauss elimination method, giving the unique solution of Equation (2.21).

Other direct methods have been developed to find the stationary probability of an irreducible Markov chain, see for instance [PSS96, Ste97] and below. In Equation (2.21), the matrix A is singular and the system is homogeneous (the right hand side is null). Since P is irreducible, A is also irreducible and there exists a LU decomposition such that

$$A = L\,U$$

where the diagonal elements of the lower triangular matrix $L \in \mathbb{R}^{n\times n}$ are equal to one and $U \in \mathbb{R}^{n\times n}$ is an upper triangular matrix with $U_{nn} = 0$ (see [PSS96, Ste97]). From this decomposition, our linear system from Equation (2.21) becomes $L\,U\pi = 0$. Since L is non singular, it leads to $U\pi = 0$. Since the solution of this singular linear system is defined up to a constant, we can fix for instance $\pi_n = 1$. Then the LU scheme for solving Equation (2.21), proposed by [PSS96, Ste97], follows.

1. Factorize the matrix of the system : $A = L\,U$,
2. set $U_{nn} = 1$,
3. solve the upper triangular system : $U\,\pi = b$ with $b_i = 0$, $i < n$ and $b_n = 1$,
4. normalize the solution : $\pi = \frac{\pi}{\|\pi\|_1}$.

2.4.2 Iterative Solver

Finding the stationary probability of a Markov Chain is a special case of consistent singular linear system. The first iterative methods for solving singular systems were proposed and studied in [MP77, NP79, Ple76]. Let first give some useful definitions and properties, known from the literature and used in the aforementioned papers.

Definition 2.7 (Semi-convergent matrix)**.** *A matrix A is said* semi-convergent *if the limit*

$$\lim_{k\to\infty} A^k$$

exists.

For a matrix A, we define

$$\gamma(A) := \max\{|\lambda| \mid \lambda \in \sigma(A), \lambda \neq 1\} \tag{2.22}$$

and we denote by $\sigma(A)$ its spectrum. Then, we have the following lemma of [Ple76],

Lemma 2.8 ([Ple76])**.** *The matrix $A \in \mathbb{R}^{n\times n}$ is semi-convergent if and only if*

- *$\gamma(A) < 1$ and*
- *If $\lambda = 1$ is an eigenvalue of A, then $\lambda = 1$ is semi-simple, that is* $\operatorname{rank}(I - A) = \operatorname{rank}((I - A)^2)$.

Another useful known definition for M-matrices is the following

Definition 2.9 (Matrix with "property c")**.** *A M-matrix A is a* matrix with "property c" *if there exists a decomposition of A*

$$A = sI - B$$

such that $s > 0$, $B \in \mathbb{R}_+^{n\times n}$ is a non-negative matrix and $T = \frac{1}{s}B$ is semi-convergent.

In our case, we define $A = I - P^T$ with P a stochastic matrix and from [BP94], we have that A is a M-matrix with "property c". Now, we introduce some basic iterative solvers.

Fixed point iteration

A straightforward iterative method to solve Equation (2.20) is a successive approximations approach. Starting with an initial approximation $\pi^{(0)} \in \mathbb{R}_+^n$ such that $\|\pi^{(0)}\|_1 = 1$, it consists in successively applying a fixed point iteration followed by a normalization step :

$$\begin{aligned} \pi^{(k+1)} &= P^T \pi^{(k)} \\ \pi^{(k+1)} &= \pi^{(k+1)} / \|\pi^{(k+1)}\|_1 \end{aligned}$$

until the desired convergence is obtained, i.e. $\|\pi^{(k+1)} - P^T \pi^{(k+1)}\| < \epsilon$, for some positive ϵ. This method converges since P is semi-convergent.

Smoothing process

As we have seen in Chapter 2.3, a key ingredient of the multigrid method is the splitting or relaxation scheme. Similarly to Section 2.2, we split the matrix $A = M - N$ with M being a non singular matrix. In this case, each smoothing step is followed by a normalization step. Starting from an initial approximation $\pi^{(0)} \in \mathbb{R}_+^n$ such that $\|\pi^{(0)}\|_1 = 1$, the splitting method consists in applying successively the two steps :

$$\pi^{(k+1)} = S\,\pi^{(k)} \quad \text{with} \quad S = M^{-1}\,N$$

$$\pi^{(k+1)} = \pi^{(k+1)} / \|\pi^{(k+1)}\|_1$$

until the desired convergence is obtained, i.e. $\|\pi^{(k+1)} - P^T\,\pi^{(k+1)}\| < \epsilon$, for some positive ϵ. Note that the fixed point iteration is a particular case with $M = I$ and $N = P^T$. A splitting method converges when the smoother operator S is semi-convergent. Moreover, a splitting is called *weak regular* if $S \geq 0$ and *regular* if $M^{-1} \geq 0$ and $N \geq 0$ (see [Var62]).

From Lemma 1 and Theorem 1 of [NP79], we have the following theorem of [BP94] :

Theorem 2.10 ([BP94])**.** *Let $A \in \mathbb{C}^{n\times n}$. Then A is a M-matrix with "property c" if and only if every regular splitting $A = M - N$ with $S = M^{-1}\,N$ satisfies $\rho(S) \leq 1$ and if $\lambda = 1$ is an eigenvalue of S, it is semi-simple.*

See also [PSS96, Ste97] for an overview of other iterative methods for finding the stationary probability of an irreducible Markov Chain. In the following section, we consider iterative methods whose concepts are closely related to multigrid methods.

2.4.3 Iterative aggregation/disaggregation for Markov Chains

We explain here a general Iterative aggregation/disaggregation (IAD) scheme to find the stationary probability of an irreducible Markov Chain as it is given in [MM03]. See also [DS00] and references therein, for a broad overview of IAD partitioning techniques, that are methods working on two levels and that are suitable for Nearly Completely Decomposable (NCD) Markov Chains. We will see in the next section, that those methods are closely related to aggregation multigrid methods.

We are looking for the stationary probability of an irreducible Markov Chain that is the solution of the linear System (2.20). Consider that a partition of the nodes in n_L aggregates $\{G_k\}_{k=1}^{n_L}$ is given. Then, we define the transfer operators: the restriction operator $\mathcal{R} : \mathbb{R}^n \to \mathbb{R}^{n_L}$

$$(\mathcal{R}\,u)_i = \sum_{j\in G_i} u_j \qquad u \in \mathbb{R}^n$$

and the interpolation operator $\mathcal{P}(w) : \mathbb{R}^{n_L} \to \mathbb{R}^n$, for $w \in (\mathbb{R}_+^*)^n$

$$(\mathcal{P}(w)\,u)_i = \frac{w_i}{(\mathcal{R}\,w)_j} u_j \qquad i \in G_j,\ j \in \{1,\cdots,n_L\},\ u \in \mathbb{R}^n,$$

that is :

$$\mathcal{P}(w) = \mathrm{diag}(w)\,\mathcal{R}^T\,\mathrm{diag}(\mathcal{R}\,w)^{-1}.$$

It follows that :

$$\mathcal{R}\,\mathcal{P}(w) \;=\; I^{|L} \tag{2.23}$$

where $I^{|L}$ is the identity operator that maps $\mathbb{R}^{n_L}$ to itself, and

$$(\mathcal{P}(w))^T\,\mathbf{1} \;=\; \mathbf{1}^{|L} \qquad \mathcal{R}^T\,\mathbf{1}^{|L} \;=\; \mathbf{1}$$

where $\mathbf{1}^{|l} = [1 \dots 1]^T$ is a vector of $\mathbb{R}^{n_l}$. Hence, the aggregate of the transition matrix P of the Markov Chain, given by :

$$(P^T)^{|L} \;=\; \mathcal{R}\,P^T\,\mathcal{P}(w) \;\in \mathbb{R}_+^{n_L \times n_L}$$

is also a stochastic matrix. We define $\mathcal{I}(w) = \mathcal{P}(w)\mathcal{R}$ the aggregation projection, we have :

$$(\mathcal{I}(w))^2 \;=\; \mathcal{I}(w).$$

Consider a splitting scheme for $A = I - P^T = M - N$ such that M is non singular and $S = M^{-1}\,N$ is non-negative, then the IAD scheme of [MM03] is given in Algorithm 2.12 applied to P, S and initial condition $u^{(0)} \in (\mathbb{R}_+^*)^n$. A global convergence result is available in [MM98] when $S = P$ and a local convergence result is available in [MM03] for the general case.

Algorithm 2.12 IAD for Markov chains : $u \leftarrow SPV(P; S; \nu_2; u)$

Step 1. Construct $\mathcal{P}(u)$ and $(P^T)^{|L} \;=\; \mathcal{R}\,P^T\,\mathcal{P}(u)$

Step 2. Find the unique solution of

$$(u^T)^{|L}\,P^{|L} \;=\; (u^T)^{|L}, \qquad \|u^{|L}\|_1 = 1, \qquad u^{|L} > 0, \in \mathbb{R}^{n_L}$$

Step 3. Disaggregate : $u \leftarrow \mathcal{P}(u)\,u^{|L}$

Step 4. Relax : $u \leftarrow Su$ ν_2 times

Step 5. Normalize : $u \leftarrow u/\|u\|_1$

Step 6. If $\|u - P^T u\| < \epsilon$ STOP and return u
Else return $u \leftarrow SPV(P; S; \nu_2; u)$.

2.4.4 Multigrid for Markov Chains

In the recent years, multigrids methods have also been introduced [HL94, DSMM$^+$08, DSMM$^+$10a, DSMM$^+$10b, DSMMS11, DSMSW10, BBB$^+$10, TY10, TY11, Vir07] for the computation of the stationary probability of irreducible Markov Chains. We focus here on [DSMM$^+$08] which is an adaptive version of [HL94]. This method is closely related to IAD schemes, explained in the previous section, except that it is given in multigrid scheme and the coarse grids are computed at each iteration. The methods in [DSMM$^+$10a,

DSMM$^+$10b, DSMMS11, DSMSW10] are variants or accelerations of [DSMM$^+$08]. Consider the linear System (2.20). As in standard AMG, the grid on a level l is represented by a set of variables: $\Omega^{|l} = [n_l]$.

The multigrid method of [DSMM$^+$08] is somewhat different from the multigrid methods of Section 2.3. Indeed, it is based on an adaptive multiplicative coarse-level correction. It is adaptive in the sense that the grids and the transfer operators are constructed at each level of a multigrid cycle. Hence, there is no setup phase. The multigrid cycles are of multiplicative type by considering the multiplicative error equation which is defined for level l by :

$$A^{|l}\ \mathrm{diag}(u^{|l})\, e^{|l}\ =\ 0 \tag{2.24}$$

where $u^{|l}$ is the current approximation of π^T and $e^{|l}$ is the corresponding multiplicative error. When the approximation converges to the solution, the multiplicative error converges to the vector $\mathbf{1}^{|l} = [1, \ldots, 1]^T \in \mathbb{R}^{n_l}$. We shall use here the same notations as in the previous section. Assuming that a partition of level l in n_{l+1} aggregates: $\{G_k^h\}_{k=1}^{n_{l+1}}$ is given, we define the piecewise constant tentative prolongator $\widetilde{\mathcal{P}}_{l+1}^l$ by Equation (2.17). Define the restriction operator $\mathcal{R}_l^{l+1} = (\widetilde{\mathcal{P}}_{l+1}^l)^T$. Then, a coarse level version of Equation (2.24) is constructed :

$$\mathcal{R}_l^{l+1}\, A^{|l}\ \mathrm{diag}(u^{|l})\widetilde{\mathcal{P}}_{l+1}^l\, e^{|l+1}\ =\ 0, \tag{2.25}$$

where $e^{|l+1}$ is the coarse approximation of $e^{|l}$ on level $l+1$. Setting $u^{|l+1} = \mathrm{diag}(\mathcal{R}_l^{l+1}u^{|l})e^{|l+1}$, then instead of solving Equation (2.25) one can solve :

$$\mathcal{R}_l^{l+1}\, A^{|l}\, \mathcal{P}_{l+1}^l(u^{|l})\, u^{|l+1}\ =\ 0$$

where $\mathcal{P}_{l+1}^l(u^{|l}) := \mathrm{diag}(u^{|l})\widetilde{\mathcal{P}}_{l+1}^l\, \mathrm{diag}(\mathcal{R}_l^{l+1}u^{|l})^{-1}$. We are looking for an improved coarse level approximation $u^{|l+1}$ of $u^{|l}$. Now, we define $A^{|l+1} := \mathcal{R}_l^{l+1}A^{|l}\mathcal{P}_{l+1}^l(u^{|l})$. Such as in Equation (2.23), we have :

$$\mathcal{R}_l^{l+1}\, \mathcal{P}_{l+1}^l(u^{|l})\ =\ I^{|l+1}$$

and

$$\begin{aligned} A^{|l+1}\ &=\ \mathcal{R}_l^{l+1}(I^{|l}\ -\ (P^T)^{|l})\mathcal{P}_{l+1}^l(u^{|l}), \\ &=\ I^{|l+1}\ -\ (P^T)^{|l+1} \end{aligned}$$

where $P^{|l+1}$ is a row stochastic matrix of $\mathbb{R}_+^{n_{l+1}\times n_{l+1}}$. The multigrid cycle MAA of [DSMM$^+$08] is then given in Algorithm 2.13. Note that the relaxation step is followed by a normalization.

The construction of the coarse levels in [DSMM$^+$08] is an aggregation based algorithm. It consists, on level l, in first constructing for all nodes, the strongly dependence sets $\left\{S_i^l\right\}_{i\in[n]}$ as defined in Equation (2.12) in which we replace the matrix $A^{|l}$ of the system by $\bar{A}^{|l} = A^{|l}\ \mathrm{diag}(u^{|l})$ where $u^{|l}$ is the current approximation of π^T and $\theta = 0.8$. Then the setup phase of [DSMM$^+$08] is given in Algorithm 2.14. It consists in selecting a seed node that corresponds to the largest value of the current approximation of π^T among

Algorithm 2.13 Aggregation multigrid for Markov chains[DSMM$^+$08] : $u^{|l} \leftarrow MAA(u^{|l})$

if $l < L-1$ **then**

pre relaxation :

$u^{|l} \leftarrow S^{|l} u^{|l}$ (on $\Omega^{|l}$) $\quad \nu_1$ times

$u^{|l} \leftarrow u^{|l} / \|u^{|l}\|_1$

coarse grid correction :

$\{G_k^l\}_{k=1}^{n_{l+1}} \leftarrow AGGRMAA(\Omega^{|l}, u^{|l})$

Build the tentative prolongator $\widetilde{\mathcal{P}}_{l+1}^l$ using $\{G_k^l\}_{k=1}^{n_{l+1}}$

$\mathcal{R}_l^{l+1} = (\widetilde{\mathcal{P}}_{l+1}^l)^T$ and $\mathcal{P}_{l+1}^l(u^{|l}) = \mathrm{diag}(u^{|l}) \, \widetilde{\mathcal{P}}_{l+1}^l \, \mathrm{diag}(\mathcal{R}_l^{l+1} \, u^{|l})^{-1}$

$A^{|l+1} = \mathcal{R}_l^{l+1} \, A^{|l} \, \mathcal{P}_{l+1}^l(u^{|l})$

$u^{|l+1} \leftarrow \mathcal{R}_l^{l+1} \, u^{|l}$

$u^{|l+1} \leftarrow MAA(u^{|l+1})$ $\quad \varsigma$ times.

$u^{|l} \leftarrow \mathcal{P}_{l+1}^l(u^{|l}) \, u^{|l+1}$

post relaxation :

$u^{|l} \leftarrow S^{|l} u^{|l}$ (on $\Omega^{|l}$) $\quad \nu_2$ times

$u^{|l} \leftarrow u^{|l} / \|u^{|l}\|_1$

else

Solve $A^{|L-1} u^{|L-1} = 0$ with $\|u^{|L-1}\|_1 = 1$

end if

all unassigned nodes. Then this seed node is used to form a new aggregate with all the unassigned nodes that are strongly influenced by it. This scheme is repeated until all nodes are assigned.

The multigrid method for Markov Chain of [DSMM$^+$08] differs from the one of [HL94] in the adaptive behavior and in the aggregation strategy. In [HL94], the grids are constructed once in a setup phase based on a initial approximation of π. A local convergence result for a two level aggregation method is given in [MM98, MM03].

2.4.5 Algebraic Multigrid Preconditioner for Markov Chains

We describe here the method of [Vir07]. In [Vir07], the algebraic multigrid method is used as a preconditioner of GMRES ([Saa03]), however in our code, we use it as a solver. This method is related to classical algebraic multigrid as explained in Section 2.3.2.

Consider the problem of finding the stationary probability of an irreducible Markov chain that is the solution of the linear System (2.21). Since the matrix $(I - P^T)$ has column sums equal to zero, the construction of the multigrid operators are based on its transpose. Hence, in the following, we consider $A = (I - P)$. The fine grid associated to the matrix $A \in \mathbb{R}^{n \times n}$ is denoted by $\Omega^{|l}$ and a partition of it in two distinct subsets: C which contains the coarse grid nodes and F which contains the nodes from the fine grid only, that is $\Omega^{|l} = C \cup F$. The coarse grid $\Omega^{|L} = C$ and contains n_L elements. As in [Stü01], the nodes, $\{1, \ldots, n\}$, of the matrix A of the system can be reorganized such

Algorithm 2.14 Aggregation [DSMM+08] : $\{G_k^l\}_{k=1}^{n_{l+1}} \leftarrow AGGRMAA(\Omega^{|l}, u^{|l})$

for all $i \in \Omega^{|l}$ **do**
  Compute the set S_i^l defined by Equation (2.12) with $\bar{A}^{|l}$
end for
Set $K = \{i \in \Omega^{|l}\}$
Set $n_C = 0$
repeat
  $n_C = n_C + 1$
  Pick up $i \in K$ such that the ith-coordinate of $u^{|l}$ has the largest value in $u^{|l}$.
  $G_{n_C}^l \leftarrow i$ and $K = K \setminus \{i\}$
  for all $k \in K$ that are strongly influenced by i (i.e. $i \in S_k^l$) **do**
    $G_{n_C}^l \leftarrow \{k\}$ and $K = K \setminus \{k\}$
  end for
until $K = \emptyset$
$n_{l+1} \leftarrow n_C$

that after a suitable permutation, the matrix rewrites :

$$A = \begin{pmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{pmatrix}.$$

In classical AMG methods [Stü01, Vir07], the coarse nodes are chosen such that the matrix A_{FF} is closed to a diagonal matrix, i.e. such that there are few connections between the nodes of the subset F. In this scope, a sparsification step is performed moving the off-diagonal elements of the submatrix A_{FF} to the submatrix A_{FC} such that the row sums of the new approximation of A are equal to the those of A. We obtain then the following approximation of A, denoted by $\tilde{A}$:

$$\tilde{A} = \begin{pmatrix} \tilde{A}_{FF} & \tilde{A}_{FC} \\ A_{CF} & A_{CC} \end{pmatrix}.$$

Then, the restriction operator $\mathcal{R} : \mathbb{R}^n \to \mathbb{R}^{n_L}$ and the interpolation operator $\mathcal{P} : \mathbb{R}^{n_L} \to \mathbb{R}^n$ are defined by :

$$\mathcal{R} = \begin{pmatrix} -A_{CF}\,\tilde{A}_{FF}^{-1} & I^{|L} \end{pmatrix}, \qquad \mathcal{P} = \begin{pmatrix} -\tilde{A}_{FF}^{-1}\,\tilde{A}_{FC} \\ I^{|L} \end{pmatrix}$$

and the coarse grid operator is given by [Vir07] :

$$A^{|L} = \mathcal{R}\,\tilde{A}\,\mathcal{P} = A_{CC} - A_{CF}\,\tilde{A}_{FF}^{-1}\,\tilde{A}_{FC} \in \mathbb{R}^{n_L \times n_L}.$$

From Lemma 1 and Theorem 2 of [Vir07], the row sums of $\mathcal{P}$ equal one, those of $A^{|L}$ equal zero and $A^{|L}$ is an M-matrix such that $A^{|L} = I^{|L} - P^{|L}$ where $P^{|L} \in \mathbb{R}^{n_L \times n_L}$ is a stochastic matrix.

Note that in our program, we use also this construction method to solve linear non singular systems involved in the policy iteration algorithm (Algorithm 1.1), in this case the matrix of the linear system is row stochastic. We use the classical AMG scheme of Section 2.3.5 for the selection of the coarse grid, and the interpolation operator (2.15). In this case, weak connected elements of $\tilde{A}_{FF}$ and $\tilde{A}_{FC}$ (in D_i^w defined in Equation (2.13)) are moved to the diagonal of $\tilde{A}_{FF}$ and the off-diagonal strongly connected elements of $\tilde{A}_{FF}$ are distributed in $\tilde{A}_{FC}$.

Chapter 3

AMGπ for discounted games

In this chapter, we present our algorithm AMGπ for solving zero-sum two player stochastic games with discounted payoff. In Section 3.1, we describe our algorithms, a shorter version has been presented in the paper [AD12]. In Section 3.2, we give numerical results on discretizations of Isaacs equations that were presented in the paper [AD12].

3.1 A multigrid algorithm for discrete dynamic programming equations arising in a discounted or stopping game

3.1.1 Policy iteration combined with algebraic multigrid method (AMGπ)

Recall that in the policy iteration algorithm for games at each step k of the interior policy iteration, we have to solve a linear System (1.8) which is of the form $v = \mu M v + r$ with M a Markov matrix and $0 < \mu < 1$ the discount factor. Since $(I - \mu M)$ are non singular M-matrices, we use AMG to solve those systems. For shortness in the sequel, we shall call the resulting algorithm AMGπ that is the combination of policy iterations and AMG. The name AMGπ refers also to the numerical implementation of this algorithm. Note that in practice, in Algorithm 1.1 (equivalently in Algorithm 1.2), the policy iterations are stopped when after Step 1, the norm of the residual, $r_v = F(v) - v$, is smaller than a given value denoted by ϵ. We used this stopping criterion in AMGπ. The algorithm AMGπ for the solution of the dynamic programming equation of a zero-sum two player stochastic game with discounted payoff is given in Algorithm 3.2. This algorithm also applies to zero-sum two player stochastic games with total payoff when the game stops in finite time almost surely. The iterations of AMGπ are summarized in the scheme represented in Figure 3.1 where $(v^{(s,k,0)}, \cdots, v^{(s,k,m)}, \cdots, v^{(s,k+1,0)})$ is a sequence of value functions generated by the multigrid solver. The algebraic multigrid methods allows us to solve linear systems arising from either the discretization of Isaacs or Hamilton-Jacobi-Bellman equations or a true finite state space zero-sum two player game.

Algorithm 3.1 $(\beta, v) \leftarrow$ AMGπone $(\beta^{(0)}, v^{(0)}, \epsilon)$

1. Compute an approximation $v^{(k+1)}$ of the value v of the game with fixed feedback policy $\beta^{(k)}$ that is solution of
$$v = F^{\beta^{(k)}}(v) \tag{3.1}$$
using an AMG solver that is
$$v^{(k+1)} \leftarrow MG(v^{(k)}, r^{\beta^{(k)}}) \ .$$
2. If $\|F(v^{(k+1)}) - v^{(k+1)}\| < \epsilon$ STOP and return $\beta^{(k)}, v^{(k+1)}$.
3. Find the optimal feedback policy $\beta^{(k+1)}$ for the value $v^{(k+1)}$, i.e. for each x in $\mathcal{X}$, choose $\beta^{(k+1)}(x)$ such that :
$$\beta^{(k+1)}(x) \ \in \ \underset{b \in \mathcal{B}(x)}{\text{Argmin}} \ F(v^{(k+1)}; x, b) \ .$$
4. Increment k by one and go to Step 1.

Algorithm 3.2 $(\alpha, \beta, v) \leftarrow$ AMGπ$(\alpha^{(0)}, \beta^{(0)}, v^{(0)}, \epsilon)$

1. Compute an approximation $v^{(s+1)}$ of the value v of the game with fixed feedback policy $\alpha^{(s)}$ that is solution of
$$v = F^{\alpha^{(s)}}(v) \tag{3.2}$$
using AMGπ for one player, that is
$$(\beta^{(s+1)}, v^{(s+1)}) \leftarrow AMG\pi one(\beta^{(k)}, v^{(s)}, \epsilon) \ .$$
2. If $\|F(v^{(s+1)}) - v^{(s+1)}\| < \epsilon$ STOP and return $\alpha^{(s)}, \beta^{(s+1)}, v^{(s+1)}$.
3. Find the optimal feedback policy $\alpha^{(s+1)}$ of MAX for the value $v^{(s+1)}$, i.e. for each x in $\mathcal{X}$, choose $\alpha^{(s+1)}(x)$ such that :
$$\alpha^{(s+1)}(x) \ \in \ \underset{a \in \mathcal{A}(x)}{\text{Argmax}} \ F(v^{(s+1)}; x, a) \ .$$
4. Increment s by one and go to Step 1.

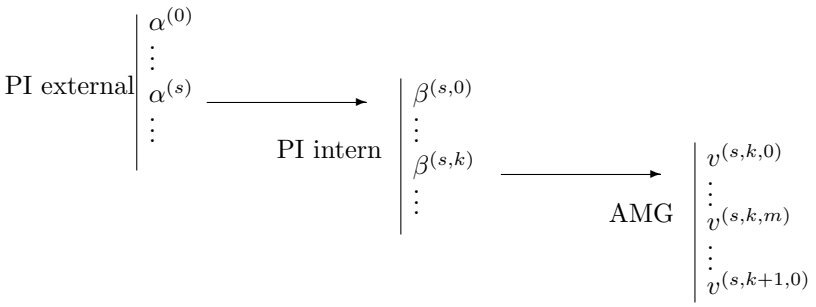

Figure 3.1: Representation of the nested iterations of AMGπ.

In the one player game case, convergence results of combination of policy iteration and geometric multigrid method have been established by Hoppe [Hop86, Hop87] and Akian [Aki90a, Aki90b]. A two-level partitioning techniques combined with policy iteration was also used by [BC89] in the context of solving Markov Decision process, i.e. the one player stochastic game, this method is more related to IAD schemes (see Section 2.4.3).

3.1.2 Full multi-level policy iteration (FAMGπ)

Recall that the number of policy iterations can be exponential in the cardinality of the state space $\mathcal{X}$. However, as for Newton's algorithm, convergence can be improved by starting the policy iterations with a good initial guess, close to the solution. With this in mind, we present a full multi-level scheme, that we shall call FAMGπ. As in standard FMG, starting from the coarsest level, it consists in solving the problem at each grid level by performing policy iterations AMGπ until a convergence criterion is verified, then to interpolate the strategies and value function to the next level, in order to initialize the policy iterations of that level. This scheme is repeated until the finest level is attained.

The algorithm FAMGπ only applies to Isaacs partial differential equations (1.13). It works as follows. The state space $\mathcal{X}$ is first discretized on sequence of L_F+1 grids : $\mathcal{X}^{|L_F} \subset \cdots \subset \mathcal{X}^{|1} \subset \mathcal{X}^{|0} = \mathcal{X}_h$ such that on grid $\mathcal{X}^{|l}$, $0 \leq l \leq L_F$, the discretization step is $h_l = 2^l h$, where h is the discretization step chosen on the finest grid $\mathcal{X}_h$. Then, the Isaacs PDE is discretized on all levels, $0 \leq l \leq L_F$, using the finite differences scheme (1.17)- (1.18). For level l, we denote by $F^{|l} : \mathcal{X}^{|l} \to \mathcal{X}^{|l}$ the dynamic programming operator, $v^{|l} : \mathcal{X}^{|l} \to \mathbb{R}$ the value of game, $x \in \mathcal{X}^{|l} \to \alpha^{|l}(x) \in \mathcal{A}(x)$ and $(x \in \mathcal{X}^{|l}, a \in \mathcal{A}(x)) \to \beta^{|l}(x,a) \in \mathcal{B}(x,a)$ the strategies of MAX and MIN respectively. We denote by $\mathcal{I}_l^{l-1}$ the linear interpolation operator which maps any vector $v^{|l}$ from $\mathbb{R}^{\mathcal{X}^{|l}}$ to $\mathbb{R}^{\mathcal{X}^{|l-1}}$:

$$\mathcal{I}_l^{l-1} v^{|l}(x) = \begin{cases} v^{|l}(x) & x \in \mathcal{X}^{|l} \\ \sum_{y \in \mathcal{N}(x)} \frac{1}{\sharp(\mathcal{N}(x))} v^{|l}(y) & x \in \mathcal{X}^{|l-1} \setminus \mathcal{X}^{|l} \end{cases}$$

where $\mathcal{N}(x) = \left\{ y \in \mathcal{X}^{|l} \mid \|x-y\|_2 <= h_l \right\}$ for $x \in \mathcal{X}^{|l-1} \setminus \mathcal{X}^{|l}$, and we denote by $\mathcal{U}_l^{l-1}$ the operator which interpolates a strategy from grid $\mathcal{X}^{|l}$ to grid $\mathcal{X}^{|l-1}$, for instance for a

strategy of MAX :

$$\mathcal{U}_l^{l-1}(\alpha^{|l}) = \begin{cases} \alpha^{|l}(x) & x \in \mathcal{X}^{|l} \\ a_0 \in \mathcal{A}(x) & x \in \mathcal{X}^{|l-1} \setminus \mathcal{X}^{|l} \end{cases}$$

where a_0 is chosen arbitrary $A(x)$ in for $x \in \mathcal{X}^{|l-1} \setminus \mathcal{X}^{|l}$. We denote by AMGπ$(\alpha, \beta, v, \epsilon)$ the algorithm AMGπ with initial strategy α for player MAX iterations, initial policy β for the first iteration of player MIN, value v as initial approximation for the first call of AMG and ϵ the stopping criterion for the policy iterations. Then FAMGπ algorithm is given in Algorithm 3.3 where $c > 0$ is a given constant.

Algorithm 3.3 FAMGπ

Given an initial $\alpha^{(0)|L_F}, \beta^{(0)|L_F}$ and $v^{(0)|L_f}$ on level L_F,
for $l = L_F$ to 1 **do**
$(\alpha^{|l}, \beta^{|l}, v^{|l}) \leftarrow$ AMGπ$(\alpha^{(0)|l}, \beta^{(0)|l}, v^{(0)|l}, ch_l^2)$ on level l
$v^{(0)|l-1} = \mathcal{I}_l^{l-1} v^{|l}$
$\alpha^{(0)|l-1} = \mathcal{U}_l^{l-1}\alpha^{|l}$ and $\beta^{(0)|l-1} = \mathcal{U}_l^{l-1}\beta^{|l}$
end for
solve $v = F(v)$ on $\mathcal{X}_h$ by using AMGπ$(\alpha^{(0)|0}, \beta^{(0)|0}, v^{(0)|0}, \epsilon)$

Figure 3.2 illustrates the FAMGπ algorithm when V-cycles are use in AMGπ. The dashed lines represent the interpolation of the solution and strategies from a coarse grid $\mathcal{X}^{|l}$ to the next fine grid $\mathcal{X}^{|l-1}$. The continuous V-lines are the V-cycles of AMGπ which are not fixed in number since at each level, AMGπ cycles are performed until a given criterion is attained.

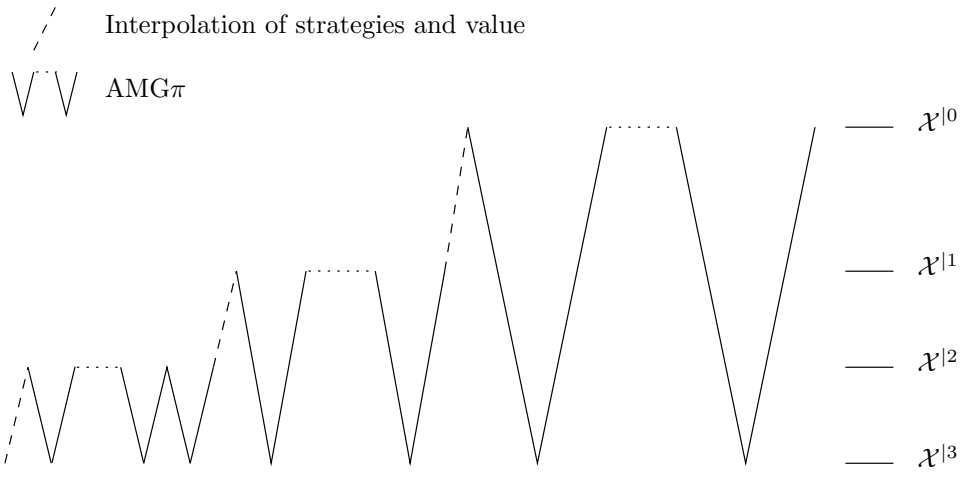

Figure 3.2: FAMGπ with AMGπ V-cycles

Note that our FAMGπ program only applies to stochastic differential games since for them coarse representation, including equations and strategies, can be easily constructed by taking different sizes of discretization step.

For one-player discounted games with infinite number of actions and under regularity

and strong convexity assumptions, it is shown in [Aki90b, Aki90a] that this kind of full multi-level policy iteration has a computing time in the order of the cardinality of $\mathcal{X}$.

3.2 Numerical results for discounted stochastic games

In this section, we apply our programs AMGπ and FAMGπ, which were implemented in C, to examples of two player zero-sum stochastic differential games. Let us first give some details about the implementation of the algorithms that we use and some notations for the numerical results.

The AMG linear solver of AMGπ implements the construction phase, including the coarsening scheme and the interpolation operator, described in [RS87] and the general recursive multigrid cycle for the solution phase (see Algorithm 2.2). In the tests, W(1,1)-cycles were used and the chosen smoother is a CF relaxation method, that is a Gauss Seidel relaxation scheme that relaxes first on C-points and then on F-points. The AMGπ program is the implementation of the method explained in section 3.1 with the above AMG linear solver. The FAMGπ program is the implementation of Algorithm 3.3.

The following notations are used in the tables: s denotes the iteration over MAX policies and $kmax$ is the corresponding number of iterations for MIN policies, that is the number of linear systems solved at iteration s. The residual error of the game is denoted by $r_v = F(v) - v$ and the exact error, when known, by $e = F(v) - u$ where u is the discretized exact solution of the game. The sup-norm $\|\cdot\|_\infty$ and discrete L_2 norm $\|\cdot\|_{L_2}$ are given for each of them.

3.2.1 Isaacs equations

The first example concerns a diffusion problem where the value $v : \overline{\mathcal{X}} \to \mathbb{R}$ of the game is solution of the following Isaacs PDE :

$$\begin{cases} \max\limits_{a\in\mathcal{A}} \min\limits_{b\in\mathcal{B}} \left(\Delta v(x) + (a \cdot \nabla v(x)) - (b \cdot \nabla v(x)) - \lambda v(x) + \dfrac{\|b\|_2^2}{2} + f(x) \right) = 0 & x \text{ in } \mathcal{X} \ , \\ v(x) = \psi_1(x) & x \text{ in } \partial\mathcal{X} \end{cases} \tag{3.3}$$

where $\mathcal{X} =]0,1[\times]0,1[$ is the unit square, $\mathcal{A} = \left\{ a \in \mathbb{R}^2 \mid \|a\|_2 \leq 1 \right\}$, $\mathcal{B} = \mathbb{R}^2$, $\psi_1(x_1, x_2) = \sin(x_1) \times \sin(x_2)$ for $(x_1, x_2) \in \partial\mathcal{X}$, and $f(x) = -(\Delta u(x) + \|\nabla u(x)\|_2 - 0.5\|\nabla u(x)\|_2^2 - \lambda u(x))$ with $u(x_1, x_2) = \sin(x_1) \times \sin(x_2)$ for $x = (x_1, x_2) \in \mathcal{X}$. Note that the exact solution is $v(x_1, x_2) = \sin(x_1) \times \sin(x_2)$ on $\mathcal{X} = [0,1] \times [0,1]$ and is represented in Figure 3.3. Indeed, by convex duality (or computation of Fenchel-Legendre transformations [Roc70]), we have that

$$\|u\|_2 = \max_{\|a\|_2 \leq 1, a \in \mathbb{R}^d} a \cdot u \qquad \text{and} \qquad \frac{1}{2}\|u\|_2^2 = \max_{b \in \mathbb{R}^d} b \cdot u - \frac{1}{2}\|b\|_2^2$$

for all $u \in \mathbb{R}^d$, $a = \frac{u}{\|u\|_2}$ and $b = u$ are optimal solutions in these equations.

To solve Equation (3.3), we first discretize the domain $[0,1] \times [0,1]$ on a grid with $m+1$ points in each direction, i.e. with a discretization step $h = \frac{1}{m}$ and we obtain a

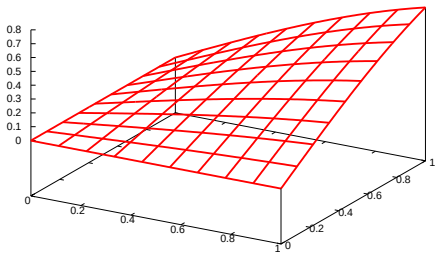

Figure 3.3: Graph of $\sin(x_1) \times \sin(x_2)$ on $\mathcal{X} = [0,1] \times [0,1]$.

discrete space $\mathcal{X}_h$ with boundary $\partial\mathcal{X}_h$. We denote by $x_i = ih$ with $i = 0, \ldots, m$ such that $\mathcal{X}_h = \{(x_i, x_j) \mid i, j \in \{1, \ldots, m-1\}\}$ and
$\partial\mathcal{X}_h = \{(x_i, x_j) \mid i \in \{0, m\}, j \in \{0, \ldots, m\} \text{ or } j \in \{0, m\}, i \in \{0, \ldots, m\}\}$. Then, using the discretization scheme (1.17)–(1.18), Equation (3.3) becomes for $(x_i, x_j) \in \mathcal{X}_h$:

$$
\begin{aligned}
0 = \max_{(a_1,a_2)\in\mathcal{A}} \min_{(b_1,b_2)\in\mathcal{B}} \Bigg\{ & \left(\frac{-4v(x_i,x_j) + v(x_{i+1},x_j) + v(x_{i-1},x_j) + v(x_i,x_{j+1}) + v(x_i,x_{j-1})}{h^2} \right) \\
& + (a_1 - b_1) \left(\frac{v(x_{i+1},x_j) - v(x_i,x_j)}{h} \right) \mathbb{I}_{(a_1-b_1)\geq 0} + (a_1 - b_1) \left(\frac{v(x_i,x_j) - v(x_{i-1},x_j)}{h} \right) \mathbb{I}_{(a_1-b_1)<0} \\
& + (a_2 - b_2) \left(\frac{v(x_i,x_{j+1}) - v(x_i,x_j)}{h} \right) \mathbb{I}_{(a_2-b_2)\geq 0} + (a_2 - b_2) \left(\frac{v(x_i,x_j) - v(x_i,x_{j-1})}{h} \right) \mathbb{I}_{(a_2-b_2)<0} \\
& - \lambda v(x_i,x_j) + \frac{b_1^2 + b_2^2}{2} + f(x_i,x_j) \Bigg\} \ ,
\end{aligned}
$$

multiply by $\frac{h^2}{c}$, where $c = 4 + h\,|a_1 - b_1| + h\,|a_2 - b_2| > 0$, and adding $v(x_i, x_j)$ on both sides, we obtain :

$$
\begin{aligned}
v(x_i,x_j) = \max_{(a_1,a_2)\in\mathcal{A}} \min_{(b_1,b_2)\in\mathcal{B}} & \left(1 + \frac{h^2}{c}\lambda\right)^{-1} \\
& \Bigg\{ \left(\frac{1}{c} + \frac{h}{c}(a_1 - b_1)\mathbb{I}_{(a_1-b_1)\geq 0} \right) v(x_{i+1},x_j) + \left(\frac{1}{c} - \frac{h}{c}(a_1 - b_1)\mathbb{I}_{(a_1-b_1)<0} \right) v(x_{i-1},x_j) \\
& + \left(\frac{1}{c} + \frac{h}{c}(a_2 - b_2)\mathbb{I}_{(a_2-b_2)\geq 0} \right) v(x_i,x_{j+1}) + \left(\frac{1}{c} - \frac{h}{c}(a_2 - b_2)\mathbb{I}_{(a_2-b_2)<0} \right) v(x_i,x_{j-1}) \\
& + \frac{h^2}{c}\,\frac{b_1^2 + b_2^2}{2} + \frac{h^2}{c}\,f(x_i,x_j) \Bigg\} \qquad \text{for } (x_i,x_j) \in \mathcal{X}_h \ , \qquad (3.4)
\end{aligned}
$$

where $v(x_i, x_j)$ is replaced by $\psi_1(x_i, x_j)$ for $(x_i, x_j) \in \partial\mathcal{X}_h$. This equation has the form of Equation (1.7) with a discount factor μ equal to $(1 + \frac{h^2}{c}\lambda)^{-1} \leq 1$. Transition probabilities

from $(x_i, x_j) \in \mathcal{X}_h$ to $(x_{i'}, x_{j'}) \in \mathcal{X}_h$ are given by :

$$p((x_{i'}, x_{j'})|(x_i, x_j), (a_1, a_2), (b_1, b_2)) = \begin{cases} \frac{1}{c} + \frac{h}{c}(a_1 - b_1)\mathbb{1}_{(a_1-b_1)\geq 0} & \text{if } i' = i+1, j' = j \ , \\ \frac{1}{c} - \frac{h}{c}(a_1 - b_1)\mathbb{1}_{(a_1-b_1)<0} & \text{if } i' = i-1, j' = j \ , \\ \frac{1}{c} + \frac{h}{c}(a_2 - b_2)\mathbb{1}_{(a_2-b_2)\geq 0} & \text{if } i' = i, j' = j+1 \ , \\ \frac{1}{c} - \frac{h}{c}(a_2 - b_2)\mathbb{1}_{(a_2-b_2)<0} & \text{if } i' = i, j' = j-1 \ , \\ 0 & \text{otherwise} \ , \end{cases} \tag{3.5}$$

and the running cost is, for $(x_i, x_j) \in \mathcal{X}_h$:

$$\begin{aligned} r((x_i, x_j), (a_1, a_2), (b_1, b_2)) \ = \ & \left(1 + \frac{h^2}{c}\lambda\right)^{-1} \left\{ \frac{h^2}{c}\left(\frac{b_1^2 + b_2^2}{2} + f(x_i, x_j)\right) \right. \\ & + \left(\frac{1}{c} + \frac{h}{c}(a_1 - b_1)\mathbb{1}_{(a_1-b_1)\geq 0}\right) \psi_1(x_{i+1}, x_j)\mathbb{1}_{(x_{i+1},x_j)\in\partial\mathcal{X}_h} \\ & + \left(\frac{1}{c} - \frac{h}{c}(a_1 - b_1)\mathbb{1}_{(a_1-b_1)<0}\right) \psi_1(x_{i-1}, x_j)\mathbb{1}_{(x_{i-1},x_j)\in\partial\mathcal{X}_h} \\ & + \left(\frac{1}{c} + \frac{h}{c}(a_2 - b_2)\mathbb{1}_{(a_2-b_2)\geq 0}\right) \psi_1(x_i, x_{j+1})\mathbb{1}_{(x_i,x_{j+1})\in\partial\mathcal{X}_h} \\ & \left. + \left(\frac{1}{c} - \frac{h}{c}(a_2 - b_2)\mathbb{1}_{(a_2-b_2)<0}\right) \psi_1(x_i, x_{j-1})\mathbb{1}_{(x_i,x_{j-1})\in\partial\mathcal{X}_h} \right\} \ . \end{aligned}$$

Note that when $i, j \in \{2, \ldots, m-2\}$ the sum of the transition probabilities from (x_i, x_j) to the points of $\mathcal{X}_h$ equals 1, when i or j is in $\{1, m-1\}$ this sum is strictly less than 1. Hence, the matrix $M^{\alpha,\beta}$ in (1.8) is substochastic, and since it is irreducible, it has a spectral radius strictly less than one. So even when $\lambda = 0$ or equivalently $\mu = 1$, the System (1.8) has an unique solution and the dynamic programing equation has also an unique solution. Hence, we shall take $\lambda = 0$ in the numerical tests. Note also that for this example, the matrices $M^{\alpha,\beta}$ in (1.8) are not symmetric but close to be symmetric when h is small, since the non-symmetric part correspond to the order one term in Equation (3.4) and are dominated by order two terms when b is optimal in (3.4).

In Table 3.1, we present numerical results when Equation (3.3) is discretized on a grid with 1025 points in each direction, i.e. with a discretization step of $h = 1/2^{10}$. The stopping criterion for the policy iterations is that the discrete L_2 norm $\|r_v\|_{L_2}$ of the residual is less than $\epsilon = 10^{-10}$. The first table of 3.1 shows the results of the policy iteration algorithm with a direct solver LU (we used the package UMFPACK [Dav04]) and the second table of 3.1 the results of AMGπ. We observe that AMGπ solves the problem faster than the policy iterations with a direct solver. In both tables, we see that only three steps on MAX policies are needed (first column) and a total of six steps on MIN policies (second column) which involves the resolution of six linear systems. The small number of iterations is due to the fact that the solution is regular. In Table 3.2, we show that the computation time

Table 3.1: Numerical results for Equation (3.3) on a 1025×1025 points grid.

Policy iteration with LU

s	$kmax$	$\|r_v\|_\infty$	$\|r_v\|_{L_2}$	$\|e\|_\infty$	$\|e\|_{L_2}$	cpu time (s)
1	3	$8.51e-7$	$5.96e-7$	$4.47e-2$	$2.48e-2$	$1.40e+2$
2	2	$2.44e-8$	$6.16e-9$	$1.84e-4$	$1.05e-4$	$2.31e+2$
3	1	$7.38e-13$	$2.03e-13$	$4.13e-6$	$2.16e-6$	$2.77e+2$

AMGπ

s	$kmax$	$\|r_v\|_\infty$	$\|r_v\|_{L_2}$	$\|e\|_\infty$	$\|e\|_{L_2}$	cpu time (s)
1	3	$8.51e-7$	$5.96e-7$	$4.47e-2$	$2.48e-2$	$2.65e+1$
2	2	$2.44e-8$	$6.16e-9$	$1.84e-4$	$1.05e-4$	$4.59e+1$
3	1	$7.92e-13$	$2.02e-13$	$4.13e-6$	$2.16e-6$	$5.56e+1$

is improved when applying FAMGπ with $c = 0.1$ to the same example. In this case, the problem is solved in approximately $18s$.

In Figure 3.4, we compare the policy iteration algorithm with a direct solver LU (UMFPACK [Dav04]) and AMGπ for solving Equation (3.4), when increasing by one the number of discretization points in each direction from $m = 5$ to $m = 1500$. The stopping criterion for the policy iterations uses $\epsilon = 10^{-10}$. In Figure 3.5, we represent the corresponding number of iterations on MIN policies, i.e the number of linear systems solved for each size of problem; this number is the same for both methods. In Figure 3.4, if we consider only the problems of size greater than 10^4, the relationship between the logarithm of the computation time and the logarithm of the size of the problem is almost linear with slope about 1.03 for AMGπ, however the slope is about 1.37 for the policy iteration algorithm with a LU solver. If we consider the 100 problems of finest discretization, the slope is about 1.04 for AMGπ while it is about 1.85 for the policy iterations with a LU solver. Hence, the computation time by AMGπ seems to grow linearly with respect to the size of the problem, whereas that of the policy iteration algorithm with a LU solver grows only polynomially with respect to the size of the problem with an exponent for large sizes strictly greater than 3/2 (which is what one may expect for a LU solver in dimension 2).

Each Table 3.3 to 3.8 contains numerical results for Equation (3.3) discretized on grids with discretization step $h = \frac{1}{2^6}$, $h = \frac{1}{2^7}$, $h = \frac{1}{2^8}$, $h = \frac{1}{2^9}$, $h = \frac{1}{2^{10}}$ and $h = \frac{1}{2^{11}}$ respectively. For these tests, the stopping criterion for the policy iterations uses $\epsilon = 0.001\, h^2$ where h is the discretization step. The stopping criterion for the linear solver AMG is $\|r\|_{L_2} < 10^{-12}$ where r is the residual for the linear system. For each line of the tables, the third column, named AMG, contains the number of iterations needed by AMG for solving each linear system ($kmax$ systems per line). We can see that the number of iterations of AMG is independent of the size of the problem. Note that the norm of the error ($\|e\|_\infty$ or $\|e\|_{L_2}$) decreases slowly when the grid becomes finer, this is because the exact solution (Figure 3.3) is smooth and a small number of points is sufficient to get a good approximation, also the non-linearity of the problem gives a worse approximation than one might expect in the

Table 3.2: Numerical results for Equation (3.3) on a 1025×1025 points grid, computed by FAMGπ with $c = 10^{-1}$.

s	$kmax$	$\|r_v\|_\infty$	$\|r_v\|_{L_2}$	$\|e\|_\infty$	$\|e\|_{L_2}$	cpu time (s)
points in each direction : 3, h $5.00e-01$						
1	2	$1.42e-01$	$1.42e-01$	$1.07e-01$	$1.07e-01$	$<< 1$
2	1	$2.34e-03$	$2.34e-03$	$2.45e-04$	$2.45e-04$	$<< 1$
points in each direction : 5, h $2.50e-01$						
1	2	$5.53e-03$	$2.84e-03$	$3.00e-03$	$1.75e-03$	$<< 1$
points in each direction : 9, h $1.25e-01$						
1	2	$2.40e-04$	$1.10e-04$	$8.20e-04$	$4.46e-04$	$<< 1$
points in each direction : 17, h $6.25e-02$						
1	2	$3.18e-05$	$7.83e-06$	$3.36e-04$	$1.90e-04$	$1.00e-02$
points in each direction : 33, h $3.12e-02$						
1	1	$5.89e-04$	$7.08e-05$	$5.05e-04$	$1.99e-04$	$1.00e-02$
points in each direction : 65, h $1.56e-02$						
1	1	$1.69e-04$	$1.25e-05$	$1.62e-04$	$4.67e-05$	$4.00e-02$
points in each direction : 129, h $7.81e-03$						
1	1	$4.28e-05$	$2.16e-06$	$4.73e-05$	$1.21e-05$	$1.80e-01$
points in each direction : 257, h $3.91e-03$						
1	1	$1.08e-05$	$3.77e-07$	$1.31e-05$	$6.07e-06$	$7.50e-01$
points in each direction : 513, h $1.95e-03$						
1	1	$2.70e-06$	$6.61e-08$	$7.29e-06$	$3.56e-06$	$3.13e+00$
points in each direction : 1025, h $9.77e-04$						
1	2	$1.23e-10$	$8.13e-13$	$4.16e-06$	$2.17e-06$	$1.85e+01$

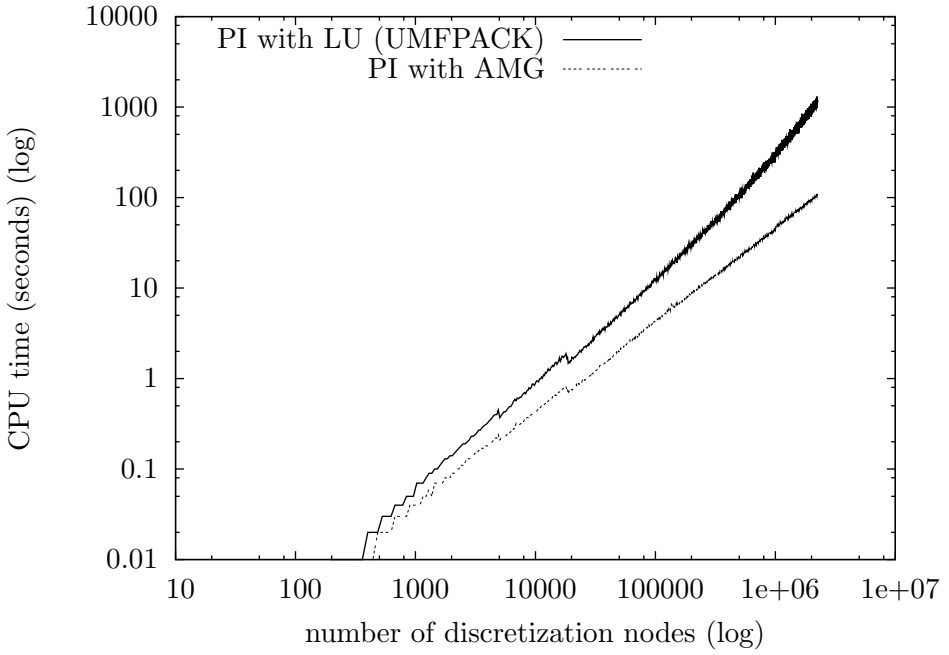

Figure 3.4: Comparison between AMGπ versus policy iteration algorithm with a LU solver for solving Equation (3.4) when increasing the size of the problem.

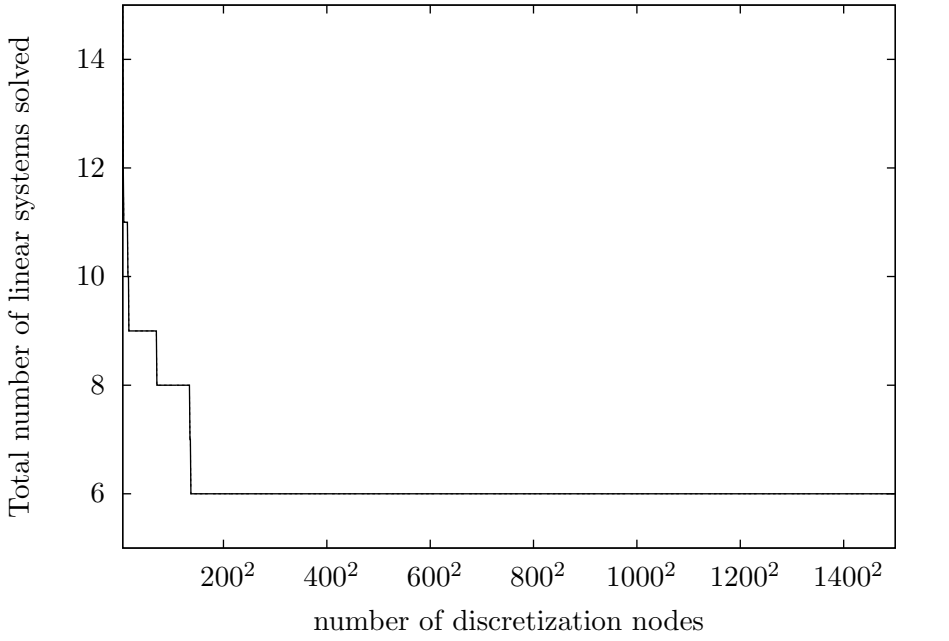

Figure 3.5: Number of iterations on MIN policies (i.e the number of linear systems solved) for solving Equation (3.4) when increasing the size of the problem, corresponding to Figure 3.4 for both methods (AMGπ and policy iteration algorithm with LU).

s	$kmax$	AMG	$\|r_v\|_\infty$	$\|r_v\|_{L_2}$	$\|e\|_\infty$	$\|e\|_{L_2}$	cpu time (s)
1	2	$5,4$	$2.15e-04$	$1.52e-04$	$4.45e-02$	$2.50e-02$	$5.00e-02$
2	2	$4,3$	$5.97e-06$	$1.59e-06$	$2.36e-04$	$1.43e-04$	$1.00e-01$
3	1	3	$3.02e-09$	$7.47e-10$	$6.49e-05$	$3.44e-05$	$1.30e-01$

Table 3.3: Numerical results with a 65×65 points grid, computed by AMGπ for Equation (3.3).

s	$kmax$	AMG	$\|r_v\|_\infty$	$\|r_v\|_{L_2}$	$\|e\|_\infty$	$\|e\|_{L_2}$	cpu time (s)
1	2	$5,4$	$5.40e-05$	$3.80e-05$	$4.46e-02$	$2.49e-02$	$2.30e-01$
2	2	$4,3$	$1.53e-06$	$3.95e-07$	$2.07e-04$	$1.23e-04$	$4.30e-01$
3	1	3	$4.08e-10$	$9.65e-11$	$3.28e-05$	$1.72e-05$	$5.40e-01$

Table 3.4: Numerical results with a 129×129 points grid, computed by AMGπ for Equation (3.3).

s	$kmax$	AMG	$\|r_v\|_\infty$	$\|r_v\|_{L_2}$	$\|e\|_\infty$	$\|e\|_{L_2}$	cpu time (s)
1	2	$5,4$	$1.35e-05$	$9.51e-06$	$4.47e-02$	$2.49e-02$	$1.06e+00$
2	2	$4,3$	$3.86e-07$	$9.86e-08$	$1.94e-04$	$1.13e-04$	$1.98e+00$
3	1	3	$5.17e-11$	$1.22e-11$	$1.65e-05$	$8.63e-06$	$2.49e+00$

Table 3.5: Numerical results with a 257×257 points grid, computed by AMGπ for Equation (3.3).

s	$kmax$	AMG	$\|r_v\|_\infty$	$\|r_v\|_{L_2}$	$\|e\|_\infty$	$\|e\|_{L_2}$	cpu time (s)
1	2	$5,4$	$3.39e-06$	$2.38e-06$	$4.47e-02$	$2.48e-02$	$4.55e+00$
2	2	$4,3$	$9.71e-08$	$2.46e-08$	$1.87e-04$	$1.08e-04$	$8.28e+00$
3	1	3	$6.26e-12$	$1.55e-12$	$8.26e-06$	$4.31e-06$	$1.04e+01$

Table 3.6: Numerical results with a 513×513 points grid, computed by AMGπ for Equation (3.3).

s	$kmax$	AMG	$\|r_v\|_\infty$	$\|r_v\|_{L_2}$	$\|e\|_\infty$	$\|e\|_{L_2}$	cpu time (s)
1	2	$5,4$	$8.48e-07$	$5.95e-07$	$4.47e-02$	$2.48e-02$	$1.85e+01$
2	2	$4,3$	$2.43e-08$	$6.15e-09$	$1.83e-04$	$1.05e-04$	$3.40e+01$
3	1	3	$7.40e-13$	$2.02e-13$	$4.13e-06$	$2.16e-06$	$4.27e+01$

Table 3.7: Numerical results with a 1025×1025 points grid, computed by AMGπ for Equation (3.3).

s	$kmax$	AMG	$\|r_v\|_\infty$	$\|r_v\|_{L_2}$	$\|e\|_\infty$	$\|e\|_{L_2}$	cpu time (s)
1	2	$5,4$	$2.12e-07$	$1.49e-07$	$4.47e-02$	$2.48e-02$	$7.46e+01$
2	2	$4,3$	$6.09e-09$	$1.54e-09$	$1.82e-04$	$1.04e-04$	$1.38e+02$
3	1	3	$1.13e-13$	$3.04e-14$	$2.07e-06$	$1.08e-06$	$1.72e+02$

Table 3.8: Numerical results with a 2049×2049 points grid, computed by AMGπ for Equation (3.3).

linear case. But a smooth solution is generally more difficult for linear iterative solvers.

3.2.2 Optimal stopping game

Next tests concern an optimal stopping time game where the value $v : \overline{\mathcal{X}} \to \mathbb{R}$ of the game is solution of the variational inequality :

$$\begin{cases} \max\left\{ \underbrace{\min_{b\in\mathcal{B}}\left(0.5\Delta v(x) - (b\cdot\nabla v(x)) + \frac{\|b\|_2^2}{2} + f(x)\right)}_{①}, \underbrace{-v(x)}_{②} \right\} = 0 & x \text{ in } \mathcal{X} \\ v(x) = u(x) & x \text{ in } \partial\mathcal{X} \end{cases} \tag{3.6}$$

where $\mathcal{X} =]0,1[\times]0,1[$ is the unit square, $\mathcal{B} = \mathbb{R}^2$, $f : \mathcal{X} \to \mathbb{R}$ satisfies for $x := (x_1, x_2) \in \mathcal{X}$:

$$f(x) = \begin{cases} -(0.5\Delta u(x) - 0.5\|\nabla u(x)\|_2^2) & \text{if } x_2 \geq (x_1 - 0.5)^2 + 0.1 \\ 0.5\Delta u(x) - 0.5\|\nabla u(x)\|_2^2 & \text{otherwise} \ , \end{cases}$$

with $u : \overline{\mathcal{X}} \to \mathbb{R}$ given by:

$$u(x_1, x_2) = \begin{cases} (x_2 - ((x_1 - 0.5)^2 + 0.1))^3 & \text{if } x_2 \geq (x_1 - 0.5)^2 + 0.1 \\ 0 & \text{otherwise} \ . \end{cases} \tag{3.7}$$

Equation (3.6) is of the form (1.14) with $\psi_1 = u$, $\psi_2 \equiv 0$, $q_{ij}(x,b) = 1/2$, $g_j(x,b) = -b$, $\lambda = 0$ and $r(x,b) = 0.5\|b\|_2^2 + f(x)$. The definitions of the functions f and ψ_1 are chosen such that the function u of (3.7), represented in Figure 3.6, is solution of (3.6) almost everywhere and such that for u, the terms ① and ② in Equation (3.6) are non positive for all $x \in \mathcal{X}$ (this condition must hold for the variational inequality to be well-defined). Recall that (3.6) can be rewritten in the form (1.13) with $\mathcal{A} = \{0, 1\}$, where ① corresponds to the action $a = 1$ of MAX (meaning that he continues to play), and ② to the action $a = 0$ of MAX (meaning that he stops the game). This example leads to a free boundary problem for the actions of MAX. Indeed, the points of the state space $\mathcal{X}$ can be divided in two parts, the points where MAX chooses action 1 and the points where MAX chooses action 0. For $x = (x_1, x_2) \in \mathcal{X}$, the optimal strategy α for MAX is $\alpha(x) = 1$ if $x_2 \geq (x_1 - 0.5)^2 + 0.1$ and $\alpha(x) = 0$ otherwise.

As for the previous example, the domain $\mathcal{X}$ is discretized on a grid with $m + 1$ points in each direction, i.e. with a discretization step $h = \frac{1}{m}$ and we obtain a discrete space $\mathcal{X}_h$ with boundary $\partial\mathcal{X}_h$. Then, Equation (3.6) is discretized by using the discretization scheme (1.17)–(1.18). After, the equations ① and ② are simplified separately by keeping equations (1.15) true. In this case, only equation ① is multiplied by $\frac{h^2}{c}$ with c an appropriate constant. After discretization, we obtain the following dynamic programming

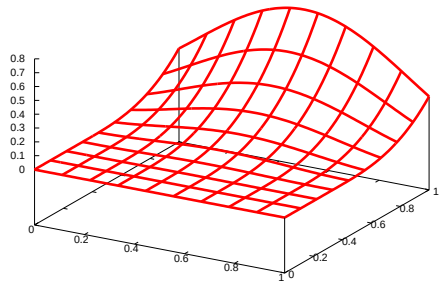

Figure 3.6: Graph of the solution of Equation (3.6).

equation for a game with state space $\mathcal{X}_h$:

$$
\begin{aligned}
v(x_i,x_j) = \quad \max \Bigg\{ & \min_{(b_1,b_2)\in\mathcal{B}} \left(\frac{1}{2c} - b_1\frac{h}{c}\mathbb{I}_{b_1\le 0}\right) v(x_{i+1},x_j) + \left(\frac{1}{2c} + b_1\frac{h}{c}\mathbb{I}_{b_1>0}\right) v(x_{i-1},x_j) \\
& + \left(\frac{1}{2c} - b_2\frac{h}{c}\mathbb{I}_{b_2\le 0}\right) v(x_i,x_{j+1}) + \left(\frac{1}{2c} + b_2\frac{h}{c}\mathbb{I}_{b_2>0}\right) v(x_i,x_{j-1}) \\
& + \frac{h^2}{c}\,\frac{b_1^2+b_2^2}{2} + \frac{h^2}{c}\,f(x_i,x_j), \quad 0 \Bigg\} \qquad \text{for } (x_i,x_j)\in\mathcal{X}_h
\end{aligned}
$$

with $c = 2 + h\,|b_1| + h\,|b_2| > 0$ and $v(x_i,x_j) = \psi_1(x_i,x_j)$ for $(x_i,x_j) \in \partial\mathcal{X}_h$. The same comments about non-symmetry and the discount factor in Equation (3.4) hold here. In particular, $\lambda = 0$ or equivalently $\mu = 1$.

The numerical results are performed for Equation (3.6) when discretized on a grid with 1025 points in each direction. In the domain $\mathcal{X}_h$, for a fixed strategy α of MAX, we represent a point x with a green color when $\alpha(x) = 1$, that is when MAX decides to continue playing, and with a blue color when $\alpha(x) = 0$, that is when MAX decides to stop the game. The optimal strategy for MAX is to have only green points above the red curve, $x_2 = (x_1 - 0.5)^2 + 0.1$, and only blue points under. We start the tests with $\alpha(x) = 0$ for all $x \in \mathcal{X}$, that is with blue points in the whole domain.

Numerical results with AMGπ are shown geometrically in Figure 3.7 where the strategies of MAX obtained after 100, 200, 300, 400, 500, 600 and 700 iterations are represented. We observe in Table 3.9 that AMGπ finds an approximation of the solution after 702 iterations and in about two hours and 15 minutes. The stopping criterion for policy iterations of AMGπ in this test uses $\epsilon = 10^{-14}$. This criterion was chosen to ensure the convergence of the policy iteration, indeed with a smaller ϵ it did not converge because the intern policy iterations did not gave a precise enough approximation.

In table 3.10, we present numerical results for the application of FAMGπ with $c = 10^{-2}$ and $\epsilon = 10^{-14}$ to Equation (3.6) for a 1025×1025 points grid. We observe that our algorithm solves the problem in about 49 seconds. Geometrical representation of the strategies of MAX obtained by AMGπ on four successive levels in the FAMGπ algorithm, are shown in Figure 3.8. We can see that on coarse grids, the algorithm can find a good

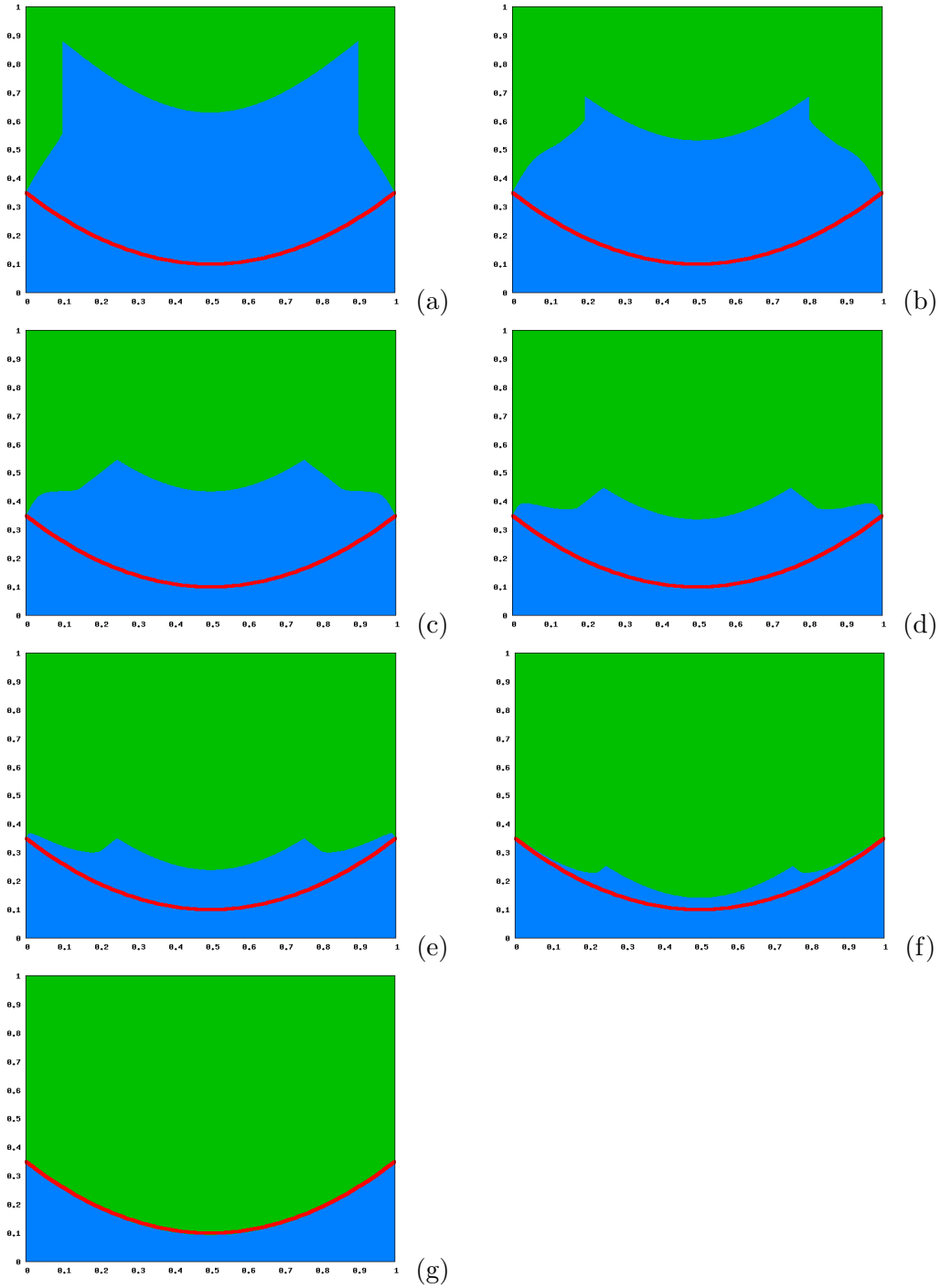

Figure 3.7: Application of AMGπ to the free boundary problem (3.6) for a 1025×1025 points grid : (a) after 100 iterations, (b) after 200 iterations, (c) after 300 iterations, (d) after 400 iterations, (e) after 500 iterations, (f) after 600 iterations and (g) after 700 iterations.

Table 3.9: Numerical results for optimal stopping time game (3.6) with a 1025×1025 points grid, computed by AMGπ with $\epsilon = 10^{-14}$.

s	$kmax$	$\|r_v\|_\infty$	$\|r_v\|_{L_2}$	$\|e\|_\infty$	$\|e\|_{L_2}$	cpu time (s)
1	0	$3.645e-01$	$9.195e-03$	$7.243e-01$	$1.998e-01$	$1.790e+00$
2	4	$1.497e-01$	$1.347e-03$	$3.782e-01$	$1.218e-01$	$1.376e+01$
3	4	$1.094e-01$	$8.839e-04$	$3.767e-01$	$1.213e-01$	$2.492e+01$
				...		
100	3	$1.744e-02$	$4.444e-05$	$2.392e-01$	$8.016e-02$	$1.009e+03$
				...		
200	3	$7.398e-03$	$1.879e-05$	$1.222e-01$	$3.996e-02$	$2.214e+03$
				...		
300	3	$2.510e-03$	$8.779e-06$	$5.614e-02$	$1.728e-02$	$3.619e+03$
				...		
400	2	$1.258e-03$	$4.363e-06$	$2.321e-02$	$6.519e-03$	$4.770e+03$
				...		
500	2	$4.761e-04$	$1.620e-06$	$6.601e-03$	$1.532e-03$	$5.861e+03$
				...		
600	2	$8.857e-05$	$2.781e-07$	$7.274e-04$	$9.598e-05$	$7.045e+03$
				...		
650	2	$1.533e-05$	$4.231e-08$	$1.538e-04$	$6.331e-05$	$7.630e+03$
				...		
700	1	$5.647e-08$	$8.734e-11$	$1.571e-04$	$6.619e-05$	$8.134e+03$
701	1	$1.207e-08$	$2.267e-11$	$1.571e-04$	$6.619e-05$	$8.141e+03$
702	1	$9.992e-16$	$7.284e-17$	$1.571e-04$	$6.619e-05$	$8.148e+03$

Table 3.10: Numerical results for optimal stopping time game (3.6) with a 1025×1025 points grid, computed by FAMGπ with $c = 10^{-2}$ and $\epsilon = 10^{-14}$.

s	$kmax$	$\|r_v\|_\infty$	$\|r_v\|_{L_2}$	$\|e\|_\infty$	$\|e\|_{L_2}$	cpu time (s)
points in each direction : 3, step size : $5.00e-01$						
1	1	$2.17e-01$	$2.17e-01$	$1.53e-01$	$1.53e-01$	$<< 1$
2	2	$2.64e-05$	$2.64e-05$	$3.92e-02$	$3.92e-02$	$<< 1$
points in each direction : 5, step size : $2.50e-01$						
1	2	$2.19e-04$	$8.41e-05$	$3.02e-02$	$1.71e-02$	$<< 1$
points in each direction : 9, step size : $1.25e-01$						
1	2	$4.99e-03$	$1.06e-03$	$1.65e-02$	$7.99e-03$	$<< 1$
2	1	$2.68e-03$	$5.41e-04$	$1.66e-02$	$8.15e-03$	$<< 1$
3	1	$2.72e-04$	$5.49e-05$	$1.68e-02$	$8.30e-03$	$<< 1$
points in each direction : 17, step size : $6.25e-02$						
1	2	$2.26e-03$	$5.44e-04$	$8.75e-03$	$3.89e-03$	$<< 1$
2	1	$7.97e-04$	$1.23e-04$	$8.84e-03$	$3.97e-03$	$<< 1$
3	1	$4.65e-04$	$5.97e-05$	$8.98e-03$	$4.11e-03$	$<< 1$
4	1	$9.57e-08$	$1.24e-08$	$9.01e-03$	$4.14e-03$	$1.00e-02$
points in each direction : 33, step size : $3.12e-02$						
1	1	$2.10e-04$	$1.90e-05$	$4.94e-03$	$2.16e-03$	$1.00e-02$
2	1	$1.05e-04$	$6.57e-06$	$4.76e-03$	$2.09e-03$	$2.00e-02$
points in each direction : 65, step size : $1.56e-02$						
1	1	$6.26e-05$	$6.43e-06$	$2.49e-03$	$1.07e-03$	$4.00e-02$
2	1	$3.64e-05$	$2.09e-06$	$2.45e-03$	$1.05e-03$	$7.00e-02$
points in each direction : 129, step size : $7.81e-03$						
1	1	$7.67e-06$	$3.88e-07$	$1.25e-03$	$5.33e-04$	$1.60e-01$
points in each direction : 257, step size : $3.91e-03$						
1	1	$2.86e-06$	$1.12e-07$	$6.28e-04$	$2.66e-04$	$6.20e-01$
points in each direction : 513, step size : $1.95e-03$						
1	1	$5.33e-07$	$1.44e-08$	$3.15e-04$	$1.33e-04$	$2.49e+00$
points in each direction : 1025, step size : $9.77e-04$						
1	2	$1.79e-07$	$3.82e-09$	$1.57e-04$	$6.62e-05$	$1.58e+01$
2	1	$9.66e-08$	$8.84e-10$	$1.57e-04$	$6.62e-05$	$2.30e+01$
3	1	$5.39e-08$	$4.10e-10$	$1.57e-04$	$6.62e-05$	$3.00e+01$
4	1	$2.86e-08$	$1.31e-10$	$1.57e-04$	$6.62e-05$	$3.70e+01$
5	1	$7.41e-09$	$1.60e-11$	$1.57e-04$	$6.62e-05$	$4.34e+01$
6	1	$8.88e-16$	$7.31e-17$	$1.57e-04$	$6.62e-05$	$4.99e+01$

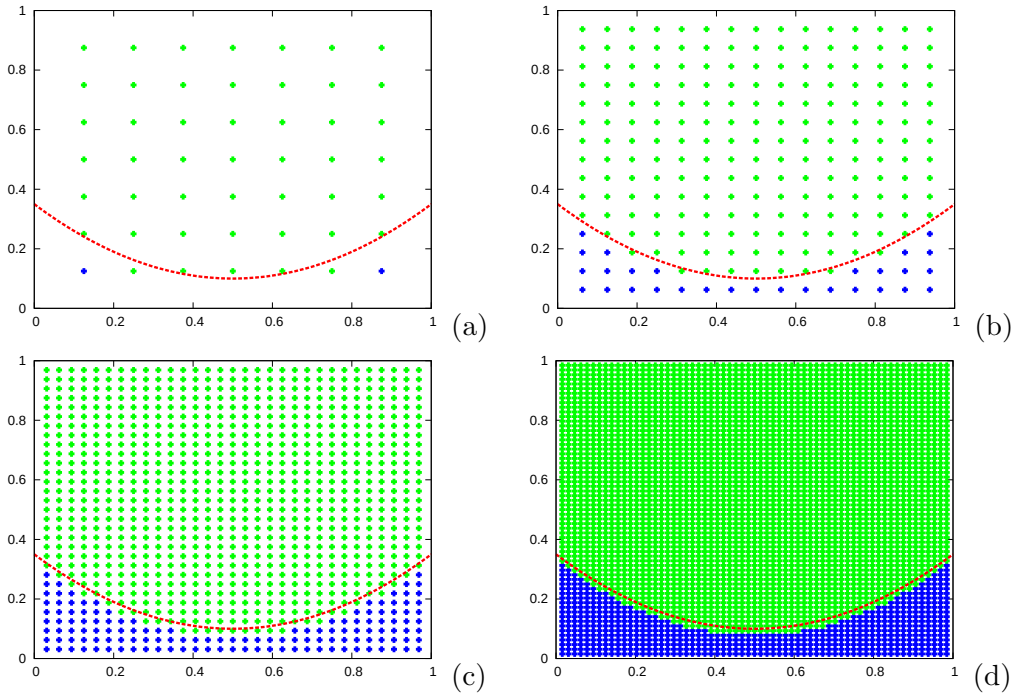

Figure 3.8: Application FAMGπ to the free boundary problem (3.6) for: (a) 9×9 points grid, (b) 17×17 points grid, (c) 33×33 points grid, (d) 65×65 points grid.

approximation of the solution in a few iterations. The interpolation of this solution and the corresponding strategies, are used to start AMGπ on the next fine level and we observe that only a small number of policy iterations are needed on each level.

With this example we show the advantage of using FAMGπ. Indeed, the computation time of the FAMGπ algorithm seems to be in the order of the number of discretization points whereas that of a AMGπ algorithm is about 160 times greater. This is due to the large number of iterations needed by AMGπ for solving this kind of games. Indeed, this number should be compared to the diameter of the graph (that is the largest number of edges which must be covered to travel from one point to another) associated to the corresponding game problem, for instance the union of all graphs of the Markov chains associated to all couple of fixed policies α and β. Due to the finite differences discretization, the arcs of the graphs are supported by edges of the grids $\mathcal{X}_h$ in $\mathbb{Z}^2$, so the diameter of the graph of the discrete game is $2m$ with $m = 1024$.

Note that, as said in Section 3.1.2, one can use a full multi-level algorithm similar to the FAMGπ Algorithm 3.3 with the policy iteration algorithm combined with a direct solver LU instead of AMGπ. Then, we should obtain the same improvement in the number of policy iterations compared to the plain policy iteration algorithm. Since the number of policy iterations at each level is small (maximum 4), and the computation time of an AMG iteration (or an AMGπ iteration) is in general in the order of the number of discretization points, the total computation time of FAMGπ algorithm should be linear in the number of discretization points. This would not be the case for the full multi-level algorithm using

a LU solver, since in this case the total computation time is greater or equal to that of the LU solver, which is not linear in the number of discretization points.

3.2.3 Stopping game with two optimal stopping

In this example, we consider a stopping game where both players have the possibility to stop the game, see [Fri73] for a complete theory about this subject. In this case, the value of the game starting in $x \in \mathcal{X}$ is given by :

$$v(x) = \sup_{\kappa_1} \inf_{\kappa_2} \left\{ \mathbb{E}_x^{\kappa_1,\kappa_2} \left[\int_0^{\kappa_1 \wedge \kappa_2} r(\xi_t)\, dt + \psi_1(\xi_{\kappa_1})\, \mathbb{1}_{\kappa_1<\kappa_2} + \psi_2(\xi_{\kappa_2})\, \mathbb{1}_{\kappa_2 \leq \kappa_1} \,\middle|\, \xi_0 = x \right] \right\}$$

where $\kappa_1 \wedge \kappa_2 = \min(\kappa_1, \kappa_2)$ and we assume $\min(\kappa_1, \kappa_2) < \tau$ ($\tau = \inf\{t \geq 0 \mid \xi_t \notin \mathcal{X}\}$, and that $\psi_2(x) > \psi_1(x)$ for all $x \in \mathcal{X}$. Then v is solution of the equation :

$$\max \left\{ \psi_1(x) - v(x),\, \min\{\psi_2(x) - v(x)\,,\, L(v;x) + r(x)\} \right\} = 0 \qquad \text{for } x \text{ in } \mathcal{X}, \quad (3.8)$$

or equivalently,

$$\begin{cases} (L(v;x) + r(x))(w(x) - v(x)) \leq 0 & \text{for } x \in \mathcal{X}, \\ \forall w,\ \psi_1 \leq w \leq \psi_2 \text{ and } \psi_1 \leq v \leq \psi_2 & , \end{cases}$$

that is

$$\text{for } x \in \mathcal{X} \quad \begin{cases} (L(v;x) + r(x)) \leq 0 & \text{if } v(x) = \psi_1(x) \\ (L(v;x) + r(x)) \geq 0 & \text{if } v(x) = \psi_2(x) \\ (L(v;x) + r(x)) = 0 & \text{if } \psi_1(x) < v(x) < \psi_2(x). \end{cases}$$

For the numerical tests, we consider the stochastic differential game whose value v is solution of :

$$\max \left\{ \psi_1(x) - v(x),\, \min\{\psi_2(x) - v(x)\,,\, 0.5\,\Delta v(x) + r(x)\} \right\} = 0 \qquad \text{for } x \text{ in } \mathcal{X}, \quad (3.9)$$

where $\mathcal{X} = [0,1]$, for all $x \in \mathcal{X}$: $\psi_1(x) = -\bar{\psi}_2$, $\psi_2(x) = \bar{\psi}_2$ with $\bar{\psi}_2 = (2\cos(0.09\pi) + \pi(0.18 - 1)\sin(0.09\pi))/2) \approx 0.6$ and $r(x) = 0.5\,\pi^2 \cos(\pi x)$. For all $x \in \mathcal{X}$, the sets of actions are $\mathcal{A} = \{0,1\}$ for MAX and $\mathcal{B} = \{0,1\}$ for MIN, where action 0 means that the player chooses to stop the game and receive ψ_1 when MAX stops or ψ_2 when MIN stops, action 1 means that the game is continuing. Here, the exact solution of Equation (3.9) in the viscosity sense is for $x \in \mathcal{X}$:

$$v(x) = \begin{cases} \psi_1(x) & \text{for } x > (1 - 0.09) \\ \psi_2(x) & \text{for } x < 0.09 \\ \cos(\pi x) + \pi \sin(0.09\pi)x + c & \text{for } 0.09 \leq x \leq (1 - 0.09) \end{cases}$$

where the constant $c = (\bar{\psi}_2 - \cos(0.09\pi) - 0.09\pi \sin(0.09\pi))$; v is represented in Figure 3.9. For all $x \in \mathcal{X}$, the optimal strategy for MAX is $\alpha(x) = 0$ if $x > (1 - 0.09)$ and $\alpha(x) = 1$ otherwise. For all $x \in \mathcal{X}$, the optimal strategy for MIN is $\beta(x) = 0$ if $x < 0.09$ and $\beta(x) = 1$ otherwise.

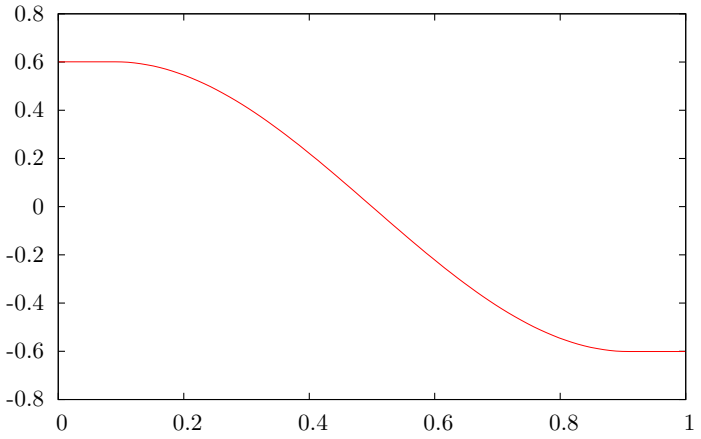

Figure 3.9: Solution of Equation (3.9)

We present numerical results for the discretization of Equation (3.9) on a grid with 2049 points in Table 3.11 when using AMGπ with $\epsilon = 10^{-10}$ and in Table 3.12 when using FAMGπ with $c = 10^{-2}$ and $\epsilon = 10^{-10}$. As in the previous example, we see the advantage of using FAMGπ for this kind of games. Indeed, FAMGπ solves the problem in about one second while AMGπ needs about 24 minutes. As for the previous example, the computation time of the FAMGπ seems to be in the order of the number of discretization points. For this example, due to the finite differences discretization, the diameter of the graph of the discrete game is m with $m = 2048$. We see in Table 3.11 that both numbers of intern and external policy iterations for AMGπ are in the order of the diameter of the graph.

Table 3.11: Numerical results for optimal stopping time game (3.9) with a 2049×2049 points grid, computed by AMGπ with $\epsilon = 10^{-10}$.

s	$kmax$	$\|r_v\|_\infty$	$\|r_v\|_{L_2}$	$\|e\|_\infty$	$\|e\|_{L_2}$	cpu time (s)
1	1	$1.20e+00$	$7.80e-01$	$7.75e-01$	$2.91e-01$	$<< 1$
2	863	$1.20e+00$	$2.66e-02$	$6.02e-01$	$3.00e-01$	$1.58e+00$
3	1025	$1.20e+00$	$2.66e-02$	$6.03e-01$	$3.00e-01$	$3.31e+00$
			...			
100	1026	$1.20e+00$	$2.66e-02$	$7.10e-01$	$2.46e-01$	$1.81e+02$
			...			
200	992	$1.20e+00$	$2.66e-02$	$8.16e-01$	$1.93e-01$	$3.75e+02$
			...			
300	947	$1.20e+00$	$2.66e-02$	$9.16e-01$	$1.44e-01$	$5.72e+02$
			...			
400	910	$1.20e+00$	$2.66e-02$	$1.00e+00$	$1.01e-01$	$7.73e+02$
			...			
500	882	$1.20e+00$	$2.66e-02$	$1.08e+00$	$6.59e-02$	$9.78e+02$
			...			
600	862	$1.20e+00$	$2.66e-02$	$1.14e+00$	$4.05e-02$	$1.19e+03$
			...			
700	849	$1.20e+00$	$2.66e-02$	$1.18e+00$	$2.84e-02$	$1.41e+03$
			...			
800	843	$1.20e+00$	$2.66e-02$	$1.20e+00$	$2.65e-02$	$1.64e+03$
			...			
839	843	$1.20e+00$	$2.66e-02$	$1.20e+00$	$2.66e-02$	$1.73e+03$
840	843	$2.03e-07$	$4.50e-09$	$5.22e-07$	$2.57e-07$	$1.74e+03$
841	1	$1.11e-16$	$6.57e-18$	$1.16e-07$	$7.40e-08$	$1.74e+03$

Table 3.12: Numerical results for optimal stopping time game (3.9) with a 2049×2049 points grid, computed by FAMGπ with $c = 10^{-2}$ and $\epsilon = 10^{-10}$.

s	$kmax$	$\|r_v\|_\infty$	$\|r_v\|_{L_2}$	$\|e\|_\infty$	$\|e\|_{L_2}$	cpu time (s)
points in each direction : 3, step size : $5.00e-01$						
1	2	$1.20e+00$	$1.20e+00$	$6.01e-01$	$6.01e-01$	$<< 1$
2	2	$0.00e+00$	$0.00e+00$	$5.55e-17$	$5.55e-17$	$<< 1$
points in each direction : 5, step size : $2.50e-01$						
1	2	$1.56e-01$	$9.02e-02$	$1.13e-01$	$1.03e-01$	$<< 1$
2	1	$1.89e-17$	$1.09e-17$	$1.13e-01$	$9.22e-02$	$<< 1$
points in each direction : 9, step size : $1.25e-01$						
1	2	$1.20e+00$	$4.54e-01$	$8.74e-01$	$3.52e-01$	$<< 1$
2	5	$1.20e+00$	$4.54e-01$	$1.09e+00$	$4.14e-01$	$<< 1$
3	5	$5.55e-17$	$2.21e-17$	$5.74e-03$	$4.35e-03$	$<< 1$
points in each direction : 17, step size : $6.25e-02$						
1	2	$1.20e+00$	$3.10e-01$	$1.16e+00$	$3.00e-01$	$<< 1$
2	10	$1.28e-03$	$3.30e-04$	$5.19e-03$	$2.83e-03$	$<< 1$
3	1	$0.00e+00$	$0.00e+00$	$3.36e-03$	$2.10e-03$	$<< 1$
points in each direction : 33, step size : $3.12e-02$						
1	2	$0.00e+00$	$0.00e+00$	$1.36e-04$	$9.50e-05$	$<< 1$
points in each direction : 65, step size : $1.56e-02$						
1	2	$1.20e+00$	$1.51e-01$	$1.20e+00$	$1.51e-01$	$<< 1$
2	28	$0.00e+00$	$0.00e+00$	$7.08e-05$	$4.94e-05$	$<< 1$
points in each direction : 129, step size : $7.81e-03$						
1	2	$1.20e+00$	$1.07e-01$	$1.20e+00$	$1.07e-01$	$<< 1$
2	54	$2.78e-17$	$3.75e-18$	$6.66e-05$	$3.85e-05$	$2.00e-02$
points in each direction : 257, step size : $3.91e-03$						
1	2	$1.20e+00$	$7.53e-02$	$1.20e+00$	$7.52e-02$	$2.00e-02$
2	108	$1.20e+00$	$7.53e-02$	$1.20e+00$	$7.53e-02$	$7.00e-02$
3	107	$1.11e-16$	$8.53e-18$	$1.61e-06$	$1.05e-06$	$1.30e-01$
points in each direction : 513, step size : $1.95e-03$						
1	2	$1.20e+00$	$5.32e-02$	$1.20e+00$	$5.32e-02$	$1.30e-01$
2	212	$1.11e-16$	$9.82e-18$	$4.53e-07$	$3.02e-07$	$3.00e-01$
points in each direction : 1025, step size : $9.77e-04$						
1	2	$1.20e+00$	$3.76e-02$	$1.20e+00$	$3.76e-02$	$3.00e-01$
2	422	$1.11e-16$	$1.15e-17$	$1.69e-07$	$1.18e-07$	$9.40e-01$
points in each direction : 2049, step size : $4.88e-04$						
1	2	$2.03e-07$	$4.50e-09$	$5.22e-07$	$2.57e-07$	$9.40e-01$
2	1	$1.11e-16$	$8.52e-18$	$1.16e-07$	$7.40e-08$	$9.50e-01$

3.3 Conclusion and perspective

In this chapter, we presented our algorithm AMGπ, which combines the policy iteration algorithm with algebraic multigrid methods, for solving two player zero-sum stochastic games. In the numerical tests, we compared AMGπ to the policy iteration algorithm with a direct solver LU on a Isaacs PDE and observed that the computation time is smaller when using AMGπ instead of the policy iteration with LU. Moreover, we noticed that the computation time needed by AMGπ increases linearly with the size of the problem.

Furthermore, we also presented a full multi-level algorithm, called FAMGπ, for solving two player zero-sum stochastic differential games. The numerical results on some stopping differential stochastic games presented here show that FAMGπ improves substantially the computation time of the policy iteration algorithm for this kind of games. Indeed the computation time of FAMGπ seems to be in the order of the number of discretization points whereas that of AMGπ algorithm is about 160 to 1700 times greater. This is due to the large number of iterations needed by AMGπ for solving this kind of games. Indeed, this number should be compared to the diameter of the graph associated to the corresponding game problem, for instance the union of all graphs of the Markov chains associated to fixed policies α and β.

The FAMGπ algorithm uses coarse grids discretizations of the partial differential equation and so cannot be applied directly to the dynamic programming equation of a two player zero-sum stochastic game with finite state space. One may ask if adapting the FAMGπ algorithm to this kind of games is possible. Indeed, the complexity of two player zero-sum stochastic games is still unsettled, one only knows that it belongs to the complexity class of NP$\cap$coNP [Pur95], and any new approach maybe useful to understand this complexity.

The results of this chapter were presented in the paper [AD12].

Chapter 4

Policy iteration algorithm for zero-sum stochastic games with mean payoff

In this chapter, we present a working paper [ADCTG12].

4.1 Introduction

The mean-payoff problem for zero-sum two player multichain games We consider a zero-sum stochastic game with finite state space $[n] := \{1, \dots, n\}$, finite action spaces $\mathcal{A}$ and $\mathcal{B}$ for the first and second player respectively, and perfect information. In the case of the finite horizon problem, in which the payoff of the game induced by a pair of strategies of the two players is defined as the expectation of the sum in finite horizon of the successive rewards (the payments of the first player to the second player), Shapley showed (see [Sha53]) that the value v_i^τ of the game with horizon T and initial state $i \in [n]$ satisfies the *dynamic programming equation* $v^{T+1} = f(v^T)$, with a dynamic programming operator $f : \mathbb{R}^n \to \mathbb{R}^n$ defined as :

$$[f(v)]_i \;=\; \min_{a\in\mathcal{A}} \left(\max_{b\in\mathcal{B}} \left(\sum_{j\in[n]} P_{ij}^{ab}\, v_j \;+\; r_i^{ab} \right) \right), \qquad \forall i \in [n], v \in \mathbb{R}^n \;\; .$$

Here, r_i^{ab} and P_{ij}^{ab} represent respectively the reward in state $i \in [n]$ and the transition probability from state i to state $j \in [n]$, when the actions of the first and second players are respectively equal to $a \in \mathcal{A}$ and $b \in \mathcal{B}$.

The above dynamic programming operator f is *order-preserving*, meaning that $v \leq w \implies f(v) \leq f(w)$ where $\leq$ denotes the partial ordering of $\mathbb{R}^n$, and *additively homogeneous*, meaning that it commutes with the addition of a constant vector. These two conditions imply that f is nonexpansive in the sup-norm (see for instance [CT80], see also [GG04] for more background on this class of nonlinear maps). Moreover, f is *polyhedral*, meaning that there is a covering of $\mathbb{R}^n$ by finitely many polyhedra such that the

restriction of f to any of these polyhedra is affine. Kohlberg [Koh80] showed that if f is a polyhedral self-map of $\mathbb{R}^n$ that is nonexpansive in some norm, then, there exist two vectors η and v in $\mathbb{R}^n$ such that $f(t\eta + v) = (t+1)\eta + v$, for all $t \in \mathbb{R}$ large enough. A map of the form $t \mapsto t\eta + v$ is called a *half-line*, and η is its *slope*. It is *invariant* if it satisfies the latter property. Moreover this property is equivalent to the following system of nonlinear equations for the couple (η, v) :

$$\begin{cases} \eta &= \hat{f}(\eta) \ , \\ \eta + v &= \acute{f}_\eta(v) \ , \end{cases} \tag{4.1}$$

where the maps $\hat{f}$ (the recession function) and $\acute{f}_\eta$ are constructed from f (see Section 4.2).

When f has an invariant half-line with slope η, the growth rate of its orbits (also called the cycle time) $\chi(f) := \lim_{k\to\infty} f^k(v)/k$ exists and is equal to η. Here, f^k denotes the k-th iterate of f, and v is an arbitrary vector of $\mathbb{R}^n$. This shows in particular that the value of the finite horizon game satisfies $\lim_{T\to\infty} v_i^T/T = \eta_i$ for any final reward. Moreover, η_i gives the value of the game with initial state i, and *mean payoff*, that is such that the payoff of the game induced by a pair of strategies of the two players is the Cesaro limit of the expectation of the successive rewards. Then a vector v such that $t \mapsto t\eta + v$ is an invariant half-line is called a *relative value* of the game, or *bias*. It is not unique, even up to an additive constant.

In this paper, we give an algorithm to find an invariant half-line, or equivalently a solution of (4.1), for general multichain games. This allows us in particular to determine the mean payoff, as well as optimal strategies for both players. By *multichain*, we mean that there is no irreducibility assumption on the Markov chains associated to the strategies of the two players, which may have in particular several invariant measures.

Classes of games solvable by earlier policy iteration algorithms Policy iteration is a general method initially introduced by Howard [How60] in the case of one player problems (Markov decision processes). The idea is to compute a sequence of strategies as well as certain valuations, which serve as optimality certificates, and to use the current valuation to improve the strategy. The algorithm bears some resemblance with the Newton method, as the strategy determines a sub or super-gradient of the dynamic programming operator. The key of the analysis of policy iteration algorithms is generally to show that the sequence of valuations which are computed satisfies a monotonicity property, from which it can be inferred that the same strategy is never selected twice. In the discounted one player case, the valuation which is maintained by the algorithm is nothing but the value vector of the current policy. For one-player games with mean-payoff, in the unichain case (in which every stochastic matrix associated to a strategy has only one final class), the valuation consists of the mean payoff of the current strategy, as well as of a relative value. In both cases, the monotonicity property is natural (it relies on the discrete maximum principle, or properties of monotonicity and contraction of the dynamic programming operator, or on the uniqueness of the invariant measure associated to a strategy). However, even for

one player games, the extension to the multichain case is more difficult. It was initially proposed by Howard [How60]. The convergence of his method was established by Denardo and Fox [DF68].

The idea of extending Howard algorithm to the two player case appeared independently in the work of Hoffman and Karp [HK66b] for a subclass of mean-payoff games with imperfect information, and in the work of Denardo [Den67] for discounted games. Both algorithms consist of nested iterations; the internal iterations are a simplified version of the one player Howard algorithm. The algorithm for discounted games appeared also, as an adaptation of the Hoffman-Karp algorithm, in the work of Rao, Chandrasekaran, and Nair [RCN73, Algorithm 1] and of Puri [Pur95] (deterministic games with perfect information). More recently, Raghavan and Syed [RS03] developed a related algorithm in which strategy improvements involve only one state at each iteration.

The Hoffman-Karp algorithm requires the game to satisfy a strong irreducibility assumption (each stochastic matrix arising from a choice of strategies of the two players must be irreducible). Without an assumption of this kind, *degenerate iterations*, at which the mean payoff vector is not improved, may occur, and so the algorithm may cycle (we shall indeed see such an example in Section 4.6). This pathology appears in particular for the important subclass of deterministic mean payoff games, for which the irreducibility assumption is essentially never satisfied.

A natural idea to solve mean-payoff games, appearing for instance in the work of Puri [Pur95], is to apply the policy iteration algorithm of Denardo [Den67] or Rao, Chandrasekaran, and Nair [RCN73, Algorithm 1] for discounted games, choosing a given discount factor α sufficiently close to one, which allows one to determine the so-called *Blackwell optimal policies*. For deterministic games, when the rewards are integers with modulus less or equal to W and the number of states is equal to n, Zwick and Paterson [ZP96] showed that taking $1-\alpha = 1/(4n^3W)$ is sufficient to determine the mean payoff by a rounding argument. However this requires high precision arithmetics. In the case of stochastic games, the situation is even worse, since examples are known in which the value of $1-\alpha$ to be used for rounding has a denominator exponential in the number of states. In particular, an approach of this kind is impracticable if one works in floating point (bounded precision) arithmetics. Hence, it is desirable to have a policy iteration algorithm for multichain stochastic games relying only on the computation of mean payoffs and relative values as in the algorithm of Howard [How60] and Denardo and Fox [DF68].

The first policy iteration algorithm not relying on vanishing discount, for general (multichain) deterministic mean payoff games, was apparently introduced by Cochet-Terrasson, Gaubert and Gunawardena [CTGG99, GG98]. The former reference concerns the special case in which the mean payoff is the same for all states at each iteration, whereas the second one covers the general case, see also [CT01]. Details of implementation, as well as experimental results were given in [DG06]. The idea of the algorithm of [CTGG99] is to handle degenerate iterations by a tropical (max-plus) spectral projector. The latter is a tropically linear retraction of the whole space onto the fixed point set of the dynamic

programming operator associated to a given strategy of the first player. When the mean payoff or the current strategy is not improved, the new relative value is obtained by applying a spectral projector to the earlier relative value. The proof of termination of the algorithm [CTGG99] relies on a key ingredient from tropical spectral theory, that a fixed point of a tropically linear map is uniquely defined by its restriction to the critical nodes (the nodes appearing infinitely often in a strategy which gives the optimal mean payoff). Then, it was shown in [CTGG99] that at each degenerate iteration, the relative value decreases, and that the set of critical nodes also decreases, from which the termination of the algorithm can be deduced.

A related class of games consists of parity games, which can be encoded as special deterministic games with mean payoff. A policy improvement algorithm for parity games was introduced by Vöge and Jurdziński [VJ00]. This algorithm differs from the one of [CTGG99, GG98] in that instead of the relative value, the algorithm maintains a set of relevant reachable vertices. Other policy algorithm for parity games or deterministic mean payoff games were introduced later on by Bjorklund, Sandberg and Vorobyov [BSV04, BV07], and by Jurdziński, Paterson, and Zwick [JPZ06]. An experimental comparison of algorithms for deterministic games was recently made by Chaloupka [Cha11, Cha09], who also gave an optimized version of the algorithm of [CTGG99, GG98, DG06].

In [BCPS04], Bielecki, Chancelier, Pliska, and Sulem used a policy iteration algorithm to solve a semi-Markov mean-payoff game problem with infinite action spaces obtained from the discretization of a quasi-variational inequality, based on the approach of [CTGG99, GG98]. Their algorithm proceeds in a Hoffman and Karp fashion.

Policy iteration algorithm for stochastic multichain zero-sum games with mean payoff Inspired by the policy iteration algorithm of [CTGG99, GG98] for deterministic games, Cochet-Terrasson and Gaubert proposed in [CTG06] a policy iteration algorithm for general stochastic games (see also [CT01] for a preliminary version). The relative values are now constructed using the nonlinear analogues of tropical spectral projectors. These nonlinear projectors where introduced by Akian and Gaubert in [AG03]. They can be thought of as a nonlinear analogues of the operation of reduction of a super-harmonic function, arising in potential theory. However, no implementation details were given in the short note [CTG06], in which the algorithm was stated abstractly, in terms of invariant half-lines.

We develop here fully the idea of [CTG06], and describe a policy iteration algorithm for multichain stochastic games with mean-payoff (see Section 4.4.2). We explain how nonlinear systems of the form (4.1) are solved at each iteration. We show in particular how non-linear spectral projections can be computed, by solving an auxiliary (one player) optimal stopping problem. This relies on the determination of the so called *critical graph*, the nodes of which (*critical nodes*) are visited infinitely often (almost surely) by an optimal strategy of a one player mean payoff stochastic game. An algorithm to compute the critical graph, based on results on [AG03], is given in Section 4.5.3.

We give the proof of the convergence theorem (which was only stated in [CTG06]). In particular, we show that the sequence $(\eta^{(k)}, v^{(k)}, C^{(k)})$ consisting of the mean payoff vector, relative value vector, and set of critical nodes, constructed by the algorithm satisfies a kind of lexicographical monotonicity property so that it converges in finite time (see Section 4.4.3). The proof of convergence exploits some results of spectral theory of convex order-preserving additively homogeneous maps, by Akian and Gaubert [AG03]. Hence, the situation is somehow analogous to the deterministic case [CTGG99], the technical results of tropical (linear) spectral theory used in [CTGG99] being now replaced by their non-linear analogues [AG03]. Note also that the convergence proof of the algorithm of [CTGG99, GG98] can be recovered as a special case of the present proof.

The convergence proof leads to a coarse exponential bound on the execution time of the algorithm: the number of iterations of the first player is bounded by its number of strategies, and the number of elementary iterations (resolutions of linear systems) is bounded by the product of the number of strategies of both players.

We also show that the specialization of this algorithm to a one-player game gives an algorithm which is similar to the multichain policy algorithm of Howard and Denardo and Fox, see Section 4.5.2.

Then, we discuss an example (see Section 4.6) involving a variant of Richman games [LLP$^+$99] (also called stochastic tug-of-war [PSSW09], related with discretizations of the infinity Laplacian [Obe05]), showing that degenerate iterations do occur and that cyclic may occur with naive policy iteration rules. Hence, the handling of degenerate iterations, that we do here by nonlinear spectral projectors, cannot be dispensed with.

The present algorithm has been implemented in the C library PIGAMES by Detournay, see [Det12] for more information. We finally report numerical experiments (see Section 4.7) carried out using this library, both on random instances of Richman type games with various numbers of states and on a class of discrete games arising from the monotone discretization of a pursuit-evasion differential game. These examples indicate that degenerate iterations are frequent, so that their treatment cannot be dispensed with. They also show that the algorithm scales well, allowing one to solve structured instances with 10^6 nodes and 10^7 actions in a few hours of CPU time on a single core processor (the bottleneck being the resolution of linear systems).

We note that our experimental results are consistent with earlier experimental tests carried out for simpler algorithms (dealing with one player or deterministic problems) of which the present one is an extension. These tests indicate that policy iteration algorithms are fast on typical instances (although instances with an exponential number of iterations have been recently constructed, as discussed in the next subsection). Indeed, in the case of one-player deterministic games (maximal circuit mean problem), Dasdan, Irani and Guptka [DIG98] concluded that the instrumentation of Howard's policy iteration algorithm by Cochet-Terrasson et al. [CTCG$^+$98], in which each iteration is carried out in linear time, was the fastest algorithm on their test suite. Dasdan latter on developed further optimizations of this method [Das04]. More recent experiments by Georgiadis, Goldberg,

Tarjan, and Werneck [GGTW09] have indicated that the class of cycle based algorithms (to which [CTCG+98, Das04] belongs) is among the best performers, close second to the tree based method of Young, Tarjan, and Orlin [YTO91]. In the deterministic two player case, Chaloupka [Cha09] compared several algorithms and observed that the one of [CTGG99, GG98, DG06], with the optimization that he introduced (see also [Cha11]), is experimentally the best performer.

Alternative algorithms and complexity issues Gurvich, Karzanov and Khachiyan [GKK88] were the first to develop a combinatorial algorithm (*pumping algorithm*) to solve zero-sum deterministic games with mean payoff. An alternative approach was developed by Zwick and Paterson [ZP96], who showed that such a game can be solved by considering the finite horizon game for a sufficiently large horizon, and applying a rounding argument. Both algorithms are pseudo-polynomial. Other algorithms, also pseudo-polynomial, based on max-plus (tropical) cyclic projections, with a value iteration flavor, have been developed by Butkovič and Cuninghame-Green [CGB03], Gaubert and Sergeev [GS07a], and Akian, Gaubert, Nitiça and Singer [AGNS11]. Deterministic mean payoff games have been recently proved to be equivalent to decision problems for tropical polyhedra (the tropical analogue of linear programming) [AGG12b]. More generally, the results there show that *stochastic* games problems with mean payoff can be cast as tropical convex (non-polyhedral) programming problems.

The pumping algorithm of [GKK88] was recently extended to the case of stochastic games with perfect information by Boros, Elbassioni, Gurvich, and Makino [BEGM10]. They showed that their algorithm is pseudo-polynomial when the number of states of the game at which a random transition occurs remains fixed. No pseudo-polynomial seems currently known without the latter restriction. Their algorithm applies to more general games than the ones covered by the irreducibility assumption of Hoffman and Karp in [HK66b], but it does not apply to all multichain games.

The question of the complexity of deterministic mean payoff games was raised in [GKK88], and it has remained open since that time. Note in this respect that such games are known to have a good characterization in the sense of Edmonds, i.e., to be in NP∩coNP. Indeed, the strategies of one player can be used as concise certificates, as observed by Condon [Con92], Paterson and Zwick [ZP96]. Such games even belong to the class UP∩coUP as shown by Jurdziński [Jur98]. We refer the reader to the discussion in [BSV04, JPZ06] for more information. The arguments of Condon [Con92] also imply that zero-sum stochastic games with perfect information (and finite state and action spaces) belong to NP∩coNP. An important subclass of deterministic games with mean payoff consists of parity games. These can be reduced to mean payoff deterministic games (Puri [Pur95]), which in turn can be reduced to discounted deterministic games. The latter ones can be reduced to simple stochastic games (Zwick and Paterson [ZP96]). In [AM09], Andersson and Miltersen generalized this result showing that stochastic mean payoff games with perfect information, stochastic parity games and simple stochastic games are polynomial time equivalent.

In particular, the decision problem corresponding to a game of any of these classes lies in the complexity class of NP∩coNP.

Friedmann has recently constructed an example [Fri09] showing that the Vöge-Jurdjińsky strategy improvement algorithm for parity games [VJ00] may require an exponential number of iterations. This also yields an exponential lower bound [Fri11] for the Hoffman-Karp strategy improvement rule for discounted deterministic games [Pur95]. The result of Friedmann has also been extended to total reward and undiscounted MDP by Fearnley [Fea10a, Fea10b] and to simple stochastic games and weighted discounted stochastic games by Andersson [And09].

Moreover, for Markov decision process with a *fixed* discount factor, some upper bound on the number of policy iterations was given in [MH86]. Recently, Ye gave a the first strongly polynomial bound [Ye05, Ye11]. The latter bound has been improved and generalized to zero-sum two player stochastic games with perfect information factor by Hansen, Miltersen and Zwick in [HMZ11], again for a fixed discount factor, giving the first strongly polynomial bound for these games. Note that a polynomial bound for mean payoff games does not follow from these results (to address the mean payoff case, we need to consider the situation in which the discount factor tends to 1).

Complexity results of a different nature have been established with motivations from numerical analysis (discretizations of PDE), exploiting in particular the relation between policy iteration and the Newton method. The policy iteration algorithm for one-player discounted games with an infinite number of actions has been proved to have a superlinear convergence around the solution under suitable assumptions (see in particular the works of Puterman and Brumelle [PB79], Akian [Aki90b], and Bokanowski, Maroso, and Zidani [BMZ09]). Chancelier, Messaoud, and Sulem [CMS07] also considered, in view of their application to quasi-variational inequalities, partially undiscounted infinite horizon problems for which they proved the contraction of the policy iteration algorithm.

The plan of the paper is the following: Section 4.2 is recalling some background on stochastic zero-sum two player games, Section 4.3 explains the construction of the nonlinear projection, Section 4.4 gives the algorithm, its practical version and its proof, Section 4.5 gives the ingredients of the algorithm, Section 4.6 shows an example with possible cycling of iterations when not using the notion of spectral projector, and Section 4.7 is for the numerical experiments.

4.2 Two player zero-sum stochastic games with discrete time and mean payoff

The class of two player zero-sum stochastic games was first introduced by Shapley in the early fifties, see [Sha53]. We recall in this section basic definitions on these games in the case of finite state space and discrete time (for more details see [Sha53, FV97, Sor03]).

We consider the finite state space $[n] := \{1, \dots, n\}$. A stochastic process $(\xi_k)_{k\geq 0}$ on $[n]$ gives the state of the game at each point time k, called stage. At each of these stages, two

players, called "MIN" and "MAX" (the minimizer and the maximizer) have the possibility to influence the course of the game.

The *stochastic game* $\Gamma(i_0)$ starting from $i_0 \in [n]$ is played in stages as follows. The initial state ξ_0 is equal to i_0 and known by the players. Player MIN plays first, and chooses an action ζ_0 in a set of possible actions $\mathcal{A}_{\xi_0}$. Then the second player, MAX, chooses an action η_0 in a set of possible actions $\mathcal{B}_{\xi_0}$. The actions of both players and the current state determine the payment $r_{\xi_0}^{\zeta_0\eta_0}$ made by MIN to MAX and the probability distribution $j \mapsto P_{\xi_0 j}^{\zeta_0\eta_0}$ of the new state ξ_1. Then the game continues in the same way with state ξ_1 and so on.

At a stage k, each player chooses an action knowing the *history* defined by $\iota_k = (\xi_0, \zeta_0, \eta_0, \cdots, \xi_{k-1}, \zeta_{k-1}, \eta_{k-1}, \xi_k)$ for MIN and (ι_k, ζ_k) for MAX. We call a *strategy* or *policy* for a player, a rule which tells him the action to choose in any situation. There are several classes of strategies. Assume $\mathcal{A}_i \subset A$ and $\mathcal{B}_i \subset B$ for some sets A and B. A *behavior or randomized strategy* for MIN (resp. MAX) is a sequence $\bar{\alpha} := (\alpha_0, \alpha_1, \cdots)$ (resp. $\bar{\beta} := (\beta_0, \beta_1, \cdots)$) where α_k (resp. β_k) is a map which to a history $h_k = (i_0, a_0, b_0, \ldots, i_{k-1}, a_{k-1}, b_{k-1}, i_k)$ with $i_\ell \in [n]$, $a_\ell \in \mathcal{A}_{i_\ell}$, $b_\ell \in \mathcal{B}_{i_\ell}$ for $0 \le \ell \le k$ (resp. (h_k, a_k)) at stage k associates a probability distribution on a probability space over A (resp. B) which support is included in the possible actions space $\mathcal{A}_{i_k}$ (resp. $\mathcal{B}_{i_k}$). A *Markovian strategy* is a strategy which only depends on the information of the current stage k: α_k (resp. β_k) depends only on i_k (resp. (i_k, a_k)), then $\alpha_k(h_k)$ (resp. $\alpha_k(h_k, a_k)$) will be denoted $\alpha_k(i_k)$ (resp. $\beta_k(i_k, a_k)$). It is said *stationary* if it is independent of k, then α_k is also denoted by α and β_k by β. A strategy of any type is said *pure* if for any stage k, the values of α_k (resp. β_k) are Dirac probability measures at certain actions in $\mathcal{A}_{i_k}$ (resp. $\mathcal{B}_{i_k}$) then we denote also by α_k (resp. β_k) the map which to the history assigns the only possible action in $\mathcal{A}_{i_k}$ (resp. $\mathcal{B}_{i_k}$).

In particular, if $\bar{\alpha}$ is a pure Markovian stationary strategy, also called *feedback* strategy, then $\bar{\alpha} = (\alpha_k)_{k\ge 0}$ with $\alpha_k = \alpha$ for all k and α is a map $[n] \to A$ such that $\alpha(i) \in \mathcal{A}_i$ for all $i \in [n]$. In this case, we also speak about pure Markovian stationary or feedback strategy for α and we denote by $\mathcal{A}_M$ the set of such maps. We adopt a similar convention for player MAX : $\mathcal{B}_M := \{\beta : [n] \times A \to B \mid \beta(i, a) \in \mathcal{B}_i \, \forall i \in [n],\, a \in \mathcal{A}_i\}$.

A strategy $\bar{\alpha} = (\alpha_k)_{k\ge 0}$ (resp. $\bar{\beta} = (\beta_k)_{k\ge 0}$) together with an initial state determines stochastic processes $(\zeta_k)_{k\ge 0}$ for the actions of MIN, $(\eta_k)_{k\ge 0}$ for the actions of MAX and $(\xi_k)_{k\ge 0}$ for the states of the game such that

$$P(\xi_{k+1} = j \,|\, \iota_k = h_k, \zeta_k = a, \eta_k = b) = P_{ij}^{ab} \tag{4.2a}$$

$$P(\zeta_k \in A' \,|\, \iota_k = h_k) = \alpha_k(h_k)(A') \tag{4.2b}$$

$$P(\eta_k \in B' \,|\, \iota_k = h_k, \zeta_k = a) = \beta_k(h_k, a)(B') \ , \tag{4.2c}$$

where $\iota_k := (\xi_0, \zeta_0, \eta_0, \ldots, \xi_{k-1}, \zeta_{k-1}, \eta_{k-1}\xi_k)$ is the history process, h_k is a history vector at time k: $h_k = (i_0, a_0, b_0, \ldots, i_{k-1}, a_{k-1}, b_{k-1}, i)$ and A' (resp. B') are measurable sets in A (resp. B). For instance, for each pair of feedback strategies (α, β) of the two players, that is such that for $k \ge 0$: $\alpha_k = \alpha$ with $\alpha \in \mathcal{A}_M$ and $\beta_k = \beta$ with $\beta \in \mathcal{B}_M$, the state

process $(\xi_k)_{k\geq 0}$ is a Markov chain on $[n]$ with transition probability

$$P(\xi_{k+1} = j \,|\, \xi_k = i) \;=\; P_{ij}^{\alpha(i)\beta(i,\alpha(i))} \qquad \text{for } i,j \in [n] \;\;,$$

and $\zeta_k = \alpha(\xi_k)$ and $\eta_k = \beta(\xi_k, \zeta_k)$.

When the strategies $\bar{\alpha}$ for MIN and $\bar{\beta}$ for MAX are fixed, the *payoff in finite horizon* τ of the game $\Gamma(i, \bar{\alpha}, \bar{\beta})$ starting from i is

$$J^\tau(i, \bar{\alpha}, \bar{\beta}) \;=\; \mathbb{E}_i^{\bar{\alpha}\bar{\beta}} \left[\sum_{k=0}^{\tau-1} r_{\xi_k}^{\zeta_k \eta_k} \right],$$

where $\mathbb{E}_i^{\bar{\alpha},\bar{\beta}}$ denotes the expectation for the probability law determined by (4.2). The *mean payoff* of the game $\Gamma(i, \bar{\alpha}, \bar{\beta})$ starting from i is

$$J(i, \bar{\alpha}, \bar{\beta}) \;=\; \limsup_{\tau\to\infty} \frac{1}{\tau} J^\tau(i, \bar{\alpha}, \bar{\beta}).$$

When the action spaces $\mathcal{A}_i$ and $\mathcal{B}_i$ are finite sets for all $i \in [n]$, the finite horizon game and the mean payoff game have a *value* which is given respectively by:

$$v_i^\tau \;=\; \inf_{\bar{\alpha}} \sup_{\bar{\beta}} J^\tau(i, \bar{\alpha}, \bar{\beta}), \tag{4.3}$$

and

$$\rho_i \;=\; \inf_{\bar{\alpha}} \sup_{\bar{\beta}} J(i, \bar{\alpha}, \bar{\beta}), \tag{4.4}$$

for all starting state $i \in [n]$, where the infimum is taken among all strategies $\bar{\alpha}$ for MIN and the supremum is taken over all strategies $\bar{\beta}$ for MAX (see [Sha53] for finite horizon games, and [LL69] for mean payoff games).

Indeed, the value v^τ of the finite horizon game satisfies the *dynamic programming equation* [Sha53]:

$$v_i^{\tau+1} \;=\; \min_{a\in\mathcal{A}_i} \left(\max_{b\in\mathcal{B}_i} \left(\sum_{j\in[n]} P_{ij}^{ab}\, v_j^\tau \;+\; r_i^{ab} \right) \right), \qquad \forall i \in [n], \tag{4.5}$$

with initial condition $v_i^0 = 0,\ i \in [n]$. Moreover, optimal strategies are obtained for both players by taking pure Markovian strategies $\bar{\alpha}$ for MIN and $\bar{\beta}$ for MAX such that, for all $k = 0, \ldots, \tau-1$, and i in $[n]$, $\alpha_k(i)$ attains the minimum in (4.5) with τ replaced by $\tau - k - 1$, and that, for all $k = 0, \ldots, \tau-1$, i in $[n]$ and a in $\mathcal{A}_i$, $\beta_k(i,a)$ attains the maximum in the expression of $F(v^{\tau-k-1}; i, a)$ defined as follows:

$$F(v; i, a) \;=\; \max_{b\in\mathcal{B}_i} \left(\sum_{j\in[n]} P_{ij}^{ab}\, v_j \;+\; r_i^{ab} \right). \tag{4.6}$$

We denote by f the *dynamic programming or Shapley operator* from $\mathbb{R}^n$ (that is here equivalent to $\mathbb{R}^{[n]}$) to itself given by:

$$[f(v)]_i := F(v; i) \;:=\; \min_{a\in\mathcal{A}_i} F(v; i, a), \qquad \forall i \in [n],\ v \in \mathbb{R}^n. \tag{4.7}$$

Then, the dynamic programming equation of the finite horizon game writes:

$$v^{\tau+1} = f(v^\tau). \tag{4.8}$$

The operator f is *order-preserving*, i.e. $v \leq w \implies f(v) \leq f(w)$ where $\leq$ denotes the partial ordering of $\mathbb{R}^n$ ($v \leq w$ if $v_i \leq w_i$ for all $i \in [n]$), and *additively homogeneous*, i.e. it commutes with the addition of a constant vector, which means that $f(\lambda + v) = \lambda + f(v)$ for all $\lambda \in \mathbb{R}$ and $v \in \mathbb{R}^n$, where $\lambda + v = (\lambda + v_i)_{i\in[n]}$. This implies that f is nonexpansive in the sup-norm (see for instance [CT80]). Note that it was observed independently by Kolokoltsov [Kol92], by Gunawardena and Sparrow (see [Gun03]) and by Rubinov and Singer [RS01] that, conversely, if $f : \mathbb{R}^n \to \mathbb{R}^n$ is order-preserving and additively homogeneous, then f can be put in the form (4.6,4.7), with possibly infinite sets A_i and B_i.

When the action spaces $\mathcal{A}_i$ and $\mathcal{B}_i$ are finite sets for all $i \in [n]$, the map f is also *polyhedral*, meaning that there is a covering of $\mathbb{R}^n$ by finitely many polyhedra such that the restriction of f to any of these polyhedra is affine. Kohlberg [Koh80] showed that if f is a polyhedral self-map of $\mathbb{R}^n$ that is nonexpansive in some norm, then, there exist two vectors v and η in $\mathbb{R}^n$ such that $f(t\eta + v) = (t + 1)\eta + v$, for all $t \in \mathbb{R}$ large enough. A map $\omega : t \in [t_0, \infty) \mapsto t\eta + v \in \mathbb{R}^n$, with $t_0 \in \mathbb{R}$, and $\eta, v \in \mathbb{R}^n$, is called a *half-line* with slope η. A *germ of half-line at infinity* is an equivalence class for the equivalence relation on half-lines $\omega \sim \omega'$ if $\omega(t) = \omega'(t)$ for $t \in \mathbb{R}$ large enough. A germ can be identified with the couple (η, v) of vectors of $\mathbb{R}^n$. Hence, in the sequel, we shall use the expression "half-line" either for a map $\omega : t \in [t_0, \infty) \mapsto t\eta + v \in \mathbb{R}^n$, for its germ, or for the couple (η, v). We shall say that it is *invariant* by f if it satisfies the latter property, that is $f(t\eta + v) = (t + 1)\eta + v$, for all $t \in \mathbb{R}$ large enough. The interest of an invariant half-line is that its slope determines the growth rate of the orbits of f, $\chi(f) := \lim_{k\to\infty} f^k(w)/k$. Here, f^k denotes the k-th iterate of f, and w is an arbitrary vector of $\mathbb{R}^n$. When it exists, the growth rate $\chi(f)$ is called the *cycle time* of f. Indeed, if $f(t\eta + v) = (t + 1)\eta + v$ for $t \geq t_0$, then $f^k(t_0\eta + v) = (t_0 + k)\eta + v$ for $k \geq 0$, hence $\lim_{k\to\infty} f^k(t_0\eta + v)/k = \eta$, and by the nonexpansiveness of f, $\lim_{k\to\infty} f^k(w)/k = \eta$ for all $w \in \mathbb{R}^n$, that is $\chi(f)$ does exist and is equal to η. For the game problem this shows that the value of the finite horizon game has a linear growth with respect to time:

$$\lim_{\tau\to\infty} \frac{1}{\tau} v_i^\tau = [\chi(f)]_i = \eta_i \ ,$$

where f is the Shapley operator defined in (4.6,4.7). Moreover, the value ρ of the mean payoff game defined in (4.4) coincides with the slope of an invariant half-line of f, and thus with the former limit:

$$\rho_i = \eta_i = [\chi(f)]_i.$$

Finally, when the action spaces are finite, one can easily see that the Shapley operator f in (4.6,4.7) satisfies for all $\eta, v \in \mathbb{R}^n$,

$$f(t\eta + v) = t\hat{f}(\eta) + \acute{f}_\eta(v) \quad \text{for } t \text{ large}, \tag{4.9}$$

where $\hat{f}$ is the *recession function* of f (see [GG04]):

$$[\hat{f}(\eta)]_i := \lim_{t\to\infty} \frac{[f(t\eta)]_i}{t} = \min_{a\in\mathcal{A}_i} \max_{b\in\mathcal{B}_i} \left(\sum_{j\in[n]} P_{ij}^{ab}\, \eta_j \right), \quad i \in [n] \ , \tag{4.10}$$

and $\acute{f}_\eta$ is what we shall call the *tangent of f at infinity* around the slope η:

$$[\acute{f}_\eta(v)]_i := \lim_{t\to\infty} [f(t\eta + v) - t\hat{f}(\eta)]_i = \min_{a\in\acute{\mathcal{A}}_{i,\eta}} \max_{b\in\acute{\mathcal{B}}_{i,a,\eta}} \left(\sum_{j\in[n]} P_{ij}^{ab}\, v_j \ + \ r_i^{ab} \right) \ , \tag{4.11a}$$

with

$$\acute{\mathcal{A}}_{i,\eta} := \operatorname*{argmin}_{a\in\mathcal{A}_i} \left\{ \max_{b\in\mathcal{B}_i} \left(\sum_{j\in[n]} P_{ij}^{ab}\, \eta_j \right) \right\} \tag{4.11b}$$

$$\acute{\mathcal{B}}_{i,a,\eta} := \operatorname*{argmax}_{b\in\mathcal{B}_i} \left\{ \sum_{j\in[n]} P_{ij}^{ab}\, \eta_j \right\} \ . \tag{4.11c}$$

Indeed, for an action $a \in \mathcal{A}_i$ and $i \in [n]$, we have from the finiteness of the sets $\mathcal{B}_i$:

$$\begin{aligned} F(t\eta + v; i, a) \ &= \max_{b\in\mathcal{B}_i} \left(\sum_{j\in[n]} P_{ij}^{ab}\, (t\eta_j + v_j) + \ r_i^{ab} \right) \\ &= \max_{b\in\mathcal{B}_i} \left(t \sum_{j\in[n]} P_{ij}^{ab}\, \eta_j + P_{ij}^{ab}\, v_j + \ r_i^{ab} \right) \\ &= \max_{b\in\mathcal{B}_i} \left(t \sum_{j\in[n]} P_{ij}^{ab}\, \eta_j \right) + \max_{b\in\acute{\mathcal{B}}_{i,a,\eta}} \left(\sum_{j\in[n]} P_{ij}^{ab}\, v_j + \ r_i^{ab} \right) \qquad \text{for } t \text{ large} \\ &= t\, \hat{F}(\eta; i, a) + \acute{F}_\eta(v; i, a) \end{aligned}$$

where one denotes :

$$\hat{F}(\eta; i, a) \ := \max_{b\in\mathcal{B}_i} \left(\sum_{j\in[n]} P_{ij}^{ab}\, \eta_j \right) \tag{4.12}$$

$$\acute{F}_\eta(v; i, a) \ := \max_{b\in\acute{\mathcal{B}}_{i,a,\eta}} \left(\sum_{j\in[n]} P_{ij}^{ab}\, v_j \ + \ r_i^{ab} \right) \ . \tag{4.13}$$

Then, using the finiteness of the sets $\mathcal{A}_i$, and

$$[f(t\eta + v)]_i = F(t\eta + v; i) \ = \min_{a\in\mathcal{A}_i} \ F(t\eta + v; i, a) \ ,$$

one obtains Equation (4.9).

From (4.9), we deduce easily that (η, v) is an invariant half-line of f if, and only if, it satisfies:

$$\begin{cases} \eta \ &= \hat{f}(\eta) \ , \\ \eta + v \ &= \acute{f}_\eta(v) \ . \end{cases} \tag{4.14}$$

This couple system of equations is what is solved in practice, when one looks for the value function $\rho = \eta$ of the mean payoff game.

4.3 Reduced super-harmonic vectors

We next present the non-linear analogue of a result of classical potential theory, on which the policy iteration algorithm for mean payoff games relies. Recall that a self-map f of $\mathbb{R}^n$ is order-preserving if $v \leq w \implies f(v) \leq f(w)$, where $\leq$ denotes the partial ordering of $\mathbb{R}^n$, and that it is additively homogeneous if it commutes with the addition of a constant vector. More generally, it is *additively subhomogeneous*, if $f(\lambda + v) \leq \lambda + f(v)$ for all $\lambda \geq 0$ and $v \in \mathbb{R}^n$. It is easy to see that an order-preserving self-map f of $\mathbb{R}^n$ is additively subhomogeneous if, and only if, it is nonexpansive in the sup-norm. (See for instance [GG04] for more background on order-preserving additively homogeneous maps.)

We shall now recall some definitions and results of [AG03], where the corresponding proofs can be found, up to an extension from additively homogeneous maps to subhomogeneous maps as in [AG03, §1.4]. To show the analogy with potential theory, we shall say that a vector $u \in \mathbb{R}^n$ is *harmonic* with respect to an order preserving, additively (sub)homogeneous map g of $\mathbb{R}^n$ if it is a fixed point of g, i.e. if $g(u) = u$, and that it is *super-harmonic* if $g(u) \leq u$. ([AG03] deals more generally with additive eigenvectors and super-eigenvectors). We shall denote by $\mathscr{H}(g)$ and $\mathscr{H}^+(g)$ the set of harmonic and super-harmonic vectors respectively.

We say that a self-map g of $\mathbb{R}^n$ is *convex* if all its coordinates $g_i : \mathbb{R}^n \to \mathbb{R}$ are convex functions. Then, the *subdifferential* of g at a point $u \in \mathbb{R}^n$ is defined as

$$\partial g(u) := \left\{ M \in \mathbb{R}^{n \times n} \mid g(v) - g(u) \geq M(v-u),\ \forall v \in \mathbb{R}^n \right\} \ .$$

Hence,

$$\partial g(u) = \left\{ M \in \mathbb{R}^{n \times n} \mid M_{i\cdot} \in \partial g_i(u) \right\} \ , \tag{4.15}$$

where $M_{i\cdot}$ denotes the i-th row of the matrix M. It can be checked that when g is order-preserving and additively homogeneous (resp. subhomogeneous), $\partial g(u)$ consists of stochastic (resp. substochastic) matrices, that is matrices with nonnegative entries and row sums equal to 1 (resp. less or equal to 1), see [AG03, Cor. 2.2 and (4)]. Assume g has a harmonic vector u. We say that a node is *critical* if it belongs to a recurrence class of some matrix $M \in \partial g(u)$, where a recurrence class of M means a (final) communication class F of M such that the $F \times F$ submatrix of M is stochastic (note that a recurrence class may not exist if g is not additively homogeneous), see [AG03, §2.3 and 1.4]. One defines also the critical graph $\mathcal{G}^{\mathrm{c}}(g)$ of g as the union of the graphs of the $F \times F$ submatrices of the matrices $M \in \partial g(u)$, such that F is a recurrence class of M. The set of critical nodes and the critical graph of g are independent of the choice of the harmonic vector u [AG03, Prop. 2.5]. Indeed, when g arises from a stochastic control problem with ergodic reward, a node is critical iff it is recurrent for some stationary optimal strategy.

If I is any subset of $[n]$, we denote by r_I the restriction from $\mathbb{R}^n$ to $\mathbb{R}^I$, such that $(r_I v)_i := v_i$, for all $i \in I$. For all $u \in \mathbb{R}^n$, we define $u_I := r_I u$, and for all self-maps g of $\mathbb{R}^n$, we define $g_I := r_I \circ g$. Let $J := [n] \setminus I$. We denote by $\imath_I$ the canonical map identifying $\mathbb{R}^I \times \mathbb{R}^J$ to $\mathbb{R}^n$, which sends (w, z) to the vector u such that $u_i = w_i$ for all $i \in I$ and

$u_i = z_i$ for all $i \in J$. Then, the transpose r_I^* of r_I is the map from $\mathbb{R}^I$ to $\mathbb{R}^n$ such that $r_I^*(w) = \imath_I(w, 0)$. Finally, for all $I, J \subset [n]$, and for all $n \times n$ matrices M, we denote by M_{IJ} the $I \times J$ submatrix of M.

Lemma 4.1. *Let g denote a convex, order preserving, and additively homogeneous self-map of $\mathbb{R}^n$. Assume that $u \in \mathbb{R}^n$ is harmonic with respect to g. Denote by C the set of critical nodes of g and by $N = [n] \setminus C$ its complement in $[n]$. Then, the map $h : \mathbb{R}^N \to \mathbb{R}^N$ with $h(w) := (r_N \circ g \circ \imath_N)(w, u_C)$ has a unique fixed point.*

Proof. Since the map g is order preserving and additively homogeneous, it is nonexpansive in the sup-norm, and so, the map h is also order preserving and nonexpansive in the sup-norm, hence it is additively subhomogeneous. Since u is harmonic with respect to g, that is a fixed point of g, u_N is a fixed point of the map h. A classical result of convex analysis (Theorem 23.9 of [Roc70]) shows in particular that if G is a finite valued convex function defined on $\mathbb{R}^d$, if A is a linear map $\mathbb{R}^p \to \mathbb{R}^d$, and if $H(v) := G(Av)$, then, $\partial H(v) = A^* \partial G(Av)$. Applying this result to every convex map G_i defined on $\mathbb{R}^n$ such that $G_i(w) := g_i(w + \imath_N(0, u_C))$, with $i \in N$, and to the linear map $A = r_N^*$, we deduce that $\partial h_i(u_N)$ is the projection on $\mathbb{R}^N$ of the subdifferential of G_i at the point $r_N^*(u_N)$, or equivalently of the subdifferential of g_i at the point $r_N^*(u_N) + \imath_N(0, u_C) = \imath(u_N, u_C) = u$. Using (4.15), this implies that $\partial h(u_N) = \{M_{NN} \mid M \in \partial g(u)\}$. Since g is order preserving and additively homogeneous, the elements of $\partial g(u)$ are stochastic matrices, and by the above equality, or since h is order preserving and additively subhomogeneous, the elements of $\partial h(u_N)$ are substochastic matrices. Recall that the set of critical nodes of h is defined as the set of nodes that belong to a final class F of some matrix $P \in \partial h(u_N)$ satisfying that P_{FF} is stochastic. Denote by F such a class. We have $F \subset N$. Moreover, since $\partial h(u_N) = \{M_{NN} \mid M \in \partial g(u)\}$, we can find a matrix $Q \in \partial g(u)$ the $N \times N$ submatrix of which, Q_{NN}, coincides with P. Since $F \subset N$, Q_{FF} coincides with P_{FF}. Hence Q_{FF} is a stochastic matrix, which implies that F is a recurrent class of Q. This shows that the nodes of F are critical nodes of g, which contradicts the fact that the set of critical nodes is C since $F \subset N = [n] \setminus C$. Therefore the set of critical nodes of h is empty. It follows from Corollary 1.3 of [AG03] that h has a unique fixed point. □

We shall need the following result of [AG03].

Lemma 4.2 ([AG03, (7) and Lemma 3.3])**.** *Let g be a convex order-preserving additively homogeneous self-map of $\mathbb{R}^n$, with at least one harmonic vector. Denote by C the set of critical nodes. If u is super-harmonic with respect to g, then $g(u) = u$ on C, and $g^\omega(u) := \lim_{k\to\infty} g^k(u)$ exists, is harmonic with respect to g and coincides with u on C. Moreover, the map $g^\omega : \mathscr{H}^+(g) \to \mathscr{H}(g)$ is order-preserving, additively homogeneous, convex, and is a projector.*

The following result gives other characterizations of $g^\omega(u)$ that allows one to compute it efficiently.

Theorem 4.3. *Let g denote a convex, order preserving, and additively homogeneous self-map of $\mathbb{R}^n$. Assume that g admits at least one harmonic vector. Let C denote the set of critical nodes of g, and let N denote its complement in $[n]$, $N = [n] \setminus C$. For a super-harmonic vector u, the following conditions define uniquely the same vector v:*

(i) $v = g^\omega(u) := \lim_{k\to\infty} g^k(u)$;

(ii) *v is harmonic and coincides with u on C;*

(iii) *v coincides with u on C and its restriction to N is a fixed point of the map $h : w \mapsto (r_N \circ g \circ \imath_N)(w, u_C)$;*

(iv) *v is the smallest super-harmonic vector that dominates u on C.*

Proof. (i)⇒(ii): This follows from Lemma 4.2.
(ii)⇒(iii): Assume that the vector v is harmonic and coincides with u on C and let h be defined as in Point (iii). Then, $v_N = h(v_N)$, showing that v_N is a fixed point of h.
(iii)⇒(i): Let v and h be as in Point (iii), hence $v_C = u_C$ and v_N is a fixed point of h. By Lemma 4.2, $w := g^\omega(u)$ is harmonic with respect to g and $w_C = u_C$. Applying Lemma 4.1 to g and w (instead of u), and using $w_C = u_C$, we get that the fixed point of h is unique, and thus equal to w_N. This shows that $v_N = w_N$, and since $v_C = u_C = w_C$, we get that $v = w = g^\omega(u)$, that is Point (i).
(ii)⇒(iv): Let v be as in Point (ii). Since v is harmonic and coincides with u on C, it is super-harmonic and dominates u on C. By ((ii)⇒(iii)), v_N is a fixed point of h, with h as in Point (iii). Assume now that w is super-harmonic and dominates u on C, that is $w_C \geq u_C$. Then, $w \geq g(w)$, and since g is order preserving, $w_N \geq g_N(w_N, w_C) \geq g_N(w_N, u_C) = h(w_N)$. Since h is order-preserving, we deduce from $w_N \geq h(w_N)$ that $w_N \geq h^1(w_N) \geq h^2(w_N) \geq \cdots$. Since h is nonexpansive and admits a fixed point, every orbit of h is bounded. Hence, $h^k(w_N)$ has a limit as k tends to infinity, and this limit is a fixed point of h. Applying Lemma 4.1 to g and v (instead of u), and using $v_C = u_C$, we get that the fixed point of h is unique and equal to v_N. It follows that $w_N \geq v_N$. Since v coincides with u on C and $w_C \geq u_C$, we deduce that $w \geq v$. This shows that v is the smallest super-harmonic vector that dominates u on C.
(iv)⇒(ii): Let v be a minimal super-harmonic vector that dominates u on C (or the smallest one if it exists). Since v is a super-harmonic vector, that is $g(v) \leq v$, and g is order-preserving, we get that $g(g(v)) \leq g(v)$, which shows that $g(v)$ is also super-harmonic. Moreover, by Lemma 4.2, $g(v)$ coincides with v on C, hence it dominates u on C. Since $g(v) \leq v$, the minimality of v implies $g(v) = v$, which shows that v is harmonic. Since u and v are super-harmonic vectors and g is order-preserving, we get that the infimum $v \wedge u$ of v and u is also a super-harmonic vector. Since v dominates u on C, we get that $v \wedge u$ equals u on C. Hence by the minimality of v, and $v \wedge u \leq v$, we obtain that $v = v \wedge u$, hence $v \leq u$. This implies that v equals u on C, hence v satisfies (ii). □

Let g^ω be defined as in Theorem 4.3. When $g(v) = Mv$ is a linear operator, and M is a stochastic matrix, $g^\omega(u)$ coincides with the *reduced* super-harmonic vector of u with

respect to the set C. When g is a max-plus linear operator, the operator g^ω coincides with the *spectral projector* which has been defined in the max-plus literature, see [CTGG99]. For this reason, we call g^ω the (nonlinear) *spectral projector* of g.

We now define a spectral projector acting on half-lines. We assume that g is a polyhedral, convex, order preserving, and additively homogeneous self-map of $\mathbb{R}^n$. This implies in particular that for all $i \in [n]$, the domain of the Legendre-Fenchel transform g_i^* of the coordinate g_i of g is included in the set of stochastic vectors, and that g_i is the Legendre-Fenchel transform of g_i^*, hence can be put in the same form as in (4.6):

$$g_i(v) \;=\; \max_{b\in\mathcal{B}_i} \left(\sum_{j\in[n]} P_{ij}^b \, v_j \;+\; r_i^b \right) \;\;, \tag{4.16}$$

where, for all $i \in [n]$, $P_{i\cdot}^b \in \mathbb{R}^n$ is a stochastic vector, $r_i^b \in \mathbb{R}$, and B_i is the domain of g_i^*, see [AG03, Prop. 2.1 and Cor. 2.2]. Since the map g_i is polyhedral, the domain of g_i^* is also a polyhedral convex set, see [Roc70, Th. 19.2], and since it is included in the set of stochastic vectors, it is compact, hence it is the convex envelope of the finite set of its extremals. Then, in (4.16), B_i can be replaced by this finite set.

Since g is polyhedral, order preserving, and additively homogeneous, we get by Kohlberg theorem [Koh80] recalled in Section 4.2, that g has an invariant half-line (η, v), η is necessarily equal to $\chi(g)$, and by (4.14), v and η satisfy $\eta = \hat{g}(\eta)$ and $\eta + v = \acute{g}_\eta(v)$, where $\hat{g}$ and $\acute{g}_\eta$ are defined in (4.10) and (4.11a) respectively. When g is given by (4.16), these maps can be rewritten as:

$$[\hat{g}(\eta)]_i = \max_{b\in\mathcal{B}_i} \left(\sum_{j\in[n]} P_{ij}^b \, \eta_j \right), \quad i \in [n] \;\;, \tag{4.17}$$

and

$$[\acute{g}_\eta(v)]_i = \max_{b\in\acute{\mathcal{B}}_{i,\eta}} \left(\sum_{j\in[n]} P_{ij}^b \, v_j \;+\; r_i^b \right) \;\;, \tag{4.18a}$$

$$\acute{\mathcal{B}}_{i,\eta} := \operatorname*{argmax}_{b\in\mathcal{B}_i} \left\{ \sum_{j\in[n]} P_{ij}^b \, \eta_j \right\} \;\;. \tag{4.18b}$$

Let us fix an invariant half-line (η, v) of g. Denote $\bar{g}(w) := \acute{g}_\eta(w) - \eta$, then v is harmonic with respect to $\bar{g}$: $\bar{g}(v) = v$. We define the set of *critical nodes* of g, $C(g)$, to be the set of critical nodes of $\bar{g}$. A half-line $w : t \mapsto t\eta + v$ is *super-invariant* if $g \circ w(t) \leq w(t+1)$, for t large enough. From (4.9), this property is equivalent to the conditions $\eta \geq \hat{g}(\eta)$ with $v_i \geq \bar{g}_i(v)$ when $\eta_i = \hat{g}_i(\eta)$. In particular when the equality $\eta = \hat{g}(\eta)$ holds, it is equivalent to $v \geq \bar{g}(v)$.

Corollary 4.4. *Assume that g is a polyhedral, convex, order preserving, and additively homogeneous self-map of $\mathbb{R}^n$. Assume that $w : t \mapsto t\eta + v$ is a super-invariant half-line of g with $\eta = \chi(g)$. Then, there exists a unique invariant half-line of g which coincides with w on the set of critical nodes of g. It is given by $t \mapsto t\eta + \bar{g}^\omega(v)$, where $\bar{g} : w \mapsto \acute{g}_\eta(w) - \eta$.*

Proof. As said above, an invariant half-line of g must be of the form $t \mapsto t\eta + z$, where $\eta = \chi(g)$ and $z \in \mathbb{R}^n$ is a fixed point of $\bar{g}$. If $w : t \mapsto t\eta + v$ is a super-invariant half-line of g with $\eta = \chi(g)$, then $\eta = \hat{g}(\eta)$, and by (4.9), we get $v \geq \bar{g}(v)$. From this, we deduce that $t \mapsto t\eta + z$ is an invariant half-line of g which coincides with w on C, if and only if z is harmonic with respect to $\bar{g}$ and coincides with v on C. By Theorem 4.3, $\bar{g}^\omega(v)$ is such a harmonic vector, and it is the unique one. The corollary follows. □

For any super-invariant half-line w of g with $\eta = \chi(g)$, we define $g^\omega(w)$ to be the half-line $t \mapsto t\eta + \bar{g}^\omega(v)$.

4.4 Policy iteration algorithm for stochastic mean payoff games

The following policy iteration scheme was introduced by Cochet-Terrasson and Gaubert in [CTG06]. We first give, in Algorithm 4.1, an abstract formulation of the algorithm similar to the one given in [CTG06], which is convenient to establish its convergence. A detailed practical algorithm will follow and the proof of the convergence of the algorithm will be given in the last subsection.

4.4.1 The theoretical algorithm

In order to present the algorithm, we assume that every coordinate of $f : \mathbb{R}^n \to \mathbb{R}^n$ is given by:

$$f_i(v) = \min_{a \in A_i} f_i^a(v) \ , \tag{4.19}$$

where A_i is a finite set, and f_i^a is a polyhedral order preserving, additively homogeneous, and convex map from $\mathbb{R}^n$ to $\mathbb{R}$. These conditions all together are indeed equivalent to the property that f is of the form (4.6,4.7), since as already observed any polyhedral order preserving, additively homogeneous and convex map g from $\mathbb{R}^n$ to $\mathbb{R}$ can be put in the form (4.16), with B_i a finite set. For all feedback strategies $\alpha \in \mathcal{A}_M = \{\alpha : [n] \to A,\ i \mapsto \alpha(i) \in A_i\}$, we denote by $f^{(\alpha)}$ the self-map of $\mathbb{R}^n$ the i-th coordinate of which is given by $f_i^{(\alpha)} = f_i^{\alpha(i)}$.

Algorithm 4.1 (Policy iteration for multichain mean payoff two player games [CTG06])**.**

Input*: A map f the coordinates of which are of the form* (4.19)*.*
Output*: An invariant half-line $w : t \mapsto t\eta + v$ of f and an optimal policy $\alpha \in \mathcal{A}_M$.*

1. Initialization*: Set $k = 0$. Select an arbitrary strategy $\alpha^{(0)} \in \mathcal{A}_M$. Compute an invariant half-line of $f^{(\alpha^{(0)})}$, $w^{(0)} : t \mapsto t\eta^{(0)} + v^{(0)}$.*
2. *If $f \circ w^{(k)}(t) = w^{(k)}(t+1)$ holds for t large enough, the algorithm stops and returns $w^{(k)}$ and $\alpha^{(k)}$.*

3. *Otherwise,* improve the strategy $\alpha^{(k)}$ *for* $w^{(k)}$*, by selecting a strategy* $\alpha^{(k+1)}$ *such that* $f \circ w^{(k)}(t) = f^{(\alpha^{(k+1)})} \circ w^{(k)}(t)$*, for* t *large enough. The choice of* $\alpha^{(k+1)}$ *must be conservative, meaning that, for all* $i \in [n]$*,* $\alpha^{(k+1)}(i) = \alpha^{(k)}(i)$ *if* $f_i \circ w^{(k)}(t) = f_i^{(\alpha^{(k)})} \circ w^{(k)}(t)$*, for* t *large enough.*
4. *Compute an arbitrary invariant half-line* $w'(t) : t \mapsto t\eta^{(k+1)} + v'$ *of* $f^{(\alpha^{(k+1)})}$*. If* $\eta^{(k+1)} \neq \eta^{(k)}$ *then set* $v^{(k+1)} = v'$*, i.e.* $w^{(k+1)} = w'$*, and go to step 6. Otherwise* $(\eta^{(k+1)} = \eta^{(k)})$*, we say that the iteration is* degenerate*.*
5. *Compute the invariant half-line* $w^{(k+1)} = (f^{(\alpha^{(k+1)})})^{\omega}(w^{(k)})$ *of* $f^{(\alpha^{(k+1)})}$*, and define* $v^{(k+1)}$ *and* $\eta^{(k+1)}$ *by* $w^{(k+1)}(t) = t\eta^{(k+1)} + v^{(k+1)}$*.*
6. *Increment* k *by one and go to step 2.*

Let us give some details about the well posedness of this algorithm. First, the existence of the invariant half-lines in Steps 1 and 4 follows from Kohlberg theorem [Koh80] applied to the polyhedral order preserving additively homogeneous maps $f^{(\alpha^{(k)})}$ with $k \geq 1$. Second, due to the finiteness of the action sets A_i and the fact that the maps f_i^a are polyhedral, the maps f and f_i^a can be rewritten in the form (4.9). Hence, the test of Step 2 and the asymptotic optimization problem of Step 3 can be rewritten as an equality test for (germs of) half-lines and the pointwise minimization of a finite set of half-lines, which are transformed into systems of equations and lexicographical optimization problems, using the representation of half-lines as couples (η, v) instead of maps $w : t \mapsto t\eta + v$, see the following section for details.

Finally, at each iteration k of Algorithm 4.1, $w^{(k)} : t \mapsto t\eta^{(k)} + v^{(k)}$ is a super-invariant half-line of $f^{(\alpha^{(k+1)})}$. Indeed, by construction of $\alpha^{(k+1)}$, and since $w^{(k)}$ is an invariant half-line of $f^{(\alpha^{(k)})}$, we get

$$f^{(\alpha^{(k+1)})}(w^{(k)}(t)) = f(w^{(k)}(t)) \leq f^{(\alpha^{(k)})}(w^{(k)}(t)) = w^{(k)}(t+1) \ , \tag{4.20}$$

for t large enough. Moreover, since $w^{(k)}$ is an invariant half-line of $f^{(\alpha^{(k)})}$, we have $\chi(f^{(\alpha^{(k)})}) = \eta^{(k)}$. Hence, in Step 5, $w^{(k)}$ is a super-invariant half-line of $f^{(\alpha^{(k+1)})}$ with slope $\eta^{(k)}$ equal to $\eta^{(k+1)} = \chi(f^{(\alpha^{(k+1)})})$. By Corollary 4.4, there exists a unique invariant half-line of $f^{(\alpha^{(k+1)})}$ which coincides with $w^{(k)}$ on the set of critical nodes of $f^{(\alpha^{(k+1)})}$ and it is given by $w^{(k+1)} = (f^{(\alpha^{(k+1)})})^{\omega}(w^{(k)}) : t \mapsto t\eta^{(k+1)} + v^{(k+1)}$ with $v^{(k+1)} = \left(\bar{f}^{(\alpha^{(k+1)})}\right)^{\omega}(v^{(k)})$. Practical computations are detailed in the following sections.

4.4.2 The practical algorithm

All the steps of Algorithm 4.1 involve equality tests or pointwise minimizations of half-lines. However, it would not be robust to do these tests on half-lines just by choosing an arbitrary large number t in the equations and inequations to be solved. We shall rather use the equivalence between the representation of a half-line as a map $w : t \mapsto t\eta + v$ with t large and that as a couple (η, v). This allows one to transform all the tests into systems of equations or optimizations of finite sets of half-lines for the pointwise lexicographic

order (which is linear, for each coordinate). This means that we are solving the system of equations (4.14). Then, using the notations of Section 4.2, the corresponding practical algorithm of the formal Algorithm 4.1 is given below in Algorithm 4.2.

Algorithm 4.2 (Policy iteration for multichain mean payoff two player games)**.**

Input*: A map f the coordinates of which are of the form* (4.19) *and the notations* (4.7,4.6) *and* (4.11–4.13)*.*

Output*: An invariant half-line (η, v) of f and an optimal policy $\alpha \in \mathcal{A}_M$.*

1. Initialization*: Set $k = 0$. Select an arbitrary strategy $\alpha^{(0)} \in \mathcal{A}_M$. Compute the couple $(\eta^{(0)}, v^{(0)})$ solution of*

$$\left\{ \begin{array}{rcl} \eta_i^{(0)} & = & \hat{F}(\eta^{(0)}; i, \alpha^{(0)}(i)) \\ \eta_i^{(0)} + v_i^{(0)} & = & \acute{F}_{\eta^{(0)}}(v^{(0)}; i, \alpha^{(0)}(i)) \end{array} \right. \quad \text{for all } i \in [n] \ . \tag{4.21}$$

2. *If $\eta^{(k)}$ and $v^{(k)}$ satisfy System* (4.14)*, or equivalently if $\alpha^{(k+1)} = \alpha^{(k)}$ is solution of* (4.22) *below, then the algorithm stops and returns $(\eta^{(k)}, v^{(k)})$ and $\alpha^{(k)}$.*

3. *Otherwise, improve the policy $\alpha^{(k)} \in \mathcal{A}_M$ for $(\eta^{(k)}, v^{(k)})$ in a conservative way, that is choose $\alpha^{(k+1)} \in \mathcal{A}_M$ such that*

$$\left\{ \begin{array}{l} \alpha^{(k+1)}(i) \ \in \ \underset{a \in \acute{\mathcal{A}}_{i,\eta^{(k)}}}{\operatorname{argmin}} \left\{ \acute{F}_{\eta^{(k)}}(v^{(k)}; i, a) \right\} \\ \alpha^{(k+1)}(i) = \alpha^{(k)}(i) \text{ if } \alpha^{(k)}(i) \text{ is optimal,} \end{array} \right. \quad \text{for all } i \in [n] \ . \tag{4.22}$$

4. *Compute a couple $(\eta^{(k+1)}, v')$ for policy $\alpha^{(k+1)}$ solution of*

$$\left\{ \begin{array}{rcl} \eta_i^{(k+1)} & = & \hat{F}(\eta^{(k+1)}; i, \alpha^{(k+1)}(i)) \\ \eta_i^{(k+1)} + v'_i & = & \acute{F}_{\eta^{(k+1)}}(v'; i, \alpha^{(k+1)}(i)) \end{array} \right. \quad \text{for all } i \in [n] \ . \tag{4.23}$$

   If $\eta^{(k+1)} \neq \eta^{(k)}$ then set $v^{(k+1)} = v'$ and go to step 6. Otherwise, the iteration is degenerate.

5.*i) Let $g := f^{(\alpha^{(k+1)})}$ ($g_i = F(\cdot; i, \alpha^{(k+1)}(i))$). Compute $C(g)$ the set of critical nodes of the map $\bar{g}$ defined by : $\bar{g} = \acute{g}_{\eta^{(k+1)}}(\cdot) - \eta^{(k+1)}$, or equivalently:*

$$\bar{g}_i(v) \ = \ \acute{F}_{\eta^{(k+1)}}(v; i, \alpha^{(k+1)}(i)) - \eta_i^{(k+1)} \quad \text{for all } i \in [n] \ ,$$

   for which v' is a harmonic vector.

5.*ii) Compute $v^{(k+1)} = \bar{g}^{\omega}(v^{(k)})$, that is the solution of:*

$$\left\{ \begin{array}{rcll} v_i^{(k+1)} & = & \acute{F}_{\eta^{(k+1)}}(v^{(k+1)}; i, \alpha^{(k+1)}(i)) - \eta_i^{(k+1)} & i \in [n] \setminus C(g) \\ v_i^{(k+1)} & = & v_i^{(k)} & i \in C(g) \ . \end{array} \right. \tag{4.24}$$

6. *Increment k by one and go to Step 2.*

It remains to precise how the steps are performed. Step 3 is just composed of lexicographic optimization problems in finite sets. The systems (4.21) and (4.23) are the dynamic programming equations of a one player multichain mean payoff game, they can be computed by applying the policy iteration algorithm for multichain Markov decision processes with mean payoff introduced by Howard [How60] and Denardo and Fox [DF68]. Note that one can also choose to solve Systems (4.21) and (4.23) by applying Algorithm 4.2 to the maps $h = f^{(\sigma_0)}$ and $h = f^{(\alpha^{(k+1)})}$ respectively, while replacing minimizations by maximizations, but in that case the algorithm is almost equivalent to that of Howard [How60] and Denardo and Fox [DF68], see Section 4.5 below. In Step 5, the set of critical nodes of g, that is that of $\bar{g}$, can be computed using a variant of the algorithm proposed in [AG03, § 6.3] described in Section 4.5.3. Finally, System (4.24) is the dynamic programming equation of an optimal control problem with infinite horizon stopped when reaching the set $C(g)$ which can be solved using the original policy iteration algorithm of Howard [How60]. We shall recall all these algorithms in Section 4.5.

4.4.3 Convergence of the algorithm

In this subsection, we show in Theorem 4.8 that Algorithm 4.1, or equivalently Algorithm 4.2 terminates after a finite number of steps. This result is proved using Theorem 4.3. Let first show some intermediate results.

The following lemma is known, see for instance Sorin [Sor04].

Lemma 4.5 (See [Sor04])**.** *Let g denote an order preserving self-map of $\mathbb{R}^n$, that is nonexpansive in the sup-norm, and has a cycle time $\chi(g)$. If $w : t \mapsto t\eta + v$ is a super-invariant half-line of g, then, $\chi(g) \leq \eta$.*

Proof. We reproduce the argument, for completeness: if $w : t \mapsto t\eta + v$ is a super-invariant half-line of g, that is $g(w(t)) \leq w(t+1)$ for $t \geq t_0$ for some $t_0 \geq 0$, then, $g^k(w(t)) \leq w(t+k)$, for all $k \geq 0$, and $t \geq t_0$, and so $\chi(g) \leq \lim_{k\to\infty} w(t_0+k)/k = \eta$, which shows Lemma 4.5. □

Since, by (4.20), $w^{(k)}$ is a super-invariant half-line of $f^{(\alpha^{(k+1)})}$, with slope $\eta^{(k)} = \chi(f^{(\alpha^{(k)})})$, it follows from Lemma 4.5 that :

Lemma 4.6. *The sequence of strategies defined in Algorithm 4.1 is such that*

$$\chi(f^{(\alpha^{(k+1)})}) \leq \chi(f^{(\alpha^{(k)})}) \ .$$

We now examine degenerate iterations.

Lemma 4.7. *Let $(\alpha^{(k)})_{k\geq 1}$ be the sequence of strategies defined in Algorithm 4.1, and assume that $\chi(f^{(\alpha^{(k+1)})}) = \chi(f^{(\alpha^{(k)})})$. Then, the following statements hold.*

1. *The half-line $w^{(k+1)}$ agrees with $w^{(k)}$ on the set of critical nodes of $f^{(\alpha^{(k+1)})}$.*
2. *Every critical node of $f^{(\alpha^{(k+1)})}$ is a critical node of $f^{(\alpha^{(k)})}$.*

3. $w^{(k+1)} \leq w^{(k)}$.

Proof. Let us use the notations: $g := f^{(\alpha^{(k+1)})}$ (as in Algorithm 4.2) and $h = f^{(\alpha^{(k)})}$. By construction and assumption, we have $\eta^{(k)} = \chi(h) = \chi(g) = \eta^{(k+1)}$, that we shall also denote by η.
Point 1: Since, by (4.20), $w^{(k)}$ is a super-invariant half-line of g, with slope $\eta^{(k)} = \chi(g)$, and since $w^{(k+1)}$ is defined as $g^\omega(w^{(k)})$, the result follows from Corollary 4.4.
Point 2: Again, since $w^{(k)}$ is a super-invariant half-line of g, with slope $\eta^{(k)} = \chi(g)$, we deduce from the definition of $\bar{g}$ and (4.9), that

$$\bar{g}(v^{(k)}) \leq v^{(k)} \ . \tag{4.25}$$

Then by Lemma 4.2, $\bar{g}(v^{(k)})$ agrees with $v^{(k)}$ on $C(\bar{g}) = C(g)$, the set of critical nodes of g, and so, the equality $g(w^{(k)}(t)) = w^{(k)}(t+1)$ holds on $C(g)$ for t large. Since $w^{(k)}$ is an invariant half line of $f^{(\alpha^{(k)})}$, we get that $f_i(w^{(k)}(t)) = f_i^{(\alpha^{(k+1)})}(w^{(k)}(t)) = w^{(k)}(t+1) = f_i^{(\alpha^{(k)})}(w^{(k)}(t))$ for t large enough and $i \in C(g)$. Hence, the conservative selection rule ensures that $\alpha^{(k+1)}(i) = \alpha^{(k)}(i)$ for all $i \in C(g)$. This implies that $g_i = h_i$ for all $i \in C(g)$, and since $\chi(g) = \chi(h)$, we get from the definitions of $\bar{g}$ and $\bar{h}$ that

$$\bar{g}_i = \bar{h}_i \qquad \text{for all } i \in C(g) \ . \tag{4.26}$$

Observe that $v^{(k+1)}$ is a fixed-point of $\bar{g}$, and that $\bar{g}$ is a polyhedral additively homogeneous order preserving convex selfmap of $\mathbb{R}^n$. Hence the critical nodes of $\bar{g}$ are the indices that belong to a final class of a matrix $M \in \partial\bar{g}(v^{(k+1)})$ (since the elements of $\bar{g}(v^{(k+1)})$ are stochastic matrices, all their final classes are recurrent). Let F be such a final class. From (4.15), the line $M_{i\cdot} \in \partial\bar{g}_i(v^{(k+1)})$ for $i \in F$, that is $\bar{g}_i(v) - \bar{g}_i(v^{(k+1)}) \geq M_{i\cdot}(v - v^{(k+1)})$ for all $v \in \mathbb{R}^n$. Since $v^{(k+1)}$ is a fixed point of $\bar{g}$, $v^{(k)}$ a fixed point of $\bar{h}$, and $v^{(k+1)}$ agrees with $v^{(k)}$ on $C(g)$ (from Point 1), we get that $\bar{g}_i(v^{(k+1)}) = v_i^{(k+1)} = v_i^{(k)} = \bar{h}_i(v^{(k)})$ for all $i \in C(g)$. From (4.26), we deduce that $\bar{g}_i(v) - \bar{g}_i(v^{(k+1)}) = \bar{h}_i(v) - \bar{h}_i(v^{(k)})$ for all $i \in C(g)$ and $v \in \mathbb{R}^n$. Now, since F is a final class of M, hence $F \subset C(g)$, and $M_{ij} = 0$ for $i \in F$ and $j \notin C(g)$, we get that $M_{i\cdot}v^{(k+1)} = M_{i\cdot}v^{(k)}$ for $i \in F$. This implies that $\bar{h}_i(v) - \bar{h}_i(v^{(k)}) \geq M_{i\cdot}(v - v^{(k)})$ for all $v \in \mathbb{R}^n$ and $i \in F$, which shows that $M_{i\cdot} \in \partial\bar{h}_i(v^{(k)})$ for $i \in F$. Let $N := [n] \setminus F$ and define the matrix Q such that $Q_{i\cdot} = M_{i\cdot}$ if $i \in F$, and $Q_{i\cdot}$ be any element of $\partial\bar{h}_i(v^{(k)})$ if $i \in N$, then $Q \in \partial\bar{h}(v^{(k)})$. Hence, the $F \times F$ submatrix of M is also a $F \times F$ submatrix of Q, and so F is a final class of Q. Since $v^{(k)}$ is a fixed point of $\bar{h}$, this implies that F is included in the set of critical nodes of $\bar{h}$, which is also by definition the set of critical nodes of h. This shows that all critical nodes of g are also critical nodes of h, and shows Point 2.
Point 3: From (4.25), we get that $\bar{g}(v^{(k)}) \leq v^{(k)}$, hence the sequence $\bar{g}^k(v^{(k)})$ is non-increasing and $\bar{g}^\omega(v^{(k)}) \leq v^{(k)}$. Since $\eta^{(k)} = \eta^{(k+1)}$, we get that $w^{(k+1)} = g^\omega(w^{(k)}) = t\eta^{(k)} + \bar{g}(v^{(k)}) \leq w^{(k)}$. □

Finally, we prove that the algorithm terminates.

Theorem 4.8. *A strategy cannot be selected twice in Algorithm 4.1, and so, the algorithm terminates after a finite number of iterations.*

Proof. Assume by contradiction that the same strategy is selected twice in Algorithm 4.1, that is $\alpha^{(s)} = \alpha^{(m)}$ for some iterations $1 \leq s < m$ of the algorithm before it stops. Then, $\chi(f^{(\alpha^{(s)})}) = \chi(f^{(\alpha^{(m)})})$ and since by Lemma 4.6, $\chi(f^{(\alpha^{(s)})}) \geq \chi(f^{(\alpha^{(s+1)})}) \geq \cdots \geq \chi(f^{(\alpha^{(m)})})$, we get the equality $\chi(f^{(\alpha^{(s)})}) = \chi(f^{(\alpha^{(s+1)})}) = \cdots = \chi(f^{(\alpha^{(m)})})$. Hence, by Lemma 4.7, Part 2, we have that $C(f^{(\alpha^{(m)})}) \subset C(f^{(\alpha^{(m-1)})}) \subset \cdots \subset C(f^{(\alpha^{(s)})})$ and since $\alpha^{(s)} = \alpha^{(m)}$, we get the equality $C(f^{(\alpha^{(m)})}) = C(f^{(\alpha^{(m-1)})}) = \cdots = C(f^{(\alpha^{(s)})})$. So by Lemma 4.7, Part 1, $w^{(s)}$ and $w^{(m)}$ are both invariant half-lines of $f^{(\alpha^{(s)})}$ with slope $\chi(f^{(\alpha^{(s)})})$, that agree on $C(f^{(\alpha^{(s)})})$. Hence by Corollary 4.4, $w^{(s)} = w^{(m)}$. Since by Lemma 4.7, Part 3, we have $w^{(s)} \geq w^{(s+1)} \geq \cdots \geq w^{(m)}$, it follows that $w^{(s)} = \cdots = w^{(m)}$. In particular, $w^{(s)} = w^{(s+1)}$. Hence, $w^{(s)}(t+1) = w^{(s+1)}(t+1) = f^{(\alpha^{(s+1)})} \circ w^{(s+1)}(t) = f^{(\alpha^{(s+1)})} \circ w^{(s)}(t) = f \circ w^{(s)}(t)$ for t large enough. It follows that $w^{(s)}$ is an invariant half-line of f, and so, the algorithm stops at step s, which contradicts the existence of iteration m, and so the same strategy cannot be selected twice in Algorithm 4.1.

Since the sets A_i are finite, the number of strategies (the elements of $\mathcal{A}_M$) is also finite, and since a strategy cannot be selected twice, Algorithm 4.1 stops after a finite number of iterations, that is bounded by the number of strategies. □

4.5 Ingredients of Algorithm 4.1 or 4.2: one player games algorithms

As said in Section 4.4.2, each basic step of the policy iteration algorithm for multichain mean payoff zero-sum two player games (Algorithm 4.1 or 4.2) concerns the solution of one player games, also called stochastic control problems or Markov decision processes, with finite state and action spaces: a mean payoff problem for Systems (4.21) and (4.23), an infinite horizon problem stopped at the boundary for System (4.24), and the set of critical nodes of the corresponding dynamic programming operator in Step 5. We recall here the policy iteration algorithm for solving stochastic control problems, with either infinite horizon or mean payoff, and the algorithm proposed in [AG03, § 6.3] for computing a critical graph, and explain how all these algorithms are applied in Algorithm 4.1 or 4.2. By doing so, we shall also see that the classical Howard / Denardo-Fox algorithm can be thought of as a special case of these algorithms, in which the second player has no choices of actions.

In all the section, we consider the following dynamic programming or Shapley operator of a one player game with finite state and action spaces: g is a map from $\mathbb{R}^n$ to itself, given by :

$$[g(v)]_i := \max_{b \in \mathcal{B}_i} G(v; i, b) \qquad \forall i \in [n],\ v \in \mathbb{R}^n \ , \tag{4.27}$$

where

$$G(v;i,b) \;=\; \sum_{j\in[n]} P^b_{ij}\, v_j \;+\; r^b_i \;\;, \tag{4.28}$$

the vectors $P^b_{i\cdot}$ are substochastic vectors, for all $i \in [n]$ and $b \in \mathcal{B}_i$, and $\mathcal{B}_i$ are finite sets, for all $i \in [n]$. Equivalently, g is a convex additively subhomogeneous order preserving polyhedral selfmap of $\mathbb{R}^n$.

Since player MIN does not exist, the set of feedback strategies for player MAX, $\mathcal{B}_M$, is given by $\mathcal{B}_M := \{\beta : [n] \to B \mid \beta(i) \in \mathcal{B}_i \,\forall i \in [n]\}$, where B contains all the sets $\mathcal{B}_i$. For each $\beta \in \mathcal{B}_M$, we denote by $g^{(\beta)}$ the self-map of $\mathbb{R}^n$ given by:

$$g^{(\beta)}_i(v) := G(v;i,\beta(i)) \qquad \forall i \in [n],\; v \in \mathbb{R}^n \;\;.$$

We also denote by $r^{(\beta)}$ the vector of $\mathbb{R}^n$ such that $r^{(\beta)}_i = r^{\beta(i)}_i$ and $P^{(\beta)}$ the $n \times n$ matrix such that $P^{(\beta)}_{ij} = P^{(\beta(i))}_{ij}$, then $g^{(\beta)} : v \mapsto P^{(\beta)}v + r^{(\beta)}$.

4.5.1 Policy iterations for one player games with discounted payoff

System (4.24) consists in finding the solution v of the equation $v = \bar{g}(v)$ with $v = u$ on $C(\bar{g})$ where $u \in \mathbb{R}^n$ is super-harmonic with respect to $\bar{g}$, $\bar{g}(u) \le u$, and g is as in (4.27) with (4.28). The solution v is thus the value of a one player game with infinite horizon stopped when reaching the set $C(\bar{g})$ whose transition probabilities are given by the P^b_{ij}, instantaneous reward is given by the r^b_i and final reward is given by u_i, when the game is in state $i \in C(\bar{g})$. This value function can be obtained using the classical policy iteration algorithm of Howard [How60] for a one player game. From Theorem 4.3, v is solution of the above equation, if and only if $v_C = u_C$ and v_N is a fixed point of the convex polyhedral additively subhomogeneous order preserving selfmap h of $\mathbb{R}^N$, with $C = C(\bar{g})$, $N = [n]\backslash C$, and h defined as in Theorem 4.3, Point (iii), with g replaced by $\bar{g}$. One can also consider the equivalent equation $v = h(v)$ with $h_i = \bar{g}_i$ for $i \in N$ and $h_i(v) = u_i$ for $i \in C$ and $v \in \mathbb{R}^n$. In that case, h is a convex polyhedral additively subhomogeneous order preserving selfmap of $\mathbb{R}^n$.

In these two settings, we need to solve an equation of the form $v = g(v)$, where g is of the form (4.27), and g has no critical node: $C(g) = \emptyset$. From [AG03, Corollary 1.3], g has a unique fixed point and all the maps $g^{(\beta)}$ with $\beta \in \mathcal{B}_M$ have a unique fixed point (since their critical nodes are necessarily critical nodes of g). The policy iteration algorithm of Howard applied to this equation is then given by Algorithm 4.3.

Algorithm 4.3 (Policy iteration of Howard [How60] for stochastic control problems)**.**

Input*: A map g of the form* (4.27) *with no critical node.*

Output*: The fixed point of g and an optimal policy $\beta \in \mathcal{B}_M$.*

1. Initialization*: Set $k = 0$. Select an arbitrary strategy $\beta^{(0)} \in \mathcal{B}_M$.*
2. *Compute the value of the game $v^{(k)}$ with fixed feedback strategy $\beta^{(k)}$, that is the solution of the linear system:*

$$v^{(k)} \;=\; g^{(\beta^{(k)})}(v^{(k)}) \;\;.$$

3. *If* $v^{(k)} = g(v^{(k)})$*, or equivalently if* $\beta^{(k+1)} = \beta^{(k)}$ *is solution of* (4.29) *below, then the algorithm stops and returns* $v^{(k)}$ *and* $\beta^{(k)}$*.*

4. *Otherwise, improve the policy* $\beta^{(k+1)} \in \mathcal{B}_M$ *for the value* $v^{(k)}$ *:*

$$\beta^{(k+1)}(i) \in \operatorname*{argmax}_{b \in \mathcal{B}_i} G(v^{(k)}; i, b) \quad \forall i \in [n]. \tag{4.29}$$

5. *Increment* k *by one and go to Step 2.*

It is known [How60] that $v^{(k+1)} \leq v^{(k)}$ and that the algorithm stops after a finite number of steps.

4.5.2 Policy iteration for multichain one player games

Consider a one player game with dynamic programming operator g given by (4.27) and mean payoff. Then, as explained in Section 4.2 in the more general two player case, the mean payoff of the game is the slope η of any invariant half line (η, v) of g, which is also any solution of the following couple system (see Equation (4.14)):

$$\begin{cases} \eta & = & \hat{g}(\eta) \\ \eta + v & = & \acute{g}_\eta(v) \ . \end{cases} \tag{4.30}$$

where $\hat{g}$ and $\acute{g}_\eta$ are defined in (4.10) and (4.11) respectively. In the present one player case, they are reduced to:

$$[\hat{g}(\eta)]_i := \max_{b \in \mathcal{B}_i} \hat{G}(\eta; i, b) \qquad \text{and} \qquad [\acute{g}_\eta(v)]_i := \max_{b \in \acute{\mathcal{B}}_{i,\eta}} G(v; i, b) \ , \tag{4.31}$$

with

$$\hat{G}(\eta; i, b) = \sum_{j \in [n]} P^b_{ij} \, \eta_j \qquad \text{and} \qquad \acute{\mathcal{B}}_{i,\eta} := \operatorname*{argmax}_{b \in \mathcal{B}_i} \left\{ \sum_{j \in [n]} P^b_{ij} \, \eta_j \right\} \ , \tag{4.32}$$

for all $\eta, v \in \mathbb{R}^n$, $i \in [n]$, $b \in B$. We refer also to [DF68, Put94] for the existence of solutions to System (4.30), and for the proof that η solution of this system is the mean payoff of the game in this one player context. The following algorithm for multichain mean payoff Markov decision processes was introduced by Howard [How60] and proved to converge by Denardo and Fox [DF68]:

Algorithm 4.4 (Policy iteration algorithm for multichain mean payoff one player games)**.**

Input*: A map* g *of the form* (4.27) *with* (4.28)*, and the notations* (4.31,4.32)*.*
Output*: An invariant half-line* (η, v) *of* g *and an optimal policy* $\beta \in \mathcal{B}_M$*.*

1. Initialization*: Set* $k = 0$*. Select an arbitrary strategy* $\beta^{(0)} \in \mathcal{B}_M$*.*

2. *For each final class F of $P^{(\beta^{(k)})}$, denote by i_F the minimal index of the elements of F, and define S as the set of all these indices i_F. Compute the couple $(\eta^{(k)}, v^{(k)})$ for policy $\beta^{(k)}$ solution of*

$$\left\{\begin{array}{rcll} \eta_i^{(k)} & = & \hat{G}(\eta^{(k)}; i, \beta^{(k)}(i)) & i \in [n] \setminus S \\ \eta_i^{(k)} + v_i^{(k)} & = & G(v^{(k)}; i, \beta^{(k)}(i)) & i \in [n] \\ v_i^{(k)} & = & 0 & i \in S \ . \end{array}\right. \tag{4.33}$$

3. *If $(\eta^{(k)}, v^{(k)})$ is solution of* (4.30)*, or equivalently if $\beta^{(k+1)} = \beta^{(k)}$ is solution of* (4.34) *below, then the algorithm stops and returns $(\eta^{(k)}, v^{(k)})$ and $\beta^{(k)}$.*

4. *Otherwise, improve the policy $\beta^{(k+1)} \in \mathcal{B}_M$ for $(\eta^{(k)}, v^{(k)})$ in a conservative way, that is choose $\beta^{(k+1)} \in \mathcal{B}_M$ such that :*

$$\left\{\begin{array}{ll} \beta^{(k+1)}(i) \ \in \ \underset{b \in \mathcal{B}_{i,\eta^{(k)}}}{\operatorname{argmax}} G(v^{(k)}; i, b) & \\ \beta^{(k+1)}(i) = \beta^{(k)}(i) \text{ if } \beta^{(k)}(i) \text{ is optimal,} & \end{array}\right. \quad \text{for all } i \in [n] \ . \tag{4.34}$$

5. *Increment k by one and go to Step 2.*

The justifications and details of Algorithm 4.4 can be found in [DF68, Put94] and are recalled in Section 4.8. Solving System (4.33) turns out to be a critical step. This can be optimized by exploiting the structure of the system, we discuss this issue in Section 4.8. As explained in Section 4.4.2, another way to solve a multichain mean payoff Markov decision process may be to use Algorithm 4.1 or 4.2 in the particular case of a one-player game, with maximizations instead of minimizations. In order to compare it with Algorithm 4.4, we rewrite below Algorithm 4.2 in that case, with the above notations. Note that in the one-player case, the map g of Step 5 of Algorithm 4.2 is affine, hence its critical graph reduces to the final graph of its tangent matrix.

Algorithm 4.5 (Specialization of Algorithm 4.2 to the one player case)**.**

Input*: A map g of the form* (4.27) *with* (4.28)*, and the notations* (4.31,4.32)*.*

Output*: An invariant half-line (η, v) of g and an optimal policy $\beta \in \mathcal{B}_M$.*

1. Initialization*: Set $k = 0$. Select an arbitrary strategy $\beta^{(0)} \in \mathcal{B}_M$. Compute the couple $(\eta^{(0)}, v^{(0)})$ solution of*

$$\left\{\begin{array}{rcl} \eta_i^{(0)} & = & \hat{G}(\eta^{(0)}; i, \beta^{(0)}(i)) \\ \eta_i^{(0)} + v_i^{(0)} & = & G(v^{(0)}; i, \beta^{(0)}(i)) \end{array}\right. \quad \text{for all } i \in [n] \ . \tag{4.35}$$

2. *If $\eta^{(k)}$ and $v^{(k)}$ satisfy System* (4.30)*, or equivalently if $\beta^{(k+1)} = \beta^{(k)}$ is solution of* (4.36) *below, then the algorithm stops and returns $(\eta^{(k)}, v^{(k)})$ and $\beta^{(k)}$.*

3. *Otherwise, improve the policy $\beta^{(k)} \in \mathcal{B}_M$ for $(\eta^{(k)}, v^{(k)})$ in a conservative way, that is choose $\beta^{(k+1)} \in \mathcal{B}_M$ such that*

$$\left\{\begin{array}{ll} \beta^{(k+1)}(i) \ \in \ \underset{b \in \mathcal{B}_{i,\eta^{(k)}}}{\operatorname{argmax}} G(v^{(k)}; i, b) & \\ \beta^{(k+1)}(i) = \beta^{(k)}(i) \text{ if } \beta^{(k)}(i) \text{ is optimal,} & \end{array}\right. \quad \text{for all } i \in [n] \ . \tag{4.36}$$

4. Compute a couple $(\eta^{(k+1)}, v')$ *for policy* $\beta^{(k+1)}$ *solution of*

$$\left\{ \begin{array}{rcl} \eta_i^{(k+1)} & = & \hat{G}(\eta^{(k+1)}; i, \beta^{(k+1)}(i)) \\ \eta_i^{(k+1)} + v'_i & = & G(v'; i, \beta^{(k+1)}(i)) \end{array} \right. \quad \text{for all } i \in [n] \ . \tag{4.37}$$

If $\eta^{(k+1)} \neq \eta^{(k)}$ *then set* $v^{(k+1)} = v'$ *and go to step 6. Otherwise, the iteration is degenerate.*

5.i) Compute C *the set of final nodes of the matrix* $P^{(\delta_{k+1})}$.

5.ii) Compute the solution $v^{(k+1)}$ *of:*

$$\left\{ \begin{array}{rcll} v_i^{(k+1)} & = & G(v^{(k+1)}; i, \beta^{(k+1)}(i)) - \eta_i^{(k+1)} & i \in [n] \setminus C \\ v_i^{(k+1)} & = & v_i^{(k)} & i \in C \ . \end{array} \right. \tag{4.38}$$

6. Increment k *by one and go to Step 2.*

Systems (4.35) and (4.37) are of the form:

$$\left\{ \begin{array}{rcl} \eta & = & P\,\eta \\ \eta + v & = & P\,v + r \ , \end{array} \right. \tag{4.39}$$

where $r = r^{(\delta)} \in \mathbb{R}^n$ and $P = P^{(\delta)}$ is a stochastic matrix, with $\delta = \beta^{(0)}$ or $\beta^{(k+1)}$. It can be shown that the solution η of such a system is unique, that one can eliminate for each final class F of P one of the equations $\eta_i = (P\eta)_i$ with index $i \in F$, and that v is defined up to an element of the kernel of $I - P$, the dimension of which is equal to the number of final classes of P. When this number is strictly greater than one, and $v^{(k+1)}$ is chosen to be any solution v' of (4.37) in Algorithm 4.5, the algorithm may cycle, see Section 4.6 for an example in the two player case. One way to handle this [DF68, Put94], is either to fix to zero the value of $\mu_F v$ for each invariant measure μ_F of P with support in a final class F of P, or to fix to zero the components of v with indices in some set S containing exactly one node of each final class of P. In these two cases, the solution v of (4.39) becomes unique. Moreover, if in Algorithm 4.5, (4.37) is combined with either the conditions $\mu_F v' = 0$ or the conditions $v'_S = 0$ with S chosen in a conservative way, that is such that the same index is chosen in F for iterations k and $k+1$, if F is a final class of $P^{(\beta^{(k+1)})}$ which is also a final class of $P^{(\beta^{(k)})}$, then $v' = v^{(k)}$ on the set of final nodes of $P^{(\beta^{(k+1)})}$ when $\eta^{(k+1)} = \eta^{(k)}$, which implies that $v' = v^{(k+1)}$, hence Step 5 of Algorithm 4.5 becomes useless. This shows that Algorithm 4.4 is equivalent to Algorithm 4.5, where (4.37) is combined with the conditions $v'_S = 0$, where S is the set of minimal indices of each final class of $P^{(\beta^{(k+1)})}$. In other words, Algorithm 4.4 is a particular realization of Algorithm 4.5, where one chooses one special solution $v' = v^{(k+1)}$ of (4.37) at each iteration of the algorithm, even when $\eta^{(k+1)} \neq \eta^{(k)}$. Denardo and Fox proved [DF68, Put94] that the sequence of couples $(\eta^{(k)}, v^{(k)})_{k \geq 1}$ of Algorithm 4.4 is non decreasing in a lexicographical order, meaning that $\eta^{(k+1)} \geq \eta^{(k)}$, with $v^{(k+1)} \geq v^{(k)}$ when $\eta^{(k+1)} = \eta^{(k)}$, and that Algorithm 4.4 stops after a finite number of iterations (when the sets of actions are finite). Indeed, the convergence of Algorithm 4.1, proved in Section 4.4.3, shows that this also holds for the little more general Algorithm 4.5.

4.5.3 Critical graph

When a degenerate iteration ($\eta^{(k+1)} = \eta^{(k)}$) occurs in Step 4 of Algorithm 4.1, one has to compute the critical nodes of $g := f^{(\alpha^{(k+1)})}$, that is that of $\bar{g}$. This can be done by applying the techniques of [AG03, § 6.3] , leading to Algorithm 4.6 below. More precisely, one applies first the followings steps to the map $\bar{g}$ and its harmonic vector v', then apply Algorithm 4.6.

Consider an additively homogeneous map g whose coordinates are defined as in (4.27) with (4.28), and u a harmonic vector of g. For any set $\mathcal{P}$ of stochastic matrices, we define $\mathcal{G}^{\mathrm{f}}(\mathcal{P})$ as the union of the graphs of the matrices M_{FF}, where $M \in \mathcal{P}$ and F is a final class of M. Define

$$\widetilde{B}_i = \{b \in \mathcal{B}_i \mid G(u; i, b) = u\} \quad \text{and} \quad \mathcal{P}_i = \left\{P^b_{i\cdot} \mid b \in \widetilde{B}_i\right\} \ . \tag{4.40}$$

Then, the critical graph of g is given by

$$\mathcal{G}^{\mathrm{c}}(g) = \mathcal{G}^{\mathrm{f}}(\partial g(u)), \quad \text{where} \quad \partial g(u) = \mathrm{co}(\mathcal{P}_1) \times \cdots \times \mathrm{co}(\mathcal{P}_n) \ , \tag{4.41}$$

and $\mathrm{co}(\cdot)$ denotes the convex hull of a set. The following algorithm computes the graph in (4.41) for a general family $\{\mathcal{P}_i\}_{i\in[n]}$, where $\mathcal{P}_i \subset \mathbb{R}^n$ is a nonempty finite set of stochastic vectors. Note that any such family $\{\mathcal{P}_i\}_{i\in[n]}$ corresponds to the map $g : \mathbb{R}^n \to \mathbb{R}^n$ such that

$$[g(v)]_i = \max_{p\in\mathcal{P}_i} pv \quad \text{for all } i \in [n] \ , \tag{4.42}$$

which has $u = 0$ as a harmonic vector, and is of the above form. Hence the algorithm below corresponds also to the computation of the critical graph of this map g.

Before writing the algorithm, we recall some definitions of graph theory (see for instance [CLRS01]). We define a graph $G := (V, E)$ as a finite set of vertices (or nodes) V and a set of edges (or arcs) $E := \{(i, j) \mid i, j \in V\}$. A path of length $l \geq 0$ is a sequence $(i_0, \ldots, i_l)$ such that $i_k \in V$ for $k \in \{0, \ldots, l\}$ and $(i_k, i_{k+1}) \in E$ for $k < l$. A strongly connected component of G is the restriction $G|_{V'}$ of G to some subset of nodes $V' \subseteq V$, that is the graph (V', E') with $E' := \{(i, j) \in E \mid i, j \in V'\}$, where V' is such that there exists a path from each node $i \in V'$ to every node $j \in V'$. A strongly connected component G' is called trivial if it consists in exactly one node and no arcs. We define a final class of $G = (V, E)$ as a non trivial strongly connected component $G' = (V', E')$ of G such that there exists no arc $(i, j) \in E$ with $i \in V'$ and $j \in V \setminus V'$. Note that the strongly connected components of a graph can be find using Tarjan algorithm, see [CLRS01].

Algorithm 4.6 (Algorithm to compute the critical graph, compare with [AG03, § 6.3])**.**

Input*: $(\mathcal{P}_1, \cdots, \mathcal{P}_n)$ where $\mathcal{P}_i \subset \mathbb{R}^n$ is a finite set of stochastic vectors for $i \in [n]$.*

Output*: A graph depending on $\mathcal{P}_1, \cdots, \mathcal{P}_n$, equal to $\mathcal{G}^{\mathrm{f}}(\mathrm{co}(\mathcal{P}_1) \times \cdots \times \mathrm{co}(\mathcal{P}_n))$ if all the $\mathcal{P}_i$ are nonempty; and its set of nodes.*

1. *Set $F^{(0)} = \emptyset$, $I^{(0)} = [n]$, $G^{(0)} = \emptyset$, $\mathcal{Q}_i^{(0)} = \mathcal{P}_i$ for $i \in [n]$, and $k = 0$.*

2. *If all the sets* $\{\mathcal{Q}_i^{(k)}\}_{i\in I^{(k)}}$ *are empty, then the algorithm stops and returns* $G^{(k)}$ *and* $F^{(k)}$.
3. *Otherwise, build the graph* $G = (I^{(k)}, E)$ *with set of nodes* $I^{(k)}$*, and set of arcs* $E = \left\{(i,j) \in I^{(k)} \times I^{(k)} \mid p_j \neq 0 \text{ for some } p \in \mathcal{Q}_i^{(k)}\right\}$. *Set* F *as the union of final classes of* G.
4. *Put* $I^{(k+1)} = I^{(k)} \setminus F$ *and* $F^{(k+1)} = F^{(k)} \cup F$.
5. *Set* $G^{(k+1)} = G^{(k)} \cup G|_F$ *where* $G|_F$ *denotes the restriction of* G *to* F.
6. *For all* $i \in I^{(k+1)}$*, define the sets* $\mathcal{Q}_i^{(k+1)} \subset \mathbb{R}^{I^{(k+1)}}$ *of row vectors obtained by restricting to* $I^{(k+1)}$ *the vectors* $p \in \mathcal{Q}_i^{(k)}$ *such that* $\sum_{j\in I^{(k+1)}} p_j = 1$.
7. *Increment* k *by one, and go to Step 2.*

The convergence (after at most n iterations) of this algorithm follows from variants of Lemmas 4.7 and 4.9 of [AG03], applied to the maps g_k constructed by (4.42) from the families $(\mathcal{Q}_i^{(k)})_{i\in[n]}$. Indeed, if all the $\mathcal{Q}_i^{(k)}$ with $i \in I^{(k)}$ are nonempty, the map g_k is a map from $\mathbb{R}^n$ to itself and Lemma 4.7 says that g_k has at least one invariant critical class, which implies that the set F of Step 3 is nonempty. Moreover, Lemma 4.9 says that, if all the $\mathcal{Q}_i^{(k+1)}$ with $i \in I^{(k+1)}$ are nonempty, the critical graph of g_k is equal to the union of $G|_F$ with the critical graph of the map g_{k+1}.

In order to generalize these arguments, one needs to extend the notion of critical graph to the case of a map g from $(\mathbb{R}\cup\{-\infty\})^n$ to itself, of the form (4.42) with general families $\{\mathcal{P}_i\}_{i\in[n]}$ of (possibly empty) finite sets of stochastic vectors (or of the form (4.27) with (4.28), with a harmonic vector $u \in (\mathbb{R}\cup\{-\infty\})^n$). For instance, define the critical graph of g as the restriction to the set of nodes $i \in [n]$ such that $\mathcal{P}_i$ is nonempty (or $u_i \neq -\infty$) of the critical graph of $g \vee \mathrm{id}$, where id is the identity map and $\vee$ denotes the supremum operation. Then, the identically $-\infty$ map has no critical class, any map g which is not identically $-\infty$ has an invariant critical class, and the above recurrence formula for critical graphs is true even if g_{k+1} takes $-\infty$ values. This shows that Algorithm 4.6 computes the critical graph of the map g associated to the family $\{\mathcal{P}_i\}_{i\in[n]}$, even if some of the sets of the family are empty.

Note that since Tarjan algorithm has a linear complexity in the number of arcs of a graph, the complexity of the above algorithm is at most in the order of nm, where m is the sum of the number of arcs of all the elements of $\mathcal{P}_i$, $i \in [n]$. This is comparable with the complexity of solving the linear systems of the form (4.33) by LU solvers, hence with the other steps of Algorithm 4.2.

4.6 An example with degenerate iterations

In this section, we present an example of zero-sum two player stochastic game for which we encounter a degenerate iteration when using the policy iteration algorithm for the mean payoff problem, and showing that Step 5 of Algorithm 4.1 is essential to obtain the convergence of the algorithm.

Before doing this, let us note that some degenerate cases may be not so problematic. Indeed, as observed before, the map $\bar{g}$ of Step 5 of Algorithm 4.2 is a polyhedral order preserving additively homogeneous convex map. By [AG03, Theorem 1.1], the set of fixed points of $\bar{g}$ is isomorphic to a convex set which dimension is the number of strongly connected components of the critical graph of $\bar{g}$ and which is invariant by the translations by a constant function. In particular, if the number of strongly connected components of the critical graph is equal to one, then the set of fixed points of $\bar{g}$ is exactly equal to the translations of v' by a constant, hence $v^{(k+1)} - v'$ is a constant function. Since all the maps considered in Algorithm 4.2 are additively homogeneous, this implies that taking v' instead of $v^{(k+1)}$, that is applying the same steps as in the nondegenerate case, does not change the sequence of policies (σ_k), and the invariant half lines are just translated by a constant after this degenerate iteration. Hence, the second part of Step 5 may be avoided in Algorithm 4.2, when one encounters only such degenerate iterations. However, to know that $\bar{g}$ has only one strongly connected component in its critical graph, one need to apply the the first part of Step 5.

We show now an example for which degenerate iterations occur with two strongly connected components of the critical graph of $\bar{g}$. We shall call these iterations *strongly degenerate*.

We consider a directed graph, with a set of nodes (or edges) $[n]$ and a set of arcs $E \subset [n] \times [n]$, in which each arc (i, j) is equipped with a weight $r_{ij} \in \mathbb{R}$, and consider the map f from $\mathbb{R}^n$ to itself, defined by:

$$f_i(v) = \frac{1}{2}\left(\max_{j:\ (i,j)\in E} (r_{ij} + v_j) \ + \ \min_{j:\ (i,j)\in E} (r_{ij} + v_j) \right) \ . \tag{4.43}$$

When the value of v is fixed at some "boundary" points, and the weights r_{ij} are independent of j, the map f arises as the dynamic programming operator of the "tug of war" game [PSSW09], which can viewed also as a discretization of the infinity Laplacian operator. Moreover the case where all the weights r_{ij} are equal to zero corresponds to a class of auction games, called Richman game [LLP$^+$99]. Therefore, the above map f appears as the dynamic programming operator of a variant of these games with additive reward and mean payoff.

We apply the policy iteration algorithm to such a game, with a graph of 5 nodes and complete set of arcs $E = [5] \times [5]$. Hence, the action spaces A_i and B_i in every state $i \in [n]$ can be identified with the set [5]. The weight of each arc $(i, j) \in E$ is defined as the entry r_{ij} of the following matrix :

$$r = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & -1 & 0 & -1 & 1 \end{pmatrix} \ ,$$

the adjacency graph of which is represented in Figure 4.1.

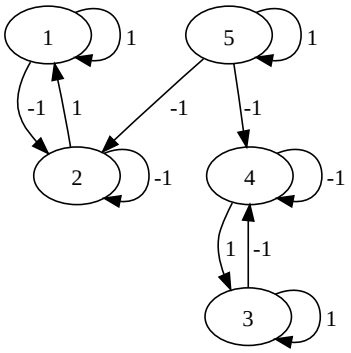

Figure 4.1: Adjacency graph of r.

Let us fix the initial strategy $\alpha^{(0)}$ for the first player, such that $\alpha^{(0)}(1) = 2$, $\alpha^{(0)}(2) = 2$, $\alpha^{(0)}(3) = 4$, $\alpha^{(0)}(4) = 4$, $\alpha^{(0)}(5) = 2$. Then, the corresponding dynamic programming operator $f^{(\alpha^{(0)})}$ is given by

$$\begin{aligned} f_1^{(\alpha^{(0)})}(v) = f_2^{(\alpha^{(0)})}(v) &= \frac{1}{2}(-1 + v_2 + \max(1 + v_1, -1 + v_2, v_3, v_4, v_5)) \\ f_3^{(\alpha^{(0)})}(v) = f_4^{(\alpha^{(0)})}(v) &= \frac{1}{2}(-1 + v_4 + \max(v_1, v_2, 1 + v_3, -1 + v_4, v_5)) \\ f_5^{(\alpha^{(0)})}(v) &= \frac{1}{2}(-1 + v_2 + \max(v_1, -1 + v_2, v_3, -1 + v_4, 1 + v_5)) \ . \end{aligned}$$

In Step 1 of Algorithm 4.1, we compute an invariant half-line of $f^{(\alpha^{(0)})}$ and obtain for instance $w^{(0)}(t) = (\eta^{(0)}, v^{(0)})$, with $v^{(0)} = (0, 0, -0.5, -0.5, 0)^T$ and $\eta^{(0)} = (0, 0, 0, 0, 0)^T$. Since $f(w^{(0)}(t)) < f^{(\alpha^{(0)})}(w^{(0)}(t))$, we need to improve the policy (Step 3) and get the unique solution (even without the conservative policy): $\alpha^{(1)}(1) = 2$, $\alpha^{(1)}(2) = 2$, $\alpha^{(1)}(3) = 4$, $\alpha^{(1)}(4) = 4$, $\alpha^{(1)}(5) = 4$. The corresponding operator is then given by :

$$\begin{aligned} f_i^{(\alpha^{(1)})} &= f_i^{(\alpha^{(0)})} \qquad 1 \le i \le 4, \\ f_5^{(\alpha^{(1)})}(v) &= \frac{1}{2}(-1 + v_4 + \max(v_1, -1 + v_2, v_3, -1 + v_4, 1 + v_5)) \ . \end{aligned}$$

We compute then (in Step 4) an invariant half-line $(\eta^{(1)}, v')$ of $f^{(\alpha^{(1)})}$, and obtain $\eta^{(1)} = (0, 0, 0, 0, 0)^T$ and for instance $v' = (0, 0, 0.5, 0.5, 0.5)^T$. Since $\eta^{(1)} = \eta^{(0)}$, the iteration is degenerate.

Hence the algorithm enters in Step 5. Set $g := f^{(\alpha^{(1)})}$. We have to compute the critical graph of $\bar{g}$, which is here equal to g, for instance by applying Algorithm 4.6 to the sets $\mathcal{P}_i$ defined in (4.40) with $u = v'$. They are given by $\mathcal{P}_1 = \mathcal{P}_2 = \{(0.5, 0.5, 0, 0, 0)\}$, $\mathcal{P}_3 = \mathcal{P}_4 = \{(0, 0, 0.5, 0.5, 0)\}$, $\mathcal{P}_5 = \{(0, 0, 0, 0.5, 0.5)\}$, then the critical graph of g is equal to the final graph of $\mathcal{P}_1 \times \cdots \times \mathcal{P}_5$, which is composed of two strongly connected components with nodes $\{1, 2\}$ and $\{3, 4\}$. Then, $v^{(1)}$ is the unique solution of:

$$\left\{ \begin{array}{lcll} v_5^{(1)} & = & f_5^{(\alpha^{(1)})}(v) = \frac{1}{2}(-1.5 + \max(0, -1, -0.5, -1.5, v_5^{(1)} + 1)) & \\ v_i^{(1)} & = & v_i^{(0)} & i \in \{1, 2, 3, 4\} \ . \end{array} \right.$$

We obtain $v^{(1)} = (0, 0, -0.5, -0.5, -0.5)$ and since $f(w^{(1)}(t)) = f^{(\alpha^{(1)})}(w^{(1)}(t))$, the algorithm stops.

However, if we do not treat the degenerate case by using Step 5, and take for instance $v^{(1)} = v'$, we obtain $f(w^{(1)}(t)) < f^{(\alpha^{(1)})}(w^{(1)}(t))$, hence we need to improve the strategy, and obtain the unique solution $\alpha^{(2)} = \alpha^{(0)}$. This means that the algorithm cycles, showing the necessity of Step 5 in the policy iterations.

4.7 Implementation and numerical results

The numerical results presented in this section were obtained with a slight modification of the policy iteration algorithm Algorithm 4.2) and of its ingredients of Section 4.5, all implemented in the C library PIGAMES, see [Det12] for more information. All the tests of this section were performed on a single processor: Intel(R) Xeon(R) W3540 - 2.93GHz with 8Go of RAM.

These slight modifications take into account the fact that (linear or nonlinear) equations may not be solved exactly (in exact arithmetics) because of the errors generated by floating-point computations, and also of the possible use of iterative methods instead of exact methods. Let us explain them briefly. For instance, the stopping criterion in Step 2 of Algorithm 4.2 can be replaced by a condition on the residual of the mean payoff, $\hat{f}(\eta^{(k)}) - \eta^{(k)}$ and the residual of the relative value, $\acute{f}_\eta(v^{(k)}) - \eta^{(k)} - v^{(k)}$. Here, we consider the infinity norm of the residual of the game that we define as $0.5 * (\|\hat{f}(\eta^{(k)}) - \eta^{(k)}\|_\infty + \|\acute{f}_\eta(v^{(k)}) - \eta^{(k)} - v^{(k)}\|_\infty)$, where $\|\cdot\|_\infty$ denotes the sup-norm. Then, we stop the policy iterations when the infinity norm of the residual of the game is smaller than a given value $\epsilon_g > 0$ or when the strategies cannot be improved. For the tests of this section, we took $\epsilon_g = 10^{-12}$. We use the same condition for the stopping criterion of the intern policy iterations, that is for Step 3 of Algorithm 4.3 and Step 3 of Algorithm 4.4. Moreover, the optimization problems in Step 3 of Algorithm 4.2 and Step 4 of Algorithms 4.3 and 4.4, are solved up to some precision. This means for instance that in Algorithm 4.2, one choose $\alpha^{(k+1)} \in \mathcal{A}_M$ such that, for all $i \in [n]$,

$$\begin{cases} \acute{F}_{\eta^{(k)}}(v^{(k)}; i, \alpha^{(k+1)}(i)) \leq \epsilon_v + \min\limits_{a \in \acute{\mathcal{A}}_{i,\eta^{(k)},\epsilon_\eta}} \left\{ \acute{F}_{\eta^{(k)}}(v^{(k)}; i, a) \right\} \text{ with} \\ \acute{\mathcal{A}}_{i,\eta,\epsilon} := \left\{ a \in \mathcal{A}_i \mid \hat{F}(\eta; i, a) \leq \epsilon + \hat{f}(\eta)]_i \right\} \\ \alpha^{(k+1)}(i) = \alpha^{(k)}(i) \text{ if } \alpha^{(k)}(i) \text{ is optimal,} \end{cases} \tag{4.44}$$

for some given ϵ_η and $\epsilon_v > 0$. Finally, the linear systems in Step 2 of Algorithms 4.3 and 4.4 are solved up to some precision, which may be lower bounded when the matrices of the systems are ill-conditioned. See the Section 4.8 for details about the solution of these linear systems.

4.7.1 Variations on tug of war and Richman games

We now present some numerical experiments on the variant of Richman games defined in Section 4.6, constructed on random graphs. As in the previous section, we consider directed graphs, with a set of nodes equal to $[n]$ and a set of arcs $E \subset [n]^2$. The dynamic programming operator is the map f defined in (4.43), where the value r_{ij} is the reward of the arc $(i, j) \in E$. In the tests of Figure 4.2 to Figure 4.4, we chose random sparse graphs with a number of nodes n between 1000 and 50000, and a number of outgoing arcs fixed to ten for each node. The reward of each arc in E has value one or zero, that is $r_{ij} = 1$ or 0. The arcs $(i, j) \in E$ and the associated rewards r_{ij} are chosen randomly (uniformly and independently). We start the experiments with a sizer of graph (number of nodes) equal to $n = 1000$, then we increase the size by 1000 until reaching $n = 10000$, after we increase the size by 10000 and end with a number of 50000 nodes. For each size that we consider, we made a sample of 500 tests. The results of the application of the policy iteration (Algorithm 4.2 with the above modifications) on those games are presented in Figures 4.2 to 4.4 and are commented below.

Figure 4.2 gives for each size n, and among the sample of 500 tests, the number of tests that encountered at least one strongly degenerate policy iteration for the first player. Hence, these games require the degenerate case issue presented in this paper, that is Step 5 of Algorithm 4.1 or 4.2. Moreover, from the data of Figure 4.2, we observe that approximately between 10 and 15 percent of the tests have at least one strongly degenerate policy iteration for the first player.

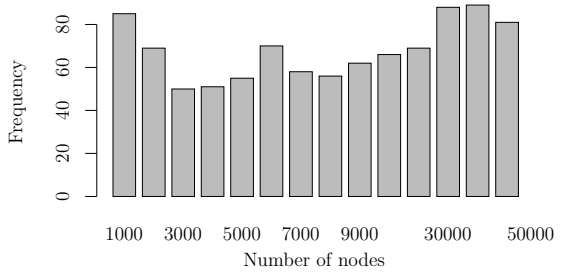

Figure 4.2: Tests on a variant of Richman games constructed on random graphs. The histogram shows for each size (number of nodes), the number of tests having at least one strongly degenerate policy iteration for the first player, among 500 tests.

In the table below we report the number of strongly degenerate iterations that occur in the global sample of tests.

Number of strongly degenerate iterations	0	1	2	3	6
Number of tests	6051	919	28	1	1

We observe that in general there is no more than one or two strongly degenerate policy iterations for our sample of tests. Note that in this section, a strongly degenerate policy iteration is to be understood as a strongly degenerate iteration for the first player only, that is for Algorithm 4.2.

In Figure 4.3, we draw on the left curves that represent the number of policy iterations for the first player, that is the number of iterations of Algorithm 4.2, as a function of the size n of the graph. The dashed lines on top and bottom are respectively the maximum and minimum value, over the sample of 500 tests, and the plain line is the average value, all as a function of the size. We observe that the average number of first player's policy iterations is almost constant as the size increases. Using the same model of representation, we show on the right of Figure 4.3 respectively the maximum, average and minimum values for the total number of policy iterations for the second player, that is the sum of the numbers of iterations of Algorithm 4.4 when applied by Algorithm 4.2, as a function of the size. We also observe that these values do not vary a lot with the size.

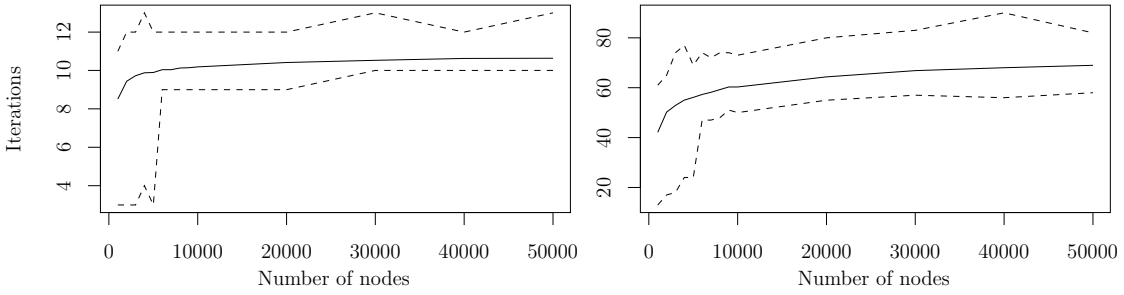

Figure 4.3: Tests on a variant of Richman games constructed on random graphs. On the left, the curves from top to bottom represent respectively the maximum, average, minimum number of first player's policy iterations, among 500 tests, as a function of the number of nodes. On the right, the curves represent the total number of second player's policy iterations.

In Figure 4.4, we present on the left the total cpu time (in seconds) needed by the policy iteration to find the solution of the game. As for the two previous figures, the curves from top to bottom show respectively the maximum, average and minimum values, over the sample of 500 tests, as a function of the size of the graphs. Finally, on the right of Figure 4.4, we give also the average of the total cpu time (in seconds) needed to solve the game but we separated the tests with strongly degenerate policy iteration(s), represented by the dashed line, from the non strongly degenerate ones, represented by the plain curve. We observe that the average cpu time is somewhat greater for the tests with strongly degenerate iteration(s). This is due to the additional steps needed for degenerate iterations. Indeed, the cpu time of a degenerate iteration should be approximately the double of that of a nondegenerate iteration, and since the number of policy iterations is around 10 in the sample of tests, the average of the total cpu time of tests with (strongly) degenerate iterations should be approximately 10 percent greater than that of the other tests.

In addition, in Table 4.1, we give numerical results for ten tests of the variant of Richman game, constructed on random large graphs with a number of nodes between 10^5 and 10^6. We observe that the number of iterations are of the same order as for the previous sample of tests presented in Figure 4.3.

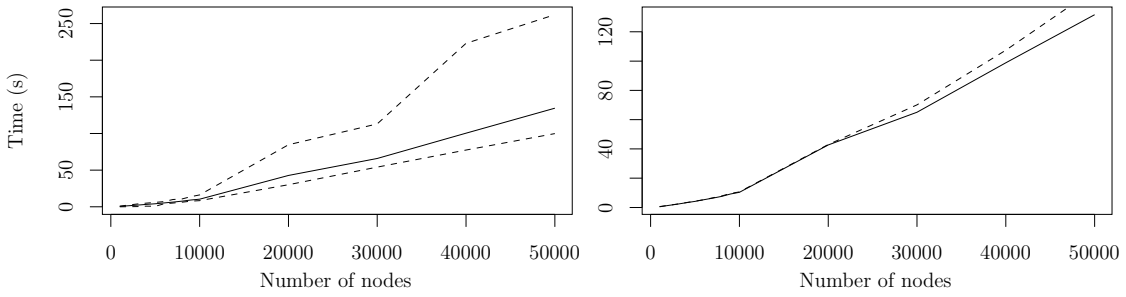

Figure 4.4: Tests on a variant of Richman games constructed on random graphs. On the left, the curves from top to bottom represent respectively the maximum, average and minimum values of the total cpu time (in seconds) taken by the policy iteration algorithm, among 500 tests, as a function of the number of nodes. On the right, the dashed line represents the average among the tests that encounter at least one strongly degenerate policy iteration for the first player, whereas the plain line represents the average among the other tests.

Table 4.1: Numerical results on a variant of Richman game constructed on random large graphs.

Number of nodes	Iterations of first player	Total number of iterations	Strongly degenerate iterations	Infinity norm of residual	CPU time (s)
100000	12	78	1	$1.44e-14$	$3.24e+02$
200000	12	74	0	$7.44e-15$	$7.90e+02$
300000	11	82	0	$1.33e-15$	$9.38e+02$
400000	12	82	1	$8.55e-15$	$1.42e+03$
500000	12	77	1	$2.00e-14$	$2.16e+03$
600000	12	77	0	$8.66e-15$	$2.61e+03$
700000	11	85	0	$3.02e-14$	$2.61e+03$
800000	12	81	1	$4.82e-14$	$6.79e+03$
900000	12	79	1	$1.27e-14$	$4.17e+03$
1000000	12	90	1	$3.33e-15$	$1.96e+04$

4.7.2 Pursuit evasion games

We consider now a pursuit evasion game with two players : a pursuer and an evader. The evader wants to maximize the distance between him and the pursuer and the pursuer has the opposite objective. See for instance [BFS94, BFS99, LCS08] for a complete description of general pursuit games. To simplify the model, we consider as state of the game, the distance between the two players. Then, the state of the game is given by $x = x_P - x_E$ where x_P is the position of the pursuer and x_E the position of the evader. We also restrict the state x to stay in a unit square centered in the 0-position, that is $x \in X := [-0.5, 0.5] \times [-0.5, 0.5]$. At each time of the game, the reward for the evader is the euclidean square norm of the distance between the two players, i.e. $\|x\|_2^2$. Such a game is a special class of differential games, the dynamic programming equation of which is an Isaacs partial differential equation. Under our simplifications and assumptions, the Hamiltonian of this equation is given by :

$$H(x,p) = \max_{a \in \mathcal{A}(x)} (a \cdot p) + \min_{b \in \mathcal{B}(x)} (b \cdot p) + \|x\|_2^2 \qquad \forall x \in X, p \in \mathbb{R}^2 \ , \tag{4.45}$$

meaning that in the case of a finite horizon problem, the Isaacs equation would be given, at least formally (but also in the viscosity sense) by :

$$-\frac{\partial v}{\partial t} + H(x, \nabla v(x)) = 0 \qquad x \in X \ .$$

Here $A(x)$ and $B(x)$ are the sets of possible directions for the evader and the pursuer respectively, when the state is equal to $x \in X$. On the boundary, we consider that only actions keeping the state of the game in the domain X are allowed, hence the above equation has to be satisfied until the boundary.

We shall consider this differential game with a mean-payoff criterion and the above reward. This means that the analogous to System (4.14) is the following system of Isaacs equations :

$$\left\{ \begin{array}{rl} \max\limits_{a \in \mathcal{A}(x)} (a \cdot \nabla \eta(x)) + \min\limits_{b \in \mathcal{B}(x)} (b \cdot \nabla \eta(x)) = 0 \ , & x \in X \ , \\ -\eta(x) + \max\limits_{a \in \acute{\mathcal{A}}_\eta(x)} (a \cdot \nabla v(x)) + \min\limits_{b \in \acute{\mathcal{B}}_\eta(x)} (b \cdot \nabla v(x)) + \|x\|_2^2 = 0 \ , & x \in X \ , \end{array} \right. \tag{4.46}$$

where

$$\acute{\mathcal{A}}_\eta(x) := \operatorname*{argmax}_{a \in \mathcal{A}(x)} (a \cdot \nabla \eta(x)) \ ,$$
$$\acute{\mathcal{B}}_\eta(x) := \operatorname*{argmin}_{b \in \mathcal{B}(x)} (b \cdot \nabla \eta(x)) \ .$$

In classical pursuit-evasion games, such as in [BFS99], the reward is constant and the value function is defined as the time (or the exponential of the opposite of the time) for the pursuer to capture the evader, then the value function is solution of the stationary Isaacs equation that is (4.46) with $\eta \equiv 0$, corresponding to the above Hamiltonian with

1 instead of $\|x\|_2^2$. In that case, the value is infinite when the pursuer's speed is smaller than the evader's speed, and it is would be difficult to compute an optimal strategy using Isaacs equation. Here by considering a mean-payoff problem, we may solve the problem even when pursuer's speed is smaller than the evader's speed, as we shall see below. Note that one may have kept the reward equal to 1, but then the optimal value η would have given less information.

A monotone discretization, for instance a finite difference discretization scheme (see [KD92]), of System (4.46) yields to System (4.14) for the dynamic programming operator f of a discrete time and finite state space game, which then may be solved using our policy iteration Algorithm 4.2.

In our tests, the domain X is discretized in each directions with a constant step size h. Then the two players of the discrete game are moving on the discretized nodes of the domain, similarly to the moves in a chess game. We assume also that the evader cannot move when the euclidean norm of the relative distance between him and the pursuer is less than 0.1, i.e when $x \in \mathcal{B}((0,0);0.1)$. We shall call the evader, the mouse and his set of possible actions at each state of the game will given by :

$$\mathcal{A}(x) := \begin{cases} \{(a_1,a_2) \,|\, a_l \in \{0,1,-1\}, \quad l=1,2\} & x \in \mathring{X} \setminus \mathcal{B}((0,0);0.1) \\ \{(0,0)\} & x \in \mathcal{B}((0,0);0.1) \ , \end{cases}$$

where $\mathring{X}$ denotes the interior of X. The pursuer, that we shall call the cat, has the following set of possible actions :

$$\mathcal{B}(x) := \{(b_1,b_2) \,|\, b_l \in \{0,\bar{b},-\bar{b}\}, \quad l=1,2\} \quad x \in \mathring{X} \ ,$$

where $\bar{b}$ is a positive real constant and represents the speed of the cat. Moreover, on the boundary of X, the sets $\mathcal{A}(x)$ and $\mathcal{B}(x)$ are restricted to avoid actions that bring the state out of X.

Numerical results for this game are presented in Table 4.2 when $\bar{b} = 0.999$, $\bar{b} = 1$ and $\bar{b} = 1.001$ respectively. Note that the solution of the discretization of Equation (4.46) may differ from the solution of the continuous equation. We observe that for $\bar{b} = 0.999$ and $\bar{b} = 1.001$, we have a strongly degenerate iteration for the first player on the last iteration.

The optimal actions for the discretized problem with $\bar{b} = 0.999$ are represented in Figure 4.5, at each node of the grid: the actions of the mouse are on the left, and that of the cat are on the right. The optimal actions are approximately the same for the two other values of $\bar{b}$. When $\bar{b} = 0.999$, the speed of the cat is smaller than the speed of the mouse $(= 1)$. The numerical results for the discretized game give an optimal mean-payoff η such that $\eta(x) = 0.492$ for $x \in X \setminus \mathcal{B}((0,0);0.1)$ and $\eta(x) = 0$ for $x \in \mathcal{B}((0,0);0.1)$. This means that the cat cannot catch the mouse when their starting positions are not too close and the mouse can keep almost the maximum distance between them. The relative value is represented on the left of Figure 4.6. When $\bar{b} = 1$, the speeds of the cat and the mouse are equal. The numerical results for the discretized game give a relative value v approximately

Table 4.2: Numerical results for the mouse and cat example where $\bar{b}$ is the speed of the cat. The second column is the index of the iteration on the cat's policies and the third column is the corresponding total number of iterations on the mouse's policies. The last column indicates if the cat's policy iteration is strongly degenerate. Number of discretization nodes: 257×257.

$\bar{b}$	Cat policy iteration index	Number of mouse policy iterations	Infinite norm of residual	CPU time (s)	Strongly degenerate iteration
0.999	1	2	$1.25e-06$	$2.59e+01$	0
	2	1	$9.93e-12$	$3.95e+01$	0
	3	1	$5.68e-14$	$7.35e+02$	1
1	1	2	$1.25e-06$	$2.60e+01$	0
	2	1	$3.39e-21$	$3.84e+01$	0
1.001	1	2	$1.25e-06$	$2.59e+01$	0
	2	1	$1.96e-14$	$6.51e+02$	1

equal to zero for every starting point and an optimal mean-payoff $\eta(x) \approx \|x\|_2^2$, meaning that the cat and mouse keep the same initial distance all along the game. In the last example, the speed of the cat $\bar{b} = 1.001$ is greater than that of the mouse ($= 1$). The numerical results for the discretized game give an optimal mean-payoff η close to zero. The relative value v is given on the right of Figure 4.6. In this case, the cat catches the mouse.

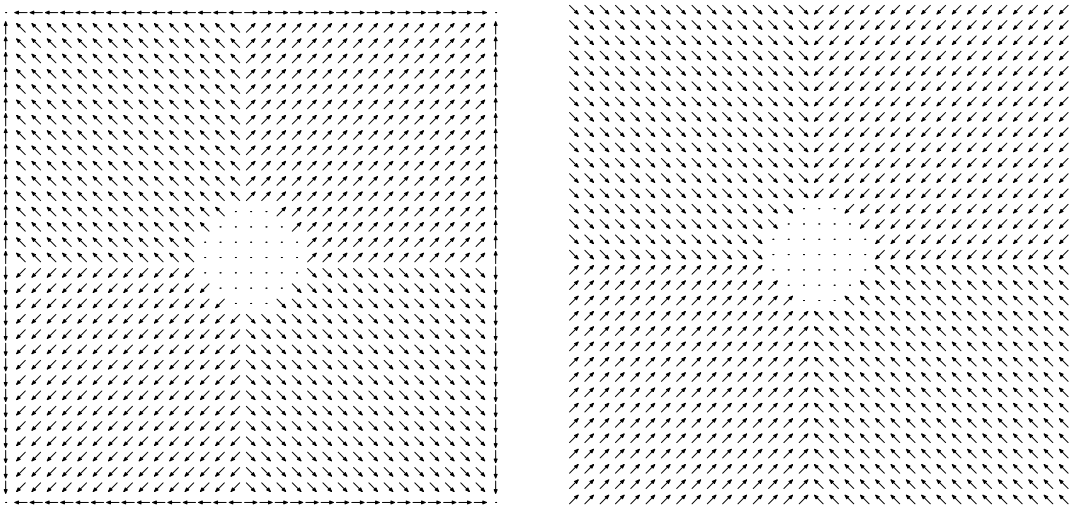

Figure 4.5: Optimal actions for the mouse on the left and for the cat on the right.

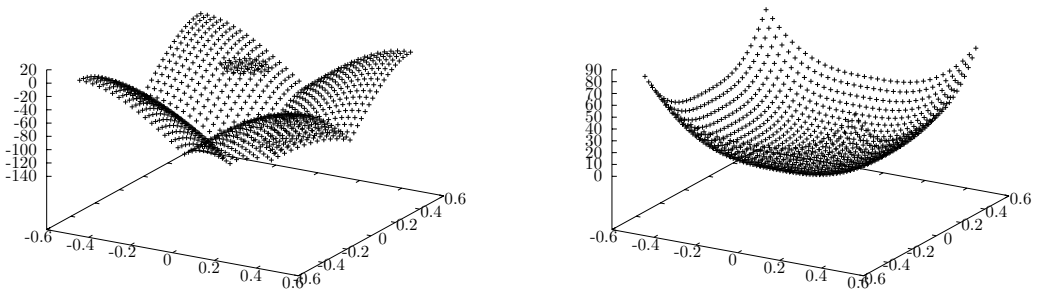

Figure 4.6: Relative value v for the mouse and cat game when the speed of the mouse is one, the speed of the cat equals 0.999 on the left and 1.001 on the right.

4.8 Details of implementation of Policy Iteration for multichain one player games

We explain in more details here why System (4.33) has a unique solution and is selecting one special solution of System (4.37) with k instead of $k+1$, and how it is solved practically (see also [DF68, Put94]).

Recall that System (4.37) is of the form (4.39) rewritten here:

$$\left\{\begin{array}{rcl} \eta & = & P\,\eta \\ \eta + v & = & P\,v + r\ , \end{array}\right.$$

where $r = r^{(\delta)} \in \mathbb{R}^n$ and $P = P^{(\delta)}$ is a stochastic matrix, with $\delta = \beta^{(k+1)}$. Moreover, System (4.33) corresponds to

$$\left\{\begin{array}{rcll} \eta_i & = & (P\eta)_i & i \in [n] \setminus S\ , \\ \eta_i + v_i & = & (Pv)_i + r_i & i \in [n]\ , \\ v_i & = & 0 & i \in S\ , \end{array}\right. \tag{4.47}$$

where r and P are as before but with $\delta = \beta^{(k)}$, and where S is composed of minimal indices i_F of each final classes F of P. Then one need to show that System 4.47 has a unique solution and is selecting one special solution of System (4.39).

First, for all final classes F of P, P_{FF} is an irreducible Markov matrix, hence the equation $\eta_F = (P\eta)_F$, which is equivalent to $\eta_F = P_{FF}\eta_F$, is also equivalent to the condition that $\eta_i = \eta_j$ for all $i, j \in F$. Moreover, P_{FF} has a unique stationary (or invariant) probability measure π_F, that is a row probability vector solution of $\pi_F = \pi_F P_{FF}$, and this vector has strictly positive coordinates. This implies that one can eliminate, for each final class F of P, one equation with index i in F in the equation $\eta = P\eta$, without changing the set of solutions. Hence, any solution of System 4.47 is also solution of (4.39).

Second, denote by $\mathcal{F}$ the union of final classes, and by $\mathcal{T}$ the union of transient classes, that is the complement in $[n]$ of $\mathcal{F}$. Then, w is in the kernel of $I - P$ if, and only if, it

satisfies $w_F = P_{FF} w_F$, for all final classes F of P and $w_{\mathcal{T}} = P_{\mathcal{T}\mathcal{T}} w_{\mathcal{T}} + P_{\mathcal{T}\mathcal{F}} w_{\mathcal{F}}$. As said before the first equations are equivalent to the conditions $w_i = w_j$ for all $i, j \in F$. Since $P_{\mathcal{T}\mathcal{T}}$ has transient classes only, it has a spectral radius strictly less than one, which implies that given the vectors w_F for all final classes F, the equation $w_{\mathcal{T}} = P_{\mathcal{T}\mathcal{T}} w_{\mathcal{T}} + P_{\mathcal{T}\mathcal{F}} w_{\mathcal{F}}$ has a unique solution $w_{\mathcal{T}}$. Hence, the dimension of the kernel of $I - P$ is equal to the number of final classes of P, and any element of this kernel which has one coordinate $i \in F$ equal to zero for each final classes F of P, has all its coordinates equal to zero. This implies that given $\eta \in \mathbb{R}^n$, the solution v of System (4.47) is unique if it exists (the difference between two such solutions satisfies the above conditions). This also implies that the codimension of the image of $I - P$ is equal to the number of final classes. Hence, the image of $I - P$ is exactly equal to the set of vectors $\eta \in \mathbb{R}^n$ such that $\pi_F \eta_F = 0$, for all final classes F of P. A vector $\eta \in \mathbb{R}^n$ is such that System (4.47) has a solution $v \in \mathbb{R}^n$ if and only if $\eta = P\eta$ and $\eta - r$ is in the image of $I - P$. These conditions are equivalent to the three conditions $\eta_i = \eta_j$ for all $i, j \in F$, for all final classes F, $\eta_{\mathcal{T}} = P_{\mathcal{T}\mathcal{T}} \eta_{\mathcal{T}} + P_{\mathcal{T}\mathcal{F}} \eta_{\mathcal{F}}$, and $\pi_F \eta_F = \pi_F r_F$, for all final classes F. The first and third conditions together are equivalent to $\eta_i = \pi_F r_F$, for all $i \in F$, and all final classes F of P, which gives a unique solution $\eta_{\mathcal{F}}$. Since the second one has a unique solution $\eta_{\mathcal{T}}$, given $\eta_{\mathcal{F}}$, we get that there is a unique vector $\eta \in \mathbb{R}^n$ such that System (4.47) has a solution $v \in \mathbb{R}^n$. In conclusion, System (4.47) has a unique solution (η, v), which finishes the proof what we wanted to show.

One may try to solve System (4.47) by using usual LU methods, however when P is not irreducible, such a method is not robust. We rather use the decomposition of P in classes, and the previous properties. In particular, since $P_{\mathcal{T}\mathcal{T}}$ has a spectral radius strictly less than 1, one can compute (η, v) solution of System (4.47) by first computing η_F and v_F, for all final classes F of P, then computing successively $\eta_{\mathcal{T}}$ and $v_{\mathcal{T}}$ which are respectively fixed points of contracting affine systems with tangent linear operator $P_{\mathcal{T}\mathcal{T}}$:

$$\begin{cases} \eta_{\mathcal{T}} &= P_{\mathcal{T}\mathcal{T}}\, \eta_{\mathcal{T}} + P_{\mathcal{T}\mathcal{F}}\, \eta_{\mathcal{F}} \\ v_{\mathcal{T}} &= P_{\mathcal{T}\mathcal{T}}\, v_{\mathcal{T}} + P_{\mathcal{T}\mathcal{F}} v_{\mathcal{F}} + r_{\mathcal{T}} - \eta_{\mathcal{T}} \ . \end{cases}$$

There exists two ways to compute η_F and v_F. One is to compute the stationary probability π_F to determine η_F by $\eta_i = \pi_F r_F$, for all $i \in F$, and then solve the following system with unknown $v_F \in \mathbb{R}^F$:

$$v_F = P_{FF}\, v_F + r_F - \eta_F \ ,$$

by eliminating one equation (since one equation is redundant) with index $j \in F$, and adding the condition $v_i = 0$ for one element $i \in F$. Another method is to consider η_F as constant, say $\eta_i = \bar{\eta}$ for $i \in F$, and solve the system with unknowns $\bar{\eta} \in \mathbb{R}$ and $v \in \mathbb{R}^F$:

$$\bar{\eta} + v_i = \sum_{j \in F} P_{ij}\, v_j + r_i, \quad i \in F \ ,$$

by adding the condition $v_i = 0$ for one element $i \in F$. In our algorithm, we choose the index $i = j \in F$ to be the minimal index of F (for a fixed total ordering of nodes). This method gives the following algorithm to solve System (4.47).

Algorithm 4.7 (Solution of System (4.47))**.** *Decompose the matrix P into irreducible classes and permute nodes without changing the order in each class, such that P takes the following form :*

$$P = \begin{pmatrix} P_{11} & P_{12} & \dots & \dots & P_{1m} \\ 0 & P_{22} & \dots & \dots & P_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & P_{m-1,m-1} & P_{m-1,m} \\ 0 & \dots & \dots & 0 & P_{mm} \end{pmatrix}$$

where m denotes the number of irreducible classes and P_{II} are square irreducible submatrices of P, for $I = 1, \dots, m$. Note that, the class corresponding to a submatrix P_{II} is final if and only if the submatrices P_{IJ} are all null for all $J \neq I$.
For each class I from m to 1, do the following :

Step 1. If I corresponds to a final class, that is P_{II} is a stochastic matrix, do one of the two following sequences of operations :

A. (a) Find the stationary probability π_I of P_{II}: $\pi_I P_{II} = \pi_I$,

(b) Set $\bar{\eta} = \pi_I r_I$ and $\eta_i = \bar{\eta} \quad i \in I$,

(c) Solve the system with unknown $v_I \in \mathbb{R}^I$:

$$\begin{cases} v_i &= \sum_{j \in I} P_{ij} v_j + r_i - \bar{\eta} \quad i \in I \setminus S \ , \\ v_i &= 0 \qquad\qquad\qquad\qquad\quad i \in S \cap I \ , \end{cases} \tag{4.48}$$

B. Solve the system with unknowns $v_I \in \mathbb{R}^I$ and $\bar{\eta} \in \mathbb{R}$:

$$\begin{cases} \bar{\eta} + v_i &= \sum_{j \in I} P_{ij} v_j + r_i \quad i \in I \ , \\ v_i &= 0 \qquad\qquad\quad i \in S \cap I \ , \end{cases}$$

and set $\eta_i = \bar{\eta} \quad i \in I$,

Step 2. if I corresponds to a transient class, that is if P_{II} is a strictly submarkovian matrix. do the following steps :

2.1 compute η_I solution of the following system :

$$\eta_I = P_{II} \eta_I + \sum_{J > I} P_{IJ} \eta_J$$

2.2 compute v_I solution of the following system :

$$v_I = P_{II} v_I + \sum_{J > I} P_{IJ} v_J + r_I - \eta_I \ .$$

In our numerical experiments, the linear system (4.33) at each intern policy iteration is solved by using Algorithm 4.7. For the numerical experiments of Section 4.7.1, on each final class, we used a SOR iterative solver to find the stationary probability π_I and also to compute the corresponding v_I in method A in Step 1 of Algorithm 4.7. For the transient class, we used the LU solver of the package [DEG+99].

The Successive Over-Relaxation (SOR) method is an iterative scheme that belongs to the class of splitting methods or relation methods, see for instance [BP94]. It is derived from the Gauss-Seidel relaxation scheme. Consider a matrix $A \in \mathbb{R}^{n\times n}$ such that $A = D - L - U$ where D, $-L$, $-U$ are respectively the diagonal, lower and upper triangular part of A. The SOR smoothing operator is defined by $S_w = M^{-1}N$ where $M = D - wL$ and $N = [(1-w)D + wU]$ for $0 < w < 2$.

Consider the irreducible stochastic matrix $P_{II} \in \mathbb{R}^{I\times I}$ and decompose $\mathcal{I} - P_{II}^T = D - L - U$ where $\mathcal{I}$ is the identity matrix of $\mathbb{R}^{I\times I}$. Starting from an initial positive approximation $\pi^{(0)} \in \mathbb{R}^I$, a SOR smoothing step to find the stationary probability of P_{II} is given by :

$$\begin{aligned} \pi^{(k)} &= S_w\, \pi^{(k-1)} \\ \pi^{(k+1)} &= \frac{\pi^{(k)}}{\left(\sum_{i\in[n]} \pi_i^{(k)}\right)} \ . \end{aligned}$$

The sequence $(\pi^{(2k)})_{k\geq 0}$ converges to the transpose of the stationary probability of P_{II} when the limit $\lim_{k\to\infty} S_w^{(k)}$ exists, see [BP94] for more details. To solve Equation (4.48), decompose $\mathcal{I} - P_{II} = D - L - U$. Then, starting from an initial approximation $v^{(0)} \in \mathbb{R}^I$, a SOR smoothing step consists in :

$$v^{(k)} = (\mathcal{I} - \mathbf{1}\mu)\,(S_w\, v^{(k-1)} + M^{-1}(r_I - \eta_I))$$

where $\mathbf{1} = (1\dots 1)^T \in \mathbb{R}^I$, and $\mu \in \mathbb{R}^I$ is a row vector such that $\mu_i = 1$ for $i \in S \cap I$, and $\mu_i = 0$ otherwise. The sequence $(v^{(k)})_{k\geq 0}$ converges to the solution of Equation (4.48) when the $\lim_{k\to\infty} S_w^{(k)}$ exists, see [BP94] for more details.

For the numerical tests of Section 4.7.2, we used the LU solver of the package [DEG+99] in both cases.

Chapter 5

Multigrid methods for particular linear systems with applications to Markov Chains and to zero-sum two player stochastic games with mean payoff

In this chapter, we present some new algebraic multigrid schemes to solve particular singular linear systems that arise for instance in the policy iteration algorithm for zero-sum two player stochastic games with mean payoff. In Section 5.2, based on one of these algebraic multigrid methods, we introduce a new method to find the stationary probability of an irreducible Markov chain using a stochastic control (i.e. a one player stochastic game) approach and the policy iterations of Howard [How60] and Denardo and Fox [DF68]. We present numerical results on transition matrices of random walks on a square uniform grid. In Section 5.3, we perform some numerical tests on ergodic differential pursuit-evasion games by using a simplified version of the policy iteration algorithm for mean payoff zero-sum two player games (Algorithm 4.2), where Step 5 is never visited, combined with an algebraic multigrid algorithm.

5.1 Solving the linear systems

In this section, we explain which algorithms we use to find a couple (v, η) of vectors of $\mathbb{R}^n$ solution of the linear system :

$$\eta = P\,\eta \tag{5.1a}$$

$$\eta + v = P\,v + r \tag{5.1b}$$

where r is a vector of $\mathbb{R}^n$ and $P \in \mathbb{R}_+^{n\times n}$ is the transition probability matrix of an irreducible Markov chain, that is $P\mathbf{1} = \mathbf{1}$ where $\mathbf{1} \in \mathbb{R}^n$ is the vector of ones and P is irreducible.

Recall that this system is replaced by the following non-singular system :

$$\begin{cases} \eta_i &= \sum_{j\in I} P_{ij}\eta_j & i \in I \setminus S \ , \\ \eta_i + v_i &= \sum_{j\in I} P_{ij}v_j + r_i & i \in I \ , \\ v_i &= 0 & i \in S \cap I \ , \end{cases} \tag{5.2}$$

in Step 1 of Algorithm 4.7, which appears at least once in each policy evaluation step of the policy iteration algorithm when solving a Markov Decision Process with mean payoff, and hence in each intern policy iteration when solving a zero-sum two player stochastic game with mean payoff.

There exists two commonly used methods to solve System (5.1) (or System (5.2)). The first approach requires the stationary probability of the irreducible Markov chain from which η can be computed and given η, it solves System (5.1b) with unknown v. The second approach consists in solving System (5.1b) by considering that η is a constant vector.

In the following sections, we shall use some numerical background of Section 2.4.

5.1.1 First Approach: using the stationary probability

This approach consists in first computing the stationary probability π of an irreducible Markov chain with transition matrix P of Equation (5.1). Here, we assume that π is a column vector and the stationary probability will be π. Hence, we need to find $\pi \in \mathbb{R}^n_+$ satisfying :

$$\pi^T P = \pi^T \qquad \text{and} \qquad \pi^T \mathbf{1} = 1. \tag{5.3}$$

From [Put94, DF68], the solution η of Equation (5.1) equals a constant vector : $\eta = \bar{\eta}\mathbf{1}$ where $\bar{\eta} = \pi^T r \in \mathbb{R}$ and $\mathbf{1} \in \mathbb{R}^n$ is the vector of ones. Hence, replacing this value in Equation (5.1b), the system becomes a singular consistent linear system that can be solved to find the solution v.

From Chapter 2, the stationary probability can be computed using a specific direct solver (see Section 2.4.1) or by iterative methods such as described in Section 2.4.2. We shall use in particular the SOR splitting that we recall below.

Decompose the matrix P such that $P^T = P_D + P_L + P_U$ where P_D, P_L and P_U are respectively the diagonal, the strictly lower triangular part and the strictly upper triangular part of P^T. Set $D = (I - P_D)$. Starting from an initial approximation $\pi^{(0)} \in (\mathbb{R}^*_+)^n$, such that $(\pi^{(0)})^T P = (\pi^{(0)})^T$ and $(\pi^{(0)})^T \mathbf{1} = 1$, a SOR smoothing step at iteration $k+1$ is given by :

$$\pi^{(k+1)} = \frac{S_w \pi^{(k)}}{(\mathbf{1}^T S_w (\pi^{(k)}))} \tag{5.4}$$

with $S_w = M^{-1}N$, $M = (D - wP_L)$, $N = [(1-w)D + wP_U]$ and $0 < w < 1$. The following result shows the semi-convergence of S_w.

Proposition 5.1. *Assume that $P \in \mathbb{R}^{n\times n}_+$ is an irreducible stochastic matrix ($P\mathbf{1} = \mathbf{1}$). Decompose the matrix P such that $P^T = P_D + P_L + P_U$ where P_D, P_L and P_U are*

respectively the diagonal, the strictly lower and the strictly upper triangular part of P^T. *Set* $D = (I - P_D)$. *Let* $S_w = M^{-1}N$ *with* $M = (D - wP_L)$ *and* $N = [(1-w)D + wP_U]$. *Then the SOR smoothing operator* S_w *is semi-convergent for* $0 < w < 1$.

Proof. For $n = 1$, the result is trivial. For $n > 1$, since $P \in \mathbb{R}_+^{n\times n}$ is irreducible and $0 < w < 1$, we have that $P_D + wP_L$ is a non negative matrix such that $(P_D + wP_L) \leq P^T$ and $(P_D + wP_L) \neq P^T$. Hence by part 4 of Theorem 2.6, $\rho(P_D + wP_L) < \rho(P) = 1$. Hence, for $0 < w < 1$, we have that $(I - (P_D + wP_L))^{-1} = \sum_{k=0}^{\infty}(P_D + wP_L)^k \in \mathbb{R}_+^{n\times n}$ and $N \in \mathbb{R}_+^{n\times n}$. It follows that $S_w \in \mathbb{R}_+^{n\times n}$ and the SOR splitting is regular. Moreover, we have that :

$$\begin{aligned} S_w &= (D - wP_L)^{-1}\,[(1-w)D + wP_U] && \text{(5.5a)}\\ &\geq (I + P_D + wP_L)\,[(1-w)D + wP_U] && \text{(5.5b)}\\ &\geq (1-w)(I+P_D)D + w(I+P_D)P_U + w(1-w)P_L\,D && \text{(5.5c)} \end{aligned}$$

Since $(I + P_D)$ and $D = (I - P_D)$ are diagonal matrices with strictly positive diagonal elements, the graph of S_w includes the graph of $(I + P^T)$. Since P is irreducible, the operator $(I + P^T)^{n-1} \in (\mathbb{R}_+^*)^{n\times n}$ is a positive matrix and hence the smoother S_w is primitive (since $S_w^{n-1} \in (\mathbb{R}_+^*)^{n\times n}$).

Moreover, the matrix NM^{-1} is column stochastic :

$$\begin{aligned} \mathbf{1}^T\,(P^T) &= \mathbf{1}^T,\\ \mathbf{1}^T\,w(P_D + P_U + P_L) &= w\mathbf{1}^T,\\ \mathbf{1}^T\,[(1-w)D + wP_U] &= \mathbf{1}^T(D - wP_L),\\ \mathbf{1}^T\,[(1-w)D + wP_U]\,(D - wP_L)^{-1} &= \mathbf{1}^T. \end{aligned}$$

Since NM^{-1} and $M^{-1}N$ are similar matrices, we have that $\rho(S_w) = \rho(M^{-1}N) = 1$. Hence, by part 8 of Theorem 2.6, the smoother S_w is semi-convergent. □

Corollary 5.2. *For* $\pi^{(0)} \in (\mathbb{R}_+^*)^n$, *the sequence of iterates* $\left(\pi^{(k)}\right)_{k\geq 0}$ *given by Equation* (5.4) *converges to the solution of Equation* (5.3) *if* $0 < w < 1$.

Proof. From Equation (5.3), we have that $\pi^{(k+1)} = c^{(k)}S_w\pi^{(k)}$ with $c^{(k)} = (\mathbf{1}^T S_w(\pi^{(k)}))^{-1}$ and $\pi^{(k+2)} = c^{(k+1)}c^{(k)}S_w^2(\pi^{(k)})$ where $c^{(k+1)}c^{(k)} = (\mathbf{1}^T S_w^2(\pi^{(k)}))^{-1}$. Hence, it is equivalent to normalize at each iteration or at the end of the iterations. From [BP94], since the smoother S_w is semi-convergent, the iterates converge to the solution for any $\pi^{(0)} \in (\mathbb{R}_+^*)^n$. □

Note that more general results of convergence were proposed and studied in [MP77, NP79, Ple76].

Solving the singular consistent linear system

It remains then to find the solution $v \in \mathbb{R}^n$ of the linear system

$$\bar{\eta} + v_i = \sum_{j\in[n]} P_{ij}v_j + r_i \qquad i \in [n]\ ,$$

where $\bar{\eta} \in \mathbb{R}$, $r \in \mathbb{R}^n$, $P \in \mathbb{R}_+^{n\times n}$ are such that $P\mathbf{1} = \mathbf{1}$ and $r - \bar{\eta}\mathbf{1}$ is in the kernel of π^T, hence in the range of $(I - P)$. If we set $b = r - \bar{\eta}\mathbf{1} \in \mathbb{R}^n$, we have to solve the singular consistent system :

$$v = Pv + b \ . \tag{5.6}$$

This singular linear system admits an infinite set of solutions. A solution v is defined up to the addition of a constant vector. Since $\pi^T(v - Pv - b) = -\pi^T b = 0$ for all v, and $\pi \in (\mathbb{R}_+^*)^n$, any equation in Equation (5.6) is a linear combination of the others. Hence, we can remove a row with index $s \in \{1, \ldots, n\}$ of Equation (5.6) without changing the set of solutions. This row equation can for instance be replaced by the equation $v_s = 0$. Then, we obtain a non singular system that can be solved by using for instance a direct solver such as the Gauss elimination, see Section 2.1.

Another approach consists in adapting the relaxation schemes (see Section 2.2) for singular systems. That is, for instance, we choose a coordinate $s \in [n]$ and after each relaxation step we remove the constant $v_s^{(k+1)}$ to all the coordinates of the current approximation $v^{(k+1)}$. In particular, we described the SOR splitting below.

Decompose the matrix P such that $P = P_D + P_L + P_U$ where P_D, P_L and P_U are respectively the diagonal, the strictly lower triangular part and the strictly upper triangular part of P. Set $D = (I - P_D)$. Choose $s \in [n]$ and let $\mu \in \mathbb{R}^n$ be the row vector such that $\mu_s = 1$ and $\mu_i = 0$ for $i \neq s$. Starting from an initial approximation $v^{(0)} \in \mathbb{R}^n$, a SOR smoothing step at iteration $k + 1$ is given by :

$$v^{(k+1)} \ = \ (I - \mathbf{1}\mu)\,(S_w\, v^{(k)} \ + \ wM^{-1}b) \tag{5.7}$$

with $S_w = M^{-1}N$, $M = (D \ - \ wP_L)$, $N = [(1 - w)D \ + \ wP_U]$ and $0 < w < 1$. We first show the following lemma.

Lemma 5.3. *Let $S \in \mathbb{R}^{n\times n}$ such that $\rho(S) = 1$, $\gamma(S) < 1$ and 1 is a simple eigenvalue of S. Let $z \in \mathbb{R}^n$ be such that $Sz = z$ and $\mu \in \mathbb{R}^n$ the row vector such that $\mu z = 1$. Then, we have $\rho((I - z\mu)S) < 1$.*

Proof. Since $Sz = z$, we have that

$$(I - z\mu)S(I - z\mu) = (I - z\mu)(S - z\mu), \tag{5.8a}$$
$$= (I - z\mu)S - (I - z\mu)z\mu, \tag{5.8b}$$
$$= (I - z\mu)S. \tag{5.8c}$$

By induction, it follows that $((I - z\mu)S)^k = (I - z\mu)S^k$.

From the Jordan decomposition of S, since $\gamma(S) < 1$ and 1 is a simple eigenvalue of S, we get that $\lim_{k\to\infty} S^k = Q$ where Q is the spectral projection of S on the eigenspace associated to the eigenvalue 1. If u is a left eigenvector of S such that $uz = 1$, we get that $Q = zu$. Since $((I - z\mu)S)^k = (I - z\mu)S^k$, we have that

$$\lim_{k\to\infty} ((I - z\mu)S)^k = (I - z\mu)zu = (z - z)u = 0.$$

This implies that all the eigenvalues of $(I - z\mu)S$ have modulus strictly less than one, hence $\rho((I - z\mu)S) < 1$.

□

Now, we show the semi-convergence of S_w of Equation (5.7).

Proposition 5.4. *Assume that $P \in \mathbb{R}_+^{n\times n}$ is an irreducible stochastic matrix ($P\mathbf{1} = \mathbf{1}$). Decompose the matrix $P = P_D + P_L + P_U$ where P_D, P_L and P_U are respectively the diagonal, the strictly lower and the strictly upper triangular part of P. Set $D = (I - P_D)$. Let $S_w = M^{-1}N$ with $M = (D - wP_L)$ and $N = [(1 - w)D + wP_U]$. Let $S_w = M^{-1}N$ with $M = (D - wP_L)$, $N = [(1 - w)D + wP_U]$ and $0 < w < 1$. Then, the operator S_w is semi-convergent.*

Proof. For $n = 1$, the result is trivial. For $n > 1$, since $P \in \mathbb{R}_+^{n\times n}$ is irreducible and $0 < w < 1$, we have that $P_D + wP_L$ is a non negative matrix such that $(P_D + wP_L) \leq P$ and $(P_D + wP_L) \neq P$. Hence by part 4 of Theorem 2.6, $\rho(P_D + wP_L) < \rho(P) = 1$. Hence, for $0 < w < 1$, we have that $(I - (P_D + wP_L))^{-1} = \sum_{k=0}^{\infty}(P_D + wP_L)^k \in \mathbb{R}_+^{n\times n}$ and $N \in \mathbb{R}_+^{n\times n}$. It follows that $S_w \in \mathbb{R}_+^{n\times n}$ and the SOR splitting is regular. Following the same argument as in Equations (5.5) with P instead of P^T, we have that the graph of S_w includes the graph of $(I + P)$. Since P is irreducible, the operator $(I + P)^{n-1} \in (\mathbb{R}_+^*)^{n\times n}$ is a positive matrix and hence the smoother S_w is primitive.

Moreover, the operator S_w is a row stochastic matrix :

$$\begin{aligned} P\,\mathbf{1} &= \mathbf{1}, \\ w(P_D + P_U + P_L)\,\mathbf{1} &= w\mathbf{1}, \\ [(1-w)D + wP_U]\,\mathbf{1} &= (D - wP_L)\,\mathbf{1}, \\ (D - wP_L)^{-1}\,[(1-w)D + wP_U]\,\mathbf{1} &= \mathbf{1}, \end{aligned}$$

and $\rho(S_w) = \rho(M^{-1}N) = 1$. Hence, by part 8 of Theorem 2.6, the operator S_w is semi-convergent and we have :

$$\lim_{k\to\infty} S_w^k = \mathbf{1}\pi_w^T,$$

where π_w is the stationary probability of S_w.

□

Corollary 5.5. *The iterates $\left(v^{(k)}\right)_{k\geq 0}$ of Equation (5.7) converge to a solution of Equation (5.6).*

Proof. We have that a solution v of Equation (5.6) satisfies

$$v = (I - \mathbf{1}\mu)\,(S_w\,v + wM^{-1}b)\ .$$

Hence, the error $e^{(k)} = v^{(k)} - v$ propagates as

$$e^{(k+1)} = (I - \mathbf{1}\mu)S_w\,e^{(k)}\ .$$

From the proof of Proposition 5.4, we have that S_w is stochastic and primitive. Hence by Lemma 5.3, we get :

$$\rho((I - \mathbf{1}\mu)S_w) < 1.$$

□

Note that in the method described in Equation (5.7), one can use any other vector $\mu \in \mathbb{R}_+^n$ such that $\mu\mathbf{1} = 1$ or a another smoother S which is semi-convergent and the above results old.

From this splitting method, we propose in Algorithm 5.1, an adaptation of the multigrid Vcycle of Algorithm 2.2 to find an approximation of the solution of Equation (5.6). We use the same notation as in Section 2.3.4. As for the non singular case, the multigrid method starts with the construction of the L grids and the corresponding transfer operators, by using for instance one of the methods of Section 2.3.5 or Section 2.3.6. Then, given an initial approximation $v^{(0)}$ on the fine grid, it consists in applying successively the Vcycle of Algorithm 5.1 until the iterates converge to the solution of the system. The resulting algorithm that we shall call AMGsingular, is given in Algorithm 5.2.

Algorithm 5.1 Vcycle for singular system : $u \leftarrow VcycleSingular(u, b)$

if $l < L$ **then**
  pre relaxation :
    $u^{|l} \leftarrow (I^{|l} - \mathbf{1}^{|l}\mu^{|l})(S^{|l}u^{|l} + w(M^{|l})^{-1}b^{|l})$ (on $\Omega^{|l}$) ν_1 times
  coarse grid correction :
    $b^{|l+1} \leftarrow \mathcal{R}_l^{l+1}(b^{|l} - A^{|l}u^{|l})$
    $u^{|l+1} \leftarrow 0$
    $u^{|l+1} \leftarrow VcycleSingular(u^{|l+1}, b^{|l+1})$ ς times
    $u^{|l} \leftarrow u^{|l} + \mathcal{P}_{l+1}^{l}u^{|l+1}$
  post relaxation :
    $u^{|l} \leftarrow (I^{|l} - \mathbf{1}^{|l}\mu^{|l})(S^{|l}u^{|l} + w(M^{|l})^{-1}b^{|l})$ (on $\Omega^{|l}$) ν_2 times
else
  Solve $A^{|L}u^{|L} = b^{|L}$ with $\mu^{|L}u^{|L} = 0$
end if

Other iterative methods can be used to solve non singular consistent systems see for instance [BP94, Saa03]. In the next section, we discuss the second approach to solve Equation (5.1).

5.1.2 Second Approach: Solving the linear system solving by considering that η is a constant vector

The second approach consists in solving directly the linear system :

$$\bar{\eta} + v_i = \sum_{j \in [n]} P_{ij}v_j + r_i \qquad i \in [n] \ , \tag{5.9}$$

Algorithm 5.2 AMG for System (5.6): $v \leftarrow AMGsingular(v^{(0)}, P, b)$

1. Set $P^{|0} = P$ and $A = I - P$.
2. Choose a splitting $A = M - N$ such that $M \in \mathbb{R}_+^{n\times n}$ is invertible and $S \in \mathbb{R}_+^{n\times n}$. The same splitting will be applied on all levels.
3. Construct the coarse grid operators for $l = 1$ to L:
   - Build $\mathcal{P}_l^{l-1}$ and $\mathcal{R}_{l-1}^l$.
   - Set $P^{|l} = \mathcal{R}_{l-1}^l P^{|l-1} \mathcal{P}_l^{l-1}$ and $A^{|l} = I^{|l} - P^{|l}$.
   - Set $S^{|l} = (M^{|l})^{-1} N^{|l}$ where $A^{|l} = M^{|l} - N^{|l}$.
   - Choose $s \in \{1, \cdots, n_l\}$ and set $\mu^{|l} \in \mathbb{R}^n$ such that $\mu_s^{|l} = 1$ and $\mu_i^{|l} = 0$ for $i \neq s$.
4. Initialize $k = 0$.
5. Compute $v^{(k+1)} = VcycleSingular(v^{(k)}, b)$.
6. If $\|b - Av^{(k+1)}\| < \epsilon$ then STOP and return $v^{(k+1)}$.
7. Else, set $k = k + 1$ and go to Step 5.

where $r \in \mathbb{R}^n$, $P \in \mathbb{R}_+^{n\times n}$ is an irreducible stochastic matrix ($P\mathbf{1} = \mathbf{1}$), and $\bar{\eta} \in \mathbb{R}$ and $v \in \mathbb{R}^n$ are the unknowns. Then, the solution η of Equation (5.1) is given by $\eta_i = \bar{\eta}$, $i \in [n]$.

One can for instance use a direct method to find a solution of this system (see for instance [Put94] and hereafter). If we fix the sth-coordinate of v to zero for $s \in [n]$, we can remove v_s from the linear System (5.9). Then, we obtain a non singular linear system with n unknowns. That is, if we choose s to be the first coordinate, the linear System (5.9) becomes :

$$\left(\begin{array}{c|ccc} 1 & A_{12} & \dots & A_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & A_{n2} & \dots & A_{nn} \end{array}\right) \begin{pmatrix} \bar{\eta} \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix}$$

with $A = I - P \in \mathbb{R}^{n\times n}$. Indeed, it is equivalent to :

$$\bar{\eta} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} + \begin{pmatrix} A_{11} & \dots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{n1} & \dots & A_{nn} \end{pmatrix} \begin{pmatrix} 0 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix}$$

and it can be solved for instance with the Gauss elimination algorithm or any algorithm for solving non singular systems.

On the other hand, one may try to use an iterative solver, such as a splitting method, on Equation (5.9). Let us apply, as in the previous section, a splitting method with $(I - P) = M - N$ directly to the system :

$$v = Pv + r,$$

with $P \in \mathbb{R}_+^{n\times n}$ an irreducible stochastic matrix ($P\mathbf{1} = \mathbf{1}$) and $r \in \mathbb{R}^n$, even when r is not in the range of $(I - P)$. This means that we define the $k+1$th-iteration by :

$$v^{(k+1)} \;=\; (I - \mathbf{1}\mu)\,(S\,v^{(k)} \;+\; M^{-1}r),$$

with $S = M^{-1}N$ with $\mu \in \mathbb{R}^n$ such that $\mu_s = 1$ and $\mu_i = 0$ for $i \neq s$. Since the system to be solved is singular and inconsistent, the iterates may not converge. Indeed, a solution of Equation (5.9) satisfies $\bar{\eta}\mathbf{1} + v = Pv + r$ or equivalently $(M - N)v = r - \bar{\eta}\mathbf{1}$, which can be rewritten as :

$$v \;=\; Sv + M^{-1}(r - \bar{\eta}\mathbf{1}).$$

Fixing the s-th coordinate of v to zero, it becomes :

$$v \;=\; (I - \mathbf{1}\mu)\,(Sv + M^{-1}(r - \bar{\eta}\mathbf{1}))\;\;.$$

Hence, the error $e^{(k)} = v^{(k)} - v$ of the splitting method propagates as

$$e^{(k+1)} \;=\; (I - \mathbf{1}\mu)(S\,e^{(k)} + M^{-1}\bar{\eta}\mathbf{1}).$$

Moreover, if $S\mathbf{1} = \mathbf{1}$, by Equations (5.8), we have

$$e^{(k+1)} \;=\; (I - \mathbf{1}\mu)(S^{k+1}\,e^{(0)} + \sum_{i=0}^{k} S^i M^{-1}\bar{\eta}\mathbf{1}).$$

Hence, if we use the SOR splitting of Theorem 5.4, that is S_w, we have that S_w is a stochastic and primitive matrix, so it is semi-convergent and the powers of S_w converge towards $\mathbf{1}\pi_w^T$ where π_w is the stationary probability of S_w. Moreover, by Perron Frobenius theory and the Jordan form, there exists a non singular matrix W such that :

$$S_w^k = W^{-1}\begin{pmatrix} 1 & 0 \\ 0 & B^k \end{pmatrix} W \qquad \text{and} \qquad \mathbf{1}\pi_w^T = W^{-1}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} W,$$

where $B \in \mathbb{R}^{n-1\times n-1}$ and $\rho(B) = \gamma(S_w) < 1$. Hence, there exist a $\gamma(S_w) < c < 1$ such that $S_w^k - \mathbf{1}\pi_w^T = O(c^k)$. It follows that $(I - \mathbf{1}\mu)S_w^k = O(c^k)$ and the limit of $e^{(k+1)}$ exists and is equal to :

$$\lim_{k\to\infty} e^{(k+1)} \;=\; \sum_{i=0}^{\infty}(I - \mathbf{1}\mu)S_w^i M^{-1}\bar{\eta}\mathbf{1}\;\;,$$

and $\mu e = 0$. Since this series may not be equal to zero, the sequence of iterates $v^{(k)}$ converges towards a vector z such that $\mu z = 0$, which is not necessary the solution of Equation (5.9). Note that if we had chosen the splitting $(I - P) = M - N$ with $M = I$ and $N = P$, then $S = P$ and we would get $e = 0$. But for the SOR splitting $e \neq 0$ in general.

An alternative method to solve the linear System (5.9) is to use an iterative solver and iterate on both solutions $\eta \in \mathbb{R}^n$ and $v \in \mathbb{R}^n$. For this purpose, we shall adapt splitting

method explained above. Set $A = I - P$ and split $A = M - N$ where M is invertible. Define the smoother operator by $S = M^{-1}N$. Choose $s \in [n]$. Then, given an initial value $v^{(0)} \in \mathbb{R}^n$ and by setting $\bar{\eta}^{(0)} = \nu\,(r - Av^{(0)})$, the $k+1$th-iteration consists in the following :

$$v^{(k+1)} = (I - \mathbf{1}\mu)(Sv^{(k)} + M^{-1}\,(r - \bar{\eta}^{(k)}\mathbf{1})), \tag{5.10a}$$
$$\bar{\eta}^{(k+1)} = \nu\,(r - Av^{(k+1)}), \tag{5.10b}$$

where $\nu = \frac{1}{n}\mathbf{1}^T$, $\mu \in \mathbb{R}^n_+$ such that $\mu_s = 1$ and $\mu_i = 0$ for $i \neq s$, and $0 < w < 1$.

Theorem 5.6. *Assume that $P \in \mathbb{R}^{n\times n}_+$ is an irreducible stochastic matrix ($P\mathbf{1} = \mathbf{1}$). Let $A = I - P$ and decompose $A = M - N$ such that $M \in \mathbb{R}^{n\times n}_+$ is invertible and $S = M^{-1}N \in \mathbb{R}^{n\times n}_+$. Consider the iterates*

$$v^{(k+1)} = (I - \mathbf{1}\mu)(Sv^{(k)} + M^{-1}\,(r - \bar{\eta}^{(k)}\mathbf{1})), \tag{5.11a}$$
$$\bar{\eta}^{(k+1)} = \nu\,(r - Av^{(k+1)}), \tag{5.11b}$$

where μ, ν are row vectors of $\mathbb{R}^n_+$ such that $\nu = \frac{1}{n}\mathbf{1}^T$, $\mu_s = 1$ and $\mu_i = 0$ for $i \neq s$. Then, the iterates converge to a solution of Equation (5.9) if $\rho((I - \mathbf{1}\nu)NM^{-1}) < 1$.

Proof. Replacing $\bar{\eta}^{(k)}$ in Equation (5.11a), the iterates of Equations (5.11) can be rewritten as :

$$\begin{aligned} v^{(k+1)} &= (I - \mathbf{1}\mu)\,(Sv^{(k)} + M^{-1}\,(r - \mathbf{1}\nu(r - Av^{(k)}))), \\ &= (I - \mathbf{1}\mu)\,(M^{-1}(N + \mathbf{1}\nu A)v^{(k)} + M^{-1}\,(I - \mathbf{1}\nu)r). \end{aligned} \tag{5.12}$$

Moreover, a solution v of Equation (5.9) also satisfies :

$$\begin{aligned} Av - r &= (\nu(Av - r))\mathbf{1}, \\ &= \mathbf{1}\nu(Av - r), \end{aligned}$$

which can be rewritten as :

$$(I - \mathbf{1}\nu)\,Av = (I - \mathbf{1}\nu)\,r, \tag{5.13}$$

that is equivalent to $(M - N - \mathbf{1}\nu A)v = (I - \mathbf{1}\nu)r$ or $v = M^{-1}(N + \mathbf{1}\nu A)v + M^{-1}(I - \mathbf{1}\nu)r$. If in addition, we choose v such that $\mu v = 0$, v satisfies :

$$v = (I - \mathbf{1}\mu)\,(M^{-1}(N + \mathbf{1}\nu A)v + M^{-1}\,(I - \mathbf{1}\nu)r).$$

Hence, the scheme defined by Equation (5.12) amounts to apply the splitting $M-(N+\mathbf{1}\nu A)$ to the operator $(I - \mathbf{1}\nu)A$ of System (5.13). Let

$$E = M^{-1}(N + \mathbf{1}\nu A),$$

then the error $e^{(k)} = v^{(k)} - v$ propagates as

$$e^{(k+1)} = (I - \mathbf{1}\mu)\,E\,e^{(k)} \ .$$

Note that E is not a regular splitting since the term $(N + \mathbf{1}\nu A)$ may not be non negative.

Since $A = I - P$ and $S = (I - M^{-1}A)$, we have that one is an eigenvalue of E and the vector $\mathbf{1}$ is a right eigenvector of E :

$$E\mathbf{1} \;=\; M^{-1}(N + \mathbf{1}\nu A)\mathbf{1} \;=\; S\mathbf{1} \;=\; \mathbf{1}.$$

Moreover, the vector $\nu M \in \mathbb{R}^n$ is a left eigenvector :

$$\begin{aligned} \nu ME &= \nu M M^{-1}(N + \mathbf{1}\nu A), \\ &= \nu N + \nu(M - N), \\ &= \nu M. \end{aligned}$$

The geometric multiplicity of the eigenvalue 1 is equal to one. Indeed, if $z \neq 0$ is such that $Ez = z$, that is $M^{-1}(N + \mathbf{1}\nu(I - P))z = z$, we have :

$$\begin{aligned} \mathbf{1}\nu(I - P)z &= (M - N)z, \\ z &= Pz + \mathbf{1}\nu(I - P)z, \end{aligned}$$

Since P is a stochastic matrix, we have that 1 is a simple eigenvalue of P and hence z is proportional to $\mathbf{1}$. Let us show that the algebraic multiplicity of the eigenvalue 1 is also equal to one. Therefore, it remains to show that $\ker(I - E)^2 = \ker(I - E)$. Assume that it is not the case, then there exists a non zero vector $u \in \ker(I - E)^2 \setminus \ker(I - E)$. Hence, we have that $(I - E)u \in \ker(I - E) \setminus \{0\}$, that implies that

$$Eu \;=\; u + c\mathbf{1} \qquad c \neq 0. \tag{5.14}$$

Left multiply both side by νM, the left eigenvector of E, we obtain $c\nu M\mathbf{1} = 0$. Hence, since $M \in \mathbb{R}_+^{n\times n}$ is invertible, the last equation is impossible. Hence, the eigenvalue 1 is a simple eigenvalue of E.

Let us show now that $\gamma(E) < 1$. First, we rewrite E :

$$\begin{aligned} E &= M^{-1}(N + \mathbf{1}\nu M\,(I - S)), \\ &= S + M^{-1}\mathbf{1}\nu M\,(I - S), \\ &= (I - M^{-1}\mathbf{1}\nu M)S + M^{-1}\mathbf{1}\nu M, \end{aligned} \tag{5.15}$$

where $M^{-1}\mathbf{1}\nu M = (M^{-1}\mathbf{1}\nu M)^2$ and $(I - M^{-1}\mathbf{1}\nu M)$ are projections. Setting $K = (I - M^{-1}\mathbf{1}\nu M)$, we have that $\mathrm{Ker}(\nu M) = \mathrm{Ker}(M^{-1}\mathbf{1}\nu M) = \mathrm{Range}(K)$ and $\mathrm{Range}(I - K) = \mathrm{Range}(M^{-1}\mathbf{1}) = \mathrm{Ker}(K)$.

From Equation (5.15), we have that $\mathrm{Range}(K) = \mathrm{Ker}(\nu M)$ is invariant by E. Let $\lambda \in \mathbb{C}$ and $u \in \mathbb{C}^n$ be such as $Eu = \lambda u$. Multiplying both sides by νM, we obtain $\nu M(E - \lambda I)u = 0$. Since $\nu ME = \nu M$, we get that $(1 - \lambda)\nu Mu = 0$. Hence, if $\lambda \neq 1$, we obtain that $u \in \mathrm{Ker}(\nu M)$, hence u is also an eigenvector of $(I - M^{-1}\mathbf{1}\nu M)S$ for the eigenvalue λ. Since the operator $(I - M^{-1}\mathbf{1}\nu M)S$ is similar to $(I - \mathbf{1}\nu)NM^{-1}$, they have the same eigenvalues. Hence, by assumption, we get that

$$\gamma(E) = \rho((I - \mathbf{1}\nu)NM^{-1}) < 1.$$

By Lemma 5.3 applied to E, we get that $\rho((I - \mathbf{1}\mu)E) < 1$. Hence, the iterates of Equations (5.11) converge to a solution of Equation (5.9)

□

Note that in Theorem 5.6, one can possibly find a weaker condition on the smoother S or on the operator M such that the eigenvalue 1 of the operator E is simple. To have the convergence of the scheme given in Theorem 5.6, it remains to show that $\rho((I - \mathbf{1}\nu)NM^{-1}) < 1$. In the following corollaries, we show some particular cases in which it is true.

Corollary 5.7. *Assume that $P \in \mathbb{R}_+^{n\times n}$ is an irreducible stochastic matrix ($P\mathbf{1} = \mathbf{1}$). Let $A = I - P$ and decompose $wA = w(I - P) = M - N$ such that $M = I$ and $N = (1 - w)I + wP$. Let $S = M^{-1}N$. Then, replacing A by wA and r by wr in Theorem 5.6, the iterates of Equations* (5.11) *converge to a solution of Equation* (5.9)*.*

Proof. We have that $S = M^{-1}N = N$ is primitive. Hence, $\rho(N) = 1$, $\gamma(N) < 1$ and 1 is a simple eigenvalue of N. By Lemma 5.3, we have that $\rho((I - \mathbf{1}\nu)N) < 1$. By Theorem 5.6, we have the convergence of the iterates. □

Corollary 5.8. *Assume that $P \in \mathbb{R}_+^{n\times n}$ is an irreducible bistochastic matrix ($P\mathbf{1} = \mathbf{1}$ and $\mathbf{1}P^T = \mathbf{1}$). Let $A = I - P$ and decompose $wA = w(I - P) = M - N$ such that $M = (D - wP_L)$ and $N = ((1 - w)D + wP_U)$ with $D = I - P_D$. Let $S = M^{-1}N$. Then, replacing A by wA and r by wr in Theorem 5.6, the iterates of Equations* (5.11) *converge to a solution of Equation* (5.9)*.*

Proof. Since, P is bistochastic, we have that $\mathbf{1}^T A = \mathbf{0}$ or equivalently $\mathbf{1}^T M = \mathbf{1}^T N$, hence $\mathbf{1}^T NM^{-1} = \mathbf{1}^T$, i.e. NM^{-1} is a column stochastic matrix. Moreover, NM^{-1} is similar to S_w which is primitive by the proof of Proposition 5.4, hence $(NM^{-1})^T$ is also primitive. By Lemma 5.3, we have that $\rho(NM^{-1}(I - \mathbf{1}\nu)) = \rho((I - \mathbf{1}\nu)(NM^{-1})^T) < 1$. We have that $((I-\mathbf{1}\nu)NM^{-1})^k = (I-\mathbf{1}\nu)(NM^{-1}(I-\mathbf{1}\nu))^{k-1}NM^{-1}$. Hence $\lim_{k\to\infty}((I-\mathbf{1}\nu)NM^{-1})^k = 0$ and $\rho((I - \mathbf{1}\nu)(NM^{-1})) < 1$. By Theorem 5.6, we have the convergence of the iterates. □

From the smoothing process given by Equation (5.10), we define a new multigrid algorithm that is given in Algorithm 5.3 and that we shall call AMGsingular2. It consists in replacing the first equation of Equation (5.10) by one multigrid Vcycle for singular system defined in Algorithm 5.1, that is replacing the splitting operator S by the operator of a multigrid V-cycle. Note that since we did not succeed to prove the convergence of the smoothing process given by Equation (5.10) in the general case, we do not have the convergence multigrid scheme AMGsingular2. We observed that for some random problems, it may not converge in all cases, see for instance the numerical results of Section 5.2. But for some regular problems, such as arising in pursuit differential games, we observed convergence of AMGsingular2 in all our tests, see for instance the numerical results in Section 5.3.

Algorithm 5.3 AMG for Equation (5.9): $(v, \bar{\eta}) \leftarrow AMGsingular2(v^{(0)}, A, r, k_M)$

1. Set $P^{|0} = P$ and $A = I - P$.
2. Choose a splitting $A = M - N$ such that $M \in \mathbb{R}_+^{n\times n}$ is invertible and $S \in \mathbb{R}_+^{n\times n}$. The same splitting will be applied on all levels.
3. Construct the coarse grid operators for $l = 1$ to L:
   - Build $\mathcal{P}_l^{l-1}$ and $\mathcal{R}_{l-1}^l$.
   - Set $P^{|l} = \mathcal{R}_{l-1}^l P^{|l-1} \mathcal{P}_l^{l-1}$ and $A^{|l} = I^{|l} - P^{|l}$.
   - Set $S^{|l} = (M^{|l})^{-1} N^{|l}$ where $A^{|l} = M^{|l} - N^{|l}$.
   - Choose $s \in \{1, \cdots, n_l\}$ and set $\mu^{|l} \in \mathbb{R}^n$ such that $\mu_s^{|l} = 1$ and $\mu_i^{|l} = 0$ for $i \neq s$.
4. Set $\nu = \frac{1}{n}\mathbf{1}^T$.
5. Initialize $k = 0$ and set $\bar{\eta}^{(0)} = \nu\,(r - Av^{(0)})$.
6. Set $b^{(k)} = r - \bar{\eta}^{(k)}$.
7. Compute $v^{(k+1)} = VcycleSingular(v^{(k)}, b^{(k)}, \mu)$.
8. Set $\bar{\eta}^{(k+1)} = \nu\,(r - Av^{(k+1)})$.
9. If $(\|(r - Av^{(k+1)}) - \bar{\eta}^{(k+1)}\mathbf{1}\| < \epsilon$ or $k = (k_M - 1))$ then STOP, return $v^{(k+1)}$ and $\bar{\eta}^{(k+1)}$.
10. Else, increment k by one and go to Step 6.

In the next section, we introduce a new approach to find the stationary probability of an irreducible Markov Chain by using a stochastic control approach, we define a policy iteration algorithm combined with the multigrid algorithm AMGsingular2. In Section 5.3, we present numerical results for some pursuit evasion games using the policy iteration algorithm for zero-sum two player stochastic games combined with the multigrid algorithm AMGsingular2.

5.2 Stochastic control for the stationary probability of an irreducible Markov Chain

Here, we present a multiplicative multigrid method to find the stationary probability of an irreducible Markov Chain which is based on a control approach and the multigrid algorithm AMGsingular2. Recall that we have to find $\pi \in \mathbb{R}_+^n$ solution of

$$\pi^T P = \pi^T, \qquad \pi^T \mathbf{1} = 1 \tag{5.16}$$

where $P \in \mathbb{R}_+^{n\times n}$ is a row stochastic and $\mathbf{1} = (1 \cdots 1)^T$ is the vector of ones in $\mathbb{R}^n$. Since $\pi \in (\mathbb{R}_+^*)^n$, we can set

$$\pi = \mathrm{Exp}(v) \qquad v \in \mathbb{R}^n,$$

where $\mathrm{Exp}(v) := (\exp(v_i))_{i\in[n]}$. Then, Equation (5.16) is equivalent to :

$$v_i \;=\; \log\left(\sum_{j\in[n]} \exp(v_j)\, P_{ji}\right) \quad := f_i(v) \qquad i \in [n].$$

One can check using Holder inequality that the maps $f_i : \mathbb{R}^n \to \mathbb{R}$ for $i \in [n]$, are all convex, hence

$$f_i(v) \;=\; \sup_{u\in\mathbb{R}^n} (uv \;-\; f_i^*(u)) \qquad v \in \mathbb{R}^n,\, i \in [n], \tag{5.17}$$

where $u \in \mathbb{R}^n$ is considered as a row vector and f^* is the Legendre-Fenchel transformation of f (see [Roc70]). The ith-coordinate of the map f^* is given by :

$$\begin{aligned} f_i^*(u) \;&=\; \sup_{w\in\mathbb{R}^n} (u\,w \;-\; f_i(w)) \\ &=\; \sup_{w\in\mathbb{R}^n} \left[u\,w \;-\; \log\left(\sum_{k\in[n]} \exp(w_k)\, P_{ki}\right)\right]. \end{aligned} \tag{5.18}$$

Since f_i is differentiable, an optimal w in the previous expression satisfies necessary:

$$u_j \;=\; \frac{\exp(w_j)P_{ji}}{\displaystyle\sum_{k\in[n]} \exp(w_k)\, P_{ki}} \qquad \text{for } j \in [n]. \tag{5.19}$$

Hence, an optimum in Equation (5.18) exists only if $u \in \mathcal{A}_i$ where

$$\mathcal{A}_i := \left\{u \in \mathbb{R}_+^n \;\middle|\; u\mathbf{1} = 1 \text{ and } u_j = 0 \text{ if } P_{ji} = 0,\; j \in [n]\right\}.$$

Indeed, since f_i is non decreasing with respect to v and is additively homogeneous, $f_i^*(u) = -\infty$ if u is not a stochastic vector (see for instance [AG03]). Moreover, if $P_{ji} = 0$, $f_i(w)$ does not depend on w_j and $f_i^*(u) = +\infty$ if there exists $j \in [n]$ such that $u_j \neq 0$. When $u \in \mathcal{A}_i$, we get that :

$$w_j \;=\; \log\left(\frac{u_j}{P_{ji}}\right) + f_i(w) \qquad \text{for } j \in [n] \text{ s.t. } P_{ji} \neq 0,$$

and replacing this expression in (5.18), we get that :

$$f_i^*(u) \;=\; \sum_{j\in[n], P_{ji}\neq 0} \log\left(\frac{u_j}{P_{ji}}\right) u_j \qquad u \in \mathcal{A}_i,\, i \in [n].$$

This map is the relative entropy of the probability vector u with respect to the positive measure $(P_{ji})_{j\in[n]}$. Note that, contrary to the usual definition of the relative entropy, $(P_{ji})_{j\in[n]}$ is not necessarily a probability vector. From this Formula, Equation (5.17) can be rewritten as

$$f_i(v) \;=\; \sup_{u\in\mathcal{A}_i} \left(u\,v \;-\; \sum_{\substack{j\in[n],\\ P_{ji}\neq 0}} \log\left(\frac{u_j}{P_{ji}}\right) u_j\right), \qquad v \in \mathbb{R}^n, i \in [n].$$

Now, for any row vector $a \in \mathcal{A}_i$, we denote its jth-element by $M^a_{ij} (= a_j)$, hence we can rewrite f as

$$f_i(v) = \max_{a \in \mathcal{A}_i} \left(\sum_{j \in [n]} M^a_{ij} v_j - f^*_i(a) \right), \qquad v \in \mathbb{R}^n, i \in [n]. \tag{5.20}$$

Since $a \in \mathcal{A}_i$ is a stochastic row vector, M^a_{ij} can be consider as the probability transition from state i to state j, where the action in state i is a. Hence, the map f is the dynamic programming operator of a Markov decision process with transition probability $p(j|i,a) = M^a_{ij}$ and reward $r^a_i = f^*_i(a)$ when the state is i and the action is a. The set of feedback policies is given by

$$\mathcal{A}_M := \{\alpha : [n] \to \mathcal{A} \mid \alpha(i) \in \mathcal{A}_i, \ i \in [n]\}.$$

The transition probability matrix $M^{(\alpha)}$ associated to a policy $\alpha \in \mathcal{A}_M$ is defined by $M^{(\alpha)} = (M^{\alpha(i)}_{ij})_{i,j \in [n]}$. and has a graph included in that of P^T. Moreover, for each $v \in \mathbb{R}^n$, the optimal policy $\alpha \in \mathcal{A}_M$ associated to v satisfies Equation (5.19) where w is replaced by v and $u = \alpha(i)$ (by duality, see [Roc70]). It implies that the transition probability matrix $M^{(\alpha)}$ is given by

$$M^{(\alpha)}_{ij} = M^{(\alpha(i))}_{ij} = (\alpha(i))_j = \frac{\exp(v_j) P_{ji}}{\sum_{k \in [n]} \exp(v_k) P_{ki}} \qquad \text{for } i, j \in [n].$$

and the associated rewards are given by

$$r^{(\alpha)}_i = -f^*_i(\alpha(i)) = \sum_{j \in [n], P_{ji} \neq 0} \log \left(\frac{M^{(\alpha)}_{ij}}{P_{ji}} \right) M^{(\alpha)}_{ij} \qquad i \in [n].$$

Hence, for any optimal policy α associated to a value v, the associated matrix $M^{(\alpha)}$ has the same graph as P^T. Then, one can restrict the optimization in Equation (5.20) to the relative interior of $\mathcal{A}_i$ which is the set of vector u stochastic with the same support as $(P_{ji})_{j \in [n]}$.

Now, we can find the fixed point $v \in \mathbb{R}^n$ of f by using the policy iteration algorithm for Markov Decision Processes with mean payoff of [How60, DF68]. In this case, the fixed point v of f is the relative value and the optimal mean payoff η of the MDP equals zero. Since all the involved matrices are irreducible, it is enough to look for a $\eta = \bar{\eta}\mathbf{1}$ with $\bar{\eta} \in \mathbb{R}$, hence to solve the equation $\bar{\eta}\mathbf{1} + v = f(v)$. We give in Algorithm 5.4 the policy iteration algorithm to solve Equation (5.20). In the policy evaluation step, we use the multigrid algorithm AMGsingular2 (Algorithm 5.3) to solve the linear system. Hence, the resulting algorithm that we shall call MCPIMG, is a new multigrid algorithm to find the stationary probability of an irreducible Markov Chain. We do not prove convergence for this algorithm since we do not have the convergence of AMGsingular2. As explained in the next section, we did not succeed to obtain convergence for numerical tests using AMGsingular for some random problems but when replacing the multigrid algorithm by the relaxation scheme of Equation (5.10) with a SOR splitting, we observed convergence in all cases. Hence, the multigrid algorithm AMGsingular2 still need some improvements.

Algorithm 5.4 Policy iteration for Markov Chains : $MCPIMG$

Input: A vector $\pi^{(0)}$ such that $\pi^{(0)} \in (\mathbb{R}_+^*)^n$, $(\pi^{(0)})^T \mathbf{1} = 1$ and an integer $k_M > 0$.

Output: An approximation of the stationary probability $\pi \in (\mathbb{R}_+^*)^n$ solution of $\pi^T P = \pi^T$ with $\pi^T \mathbf{1} = 1$.

1. *Initialization*: Set $k = 0$ and $v^{(0)} = \mathrm{Log}(\pi^{(0)})$ (where $\mathrm{Log}(\pi) := (\log(\pi_i))_{i\in[n]}$).
2. *Improve the policy by computing* :

$$M_{ij} = \frac{\pi_j^{(k)} P_{ji}}{(\pi^{(k)})^T P_{\cdot i}} \qquad i,j \in [n]$$

and

$$r_i = -\sum_{j\in[n]} \left(\log\left(\frac{\pi_j^{(k)}}{(\pi^{(k)})^T P_{\cdot i}} \right) M_{ij} \right) \qquad i \in [n].$$

3. *Policy evaluation*: find an approximation of the solution $(\bar{\eta}, v) \in \mathbb{R} \times \mathbb{R}^n$ of

$$\bar{\eta}\, \mathbf{1} + v = Mv + r \qquad \bar{\eta} \in \mathbb{R}. \tag{5.21}$$

by using a linear solver, for instance :

$$(v^{(k+1)}, \bar{\eta}^{(k+1)}) \leftarrow AMGsingular2(v^{(k)}, M, r, k_M)$$

4. Set $\pi^{(k+1)} = \mathrm{Exp}(v^{(k+1)})$.
5. If $\left\| \left(\frac{\pi^{(k+1)}}{(\pi^{(k+1)})^T \mathbf{1}} \right)^T P - \left(\frac{\pi^{(k+1)}}{(\pi^{(k+1)})^T \mathbf{1}} \right)^T \right\| < \epsilon$ then the algorithm stops and returns $\frac{\pi^{(k+1)}}{(\pi^{(k+1)})^T \mathbf{1}}$.
6. Otherwise, increment k by one and go to Step 2.

5.2.1 Numerical tests

In this section, we shall compare our algorithm MCPIMG (Algorithm 5.4) with the algorithm MAA of [DSMM$^+$08] (Algorithm 2.13). Let us first give some details about the parameters of both methods. The implementation is in C.

In MAA and in the multigrid solver AMGsingular2 of MCPIMG, we use $W(1,2)$-cycles (i.e. $\varsigma = 2$, $\nu_1 = 1$ and $\nu_1 = 2$ in Algorithm 5.1 and Algorithm 2.13). For MAA, we use the SOR smoother adapted to Markov Chains of Equations (5.4) with $w = 0.8$. For algorithm AMGsingular2 (in MCPIMG), we use the SOR smoother of Equations (5.10) also with $w = 0.8$. For both algorithms, the initial approximation is the vector $(\frac{1}{n} \cdots \frac{1}{n}) \in \mathbb{R}^n$ when the size of the problem is n. The iterations of MAA and MCPIMG stop when the sup-norm of the residual is smaller than 10^{-14}, that is $\|(\pi^{(k+1)})^T P - (\pi^{(k+1)})^T\| < 10^{-14}$. We use the same type of aggregation scheme for both MAA and MCPIMG algorithms.

When we performed the tests using the MCPIMG algorithm, we encountered some difficulties of convergence due to the fact that the multigrid algorithm AMGsingular2 may slightly diverge. When instead of the multigrid solver, we used the SOR smoother of Equations (5.10), we observed that the MCPIMG algorithm always converges. However, after experimental tests, the divergence or slowly convergence of the multigrid solver occurs in the first policy iterations only. Moreover, stopping the multigrid iterations and go forward with the policy iterations does not affect the convergence of the policy iterations and we observed that it is more efficient to fix the number of iterations of the multigrid solver. Indeed in the first iterations, the policy evaluation step does not need to be accurate. This is due to the fact that, the maps f_i being regular, the policy iteration algorithm is equivalent to the Newton method, and in this case, the number of multigrid iterations should even increase exponentially with policy iterations (see for instance [Aki90a]). Hence, we fixed the maximum number of $W(1,2)$-cycles iterations for each policy evaluation step in MCPIMG to five ($k_M = 5$) and the precision for the multigrid solver to $\epsilon = 10^{-14}$.

Random walks on a square grid

Here, we present some numerical results on random matrices that represent random walks on an uniform square grid. This problem is inspired from an example of the Markov model of a random walk on a triangular grid from the book of [Saa92]. In our case, we consider a square grid, represented in Figure 5.1 for 25 nodes, and a particle that moves randomly on that graph from one node to one of its neighbors following the directed edges. The transition probabilities between each node and its neighbors are chosen randomly. The numbering of the nodes goes from bottom to top and from left to right. Then, the corresponding transition probability matrix $P \in \mathbb{R}_+^{n\times n}$, whose elements P_{ij} are the probabilities that the particle jumps from node i to node j for $i, j \in [n]$, has a sparsity pattern represented in Figure 5.1 for $n = 25$.

The numerical results for random Markov walk on graphs are presented in Figure 5.2 and Figure 5.3 for a sample of 901 tests. We choose random probability transitions for

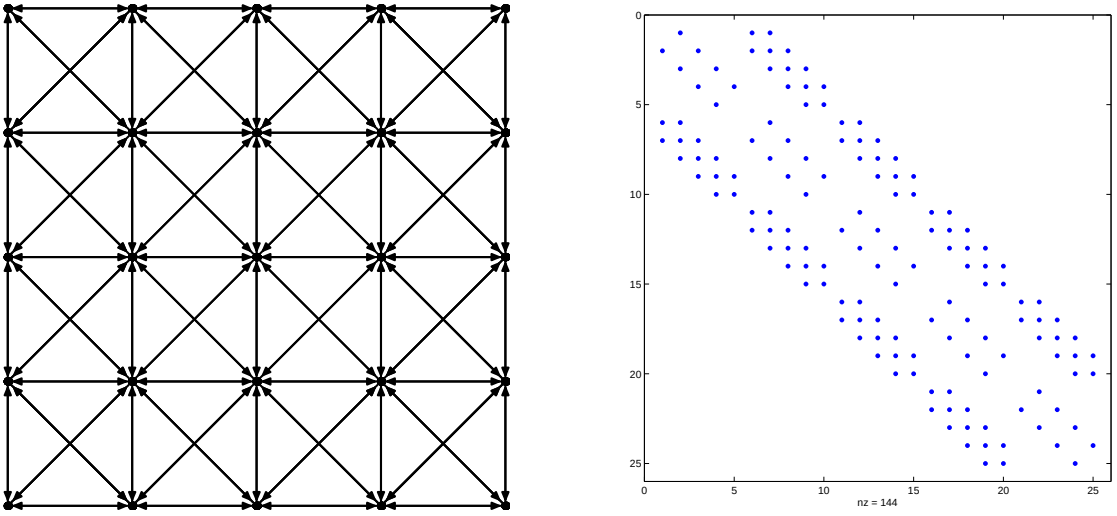

Figure 5.1: On the left, a random walk on a 5×5 square grid and on the right, the sparsity pattern of the corresponding transition probability matrix.

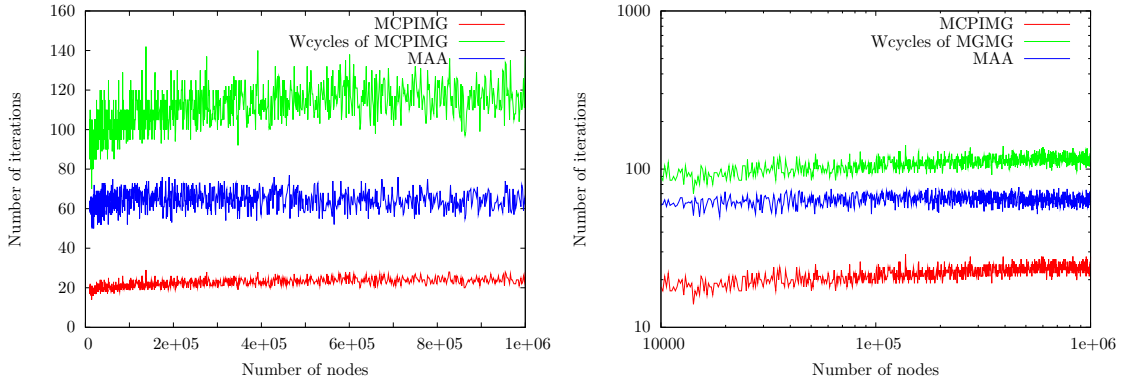

Figure 5.2: Number of iterations, given in log-scale on the right.

square grid graphs having a number of nodes n between 100^2 and 1000^2. We increase by one the number of points in each direction of the grid from 100 to 1000. In Figure 5.2, we draw the number of iterations taken by both algorithms, we observe that the MCPIMG take between 14 and 29 policy iterations with a mean value of 22.29, and the total number of $W(1,2)$-cycles iterations is between 70 and 142 with a mean value of 110.09. The number of $W(1,2)$-cycles for MAA is between 50 and 70 with a mean value of 64.40. For both algorithms, when the test problems have large sizes, the number of iterations oscillate around a constant value which is expected for a multigrid scheme.

In Figure5.3, we represent the cpu time (in seconds) needed by the MCPIMG and MAA to find the stationary probability for the 901 tests of random walks. We can see that MAA needs more time to solve a problem than MCPIMG, however when the size of the problems increase, the two curves merge. If we consider only the problems of size greater than 150000, the relationship between the logarithm of the computation time and the logarithm of the size of the problem is almost linear for both algorithms with slope

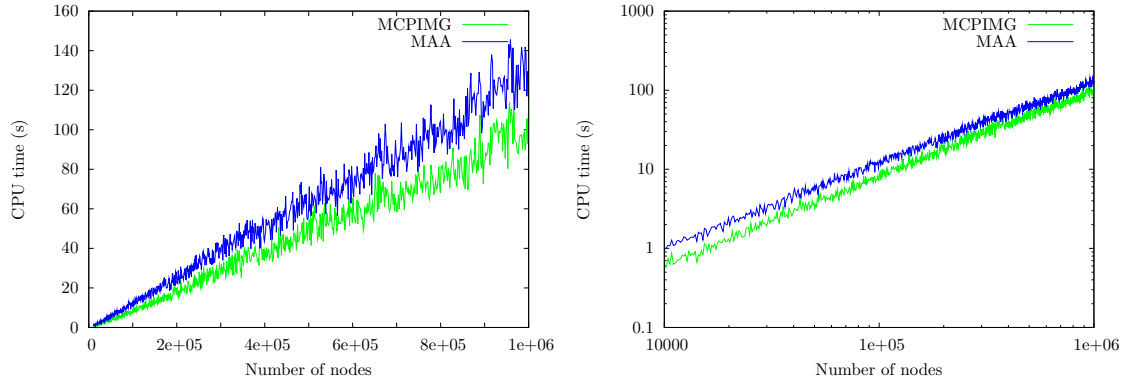

Figure 5.3: Time, given in log-scale on the right.

about 1.05 for MCPIMG and a slope of 0.99 for MAA. Hence, the computation time of both algorithms seems to grow linearly with respect to the size of the problem, which is expected for multigrid algorithms.

Comparing both algorithms, we conclude that they share both the properties expected for multigrid algorithms. Note that some acceleration algorithms exist for the MAA method in [DSMM$^+$10a, DSMM$^+$10b, DSMMS11, DSMSW10]. The MCPIMG may also need some improvements by determining the best parameter k_M multigrid iterations or by increasing the number of multigrid iterations exponentially (as for instance in [Aki90a]) and by improving the AMGsingular2 algorithm.

5.3 Ergodic differential stochastic games : Isaacs equation

In this section, we shall apply policy iterations combined with the AMGsingular2 algorithm (Algorithm 5.3) to solve some ergodic differential games with mean payoff.

We first make some recalls about ergodic differential stochastic games with mean payoff, i.e. the value of the game η is independent of the initial state, and their associated Bellman-Isaacs equation. We refer also to [AB07, Bar09] for precise definitions of these games that are called ergodic differential stochastic games with long-time-average reward.

5.3.1 Isaacs equations for mean payoff differential games

As in Section 1.3, we assume that the state space is a regular open subset $\mathcal{X}$ of $\mathbb{R}^d$ and we suppose a probability space Ω is given, as well as a filtration $(\mathcal{F}_t)_{t\geq 0}$ over it. We consider games which dynamics is governed by the following stochastic differential equation :

$$d\xi_t \;=\; g(\xi_t, \zeta_t, \eta_t)\, dt + \sigma(\xi_t, \zeta_t, \eta_t)\, dW_t, \tag{5.22}$$

with initial state $\xi_0 = x \in \mathcal{X}$. Here W_t is a d'-dimensional Wiener process on $(\Omega, (\mathcal{F}_t)_{t\geq 0})$; ζ_t and η_t are stochastic processes taking values in closed subsets $\mathcal{A}$ and $\mathcal{B}$ of $\mathbb{R}^p$ and $\mathbb{R}^q$ respectively; $(x, a, b) \in \mathcal{X} \times \mathcal{A} \times \mathcal{B} \mapsto g(x, a, b) \in \mathbb{R}^d$ and $(x, a, b) \in \mathcal{X} \times \mathcal{A} \times \mathcal{B} \mapsto \sigma(x, a, b) \in \mathbb{R}^{d\times d'}$ are given functions. The processes ζ_t and η_t are assumed to be adapted

to the filtration $(\mathcal{F}_t)_{t\geq 0}$ with $\zeta_t \in \mathcal{A}(\xi_t)$ and $\eta_t \in \mathcal{B}(\xi_t)$ for all $t \geq 0$, where $\mathcal{A}(x) \subset \mathcal{A}$ and $\mathcal{B}(x) \subset \mathcal{B}$ are the sets of actions of the first and second players respectively when the state of the game is equal to $x \in \mathcal{X}$. We also consider strategies $\bar{\alpha} = (\alpha_t)_{t\geq 0}$ (resp. $\bar{\beta} = (\beta_t)_{t\geq 0}$) of player MAX (resp. MIN) determining the process $(\zeta_t)_{t\geq 0}$ (resp. $(\eta_t)_{t\geq 0}$). In particular, for pure Markovian stationary strategies, one has $\zeta_t = \alpha(\xi_t)$ and $\zeta_t = \beta(\xi_t, \zeta_t)$.

When the strategies $\bar{\alpha}$ for MIN and $\bar{\beta}$ for MAX are fixed, the payoff in finite horizon τ of the game $\Gamma(x, \alpha, \beta)$ starting from x is

$$J^\tau(x; \alpha, \beta) = \mathbb{E}_x^{\alpha,\beta}\left[\int_0^\tau r(\xi_t, \zeta_t, \eta_t)dt\right]$$

where $\mathbb{E}_x^{\alpha,\beta}$ denotes the expected value and r is the running reward. The long-time-average reward of the game $\Gamma(x, \alpha, \beta)$ is

$$J(x; \alpha, \beta) = \limsup_{\tau\to\infty} \frac{1}{2} J^\tau(x; \alpha, \beta) \ .$$

The mean payoff of the stochastic differential game starting at x is given by

$$\eta(x) = \inf_\alpha \sup_\beta J(x; \alpha, \beta)$$

where the infimum is taken among all strategies α for the minimizing player and the supremum is taken over all strategies β for the maximizing player.

Now, we consider the second order Hamiltonian given by

$$H(x, p, K) = \min_{a\in\mathcal{A}(x)} \max_{b\in\mathcal{B}(x)} \left[\frac{1}{2}\mathrm{tr}(\sigma\sigma^T(x, a, b)\, K) + p \cdot g(x, a, b) + r(x, a, b)\right]$$

where $x \in \mathcal{X}, p \in \mathbb{R}^n, K \in \mathbb{R}^{n\times n}$. We denote by $D_x v$ (resp. $D_x^2 v$) the vector (resp. matrix) of the first (resp. second) partial derivatives of v. Then assuming regularity assumptions on $\mathcal{X}$ and on the functions g, σ and r, when $\eta(x)$ is independent of the starting point x, the mean payoff η of the game is the unique solution of the following dynamic programming equation (see [AB07, Bar09]) :

$$\eta - H(x, D_x v, D_x^2 v) = 0, \quad x \in \mathcal{X}. \tag{5.23}$$

This partial differential equation is called Bellman-Isaacs equation. For problems with mean-payoff, one can have either natural boundary conditions as in Section 4.7.2 (which means that (5.23) is fulfilled up to the boundary of $\mathcal{X}$, but since the sets $\mathcal{A}(x)$ and $\mathcal{B}(x)$ may be different there, the Hamiltonian H is not necessarily continuous), or Neuman boundary conditions as in the following section (which means that Equation (5.22) is modified accordingly), or periodic boundary conditions.

In the general case, the mean payoff $\eta(x)$ may depend on the initial state x, then, as in Section 4.7.2, Equation (5.23) should be replaced by a system of equations involving η and v similar to the discrete equation (4.14). First, we define some notation. For an action $a \in \mathcal{A}(x)$ and an action $b \in \mathcal{B}(x)$, we denote

$$H^{ab}(x, p, K) := p \cdot g(x, a, b) + \frac{1}{2}\mathrm{tr}(\sigma\sigma^T(x, a, b)\, K) \qquad x \in \mathcal{X} \ .$$

We define the Hamiltonian $\hat{H}$ by :

$$\hat{H}(x,p,K) := \min_{a\in\mathcal{A}(x)} \max_{b\in\mathcal{B}(x)} H^{ab}(x,p,K) \qquad x\in\mathcal{X}$$

and the Hamiltonian $\acute{H}_\eta$ by :

$$\acute{H}_\eta(x,p,K) := \min_{a\in\acute{\mathcal{A}}_\eta(x)} \max_{b\in\acute{\mathcal{B}}_\eta(x,a)} \Big[H^{ab}(x,p,K) + r(x,a,b)\Big] \qquad x\in\mathcal{X}$$

where

$$\acute{\mathcal{A}}_\eta(x) := \operatorname*{argmin}_{a\in\mathcal{A}(x)} \left\{ \max_{b\in\mathcal{B}(x)} H^{ab}(x, D_x\eta, D_x^2\eta) \right\}$$

and

$$\acute{\mathcal{B}}_\eta(x,a) := \operatorname*{argmax}_{b\in\mathcal{B}(x)} \left\{ H^{ab}(x, D_x\eta, D_x^2\eta) \right\} \ .$$

Then, one is looking for a couple of values (η, v) satisfying the system of dynamic programing equations given by :

$$\left\{ \begin{array}{rcll} \hat{H}(x, D_x\eta, D_x^2\eta) & = & 0 & x\in\mathcal{X} \\ -\eta(x) + \acute{H}_\eta(x, D_x v, D_x^2 v) & = & 0 & x\in\mathcal{X}. \end{array} \right. \tag{5.24}$$

Clearly, due to the definition of $\acute{H}_\eta$, η needs to be a $\mathcal{C}^2$ function, however the solution v of the second equation of System (5.24) may be considered in the viscosity sense. When a solution (η, v) is such that both η and v are $\mathcal{C}^2$ functions, one may expect that η gives the mean payoff of the game.

The discretization of Bellman-Isaacs Equation (5.23) or (5.24) with monotone schemes in the sense of [BS91], e.g. the difference scheme of [Kus77] (see Section 1.3.3), yields the dynamic programming operator of a two player zero-sum stochastic game with discrete state space and mean payoff. Hence the policy iteration algorithm of Section 4.4 can be used to solve the resulting discrete equations.

In Section 4.7.2, we already solved an equation of the form (5.24), whereas below we shall consider an ergodic stochastic game, so that Equation (5.23) will be sufficient.

5.3.2 Numerical results

We consider almost the same game model as in Section 4.7.2, with the difference that here we assume as in [LCS08] that the movements are random by adding a fixed noise to the differential equations. Recall that the game has two players : a pursuer and an evader. The evader wants to maximize the distance between him and the pursuer and the pursuer has the opposite objective. To simplify the model, we will consider as state of the game, the distance between the two players, i.e. the state of the game will be given by $x = x_P - x_E$ where x_P is the position of the pursuer and x_E the position of the evader. We also restrict x to stay in a unit square centered in the 0-position, that is $x\in\mathcal{X} :=]-0.5, 0.5[\times]-0.5, 0.5[$. The payoff of the game at each time is the euclidean square norm of the distance between the two players, i.e. $\|x\|_2^2$.

Since the noise ϵ is constant, the optimal mean-payoff $\eta(x)$ does not depend on the initial position x. After discretization, the stochastic matrix P in System (4.39), associated to any couple of possible stationary strategies, is irreducible. Then the Bellman-Isaacs equation is of the form (5.23) and is given in the present problem by

$$-\eta + \epsilon\Delta v + \max_{a\in\mathcal{A}}(a\cdot\nabla v) + \min_{b\in\mathcal{B}}(b\cdot\nabla v) + \|x\|_2^2 = 0 \text{ in } \mathcal{X} \tag{5.25}$$

where $\mathcal{A}$ and $\mathcal{B}$ are the set of possible directions respectively for the evader and the pursuer. In our tests, the domain $\mathcal{X}$ is discretized in each directions with a constant step size h. Then, the two players are moving on the discretized nodes of the domain, similarly to the moves in a chess game.

The king and the horse

We performed the tests of this section by using the policy iterations for mean payoff two player games (see Algorithm 4.2 where the Step 5 is never visited since we have only one irreducible class). In the policy evaluation step, we use AMGsingular2 (Algorithm 5.3) to compute the couple of value η and relative value v of the game. We use the construction of grids and coarse operators of [RS87, BHM00] which is recalled in Algorithm 2.6 of Section 2.3.5 with the interpolation weights defined by Equation (2.15). We use $V(1,2)$-cycles and the stopping criterion for the iterations of AMGsingular2 is $\|r\|_\infty < 10^{-15}$ where r is the residual for the linear system.

In the first example, we solve Equation (5.25) with $\epsilon = 0.5$ and Neumann boundary conditions. The evader is called the king and his set of possible actions at each state of the game is :

$$\mathcal{A} := \{(a_1, a_2)\,|\,a_i \in \{0, 1, -1\},\quad i = 1, 2\}.$$

The pursuer, called the horse, has the following set of possible actions :

$$\mathcal{B} := \{(0,0), (\pm1, \pm2), (\pm2, \pm1)\}.$$

Note that contrarily to the example of Section 4.7.2, the sets $\mathcal{A}$ and $\mathcal{B}$ do not depend on the state $x \in \mathcal{X}$ and so are not reduced to $\{(0,0)\}$ around 0, so that the game does not stop when the horse catches the king. The numerical results presented in Table 5.1, were performed on Equation (5.25) discretized on a grid with a constant step size $h = 1/256$ in each directions. The optimal actions for the discretized problem at each node of the grid, are represented in Figure 5.6 for both players. The optimal mean payoff for the discretized problem is : $\eta(x) = 0.141\ \forall\, x \in \mathcal{X}$ and the relative value v is represented in Figure 5.5 on the left.

For the second example, we consider the same problem but we limited the actions of the horse, which are now :

$$\mathcal{B} := \{(0,0), (1,2), (2,1)\}.$$

The optimal actions for the discretized problem at each node of the grid, are represented in Figure 5.7 for both players. The optimal mean payoff for the discretized problem is :

King policy iteration index	Number of horse policy iterations	Sup-norm of residual	CPU time (s)
1	4	$3.09e-07$	$6.36e+01$
2	3	$1.73e-17$	$1.11e+02$

Table 5.1: Numerical results for the horse and the king example, performed on a 257×257 grid . Actions for the king : $\{(a_1, a_2) \,|\, a_i \in \{0, 1, -1\}, \quad i = 1, 2\}$, actions for the horse : $\{(0, 0), (\pm 1, \pm 2), (\pm 2, \pm 1)\}$.

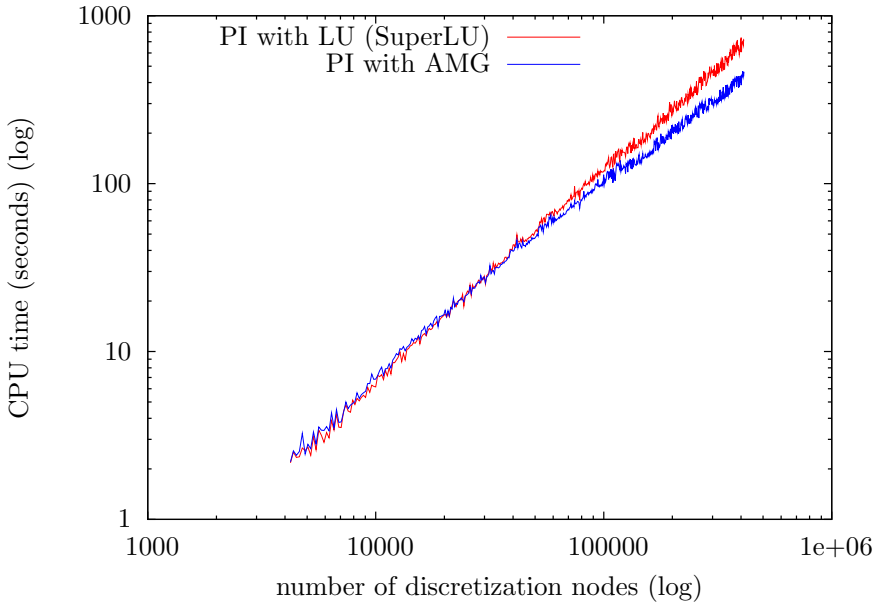

Figure 5.4: Numerical results for the king and horse example. Comparison between policy iteration algorithm with the AMGsingular2 algorithm versus a LU solver (SuperLU) when increasing the size of the problem.

$\eta(x) = 0.194 \; \forall \, x \in \mathcal{X}$ and the relative value v is represented in Figure 5.5 on the right. In Table 5.2, we present numerical results when the Equation (5.25) is discretized on grids with discretization step $h = \frac{1}{2^6}$, $h = \frac{1}{2^7}$, $h = \frac{1}{2^8}$ and $h = \frac{1}{2^9}$ respectively. From the fourth column, we can see that fewer $V(1, 2)$-cycles are needed to solve the linear systems as the algorithm progresses in the exploration of strategies of the second player. Moreover, these numbers of $V(1, 2)$-cycles are essentially independent of the size of the grid, what is expected for a multigrid scheme.

In Figure 5.4, we compare the policy iteration algorithm with a direct solver LU (SuperLU) that use the method of [PSS96, Ste97] (Section 2.4.1) to find the stationary probabilities, versus the AMGsingular2 algorithm, when increasing by one the number of discretization points in each direction from $m = 65$ to $m = 642$. The stopping criterion for the policy iterations used is 10^{-10}. Note that for each test, the number of total policy iterations (that is the number of linear systems to solve) is the same for both methods and is around ten iterations. In Figure 5.4, if we consider the 298 problems of finest discretiza-

Size of the grid	King policy iteration index	Number of horse policy iterations	Number of AMG iterations	Sup-norm of residual	CPU time (s)
65×65	1	5	$9, 10, 9, 7, 1$	$5.82e-06$	$1.25e+00$
	2	3	$11, 8, 5$	$6.07e-07$	$2.04e+00$
	3	2	$8, 1$	$3.17e-17$	$2.52e+00$
129×129	1	5	$9, 10, 8, 6, 1$	$1.45e-06$	$6.73e+00$
	2	3	$11, 9, 5$	$1.76e-07$	$1.09e+01$
	3	2	$8, 4$	$1.42e-10$	$1.37e+01$
	4	1	1	$6.54e-13$	$1.49e+01$
257×257	1	5	$9, 13, 9, 6, 1$	$3.64e-07$	$3.22e+01$
	2	3	$12, 9, 4$	$4.70e-08$	$5.20e+01$
	3	2	$9, 4$	$3.97e-10$	$6.47e+01$
	4	1	4	$2.08e-16$	$7.08e+01$
513×513	1	5	$9, 12, 9, 6, 1$	$9.09e-08$	$1.39e+02$
	2	4	$10, 7, 4, 1$	$1.31e-08$	$2.47e+02$
	3	2	$7, 1$	$1.10e-10$	$2.98e+02$
	4	1	1	$8.14e-13$	$3.22e+02$

Table 5.2: Numerical results for the king and horse example. Actions for the king : $\{(a_1, a_2) \,|\, a_i \in \{0, 1, -1\}, \quad i = 1, 2\}$, actions for the horse : $\{(0, 0), (1, 2), (2, 1)\}$. Policy iterations with AMGsingular2 algorithm (V$(1, 2)$-cycle and precision 10^{-15}).

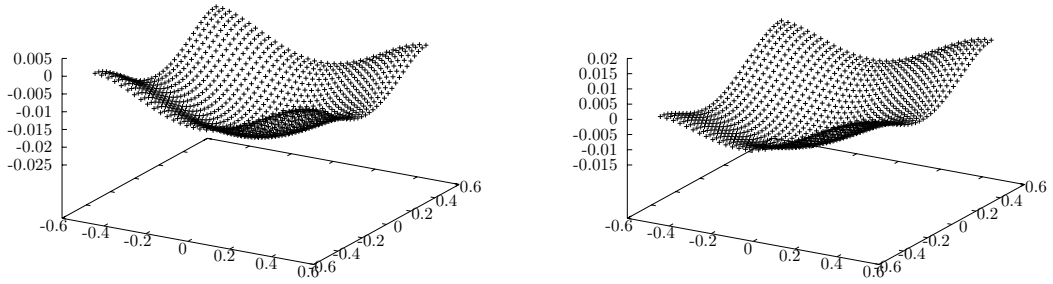

Figure 5.5: Relative solution for the horse and king example. Actions for the king : $\{(a_1, a_2) \,|\, a_i = \pm 1 \text{ or } 0\}$. Actions for the horse : $\{(0,0), (\pm 1, \pm 2), (\pm 2, \pm 1)\}$ on the left, $\{(0,0), (1,2), (2,1)\}$ on the right. Neumann conditions and $\epsilon = 0.5$ in (5.25).

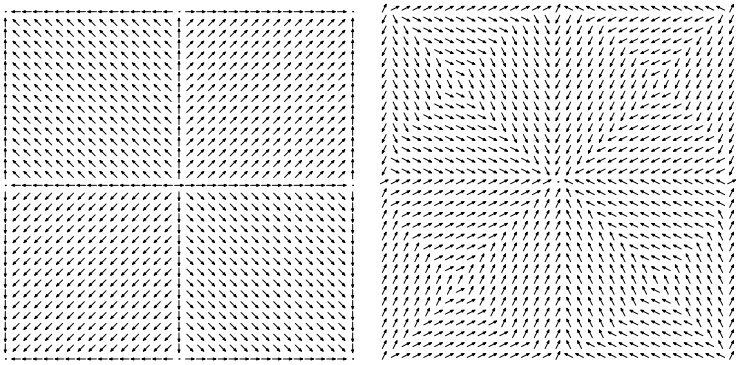

Figure 5.6: Optimal actions for the king on the left and for the horse on the right.

tion (the half of the tests), the relationship between the logarithm of the computation time and the logarithm of the size of the problem is almost linear with slope about 1.06 when using the AMGsingular2 algorithm, however the slope is about 1.28 for the policy iteration algorithm with a LU solver. Hence, the computation time while using the policy iterations with a AMGsingular2 algorithm seems to grow linearly with respect to the size of the problem.

5.4 Conclusion

In this chapter, we presented some new algebraic multigrid algorithms to solve particular singular linear systems that arise for instance in the policy iteration algorithm when solving zero-sum two player stochastic games with mean payoff. In particular, we presented our algorithm AMGsingular2.

Furthermore, we introduced a new method to find the stationary probability of an irreducible Markov chain using a stochastic control approach and presented our algorithm MGPIMG which combines the policy iterations of [How60, DF68] and AMGsingular2.

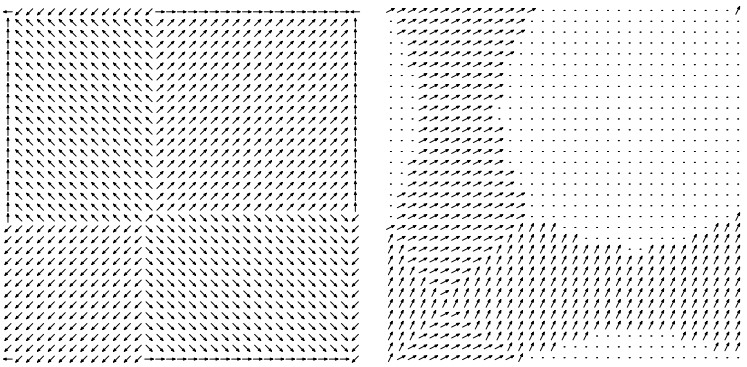

Figure 5.7: Optimal actions for the king on the left and for the horse on the right.

In the numerical tests, we compared our algorithm MCPIMG with the algorithm MAA of [DSMM$^+$08] and showed that they share both the properties expected for multigrid algorithms. We observed that MCPIMG is faster than MAA, however there exist acceleration algorithms for the MAA method in [DSMM$^+$10a, DSMM$^+$10b, DSMMS11, DSMSW10]. We may also expect to improvement the MCPIMG algorithm for instance by increasing the number of multigrid iterations exponentially at each policy iteration (as for instance in [Aki90a]) or by improving the AMGsingular2 algorithm.

In the last section, we applied the policy iterations combined with AMGsingular2 to solve some ergodic differential games with mean payoff. We compared the policy iteration algorithm with a direct solver LU versus the policy iteration algorithm with AMGsingular2 on pursuit evasion games. We observed that the policy iterations with AMGsingular2 grow linearly with respect to the size of the problem which is expected for a multigrid scheme.

We also mentioned that the multigrid algorithm AMGsingular2 sometimes diverges for random problems. However, we observed that for regular problems such as for pursuit evasion games, the AMGsingular2 converges in practice for all our tests. Hence, our AMGsingular2 algorithm need some improvements.

Chapter 6

Modeling and implementation

In this chapter, we describe our modeling of a zero-sum two player stochastic game of perfect information with discounted or mean payoff. We also present the main ideas behind the implementation of our program *PIGAMES* .

For stochastic game with discounted payoff, we shall use the same notations as in Chapter 1. Whereas for stochastic game with mean payoff, we shall use those of Chapter 4.

6.1 Modeling of a zero-sum two player stochastic game with perfect information

6.1.1 Discounted Payoff

Here, we consider a zero-sum two player stochastic game with perfect information, discounted payoff and discrete state space $\mathcal{X} = \{1, \ldots, n\}$ as defined in Chapter 1. Recall that the value function v of such game Γ_μ, defined by (1.6), is the unique solution $v : \mathcal{X} \to \mathbb{R}$ of the following dynamic programming equation:

$$v(x) = \underbrace{\max_{a \in \mathcal{A}(x)} \left(\min_{b \in \mathcal{B}(x,a)} \left(\sum_{y \in \mathcal{X}} \mu\, p(y \,|\, x, a, b)\, v(y) \;+\; r(x, a, b) \right) \right)}_{= F(\mu v; x)} \qquad \forall x \in \mathcal{X}. \tag{6.1}$$

For implementation purpose, when the actions sets $\mathcal{A}(x)$ and $\mathcal{B}(x, a)$ are finite for all $x \in \mathcal{X}$ and $a \in \mathcal{A}(x)$, it is convenient to decompose Equation (6.1) by splitting the non-linear and linear terms. Then, Equation (6.1) rewrites for each $x \in \mathcal{X}$:

$$\begin{cases} v(x) & = & \max\limits_{a \in \mathcal{A}(x)} v(x, a) & \\ v(x, a) & = & \min\limits_{b \in \mathcal{B}(x,a)} \{v(x, a, b) \;+\; r(x, a, b)\} & a \in \mathcal{A}(x) \\ v(x, a, b) & = & \sum\limits_{y \in \mathcal{X}} \mu\, p(y|x, a, b)\, v(y) & a \in \mathcal{A}(x),\; b \in \mathcal{B}(x, a)\;. \end{cases} \tag{6.2}$$

We interpret this decomposition in the following way. All nodes x from the state space $\mathcal{X}$ are the nodes where MAX plays (that is where MAX chooses an action a in $\mathcal{A}(x)$) and

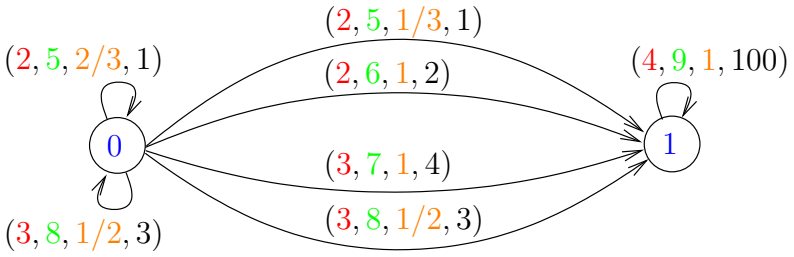

Figure 6.1: Example of a stochastic game.

are called MAX nodes. To a MAX node x and for each action $a \in \mathcal{A}(x)$, we associate an artificial node, denoted by (x, a) :

$$x \to (x, a) \quad a \in \mathcal{A}(x) \ .$$

This node represents the node where MIN plays (that is where MIN chooses an action b in $\mathcal{B}(x, a)$) and is called MIN node. To a MIN node (x, a) and for each action b in $\mathcal{B}(a, x)$, we associate an artificial node (x, a, b) :

$$(x, a) \to (x, a, b) \quad b \in \mathcal{B}(x, a) \ .$$

This node represents a distribution of probability on $\Delta(\mathcal{X})$ when in state x, player MAX chooses action a and player MIN chooses action b and is called PROBA node :

$$(x, a, b) \to \Delta(\mathcal{X}) \ .$$

We also denote by $\mathcal{X}_{player}$ the set of all nodes associated to *player* where *player* is one of MAX, MIN or PROBA. For instance, for the decomposition (6.2), the sets $\mathcal{X}_{\text{MAX}} = \mathcal{X}$, $\mathcal{X}_{\text{MIN}} = \{(x, a) \mid a \in \mathcal{A}(x), x \in \mathcal{X}\}$ and $\mathcal{X}_{\text{PROBA}} = \{(x, a, b) \mid b \in \mathcal{B}(x, a), a \in \mathcal{A}(x), x \in \mathcal{X}\}$. Then, we denote by $\bar{\mathcal{X}}$ the extended state space defined by $\bar{\mathcal{X}} = \mathcal{X}_{\text{MAX}} \dot{\cup} \mathcal{X}_{\text{MIN}} \dot{\cup} \mathcal{X}_{\text{PROBA}}$ and we denote by $\bar{n}$ the cardinality of $\bar{\mathcal{X}}$.

Example 6.1. *Let* $n = 2$, $\mathcal{X} = \{0, 1\}$, $\mathcal{A}(0) = \{2, 3\}$, $\mathcal{A}(1) = \{4\}$, $\mathcal{B}(0, 2) = \{5, 6\}$, $\mathcal{B}(0, 3) = \{7, 8\}$, $\mathcal{B}(1, 4) = \{9\}$, $p(0 \,|\, 0, 2, 5) = 2/3$, $p(0 \,|\, 0, 3, 8) = 1/2$, $p(0 \,|\, 0, 2, 5) = 2/3$, $p(1 \,|\, 0, 2, 5) = 1/3$, $p(1 \,|\, 0, 3, 8) = 1/2$, $p(1 \,|\, 1, 4, 9) = 1$, $r(0, 2, 5) = 1$, $r(0, 2, 6) = 2$, $r(0, 3, 7) = 4$, $r(0, 3, 8) = 3$ *and* $r(1, 4, 9) = 100$. *This example is represented by a graph in Picture 6.1 where the labels above the edges going from nodes* x *to* y *are* $(a, b, p(y \,|\, x, a, b), r(x, a, b))$, *for all* $a \in \mathcal{A}(x)$, $b \in \mathcal{B}(x, a)$ *and* $x, y \in \mathcal{X}$. *The decomposition in* MAX-MIN-PROBA *nodes is represented in Picture 6.2 where the* MAX *nodes* , $x \in \mathcal{X}_{\text{MAX}} = \{0, 1\}$, *are represented in blue circles, the* MIN *nodes,* $a \in \mathcal{X}_{\text{MIN}} = \{2, 3, 4\}$, *are in red circles and the* PROBA *nodes,* $b \in \mathcal{X}_{\text{PROBA}} = \{5, 6, 7, 8, 9\}$, *are in green circles. The label on the edge from a* MAX *node* x *to a* MIN *node* a *is the cost* $r(x, a)$ *and from a* MIN *node* a *to a* PROBA *node* b *is the cost* $r(x, a, b)$. *The labels on the edges going from a node* $b \in \mathcal{X}_{\text{PROBA}}$ *to a node* $y \in \mathcal{X}_{\text{MAX}}$ *are the transition probabilities* $p(y \,|\, x, a, b)$.

When some action spaces are not finite, one cannot use the decomposition (6.2) anymore. Hence, we introduce a new type of artificial node that we call ORACLE node. If

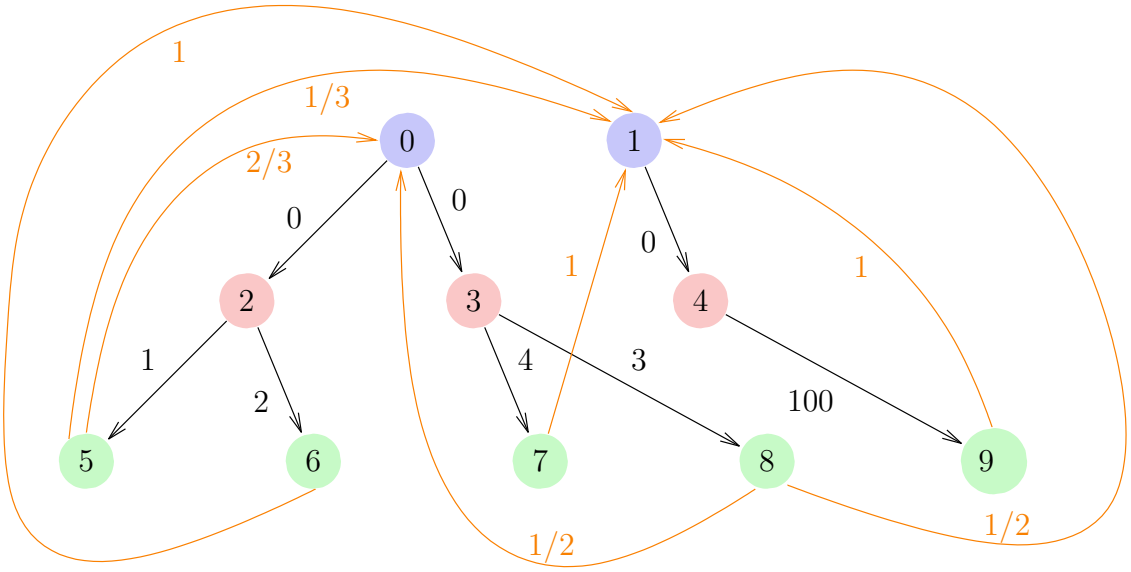

Figure 6.2: Example of a decomposition of the stochastic game represented in Figure 6.1 in MAX-MIN-PROBA nodes.

all the action spaces are not finite, we keep the dynamic programming equation in its original form Equation (6.1) and $\bar{\mathcal{X}} = \mathcal{X}$. If some of the action spaces are finite, we can separate the nodes with non-linear parts from the nodes with linear parts. For instance, if we consider Equation (6.1) with finite sets $\mathcal{A}(x)$ for $x \in \mathcal{X}$ and where the sets $\mathcal{B}(x,a)$ are not finite for $x \in \mathcal{X}$ and $a \in \mathcal{A}(x)$, we obtain the following decomposition :

$$\left\{ \begin{array}{lcll} v(x) & = & \max\limits_{a \in \mathcal{A}(x)} v(x,a) & \\ v(x,a) & = & \min\limits_{b \in \mathcal{B}(x,a)} \left\{ \sum\limits_{y \in \mathcal{X}} \mu\, p(y|x,a,b)\, v(y) \,+\, r(x,a,b) \right\} & a \in \mathcal{A}(x) \end{array} \right. \tag{6.3}$$

where the nodes $x \in \mathcal{X}$ are MAX nodes, the nodes (x,a) with $x \in \mathcal{X}, a \in \mathcal{A}(x)$ are MIN ORACLE nodes and $\bar{\mathcal{X}} = \mathcal{X} \dot{\cup} \{(x,a) \mid x \in \mathcal{X},\, a \in \mathcal{A}(x)\}$. Similarly, we can define MAX ORACLE nodes and ORACLE SELLE nodes.

Note that the dynamic programming Equation (6.1) may be given in another form then MAX-MIN-PROBA, for instance we can have the forms MIN-MAX-PROBA, PROBA-MAX-MIN, etc, then we decompose the dynamic programming equation according these forms.

6.1.2 Mean Payoff

Now, we consider a zero-sum two player stochastic game with perfect information, mean payoff and finite state space $[n] = \{1, \cdots, n\}$, as defined in Chapter 4. Recall that the mean value $\eta : [n] \to \mathbb{R}$ and relative value $v : [n] \to \mathbb{R}$ of the game are solution of the

following system, for $i \in [n]$:

$$\begin{cases} \eta_i = \min\limits_{a\in\mathcal{A}_i} \max\limits_{b\in\mathcal{B}_{i,a}} \sum\limits_{j\in[n]} P_{ij}^{ab}\, \eta_j \\ \eta_i + v_i = \min\limits_{a\in\acute{\mathcal{A}}_{i,\eta}} \max\limits_{b\in\acute{\mathcal{B}}_{i,a,\eta}} \left\{ \sum\limits_{j\in[n]} P_{ij}^{ab}\, v_j + r_{i,a,b} \right\} \end{cases} \tag{6.4}$$

As for discounted payoff case, when the actions sets $\mathcal{A}_i$ and $\mathcal{B}_{i,a}$ are finite for all $i \in [n]$, $a \in \mathcal{A}_i$, one can decompose the dynamic programming equations (6.4) by splitting the non-linear and linear terms. As in Section 6.1, we denote $[n]_{player}$ the set of all nodes associate to *player* where *player* is one of MAX, MIN or PROBA. For instance, for the decomposition (6.2), the sets $[n]_{\text{MAX}} = [n]$, $[n]_{\text{MIN}} = \{(i,a) \mid a \in \mathcal{A}_i,\, i \in [n]\}$ and $[n]_{\text{PROBA}} = \{(i,a,b) \mid b \in \mathcal{B}_{i,a},\, a \in \mathcal{A}_i,\, i \in [n]\}$. Then, we denote by $[\bar{n}]$ the extended state space defined by $[\bar{n}] = [n]_{\text{MAX}} \cup [n]_{\text{MIN}} \cup [n]_{\text{PROBA}}$ and we denote by $\bar{n}$ the cardinality of $[\bar{n}]$.

Then, we obtain the following decomposition of Equation (6.4) in MAX-MIN-PROBA, for each $i \in [n]$:

$$\begin{cases} \bar{\eta}_i = \min\limits_{a\in\mathcal{A}_i} \bar{\eta}_{i,a} & \\ \bar{\eta}_{i,a} = \max\limits_{b\in\mathcal{B}_{i,a}} \bar{\eta}_{i,a,b} & a \in \mathcal{A}_i \\ \bar{\eta}_{i,a,b} = \sum\limits_{j\in[n]} P_{ij}^{ab}\, \bar{\eta}_j & a \in \mathcal{A}_i,\ b \in \mathcal{B}_{i,a} \end{cases} \tag{6.5}$$

and

$$\begin{cases} \bar{\eta}_i + v_i = \min\limits_{a\in\acute{\mathcal{A}}_{i,\eta}} v_{i,a} & \\ \bar{\eta}_{i,a} + v_{i,a} = \max\limits_{b\in\acute{\mathcal{B}}_{i,a,\eta}} \{v_{i,a,b} + r_{i,a,b}\} & a \in \acute{\mathcal{A}}_{i,\eta} \\ \bar{\eta}_{i,a,b} + v_{i,a,b} = \sum\limits_{j\in[n]} P_{ij}^{ab}\, v_j & a \in \acute{\mathcal{A}}_{i,\eta},\ b \in \acute{\mathcal{B}}_{i,a,\eta} \end{cases} \tag{6.6}$$

where $\bar{\eta}_i = \frac{1}{3}\eta_i$.

When the mean payoff of the game is independent of the initial state, that is $\eta_i = \eta$ for $i \in [n]$, we can see that Equation (6.4) is equivalent to Equation (6.5)-(6.6) with $\bar{\eta}_i = \frac{1}{3}\eta$ for i in $[n]$. Indeed, replacing $\bar{\eta}_i$ by $\frac{1}{3}\eta$, Equation (6.5)-(6.6) rewrites, for each $i \in [n]$:

$$\begin{cases} \frac{1}{3}\eta + v_i = \min\limits_{a\in\mathcal{A}_i} v_{i,a} & \\ \frac{1}{3}\eta + v_{i,a} = \max\limits_{b\in\mathcal{B}_{i,a}} \{v_{i,a,b} + r_{i,a,b}\} & a \in \mathcal{A}_i \\ \frac{1}{3}\eta + v_{i,a,b} = \sum\limits_{j\in[n]} P_{ij}^{ab}\, v_j & a \in \mathcal{A}_i,\ b \in \mathcal{B}_{i,a} \end{cases}$$

that is equivalent to :

$$\frac{1}{3}\eta + v_i = \min_{a\in\mathcal{A}_i}\left\{\max_{b\in\mathcal{B}_{i,a}}\left\{\sum_{j\in[n]} P_{ij}^{ab}\, v_j - \frac{1}{3}\eta + r_{i,a,b}\right\} - \frac{1}{3}\eta\right\},$$

$$\eta + v_i = \min_{a\in\mathcal{A}_i}\max_{b\in\mathcal{B}_{i,a}}\left\{\sum_{j\in[n]} P_{ij}^{ab}\, v_j + r_{i,a,b}\right\}.$$

When the mean payoff of the game depends on the initial state, the same equivalence holds with $\bar{\eta}_i = \frac{1}{3}\eta_i$ for $i \in [n]$. Indeed, replacing $\bar{\eta}_i$ by $\frac{1}{3}\eta_i$ for $i \in [n]$, in Equation (6.5), gives :

$$\frac{1}{3}\eta_i = \min_{a\in\mathcal{A}_i}\max_{b\in\mathcal{B}_{i,a}}\sum_{j\in[n]} P_{ij}^{ab}\,\frac{1}{3}\eta_j, \tag{6.7}$$

and in Equation (6.6), gives :

$$\frac{1}{3}\eta_i + v_i = \min_{a\in\acute{\mathcal{A}}_{i,\eta}}\left\{\max_{b\in\acute{\mathcal{B}}_{i,a,\eta}}\left\{\sum_{j\in[n]} P_{ij}^{ab}\, v_j - \bar{\eta}_{i,a,b} + r_{i,a,b}\right\} - \bar{\eta}_{i,a}\right\},$$

since a, b are chosen in $\acute{\mathcal{A}}_{i,\eta}$, $\acute{\mathcal{B}}_{i,a,\eta}$ respectively, we have $\bar{\eta}_{i,a} = \bar{\eta}_{i,a,b} = \frac{1}{3}\eta_i$ and :

$$\eta_i + v_i = \min_{a\in\acute{\mathcal{A}}_{i,\eta}}\max_{b\in\acute{\mathcal{B}}_{i,a,\eta}}\left\{\sum_{j\in[n]} P_{ij}^{ab}\, v_j + r_{i,a,b}\right\}.$$

Note that similar decompositions exist when the dynamic programing equations are given in other forms than MAX-MIN-PROBA, then the "coefficient $\frac{1}{3}$" must be adapted to the decomposition.

In the next section, we describe how this modeling is used in the implementation of our program.

6.2 Technical details of implementation of AMGπ algorithm

In this section, we give the main ideas of how the program *PIGAMES* is implemented in term of structures (objects) and algorithms used. We give the definitions of the structures for the game given in the modeling of the previous section, then we explain how we adapt the policy iteration algorithm these structures. We first consider stochastic games with discounted payoff and then with mean payoff.

6.2.1 Discounted Payoff

Structure

The data of the game and the options for the algorithms are gathered in a structure named **Game problem** which is represented in Structure 1. The nodes of the extended state space $\bar{\mathcal{X}}$ are stored in some linked lists of **Node** Structures 2 (explained below). In

the **Game problem** structure, *lists* is an array of pointers to linked **Node** lists, the size of this array is nl, to each pointer $lists[i]$ $(1 \leq i \leq nl)$ correspond a type $type[i]$ which can have the following value : MAX, MIN, PROBA, MAX ORACLE or MIN ORACLE, all the nodes of the same $lists[i]$ have the same type(s). When we have a list of ORACLE SELLE nodes then it is linked twice in the array *lists*, once with type MAX ORACLE and once with type MIN ORACLE that is for instance $lists[1] = lists[2]$ with $type[1]$ = MAX ORACLE and $type[2]$ = MIN ORACLE. The order of the lists in the array *lists* has an importance, it depends of the form of the dynamic programming equation and $lists[1]$ always points to the list of nodes that represents the original state space $\mathcal{X}$. The arrays *maxit* and ϵ of size nl are the maximum numbers of iterations and the stopping criterion for the policy iteration algorithm. The variable *linear system* is the reference of the linear system solver chosen to solve the linear system in Step 1 of the policy iteration Algorithm 3.1. The matrix $\bar{P} : \mathbb{R}^{\bar{n} \times \bar{n}}$ is the probability matrix corresponding to the selected strategies on the extended state space $\bar{\mathcal{X}}$. The vector $r : \bar{\mathcal{X}} \to \mathbb{R}$ is the corresponding cost vector. The vector $v : \bar{\mathcal{X}} \to \mathbb{R}$ of the **Game problem** structure is the current approximation of the solution of the game defined for the extended state space $\bar{\mathcal{X}}$ and the vector $fv : \bar{\mathcal{X}} \to \mathbb{R}$ contains the application of dynamic programming operator on that approximation, that is $F(v)$.

Game Problem	
int	**nl**
	number of lists
Node	**(*lists)[nl]**
	lists of nodes
int	**type[nl]**
	*type[i] is the type of the above lists (*MAX*,* MIN*,* PROBA*,* MAX ORACLE *or* MIN ORACLE*)*
int	**maxit[nl]**
	maxit[i] is the maximum policy iterations for list i
double	**ϵ[nl]**
	ϵ[i] is the precision for policy iterations for list i
int	**linear solver**
	the reference of the linear solver chosen
matrix	$\bar{\mathbf{P}}$
	probability matrix
vector	$\bar{\mathbf{r}}$
	cost vector
vector	**v**
	value of the game
vector	**fv**

$F(v)$

Structure 1.

For instance, for a game for which the dynamic programming Equation (6.1) is decomposed in Equation (6.2), we construct three lists ($nl = 3$), the first one $lists[1]$ is of type $type[1] =$MAX and represents the nodes $x \in \mathcal{X}_{\text{MAX}}$ of the original state space $\mathcal{X}$, the second list $lists[2]$ is of type $type[2] =$MIN and contains all the MIN nodes $a \in \mathcal{X}_{\text{MIN}}$, the last list $lists[3]$ is of type $type[3] =$PROBA and contains all the PROBA nodes $b \in \mathcal{X}_{\text{PROBA}}$.

The Structure of a **Node** is represented in Structure 2. All the nodes are numbered from one to $\bar{n}$, this number is called the *label* of the node and it also represents its coordinate in the vector $\bar{v}$, the solution of the game extended to $\bar{\mathcal{X}}$. The interpretation of the other elements of the structure **Node** depends on the type of the node and are described below.

Node	
int	**label**
	label of the node
int	**np**
	number of nodes to which it points
int	**pointers[np]**
	nodes to which it points (correspond to the actions)
double	**probabilities[np]**
	probabilities[i] is the probability of pointers[i] (if any)
double	**costs[np]**
	costs[i] is the cost of pointers[i] (if any)
int	**strategy**
	chosen strategy in pointers
oracle	**oracle**
	special structure with information for oracle nodes
Node	***next**
	pointer to the next node on the list

Structure 2.

For a MAX node x, the element np denotes the number of nodes to which it points, that is the the cardinality of the action space $\mathcal{A}(x)$, the array *pointers* and *costs* of size np contains respectively the labels of the nodes corresponding to the actions in $\mathcal{A}(x)$ and the costs related to those actions. The strategy is the chosen action $\alpha(x)$ of the array *pointers*. The same holds for a MIN node. For instance, for a MAX node x such as defined in Equation (6.2), np equals the cardinally of $\mathcal{A}(x)$, if the actions a in $\mathcal{A}(x)$ are numbered from one to np, then $pointers[i]$ is the label of the ith-node $a \in \mathcal{X}_{\text{MIN}}$ and $costs[i] = r(x, a)$ for $i \in \{1, \cdots, np\}$. The element *strategy* is a chosen action a in $\mathcal{A}(x)$.

For a PROBA node x, the element np is the number of nodes y in $\bar{\mathcal{X}}$ for which the probability going from node x to this node y is nonzero, the arrays *pointers* and *probabilities* of size np contain respectively the labels of those nodes y and the probabilities of going from x to node y. For instance, for a PROBA node $(x, a, b) \in \mathcal{X}_{\text{PROBA}}$ such as defined in Equation (6.2), np equals the number of $y \in \mathcal{X}$ such that $p(y|x, a, b) \neq 0$, if those nodes y are numbered from one to np, $pointers[i]$ is the label of one of those y and $probabilities[i] = p(y|x, a, b)$ for $i \in \{1, \cdots, np\}$.

For a MAX ORACLE node x, the special element *oracle* contains some information about the node that may be used in a special oracle function when improving the strategy in Step 3 of Algorithm 3.1 or Algorithm 3.2. In this case, np is the size of the arrays *pointers* and *probabilities*, they contain respectively the labels and the corresponding non null probabilities. The array *costs* is of size one and contains the reward at node x for the chosen action. Those values are computed and given at each iteration of the Algorithm 3.1 or Algorithm 3.2 by a special oracle function provided by the user. The same hold for a MIN ORACLE node. For instance, for a MIN ORACLE node (x, a) as defined in Equation (6.3), a special oracle function returns, in the improvement Step 3 of Algorithm 3.1, the cost $costs = r(x, a, b)$ and the array *probabilities* of non null probabilities $p(y|x, a, b)$ with the corresponding array *pointers* of labels.

Policy iteration algorithm

An adaptation of the Algorithm 3.2 for the solution of the dynamic programming Equation (6.1) of a zero-sum two player stochastic game to the above decomposition and structure, is written in pseudo code in Algorithm 6.1, it starts with parameters: $player = 1$ and the **Game Problem** $GProblem$ that contains the game data. The algorithm is constructed in a recursive form based on the lists of nodes of the **Game Problem** structure to allow working with different decomposition models for different forms of Equations (6.1). Starting from the first list, it recursively applies the policy iteration algorithm (that is successively applies a policy evaluation step followed by a policy improvement step until convergence) on each list and hence produce nested iterations (see Section 3.1 and Figure 3.1). When the algorithm reaches the last list, that is the most internal loop of the nested policy iterations, the policy evaluation step amounts to solve a linear system. Note that in the case of a loop on a list of type PROBA, there is no optimization needed in the improvement step and the policy evaluation step consists in applying a chosen linear solver $linearSolver()$ to find an approximation of the solution of the system $v = \bar{P}v + \bar{r}$. For other lists, the policy improvement step consists in finding the optimal strategy for the current value of the game and player. It also sets the corresponding lines in the matrix $\bar{P}$. This is done in Algorithm 6.1 for each node of the current list, depending on the type of the player. For a MAX or MIN node, the function $argminmax()$ returns the optimal action for the fixed strategy and the corresponding new row for the matrix $\bar{P}$. For an ORACLE node, a function $specialOracleFunction()$ provided by the user also return the optimal action for the node and the corresponding probability row.

Algorithm 6.1a $(v) \leftarrow GPI(player, GProblem)$

$r_v = 1$;
while $(|r_v| > GProblem.\epsilon[player]$ **or** $k < GProblem.maxit[player])$ **do**
  if $player < (GProblem.nl - 1)$ **then**
    $(v) \leftarrow GPI(player + 1, GProblem)$;
  else
    $(v) \leftarrow linearSolver(v, \bar{P}, \bar{r})$;
  end if
  $PImprovement(player, GProblem)$;
  $r_v = F_{player}(v) - v$;
end while
return v;

Algorithm 6.1b $PImprovement(player, GProblem)$

for each $node$ **in** $Gproblem.lists[player]$ **do**
  switch $(GProblem.type[player])$
    case MAX:
    case MIN:
      $(strategy, \bar{P}(label, :)) \leftarrow argminmax(node, GProblem.type[player])$
      break;
    case PROBA:
      $fvalProba(node, GProblem)$
      break;
    case MAX ORACLE:
    case MIN ORACLE:
      $(strategy, \bar{P}(label, :)) \leftarrow specialOracleFunction(node, GProblem.type[player])$
      break;
  end switch
end for

6.2.2 Mean payoff

Now we consider zero-sum two player games with mean payoff. We first define the structures of the game, then we explain how we adapt the policy iteration algorithm our modeling of the game.

Structures

In this case, we use the same structure **Node** (Structure 2) and the structure **Game problem** (Structure 1) to which we add some new elements : $\eta : [\bar{n}] \to \mathbb{R}$ the mean value of the game, $v : [\bar{n}] \to \mathbb{R}$ becomes the relative value of the game and two new arrays $optimalSet[]$ and $optimalSet_\eta[]$. In the optimization step of the policy iteration for stochastic games with mean payoff, for each node i, the array $pointers[]$ is sorted such that the first $optimalSet[i]$ elements contains the current optimal actions for that node, similarly for $optimalSet_\eta[i]$ which determines the optimal set of node i for the current value of η. The arrays $optimalSet[\cdot]$ and $optimalSet_\eta[\cdot]$ correspond to the sets $\mathcal{A}_{(\cdot)}, \mathcal{B}_{(\cdot)}$ and $\acute{\mathcal{A}}_{(\cdot),\eta}, \acute{\mathcal{B}}_{(\cdot),\eta}$ respectively. The entire structure **Game problem** is given below.

Game Problem	
int	**nl**
	number of lists
Node	**(*lists)[nl]**
	lists of nodes
int	**type[nl]**
	*type[i] is the type of the above lists (*MAX*,* MIN*,* PROBA*,* MAX ORACLE *or* MIN ORACLE*)*
int	**maxit[nl]**
	maxit[i] is the maximum policy iterations for list i
double	**ϵ[nl]**
	ϵ[i] is the precision for policy iterations for list i
int	**linear solver**
	the reference of the linear solver chosen
matrix	$\bar{\mathbf{P}}$
	probability matrix
vector	$\bar{\mathbf{r}}$
	cost vector
vector	$\bar{\mathbf{v}}$
	relative value of the game
vector	$\boldsymbol{\eta}$
	mean value of the game
vector	$\bar{\mathbf{f}}\mathbf{v}$
	$F(v)$

int	***optimalSet*[n]**
	optimalSet[i] is the optimal set for node i
int	***optimalSet*$_\eta$[n]**
	optimalSet$_\eta$[i] is the optimal set for node i for η

Structure 3.

Algorithms

In this section, we adapt the algorithms of Chapter 4 and 5, to give the idea of how it is implemented in the program *PIGAMES* .

Policy iteration algorithm

We present in Algorithm 6.2, a policy iteration algorithm for solving zero-sum two player stochastic games with mean payoff which is an adaptation of Algorithm 4.2. We also give in Algorithm 6.3 an adaptation of the policy iteration for MDP with mean payoff of Algorithm 4.5. Both algorithm Algorithm 6.2- 6.3 use floating points and may use iterative linear solvers. For this purpose, we introduce some precision parameters : $\epsilon_\eta, \epsilon_v$ and ϵ_{LS}. The parameter ϵ_{LS} is used as stopping criterion for the iterative linear solvers called in Algorithm 6.4. The parameters : $\epsilon_\eta, \epsilon_v$ are used for the comparison tests of the values $\eta^{(k)}$ and $v^{(k)}$ respectively in the stopping criterion in Step 2 of Algorithm 6.2 and Algorithm 6.3 and in the improvement of the strategies represented in Algorithm 6.5. We do not give here the details of the implementation of the linear systems, they use the algorithms described in Chapter 5.

The algorithm for the computation of the critical graph which is called in the degenerate iterations of Algorithm 6.2, is given in the next section.

Algorithm 6.2 $(\alpha, \eta, v) \leftarrow PImultichainTwo(\alpha^{(0)}, \beta^{(0)}, \eta^{(0)}, v^{(0)})$

1. Compute an approximation $(\eta^{(k+1)}, v^{(k+1)})$ of the solution (η, v) of the game for the fixed policy $\alpha^{(k)}$:

$$(\beta^{(k+1)}, \eta^{(k+1)}, v^{(k+1)}, status) \leftarrow PImultichainOne(\beta^{(k)}, \eta^{(k)}, v^{(k)}, player2)$$

2. 2.1 **if** $(\|\eta^{(k)} - \eta^{(k+1)}\| < \epsilon_\eta$ **and** $\|v^{(k)} - v^{(k+1)}\| < \epsilon_v)$ **then**
   STOP and return $(\alpha^{(k)}, \eta^{(k+1)}, v^{(k+1)})$

   2.2 **if** $\|\eta^{(k)} - \eta^{(k+1)}\| < \epsilon_\eta$ **and** $k > 0$ **then**

   i. Define $\bar{P}$ from Equation (6.10) with $\alpha^{(k)}$ and $\{\tilde{\mathcal{B}}_i\}_{i \in [n]_{pl2}}$

   ii. $C(g) \leftarrow CriticalGraph(\bar{P})$

   iii. Solve

$$\begin{cases} v_i^{(k+1)} &= \acute{F}_{\eta^{(k+1)}}(v^{(k+1)}; i, \alpha_{k+1}(i)) - \eta_i^{(k+1)} & i \in [n] \setminus C(g) \\ v_i^{(k+1)} &= v_i^{(k)} & i \in C(g) \ . \end{cases}$$

   with the policy iteration for MDP with discounted payoff.

3. Improve the policy $\alpha^{(k)}$ for $(\eta^{(k+1)}, v^{(k+1)})$ in a conservative way :

$$(\alpha^{(k+1)}) \leftarrow PIimprove(\eta^{(k+1)}, v^{(k+1)}, player1) \tag{6.8}$$

4. **if** $(\alpha^{(k)}(i) = \alpha^{(k+1)}(i)\ i \in [n])$ **then**
   return $(\alpha^{(k+1)}, \beta^{(k+1)}, \eta^{(k+1)}, v^{(k+1)})$

5. Increment k by one and go to Step 1.

Algorithm 6.3 $(\beta, \eta, v) \leftarrow PImultichainOne(\beta^{(0)}, \eta^{(0)}, v^{(0)}, player)$

1. Compute an approximation $(\eta^{(k+1)}, v^{(k+1)})$ of the solution (η, v) of the game for the fixed policy $\beta^{(k)}$:

$$(\eta^{(k+1)}, v^{(k+1)}) \leftarrow LSsolve(\eta^{(k)}, v^{(k)}, P^{(\beta^{(k)})}, r^{(\beta^{(k)})}, \epsilon_{LS})$$

2. 2.1 **if** $(\|\eta^{(k)} - \eta^{(k+1)}\| < \epsilon_\eta)$ **and** $(\|v^{(k)} - v^{(k+1)}\| < \epsilon_v)$ **then**
   STOP and return $(\beta^{(k)}, \eta^{(k+1)}, v^{(k+1)})$

3. Improve the policy $\beta^{(k)}$ for $(\eta^{(k+1)}, v^{(k+1)})$ in a conservative way :

$$(\beta^{(k+1)}) \leftarrow PIimprove(\eta^{(k+1)}, v^{(k+1)}, player) \tag{6.9}$$

4. **if** $(\beta^{(k+1)}(i) = \beta^{(k)}(i)\ i \in [n])$ **then** return $(\beta^{(k+1)}, \eta^{(k+1)}, v^{(k+1)})$

5. Increment k by one and go to Step 1.

Algorithm 6.4 $(\eta, v) \leftarrow LSsolve(\eta, v, \beta^{(k)}, P, r, \epsilon_{LS})$

$(P, type) \leftarrow tarjan(P)$
for $I \in \{1, \cdots, m\}$ **do**
  if $type(I) = FINAL$ **then**
    $(\eta_I, v_I) \leftarrow singularSolver(P_{II}, \eta_I, v_I, r_I, \epsilon_{LS})$
  else if $type(I) = TRANSIANT$ **then**
    $rhs = \sum_{J>I} P_{IJ}\eta_J$
    $\eta_I \leftarrow livearSolver(P_{II}, rhs, \eta_I, \epsilon_{LS})$
    $rhs = \sum_{J>I} P_{IJ}v_J + r_I - \eta_I$
    $v_I \leftarrow livearSolver(P_{II}, rhs, v_I, \epsilon_{LS})$
  end if
end for

The critical graph algorithm

Now, we give the adaptation of Algorithm 4.6 for the computation of the critical graph to our model. This algorithm depends on the structure of the model. We consider here the algorithm the MAX-MIN-PROBA decomposition, that is Equation (6.6)-(6.5). Suppose MAX is the first player and MIN the second player.

Recall that in Step 5 of Algorithm 4.2, one has to compute the critical nodes of the map $\bar{g}$ defined by :

$$\bar{g}_i(v) := \acute{F}_{\eta^{(k+1)}}(v; i, \alpha(i)) - \eta_i^{(k+1)} \quad \text{for all } i \in [n] \ ,$$

with $\alpha = \alpha^{k+1}$. Given an harmonic vector u, we denote the set

$$\widetilde{B}_i^a = \left\{ b \in \bar{\mathcal{B}}_{i,\eta^{(k+1)}} \mid -\eta_i^{(k+1)} + r_i^{ab} + \sum_{j\in[n]} P_{ij}^{ab}\, u_i = u_i \right\},$$

and $\mathcal{P}_i = \left\{ (P_{ij}^{\alpha(i)b})_{j\in[n]} \mid b \in \widetilde{B}_i^{\alpha(i)} \right\}$. The adaptation of the critical graph Algorithm 4.6 is given in Algorithm 6.6. First we define the matrix $\bar{P} \in \mathbb{R}^{\bar{n}\times\bar{n}}$ such that its non-null elements are given by, for all $i \in [n]$:

$$\bar{P}(i, (i, a)) = 1 \qquad \text{for } a = \alpha(i) \tag{6.10}$$

$$\bar{P}((i, a), (i, a, b)) = 1 \qquad \text{for } a = \alpha(i) \text{ and } b \in \widetilde{B}_i^a \tag{6.11}$$

$$\bar{P}((i, a, b), j) = P_{ij}^{ab} \qquad \text{for } a = \alpha(i),\ b \in \widetilde{B}_i^a \text{ and } j \in [n]. \tag{6.12}$$

This matrix represents the sets $\mathcal{P}_i$ for $i \in [n]$ that is, for a node $i \in [n]$, the element $\bar{P}((i, \alpha(i)), (i, \alpha(i), b))$ equals one if the row vector $(\bar{P}((i, \alpha(i), b)), j)_{j\in[n]}$ belongs to $\mathcal{P}_i$. Similarly, the matrix $\bar{P}^{(k)}$ represents the sets $\mathcal{Q}^{(k)}$ in Algorithm 4.6. Indeed, removing a row vector of $\mathcal{Q}_i^{(k)}$ which has a rowsum less than one in Algorithm 4.6 corresponds to set the corresponding value $\bar{P}^{(k)}((i, \alpha(i)), (i, \alpha(i), b))$ to zero in Algorithm 6.6. Note that in our case, we only need the critical nodes and not the complete critical graph.

Algorithm 6.5 $(\beta^{(k+1)}, \{\acute{\mathcal{B}}_{i,\eta}\}_{i\in[n]_{pl}}, \{\tilde{\mathcal{B}}_i\}_{i\in[n]_{pl}}) \leftarrow PIimprove(\eta, v, \beta^{(k)}, player)$

for $i \in [n]_{pl}$ **do**
  $\acute{\mathcal{B}}_{i,\eta} = \emptyset$; $\tilde{\mathcal{B}}_i = \emptyset$; $\beta^{(k+1)}(i) = \beta^{(k)}(i)$
  for $b \in \mathcal{B}_i$ **do**
    if $(P_{i\cdot}^{b}\eta < P_{i\cdot}^{\beta^{(k+1)}(i)}\eta$ **and** *player* is MIN**)** **or** $(P_{i\cdot}^{b}\eta > P_{i\cdot}^{\beta^{(k+1)}(i)}\eta$ **and** *player* is MAX**)**
    then
      $\beta^{(k+1)}(i) = b$
    end if
  end for
  for $b \in \mathcal{B}_i$ **do**
    if $\|P_{i\cdot}^{b}\eta - P_{i\cdot}^{\beta^{(k+1)}(i)}\eta\| < \epsilon_\eta$ **then**
      $\acute{\mathcal{B}}_{(i),\eta} \leftarrow b$
    end if
  end for
  for $b \in \acute{\mathcal{B}}_{(i),\eta}$ **do**
    if $((P_{i\cdot}^{b}v + r_i^b) < (P_{i\cdot}^{\beta^{(k+1)}(i)}v + r_i^{\beta^{(k+1)}(i)}))$ **and** *player* is MIN**)** **or** $((P_{i\cdot}^{b}v + r_i^b) > (P_{i\cdot}^{\beta^{(k+1)}(i)}v + r_i^{\beta^{(k+1)}(i)})$ **and** *player* is MAX**)** **then**
      $\beta^{(k+1)}(i) = b$
    end if
  end for
  for $b \in \acute{\mathcal{B}}_{i,\eta}$ **do**
    if $(\|(P_{i\cdot}^{b}v + r_i^b) - (P_{i\cdot}^{\beta^{(k+1)}(i)}v + r_i^{\beta^{(k+1)}(i)})\| < \epsilon_v)$ **then**
      $\tilde{\mathcal{B}}_i \leftarrow b$
    end if
  end for
  if $\beta^{(k)}(i) \in \tilde{\mathcal{B}}_i$ **then**
    $\beta^{(k+1)}(i) = \beta^{(k)}(i)$
  end if
end for

Algorithm 6.6a $F \leftarrow CriticalGraph(\bar{P})$

Set $N = \emptyset$, $F = \emptyset$, $\bar{P}^{(0)} = \bar{P}$ and $k = 0$
$Tarjan(\bar{P})$, put the final nodes in F and the transient nodes in N
$\mathcal{Q}' \leftarrow N$
while $\mathcal{Q}' \neq \emptyset$ **do**
  Compute the set $\mathcal{Q}'$ using Algorithm 6.6b and construct $\bar{P}^{(k+1)}$ by tacking from $\bar{P}^{(k)}$ the rows and columns corresponding to the nodes in $\mathcal{Q}'$. (That are the nodes $i \in [n]$ such that $(i,a) \in \mathcal{Q}'$, the nodes $(i,a) \in \mathcal{Q}'$ and the nodes (i,a,b) such that $\bar{P}^{(k)}((i,a),(i,a,b)) = 1$ for all $(i,a) \in \mathcal{Q}'$.)
  $Tarjan(\bar{P}^{(k+1)})$, put the final nodes in F' and set $F = F \cup F'$
  $k++$
end while
return F the set of critical nodes

Algorithm 6.6b Compute $\mathcal{Q}'$ set

for all node (i,a) in $\mathcal{Q}'$ (from player 2) **do**
  if (i,a) is ergodic **then**
    $\mathcal{Q}' = \mathcal{Q}' \setminus \{(i,a)\}$
    $N = N \setminus \{(i,a)\}$
  else
    for all node (i,a,b) such that $\bar{P}((i,a),(i,a,b)) \neq 0$ (to proba) **do**
      for all node y such that $\bar{P}((i,a,b),y) \neq 0$ (to player 1) **do**
        if j is an ergodic node **then**
          $\bar{P}((i,a),(i,a,b)) = 0$
          break;
        end if
      end for
    end for
    if there is no (i,a,b) such that $\bar{P}((i,a),(i,a,b)) \neq 0$ (a-th row is null) **then**
      $\mathcal{Q}' = \mathcal{Q}' \setminus \{(i,a)\}$
    end if
  end if
end for

6.3 The linear solver: problems and issues

6.3.1 Tarjan Algorithm

The decomposition of Equation (6.1) in Equation (6.2) has an impact on the structure of the matrix $\bar{P}$ of the policy iteration algorithm, that arises in the linear system of the most internal loop of the nested policy iterations. This matrix has dimension $\bar{n}$ and corresponds to the set of Equations (6.2). Indeed, when for each $x \in \mathcal{X}_{\text{MAX}}$ we fix an action $a \in \mathcal{A}(x)$ (that is the actions that determines the strategy for player MAX) and for each node $(x, a) \in \mathcal{X}_{\text{MIN}}$ we fix an action $b \in \mathcal{B}(x, a)$ (that determines the strategy for player MIN), the non-linear terms in Equation (6.2) vanish and form the linear system $v = \bar{P}v + \bar{r}$. The adjacency graph of the matrix $\bar{P}$ induced by this decomposition is highly reducible and it may compromise the efficiency of the chosen linear system solver. Therefore, we decompose the linear System (6.2) into irreducible classes using Tarjan algorithm [Tar72] (which is explained below). The system is then solved successively on each irreducible classes.

We recall here the Tarjan algorithm in Algorithm 6.7 to construct strongly connected components of a graph $G = (V, E)$, this algorithm [Tar72] requires $O(V + E)$ space and time where V is the number of vertices and E is the number of edges of the graph G. Note that for efficiently this algorithm is implemented in a equivalent non-recursive form in our program *PIGAMES* .

Algorithm 6.7a Tarjan [Tar72]

for $x \in G$ **do**
  $number[x] \leftarrow -1$;
  $lowlink[x] \leftarrow -1$;
end for
$i \leftarrow 0$;
for $x \in G$ **do**
  if $number[x] == -1$ **then**
    StrongConnect $(x, i, number[], lowlink[])$;
  end if
end for

Algorithm 6.7b StrongConnect $(x, i, number[], lowkink[])$

```
lowlink[x] ← i;
number[x] ← i;
i++;
put x on Stack;
for y in adjacency list of x do
    if number[y] == −1 then
        StrongConnect (y, i, number, lowlink);
        lowlink[y] ← min {lowlink[x], lowlink[y]};
    else if number[y] < number[x] then
        if y is on Stack then
            lowlink[x] ← min {lowlink[x], lowlink[y]};
        end if
    end if
end for
if lowlink[x] == number[x] then
    Start new strongly connect component SCC:
    while number[top on Stack] ≥ number[x] do
        y ← pop Stack;
        put y on SCC;
    end while
end if
```

6.4 The *PIGAMES* package

We present here the readme file of the *PIGAMES* package.

6.5 Contents of the package

The package *PIGAMES* contains the following subdirectories :

game/	main source code.
manual/	contains the manual files.
soft/	interface code for external packages.

The main directory *PIGAMES* also contains a `README` file with the instructions for the installation of the package, a `makefile` and a file `makefile.mk` which contains some specifications to be completed by the user. The program requires that the libraries `BLAS` and `LAPACK` (see below).

6.5.1 Installation

- Complete makefile.mk with your directories, libraries and flags.
- If some libraries are not installed on your system, modify solver.h, comment the libraries that you do not have.
- Run `make all` to build the package (see the makefile for other commands).

The command `make game_documentation` builds the documentation in HTML pages form that is available in `pigames/manual/html/index.html`. This pages can be accessed using a usual browser.

6.5.2 Run

The main command is `./pigames`, to get an overview of the possible commands, the user can `./pigames help` or simply `./pigames`. The general form of call of the program is

```
./pigames choice file_name(s)
```

where `choice` is an integer explained below, `file_name(s)` are input files and `[dessin]` is an option for plotting a graph (if available for the chosen application). The different options are the following:

- `./pigames 7 input_options input_data`
  Run the program to solve a stochastic game (discounted or mean payoff) whose data is given in the file `input_data` and the options for the solver in the file `input_options`.
- `./pigames 2 inputs/input_random_test.options`
  Run the program who generates a random two player zero-sum stochastic game with discounted payoff, the options are in the file `inputs/input_random_test.options`.

- `./pigames 3 inputs/input_richman_test.options`
  Run the program for a random Richman game with the options in the file `inputs/input_richman_test.options`
- `./pigames 4 input_file`
  Run the program AMGπ to solve a HJB-Bellman equation (one player), the options are in the input file, see for instance the file `inputs/input_edp1_test.options`.
- `./pigames 5 inputs_file`
  Same as choice 4 but instead the solver FAMGπ is used.
- `./pigames 21 input_file`
  Run the program AMGπ to solve an Isaacs equation, the options are in the input file, for instance the file `inputs/input_edpjeux1.options`.
- `./pigames 22 input_file`
  Same as choice 21 but instead the solver FAMGπ is used.
- `./pigames 111 name_problem input_file` Run the program to solve a stochastic game with mean payoff of type `name_problem`, the options are in the file `input_file`. For instance:
  - `./pigames 111 richmanCTG` example of the Richman game with mean payoff the paper of Cochet-Terrasson and Gaubert [CTG06], see also the file `inputs/input_ergodic_richmanCTG` for the options.
  - `./pigames 111 saad` example of zero-sum two player stochastic games where transitions matrices are build using a modified version of the random walk code of Saad Sparse-Kit package, see also the file `inputs/input_ergodic_saad` for options.
  - `./pigames 111 random` example of zero-sum two player stochastic games with random transitions matrices, see also the file `inputs/input_ergodic_random` for options.
  - `./pigames 111 pursuit inputfile` example of pursuit games of Section 5.3.2, see also the file `inputs/input_edp_pursuit` for options.
  - `./pigames 111 richman inputfile` example of a random mean payoff Richman game, see also the file `inputs/input_ergodic_richman` for options.
- `./pigames 222` (use for debug functions only)

The above input option files contain default value and can be edited.

6.5.3 Input data file for discrete stochastic games

We describe how to use the following command :

`./pigames 7 input_options input_data`

which allows to run the program for solving a stochastic game (discounted or mean payoff) with data given in the `input_data` and `input_options` files. In this case, the problem must be entered in a MAX-MIN-PROBAform or similar without oracle nodes (see Section 6.1)

and the `input_data` has the following structure :

- (number of node lists)
- (types of the lists)*
- <list of lists>:
  - <list of nodes>:
    - (label) (type of the list)
    - (number of arcs to which it points) (initial strategy)
    - <array of pointers>: (size) (number)*
    - <array of probabilities>: (size) (number)*
    - <array of costs>: (size) (number)*

where the first line of this file contains the number of lists, the second line contains the type of the lists in order (the first list is always the initial state space). The next lines contain the definition of all the nodes that can be given in any order. Each node is defined by a block of five lines as followed. The first line contains the label of the node followed by the type of the list to which it belongs. The second line contains the number nodes to which it points, that we shall call np, followed by the position of the strategy, that we shall call st, in the array of pointers defined in the third line, $0 \leq st < np$. The third, fourth and fifth lines contain respectively, the arrays of pointers, corresponding probabilities and costs. Each of these arrays are defined by the size of the array followed by its elements. When the array is empty it is defined by size 0. For instance, if we consider Example 6.1, illustrated in Figure 6.1 and in MAX-MIN-PROBA form in Figure 6.2, we have the following input file:

```
3               # number of list (nodes and arcs)
-12 -11 -13     # === type of the lists player1, player2, proba (-11 MIN, -12 MAX, -13 PROBA)
0  -12          # === new node  MAX NODE ===
2 0
2 2 3
0
2 0 0
1  -12          # === new node  MAX NODE ===
1 0
1 4
0
1 0
2  -11          # === new node  MIN NODE ===
2 0
2 5 6
0
2 1 2
3  -11          # === new node  MIN NODE ===
2 0
2 7 8
0
2 4 3
4  -11          # === new node  MIN NODE ===
```

```
1 0
1 9
0
1 100
5  -13           # === new node  PROBA NODE ===
2 0
2 0 1
2 0.666666 0.333333
0
6  -13           # === new node  PROBA NODE ===
1 0
1 1
1 1
0
7  -13           # === new node  PROBA NODE ===
1 0
1 1
1 1
0
8  -13           # === new node  PROBA NODE ===
2 0
2 0 1
2 0.5 0.5
0
9  -13           # === new node  PROBA NODE ===
1 0
1 1
1 1
0
```

The corresponding options input file is given, for instance, by :

```
1         # linear system solver
1e-16     # precision for the linear solver if iterative
50        # max iterations for the linear solver solver if iterative
1         # game solver (1 Policy iteration)
100 100 100            # maximum policy iterations per list
1e-12 1e-12 1e-12      # precision tolerance for policy iterations per list

0 0.001   # 1 if mean payoff game, 0 if discounted payoff and lambda
13        # ergodic solver choice
0 1 2     # (first player, second player and proba list correspond to second line of file.data)
```

6.5.4 Option files for linear systems

In each of the options file called when running the program *PIGAMES* , there are options for choosing the linear solvers to use. We give the values of those options below.

- Options for the linear solver:
  - 11 : AMG home made solver (it can be parametrize using the file `MG.options` explained hereafter).

  - 13 : ML∗ smooth aggregation solver of the ML external package.
  - 14 : UMFPACK∗ LU direct solver for sparse matrix from the UMFPACK external package.
  - 15 : AGMG∗ algebraic multigrid solver of aggregation type with K-cycle from the AGMG external package
  - 16 : SuperLU∗ LU direct solver for sparse matrix from the SuperLU external package.
  - 17 : Lapack∗ LU direct solver for dense matrix (use for small sizes) from the LAPACK external package.

  One of the above options minus 10 runs the Tarjan algorithm combined with the chosen solver.
- Options for the ergodic linear solver:
  - 11: SuperLU∗ old (2)
  - 12: Home made multigrid for Markov chain (2)
  - 13: SuperLU∗ for the stationary probability of the Markov chain and home multigrid for the singular linear system (2).
  - 15: Home made adapted multigrid (1).
  - 16: Multigrid for Markov chain (2) multiplicative special (MA)
  - 17: LAPACK∗ (2)
  - 18: LAPACK∗ (1)
  - 19: SuperLU∗ (1)
  - 20: SuperLU∗ (2)
  - 21: UMFpack∗ (1)
  - 22: UMFpack∗ (2)

  When the solver name is followed by
  - (2) it means that the computation is made in two times, first the mean value of the game is computed using the stationary probability of the Markov Chain and then the relative value is computed using a singular linear solver.
  - (1) then the two values are computed simultaneous.

  The options 12, 13, 15, 16 should be used with the `MGS.options` file for the linear singular solver options and for the options 13, 16, together with the `MGMC.options` for the solver for
- Note that all the above options with a "∗" are external solvers from external libraries that need to be installed separately. The library LAPACK is an LU solver for dense matrices and the libraries SuperLU and UMFpack are LU solvers for sparse matrices.

In the the main source directory `pigames/game/`, one can find the option files for the different home made multigrid solvers:

- `MG.options` contains the options for the multigrid solver that was implemented for this thesis.
- `MGMC.options` contains the options for the multigrid solver that finds the stationary probability of a Markov chain.

– `MGS.options` contain the options for the multigrid solver for singular systems.

Those files are prefilled with standard values that can be changed, we explain below some particular options.

– The value Multigrid Construction for the construction of the grids, in `MG.options` and `MGS.options`, can have the following values :
  – -12: for an Aggregation algorithm of [VMB96]
  – -11: for a standard AMG construction.
  – -14: for an aggregation type (for MGS.options only)
  – 22: adaptive additive (for MGS.options only)
  – 31: Smoother only (do not construct any grids).
– The value Interpolation for the construction of the interpolation operator, in `MG.options` and for `MGS.options` if Construction is -11 or -12, can have the following values :
  – 1: interpolation defined in the book of [BHM00]
  – 2: interpolation positive negative AMG of the report of [Stü01]
  – 3: interpolation positive negative AMG with truncation of the report of [Stü01]
  – 4: interpolation defined in the book [RS87]
  – 5: constant interpolation with selection of aggregation like in the book of [RS87]
  – 15: Aggregation as in [VMB96] constant operator
  – 16: Smooth Aggregation, aggregation as in [VMB96] with relaxed operator.
– The value Interpolation for the construction of the interpolation operator, in `MGMC.options` and for `MGS.options` if Construction is -14 or 22, can have the following values :
  – 20: Aggregation as in [VMB96] with constant operator
  – 21: Aggregation with scaling as in [DSMM$^+$08] and selection using first coloring scheme of [RS87].
  – 22: Aggregation as in [VMB96] and selection using first coloring scheme of [RS87].
– The value Galerkin operator, in `MGS.options` and `MG.options`, can be :
  – 1 : the restriction operator is the transpose of interpolation
  – 0: the transfer operators are defined as in [Vir07].

6.5.5 External packages

Here follow the external packages corresponding to the solvers of the previous section.

– LAPACK : http://www.netlib.org/lapack/#_software/ (from [ABB$^+$99]).
– AGMG : http://homepages.ulb.ac.be/ ynotay/ (from [Not10a]).
– SuperLU for sequential machines : http://crd.lbl.gov/ xiaoye/SuperLU/ (from [DEG$^+$99])
– UMFpack: http://www.cise.ufl.edu/research/sparse/umfpack/ (from [Dav04]).
– ML of the trilinos package: http://trilinos.sandia.gov/download/trilinos-9.0.html (from [HBH$^+$03]).
– sparseKit : http://www-users.cs.umn.edu/ saad/software/SPARSKIT/index.html (from [Saa03]).

Note that for some Linux distribution some rpm of debian packages may exists, in particular for LAPACK.

Bibliography

[AB07] Olivier Alvarez and Martino Bardi. Ergodic problems in differential games. In *Advances in dynamic game theory*, volume 9 of *Ann. Internat. Soc. Dynam. Games*, pages 131–152. Birkhäuser Boston, Boston, MA, 2007.

[ABB+99] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[AD12] Marianne Akian and Sylvie Detournay. Multigrid methods for two-player zero-sum stochastic games. *Numerical Linear Algebra with Applications*, 19(2):313–342, 2012.

[ADCTG12] Marianne Akian, Sylvie Detournay, Jean Cochet-Terrasson, and Stéphane Gaubert. Policy iteration algorithm for zero-sum stochastic games with mean payoff. *preprint*, 2012.

[AG03] M. Akian and S. Gaubert. Spectral theorem for convex monotone homogeneous maps, and ergodic control. *Nonlinear Analysis. Theory, Methods & Applications*, 52(2):637–679, 2003.

[AGG08] A. Adjé, S. Gaubert, and É. Goubault. Computing the smallest fixed point of order-preserving nonexpansive mappings arising in game theory and static analysis of programs. In *Proceedings of MTNS'08*, 2008. See also arXiv:0806.1160.

[AGG10] Assalé Adjé, Stéphane Gaubert, and Eric Goubault. Coupling policy iteration with semi-definite relaxation to compute accurate numerical invariants in static analysis. In Andrew Gordon, editor, *Programming Languages and Systems*, volume 6012 of *Lecture Notes in Computer Science*, pages 23–42. Springer Berlin / Heidelberg, 2010.

[AGG12a] A. Adjé, S. Gaubert, and É. Goubault. Coupling policy iteration with semi-definite relaxation to compute accurate numerical invariants in static analysis. *Logical methods in computer science*, 8(1):1–32, 2012.

[AGG12b] M. Akian, S. Gaubert, and A. Guterman. Tropical polyhedra are equivalent to mean payoff games. *International of Algebra and Computation*, 22(1):125001 (43 pages), 2012. doi10.1142/S0218196711006674, arxiv0912.2462.

[AGL08] Marianne Akian, Stéphane Gaubert, and Asma Lakhoua. The max-plus finite element method for solving deterministic optimal control problems: basic properties and convergence analysis. *SIAM J. Control Optim.*, 47(2):817–848, 2008.

[AGNS11] M. Akian, S. Gaubert, V. Nitica, and I. Singer. Best approximation in max-plus semimodules. *Linear Algebra and its Applications*, 435(12):3261–3296, 2011. doi10.1016/j.laa.2011.06.009, arxiv1012.5492.

[Aki90a] Marianne Akian. Analyse de l'algorithme multigrille FMGH de résolution d'équations d'Hamilton-Jacobi-Bellman. In *Analysis and optimization of systems (Antibes, 1990)*, volume 144 of *Lecture Notes in Control and Inform. Sci.*, pages 113–122. Springer, Berlin, 1990.

[Aki90b] Marianne Akian. *Méthodes multigrilles en contrôle stochastique*. Institut National de Recherche en Informatique et en Automatique (INRIA), Rocquencourt, 1990. Thèse, Université de Paris IX (Paris-Dauphine), Paris, 1990.

[Alt94] E. Altman. Flow control using the theory of zero sum Markov games. *IEEE Trans. Automat. Control*, 39(4):814–818, 1994.

[AM09] Daniel Andersson and Peter Miltersen. The complexity of solving stochastic games on graphs. In Yingfei Dong, Ding-Zhu Du, and Oscar Ibarra, editors, *Algorithms and Computation*, volume 5878 of *Lecture Notes in Computer Science*, pages 112–121. Springer Berlin / Heidelberg, 2009.

[And09] D. Andersson. Extending Friedmann s lower bound to the hoffman-karp algorithm. *preprint, June*, 2009.

[Bar09] Martino Bardi. On differential games with long-time-average cost. In *Advances in dynamic games and their applications*, volume 10 of *Ann. Internat. Soc. Dynam. Games*, pages 3–18. Birkhäuser Boston Inc., Boston, MA, 2009.

[BB95] T. Başar and P. Bernhard. *H^∞-optimal control and related minimax design problems*. Systems & Control: Foundations & Applications. Birkhäuser Boston Inc., Boston, MA, second edition, 1995. A dynamic game approach.

[BBB+10] M. Bolten, A. Brandt, J. Brannick, A. Frommer, K. Kahl, and I. Livshits. A Bootstrap Algebraic Multilevel method for Markov Chains. *ArXiv e-prints*, April 2010.

[BC89] Dimitri P. Bertsekas and David A. Castañon. Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Trans. Automat. Control*, 34(6):589–598, 1989.

[BCPS04] T.R. Bielecki, J.-P. Chancelier, S.R. Pliska, and A. Sulem. Risk sensitive portfolio optimization with transaction costs. *Journal of computational Finance*, 8(1):39–65, 2004.

[BEGM10] Endre Boros, Khaled Elbassioni, Vladimir Gurvich, and Kazuhisa Makino. A pumping algorithm for ergodic stochastic mean payoff games with perfect information. In *Integer programming and combinatorial optimization*, volume 6080 of *Lecture Notes in Comput. Sci.*, pages 341–354. Springer, Berlin, 2010.

[Bel57] Richard Bellman. *Dynamic programming*. Princeton University Press, Princeton, N. J., 1957.

[Ber87] D. P. Bertsekas. *Dynamic programming*. Prentice Hall Inc., Englewood Cliffs, NJ, 1987. Deterministic and stochastic models.

[BFS94] M. Bardi, M. Falcone, and P. Soravia. Fully discrete schemes for the value function of pursuit-evasion games. In *Advances in dynamic games and applications (Geneva, 1992)*, volume 1 of *Ann. Internat. Soc. Dynam. Games*, pages 89–105. Birkhäuser Boston, Boston, MA, 1994.

[BFS99] Martino Bardi, Maurizio Falcone, and Pierpaolo Soravia. Numerical methods for pursuit-evasion games via viscosity solutions. In *Stochastic and differential games*,

volume 4 of *Ann. Internat. Soc. Dynam. Games*, pages 105–175. Birkhäuser Boston, Boston, MA, 1999.

[BHM00] William L. Briggs, Van Emden Henson, and Steve F. McCormick. *A multigrid tutorial*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2000.

[BKS00] Martino Bardi, Shigeaki Koike, and Pierpaolo Soravia. Pursuit-evasion games with state constraints: dynamic programming and discrete-time approximations. *Discrete Contin. Dynam. Systems*, 6(2):361–380, 2000.

[BL78] A. Bensoussan and J.-L. Lions. *Applications des inéquations variationnelles en contrôle stochastique*. Dunod, Paris, 1978. Méthodes Mathématiques de l'Informatique, No. 6.

[BL82] A. Bensoussan and J.-L. Lions. *Contrôle impulsionnel et inéquations quasi variationnelles*, volume 11 of *Méthodes Mathématiques de l'Informatique [Mathematical Methods of Information Science]*. Gauthier-Villars, Paris, 1982.

[BMR85] A. Brandt, S. McCormick, and J. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. In *Sparsity and its applications (Loughborough, 1983)*, pages 257–284. Cambridge Univ. Press, Cambridge, 1985.

[BMZ09] Olivier Bokanowski, Stefania Maroso, and Hasnaa Zidani. Some convergence results for Howard's algorithm. *SIAM J. Numer. Anal.*, 47(4):3001–3026, 2009.

[BP94] Abraham Berman and Robert J. Plemmons. *Nonnegative matrices in the mathematical sciences*, volume 9 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1994. Revised reprint of the 1979 original.

[BR82] Randolph E. Bank and Donald J. Rose. Analysis of a multilevel iterative method for nonlinear finite element equations. *Math. Comp.*, 39(160):453–465, 1982.

[Bra86] Achi Brandt. Algebraic multigrid theory: the symmetric case. *Appl. Math. Comput.*, 19(1-4):23–56, 1986. Second Copper Mountain conference on multigrid methods (Copper Mountain, Colo., 1985).

[BS91] G. Barles and P. E. Souganidis. Convergence of approximation schemes for fully nonlinear second order equations. *Asymptotic Anal.*, 4(3):271–283, 1991.

[BSV04] H. Bjorklund, S. Sandberg, and S. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. Technical Report 05, DIMACS, 2004.

[BV07] H. Bjorklund and S. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discrete Appl. Math.*, 155:210229, 2007.

[BZ03] J. Frédéric Bonnans and Housnaa Zidani. Consistency of generalized finite difference schemes for the stochastic HJB equation. *SIAM J. Numer. Anal.*, 41(3):1008–1021 (electronic), 2003.

[CF09] Emiliano Cristiani and Maurizio Falcone. Fully-discrete schemes for the value function of pursuit-evasion games with state constraints. In *Advances in dynamic games and their applications*, volume 10 of *Ann. Internat. Soc. Dynam. Games*, pages 177–206. Birkhäuser Boston Inc., Boston, MA, 2009.

[CGB03] R.A. Cuninghame-Green and P. Butkovič. The equation $a \otimes x = b \otimes y$ over (max,+). *Theoretical Computer Science*, 293:3–12, 2003.

[CGG+05] A. Costan, S. Gaubert, E. Goubault, M. Martel, and S. Putot. A policy iteration algorithm for computing fixed points in static analysis of programs. In Kousha Etessami and Sriram Rajamani, editors, *Computer Aided Verification*, volume 3576 of *Lecture Notes in Computer Science*, pages 215–226. Springer Berlin / Heidelberg, 2005.

[Cha09] J. Chaloupka. Parallel algorithms for mean-payoff games: an experimental evaluation. In *Algorithms—ESA 2009*, volume 5757 of *Lecture Notes in Comput. Sci.*, pages 599–610. Springer, Berlin, 2009.

[Cha11] J. Chaloupka. *Algorithms for Mean-Payoff and Energy Games*. Phd thesis, Masaryk University, 2011.

[CIL92] Michael G. Crandall, Hitoshi Ishii, and Pierre-Louis Lions. User's guide to viscosity solutions of second order partial differential equations. *Bull. Amer. Math. Soc. (N.S.)*, 27(1):1–67, 1992.

[CLRS01] T. H. Cormen, Ch. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, second edition, 2001.

[CM69] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, ACM '69, pages 157–172, New York, NY, USA, 1969. ACM.

[CMS07] J.-P. Chancelier, M. Messaoud, and A. Sulem. A policy iteration algorithm for fixed point problems with nonexpansive operators. *Math. Methods Oper. Res.*, 65(2):239–259, 2007.

[Con92] A. Condon. The complexity of stochastic games. *Inform. and Comput.*, 96(2):203–224, 1992.

[Con93] Anne Condon. On algorithms for simple stochastic games. In *Advances in computational complexity theory (New Brunswick, NJ, 1990)*, volume 13 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 51–71. Amer. Math. Soc., Providence, RI, 1993.

[CT80] M.G. Crandall and L. Tartar. Some relations between non expansive and order preserving maps. *Proceedings of the AMS*, 78(3):385–390, 1980.

[CT01] J. Cochet-Terrasson. *Algorithmes d'itération sur les politiques pour les applications monotones contractantes*. Thèse, École des Mines de Paris, 2001.

[CTCG+98] Jean Cochet-Terrasson, Guy Cohen, Stéphane Gaubert, Michael Mc Gettrick, and Jean-Pierre Quadrat. Numerical computation of spectral elements in max-plus algebra. In *Proc. of the IFAC Conference on System Structure and Control*, Nantes, July 1998.

[CTG06] Jean Cochet-Terrasson and Stéphane Gaubert. A policy iteration algorithm for zero-sum stochastic games with mean payoff. *C. R. Math. Acad. Sci. Paris*, 343(5):377–382, 2006.

[CTGG99] J. Cochet-Terrasson, S. Gaubert, and J. Gunawardena. A constructive fixed point theorem for min-max functions. *Dynamics and Stability of Systems*, 14(4):407–433, 1999.

[Das04] A. Dasdan. Experimental analysis of the fastest optimum cycle ratio and mean algorithms. *ACM Transactions on Design Automation of Electronic Systems*, 9(4):385–418, 2004.

[Dav04] Timothy A. Davis. Algorithm 832: Umfpack v4.3—an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, 30:196–199, June 2004.

[Dav06] Timothy A. Davis. *Direct methods for sparse linear systems*, volume 2 of *Fundamentals of Algorithms*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2006.

[DEG+99] James W. Demmel, Stanley C. Eisenstat, John R. Gilbert, Xiaoye S. Li, and Joseph W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Analysis and Applications*, 20(3):720–755, 1999.

[Den67] Eric V. Denardo. Contraction mappings in the theory underlying dynamic programming. *SIAM Rev.*, 9:165–177, 1967.

[Det12] Sylvie Detournay. *Multigrid for zero-sum two player stochastic games*. Phd thesis, École Polytechnique, Palaiseau, France, 2012.

[DF68] E. V. Denardo and B. L. Fox. Multichain Markov renewal programs. *SIAM J. Appl. Math.*, 16:468–487, 1968.

[DG06] V. Dhingra and S. Gaubert. How to solve large scale deterministic games with mean payoff by policy iteration. In *Valuetools '06: Proceedings of the 1st international conference on Performance evaluation methodologies and tools*, page 12, New York, NY, USA, 2006. ACM Press.

[DIG98] A. Dasdan, S. S. Irani, and R. K. Guptka. An experimental study of minimum mean cycle algorithms. Technical report # 98-32, UCI-ICS, 1998.

[DS00] Tuğrul Dayar and William J. Stewart. Comparison of partitioning techniques for two-level iterative solvers on large, sparse Markov chains. *SIAM J. Sci. Comput.*, 21(5):1691–1705 (electronic), 2000. Iterative methods for solving systems of algebraic equations (Copper Mountain, CO, 1998).

[DSMM+08] H. De Sterck, Thomas A. Manteuffel, Stephen F. McCormick, Quoc Nguyen, and John Ruge. Multilevel adaptive aggregation for Markov chains, with application to web ranking. *SIAM J. Sci. Comput.*, 30(5):2235–2262, 2008.

[DSMM+10a] H. De Sterck, T. A. Manteuffel, S. F. McCormick, K. Miller, J. Pearson, J. Ruge, and G. Sanders. Smoothed aggregation multigrid for Markov chains. *SIAM J. Sci. Comput.*, 32(1):40–61, 2010.

[DSMM+10b] H. De Sterck, T. A. Manteuffel, S. F. McCormick, K. Miller, J. Ruge, and G. Sanders. Algebraic multigrid for Markov chains. *SIAM J. Sci. Comput.*, 32(2):544–562, 2010.

[DSMMS11] H. De Sterck, K. Miller, T. Manteuffel, and G. Sanders. Top-level acceleration of adaptive algebraic multilevel methods for steady-state solution to Markov chains. volume 35, pages 375–403, 2011.

[DSMSW10] H. De Sterck, K. Miller, G. Sanders, and M. Winlaw. Recursively accelerated multilevel aggregation for Markov chains. *SIAM J. Sci. Comput.*, 32(3):1652–1671, 2010.

[ES11] R. J. Elliott and T.K̃. Siu. A stochastic differential game for optimal investment of an insurer with regime switching. *Quant. Finance*, 11(3):365–380, 2011.

[Fal06] M. Falcone. Numerical methods for differential games based on partial differential equations. *Int. Game Theory Rev.*, 8(2):231–272, 2006.

[Fea10a] John Fearnley. Exponential lower bounds for policy iteration. In *Automata, Languages and Programming*, pages 551–562, 2010.

[Fea10b] John Fearnley. *Strategy Algorithms for Games and Markov Decision Processes*. PhD thesis, The University of Warwick, Coventry, United Kingdom, 2010.

[Fle06] Wendell H. Fleming. Risk sensitive stochastic control and differential games. *Commun. Inf. Syst.*, 6(3):161–177, 2006.

[FM00] Wendell H. Fleming and William M. McEneaney. A max-plus-based algorithm for a Hamilton-Jacobi-Bellman equation of nonlinear filtering. *SIAM J. Control Optim.*, 38(3):683–710 (electronic), 2000.

[Fri73] Avner Friedman. Stochastic games and variational inequalities. *Arch. Rational Mech. Anal.*, 51:321–346, 1973.

[Fri09] Oliver Friedmann. An exponential lower bound for the parity game strategy improvement algorithm as we know it. In *LICS*, pages 145–156. IEEE Computer Society, 2009.

[Fri11] Oliver Friedmann. An exponential lower bound for the latest deterministic strategy iteration algorithms. May 2011.

[FS89] W. H. Fleming and P. E. Souganidis. On the existence of value functions of two-player, zero-sum stochastic differential games. *Indiana Univ. Math. J.*, 38(2):293–314, 1989.

[FS06] W. H. Fleming and H. M. Soner. *Controlled Markov processes and viscosity solutions*, volume 25 of *Stochastic Modelling and Applied Probability*. Springer, New York, second edition, 2006.

[FV97] Jerzy Filar and Koos Vrieze. *Competitive Markov decision processes*. Springer-Verlag, New York, 1997.

[FVZ05] Robert D. Falgout, Panayot S. Vassilevski, and Ludmil T. Zikatanov. On two-grid convergence estimates. *Numer. Linear Algebra Appl.*, 12(5-6):471–494, 2005.

[GG98] S. Gaubert and J. Gunawardena. The duality theorem for min-max functions. *C.R. Acad. Sci.*, 326(1):43–48, 1998.

[GG04] S. Gaubert and J. Gunawardena. The Perron-Frobenius theorem for homogeneous, monotone functions. *Trans. of AMS*, 356(12):4931–4950, 2004.

[GGTW09] L. Georgiadis, A. V. Goldberg, R. E. Tarjan, and R. F. F. Werneck. An experimental study of minimum mean cycle algorithms. In *Proceedings of the Eleventh Workshop on Algorithm Engineering and Experiments (ALENEX09)*, pages 1–13, 2009.

[GGTZ07] Stephane Gaubert, Eric Goubault, Ankur Taly, and Sarah Zennou. Static analysis by policy iteration on relational domains. In Rocco De Nicola, editor, *Programming Languages and Systems*, volume 4421 of *Lecture Notes in Computer Science*, pages 237–252. Springer Berlin / Heidelberg, 2007.

[GKK88] V. A. Gurvich, A. V. Karzanov, and L. G. Khachiyan. Cyclic games and finding minimax mean cycles in digraphs. *Zh. Vychisl. Mat. i Mat. Fiz.*, 28(9):1407–1417, 1439, 1988.

[GL81] A. George and J.W.H. Liu. *Computer solution of large sparse positive definite systems*. Prentice-Hall series in computational mathematics. Prentice-Hall, 1981.

[GS07a] S. Gaubert and S. Sergeev. Cyclic projectors and separation theorems in idempotent convex geometry. *Fundamentalnaya i prikladnaya matematika*, 13(4):33–52, 2007. Engl. translation in Journal of Mathematical Sciences (Springer, New-York), Vol. 155, No. 6, pp.815–829, 2008.

[GS07b] Thomas Gawlitza and Helmut Seidl. Precise fixpoint computation through strategy iteration. In Rocco De Nicola, editor, *ESOP*, volume 4421 of *Lecture Notes in Computer Science*, pages 300–315. Springer, 2007.

[GS07c] Thomas Gawlitza and Helmut Seidl. Precise relational invariants through strategy iteration. In Jacques Duparc and Thomas Henzinger, editors, *Computer Science Logic*, volume 4646 of *Lecture Notes in Computer Science*, pages 23–40. Springer Berlin / Heidelberg, 2007.

[GSA+12] Thomas Martin Gawlitza, Helmut Seidl, Assalé Adjé, Stéphane Gaubert, and Éric Goubault. Abstract interpretation meets convex optimization. *Journal of Symbolic Computation*, 47(12):1416 – 1446, 2012. International Workshop on Invariant Generation.

[Gun03] J. Gunawardena. From max-plus algebra to nonexpansive maps: a nonlinear theory for discrete event systems. *Theoretical Computer Science*, 293:141–167, 2003.

[GVL96] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[HBH+03] Michael Heroux, Roscoe Bartlett, Vicki Howle Robert Hoekstra, Jonathan Hu, Tamara Kolda, Richard Lehoucq, Kevin Long, Roger Pawlowski, Eric Phipps, Andrew Salinger, Heidi Thornquist, Ray Tuminaro, James Willenbring, and Alan Williams. An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, 2003.

[HK66a] A. J. Hoffman and R. M. Karp. On nonterminating stochastic games. *Management Sci.*, 12:359–370, 1966.

[HK66b] A. J. Hoffman and R. M. Karp. On nonterminating stochastic games. *Management sciences*, 12(5):359–370, 1966.

[HL94] Graham Horton and Scott T. Leutenegger. A multi-level solution algorithm for steady-state markov chains. In *SIGMETRICS '94: Proceedings of the 1994 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, pages 191–200, New York, NY, USA, 1994. ACM.

[HMZ11] T.D. Hansen, P.B. Miltersen, and U. Zwick. Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. In *Innovations in Computer Science 2011*, pages 253–263. Tsinghua University Press, 2011.

[Hop86] Ronald H. W. Hoppe. Multigrid methods for Hamilton-Jacobi-Bellman equations. *Numer. Math.*, 49(2-3):239–254, 1986.

[Hop87] Ronald H. W. Hoppe. Multigrid algorithms for variational inequalities. *SIAM J. Numer. Anal.*, 24(5):1046–1065, 1987.

[How60] Ronald A. Howard. *Dynamic programming and Markov processes*. The Technology Press of M.I.T., Cambridge, Mass., 1960.

[JPZ06] M. Jurdziński, M. Paterson, and U. Zwick. A deterministic subexponential algorithm for solving parity games. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA 2006)*, January 2006.

[Jur98] Marcin Jurdziński. Deciding the winner in parity games is in UP ∩ co-UP. *Inform. Process. Lett.*, 68(3):119–124, 1998.

[KD92] Harold J. Kushner and Paul G. Dupuis. *Numerical methods for stochastic control problems in continuous time*, volume 24 of *Applications of Mathematics (New York)*. Springer-Verlag, New York, 1992.

[KK71] H. Kushner and A. Kleinman. Accelerated procedures for the solution of discrete markov control problems. *Automatic Control, IEEE Transactions on*, 16(2):147 – 152, apr 1971.

[Koh80] E. Kohlberg. Invariant half-lines of nonexpansive piecewise-linear transformations. *Math. Oper. Res.*, 5(3):366–372, 1980.

[Kol92] V. N. Kolokoltsov. On linear, additive, and homogeneous operators in idempotent analysis. In *Idempotent analysis*, volume 13 of *Adv. Soviet Math.*, pages 87–101. Amer. Math. Soc., Providence, RI, 1992.

[Kus77] Harold J. Kushner. *Probability methods for approximations in stochastic control and for elliptic equations*. Academic Press [Harcourt Brace Jovanovich Publishers], New York, 1977. Mathematics in Science and Engineering, Vol. 129.

[LCS08] Dongxu Li, Jose B. Cruz, Jr., and Corey J. Schumacher. Stochastic multi-player pursuit-evasion differential games. *Internat. J. Robust Nonlinear Control*, 18(2):218–247, 2008.

[LL69] T. M. Liggett and S. A. Lippman. Stochastic games with perfect information and time average payoff. *SIAM Rev.*, 11:604–607, 1969.

[LLP+99] A. J. Lazarus, D. E. Loeb, J. G. Propp, W. R. Stromquist, and D. H. Ullman. Combinatorial games under auction play. *Games Econom. Behav.*, 27(2):229–264, 1999.

[LLPU96] Andrew J. Lazarus, Daniel E. Loeb, James G. Propp, and Daniel Ullman. Richman games. In *Games of no chance (Berkeley, CA, 1994)*, volume 29 of *Math. Sci. Res. Inst. Publ.*, pages 439–449. Cambridge Univ. Press, Cambridge, 1996.

[LM80] P.-L. Lions and B. Mercier. Approximation numérique des équations de Hamilton-Jacobi-Bellman. *RAIRO Anal. Numér.*, 14(4):369–393, 1980.

[McC87] Stephen F. McCormick, editor. *Multigrid methods*, volume 3 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1987.

[MFL04] W.M. McEneaney, B.G. Fitzpatrick, and I.G. Lauko. Stochastic game approach to air operations. *IEEE Trans. Aero. Elec. Systems*, 40:1191–1216, 2004.

[MH86] U. Meister and U. Holzbaur. A polynomial time bound for howard's policy improvement algorithm. *OR Spectrum*, 8:37–40, 1986. 10.1007/BF01720771.

[MM98] Ivo Marek and Petr Mayer. Convergence analysis of an iterative aggregation/disaggregation method for computing stationary probability vectors of stochastic matrices. *Numer. Linear Algebra Appl.*, 5(4):253–274, 1998.

[MM03] Ivo Marek and Petr Mayer. Convergence theory of some classes of iterative aggregation/disaggregation methods for computing stationary probability vectors of stochastic matrices. *Linear Algebra Appl.*, 363:177–200, 2003. Special issue on nonnegative matrices, M-matrices and their generalizations (Oberwolfach, 2000).

[MN08] C. Mense and R. Nabben. On algebraic multi-level methods for non-symmetric systems—comparison results. *Linear Algebra Appl.*, 429(10):2567–2588, 2008.

[MP77] Carl D. Meyer, Jr. and R. J. Plemmons. Convergent powers of a matrix with applications to iterative methods for singular linear systems. *SIAM J. Numer. Anal.*, 14(4):699–705, 1977.

[MZ05] Rémi Munos and Hasnaa Zidani. Consistency of a simple multidimensional scheme for Hamilton-Jacobi-Bellman equations. *C. R. Math. Acad. Sci. Paris*, 340(7):499–502, 2005.

[Not10a] Yvan Notay. An aggregation-based algebraic multigrid method. *Electronic Transactions on Numerical Analysis*, 37:123–146, 2010.

[Not10b] Yvan Notay. Algebraic analysis of two-grid methods: The nonsymmetric case. *Numer. Linear Algebra Appl.*, 17(1):73–96, 2010.

[NP79] M. Neumann and R. J. Plemmons. Convergent nonnegative matrices and iterative methods for consistent linear systems. *Numer. Math.*, 31(3):265–279, 1978/79.

[NS03] A. Neyman and S. Sorin. *Stochastic games and applications*, volume 570. Springer Netherlands, 2003.

[NV08] Yvan Notay and Panayot S. Vassilevski. Recursive Krylov-based multigrid cycles. *Numer. Linear Algebra Appl.*, 15(5):473–487, 2008.

[Obe05] A. M. Oberman. A convergent difference scheme for the infinity Laplacian: construction of absolutely minimizing Lipschitz extensions. *Math. Comp.*, 74(251):1217–1230, 2005.

[OZ05] N. Shimkin O. Ziv. Multigrid methods for policy evaluation and reinforcement learning. In *Proc. IEEE International Symposium on Intelligent Control (ISIC05)*. IEEE, 2005.

[PAI69] M. A. Pollatschek and B. Avi-Itzhak. Algorithms for stochastic games with geometrical interpretation. *Management Sci.*, 15:399–415, 1968/1969.

[PB74] George Poole and Thomas Boullion. A survey on M-matrices. *SIAM Rev.*, 16:419–427, 1974.

[PB79] Martin L. Puterman and Shelby L. Brumelle. On the convergence of policy iteration in stationary dynamic programming. *Math. Oper. Res.*, 4(1):60–69, 1979.

[Ple76] R. J. Plemmons. M-matrices leading to semiconvergent splittings. *Linear Algebra and Appl.*, 15(3):243–252, 1976.

[Por75] Evan L. Porteus. Bounds and transformations for discounted finite Markov decision chains. *Operations Res.*, 23(4):761–784, 1975.

[Por80] E. L. Porteus. Improved iterative computation of the expected discounted return in Markov and semi-Markov chains. *Z. Oper. Res. Ser. A-B*, 24(5):155–170, 1980.

[Por72] Evan L. Porteus. Some bounds for discounted sequential decision processes. *Management Sci.*, 18:7–11, 1971/72.

[PSS96] Bernard Philippe, Youcef Saad, and William J. Stewart. Numerical methods in markov chain modelling. *Operations Research*, 40:1156–1179, 1996.

[PSSW09] Yuval Peres, Oded Schramm, Scott Sheffield, and David B. Wilson. Tug-of-war and the infinity Laplacian. *J. Amer. Math. Soc.*, 22(1):167–210, 2009.

[Pur95] Anuj Puri. *Theory of hybrid systems and discrete event systems.* PhD thesis, University of California at Berkeley, Berkeley, CA, USA, 1995.

[Put94] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming.* Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics. John Wiley & Sons Inc., New York, 1994.

[Rag03] T. E. S. Raghavan. Finite-step algorithms for single-controller and perfect information stochastic games. In *Stochastic games and applications (Stony Brook, NY, 1999)*, volume 570 of *NATO Sci. Ser. C Math. Phys. Sci.*, pages 227–251. Kluwer Acad. Publ., Dordrecht, 2003.

[RCN73] S. S. Rao, R. Chandrasekaran, and K. P. K. Nair. Algorithms for discounted stochastic games. *J. Optimization Theory Appl.*, 11:627–637, 1973.

[Roc70] R. T. Rockafellar. *Convex analysis.* Princeton University Press, 1970.

[RS87] J. W. Ruge and K. Stüben. Algebraic multigrid. In Stephen F. McCormick, editor, *Multigrid methods*, volume 3 of *Frontiers Appl. Math.*, pages 73–130. SIAM, Philadelphia, PA, 1987.

[RS01] A. M. Rubinov and I. Singer. Topical and sub-topical functions, downward sets and abstract convexity. *Optimization*, 50(5-6):307–351, 2001.

[RS03] T. E. S. Raghavan and Zamir Syed. A policy-improvement type algorithm for solving zero-sum two-person stochastic games of perfect information. *Math. Program.*, 95(3, Ser. A):513–532, 2003.

[RTV85] T. E. S. Raghavan, S. H. Tijs, and O. J. Vrieze. On stochastic games with additive reward and transition structure. *J. Optim. Theory Appl.*, 47(4):451–464, 1985.

[Saa92] Youcef Saad. *Numerical methods for large eigenvalue problems.* Algorithms and Architectures for Advanced Scientific Computing. Manchester University Press, Manchester, 1992.

[Saa03] Yousef Saad. *Iterative methods for sparse linear systems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2003.

[Sha53] L. S. Shapley. Stochastic games. *Proc. Nat. Acad. Sci. U. S. A.*, 39:1095–1100, 1953.

[Sor03] Sylvain Sorin. Classification and basic tools. In *Stochastic games and applications (Stony Brook, NY, 1999)* [NS03], pages 27–36.

[Sor04] S. Sorin. Asymptotic properties of monotonic nonexpansive mappings. *Discrete Event Dyn. Syst.*, 14(1):109–122, 2004.

[Ste97] William J. Stewart. Numerical methods for computing stationary distributions of finite irreducible markov chains. In *Advances in Computational Probability.* Kluwer Academic Publishers, 1997.

[Stü01] K. Stüben. An introduction to algebraic multigrid. In U. Trottenberg, C. W. Oosterlee, and A. Schüller, editors, *Multigrid.* Academic Press Inc., San Diego, CA, 2001.

[Świ96] Andrzej Świech. Another approach to the existence of value functions of stochastic differential games. *J. Math. Anal. Appl.*, 204(3):884–897, 1996.

[Tar72] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

[TR97] Frank Thuijsman and Thirukkannamangai E. S. Raghavan. Perfect information stochastic games and related classes. *Internat. J. Game Theory*, 26(3):403–408, 1997.

[TY10] Eran Treister and Irad Yavneh. Square and stretch multigrid for stochastic matrix eigenproblems. *Numer. Linear Algebra Appl.*, 17(2-3):229–251, 2010.

[TY11] Eran Treister and Irad Yavneh. On-the-fly adaptive smoothed aggregation multigrid for Markov chains. *SIAM J. Sci. Comput.*, 33(5):2927–2949, 2011.

[Var62] Richard S. Varga. *Matrix iterative analysis*. Prentice-Hall Inc., Englewood Cliffs, N.J., 1962.

[VBM01] Petr Vaněk, Marian Brezina, and Jan Mandel. Convergence of algebraic multigrid based on smoothed aggregation. *Numer. Math.*, 88(3):559–579, 2001.

[vdW77] J. van der Wal. Discounted Markov games; successive approximation and stopping times. *Internat. J. Game Theory*, 6(1):11–22, 1977.

[vdW78] J. van der Wal. Discounted Markov games: generalized policy iteration method. *J. Optim. Theory Appl.*, 25(1):125–138, 1978.

[vdW80] J. van der Wal. Successive approximations for average reward Markov games. *Internat. J. Game Theory*, 9(1):13–24, 1980.

[Vir07] Elena Virnik. An algebraic multigrid preconditioner for a class of singular M-matrices. *SIAM J. Sci. Comput.*, 29(5):1982–1991 (electronic), 2007.

[VJ00] J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games. In *Proceedings of 12th Int. Conf. on Computer Aided Verification (CAV'2000)*, July 2000.

[VMB96] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56(3):179–196, 1996. International GAMM-Workshop on Multi-level Methods (Meisdorf, 1994).

[vN76] J. A. E. E. van Nunen. A set of successive approximation methods for discounted markovian decision problems. *Mathematical Methods of Operations Research*, 20:203–208, 1976. 10.1007/BF01920264.

[VNM44] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton University Press, Princeton, 1944.

[Vri03] O. J. Vrieze. Stochastic games, practical motivation and the orderfield property for special classes. In *Stochastic games and applications (Stony Brook, NY, 1999)* [NS03], pages 215–225.

[Yan81] Mihalis Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM J. Algebraic Discrete Methods*, 2(1):77–79, 1981.

[Ye05] Yinyu Ye. A new complexity result on solving the Markov decision problem. *Mathematics of Operations Research*, 30(3):733–749, 2005.

[Ye11] Yinyu Ye. The simplex and policy-iteration methods are strongly polynomial for the markov decision problem with a fixed discount rate, 2011.

[YTO91] N. Young, R.E. Tarjan, and J.B. Orlin. Faster parametric shortest path and minimum balance algorithms. *Networks*, 21:205–221, 1991.

[ZP96] Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theoret. Comput. Sci.*, 158(1-2):343–359, 1996.