NTNU – Trondheim
Norwegian University of
Science and Technology

Numerical Methods and Brute Force Optimisation for a General Formulation of the Mean Field Game Equations

Snorre Alexander
Berthelsen Husby

Master of Science in Physics and Mathematics
Submission date: July 2015
Supervisor: Espen Robstad Jakobsen, MATH

Norwegian University of Science and Technology
Department of Mathematical Sciences

Numerical Methods and Brute Force Optimisation for a General Formulation of the Mean Field Game Equations

Author:
Snorre Alexander Berthelsen Husby

Supervisor:
Espen Robstad Jakobsen

NTNU
Department
of Mathematical Sciences

Summary

This thesis considers the numerical solution of a general formulation of the mean field game (MFG) equations. MFG are a relatively new field with few general results but with many modelling applications. The MFG equations consist of a Hamilton-Jacobi-Bellman equation (HJB) and a Fokker-Planck equation (FP) which are coupled by an optimal control.

In established theory and existing numerical methods for MFG, this optimal control is assumed known as a function of the Hamiltonian in the HJB equation. This reduces the generality of the methods. In this thesis, we instead consider the general formulation for which the optimal control is unknown. We develop brute force optimisation methods to directly compute this based on the discretised Hamiltonian. We also develop robust numerical schemes for the HJB and FP equations that, together with the brute force methods, allow the computation of the solutions of the general formulation of the MFG equations. Of particular note here are methods to evaluate a diffusion tensor in the FP equation.

Due to the coupled nature of the MFG equations, the numerical computation of their solutions require a solution procedure. In this procedure the HJB and optimal control are solved at the same time, before the solution of these are used to compute the solution of the FP equation. This solution is again used to solve for the HJB equation and optimal control. It is expected that computing several iterations of this solution procedure is necessary. However, we encountered several cases in which this procedure failed to converge. It is important to remark that this is not a phenomenon unique to our thesis, but is reported for other numerical methods for less general forms of the MFG equations by other authors.

After a wide range of numerical tests, we develop an intuition for the causes of this lack of convergence. In addition, we introduce alternative solution procedures with slightly better convergence properties. We are also able to produce solutions that converge with a refined mesh for some cases of the MFG equations for which there to our knowledge does not exist existence theorems for. These results remain speculative.

Based on our experiences, we propose topics of future work to deal with the numerical solution of the MFG equations. We also present some ideas for improvements to our solution methods.

Sammendrag

I denne masteroppgaven tar vi for oss numeriske løsninger av en generell formulering av "mean field game"-likningene (MFG). MFG er et relativt nytt felt med få generelle resultater, men med åpenbar bruk for matematisk modellering. MFG-likningene består av en Hamilton-Jacobi-Bellman likning (HJB) og en Fokker-Planck likning (FP) som er koblet via en optimal kontroll.

I etablert teori og eksisterende numeriske metoder for MFG, er denne optimale kontrollen antatt kjent som en funksjon av den hamiltonske funksjonen i HJB-likningen. I denne masteroppgaven ser vi på en generell formulering av MFG der den optimale kontrollen er ukjent. Vi utvikler "brute force" optimeringsmetoder for å beregne den optimale kontrollen direkte basert på den diskretiserte hamiltonske funksjonen. We utvikler også robuste numeriske skjemaer for HJB- og FP-likningene som, sammen med optimeringsmetodene, tillater beregningen av løsninger for den generelle formuleringen av MFG-likningene. Verdt å merke her er ulike metoder for å håndtere en diffusjonstensor som opptrer i FP-likningen.

På grunn av den koblede naturen av MFG-likningene, vil numeriske beregninger av løsningene deres trenge en løsningsprosedyre. I denne prosedyren er HJB-likningen og den optimale kontrollen løst for samtidig, før løsningen av disse blir brukt til å beregne løsningen av FP-likningen. Løsningen av FP-likningen blir så brukt for å løse HJB-likningen og den optimale kontrollen. Det bør derfor forventes at det trengs flere iterasjoner av denne løsningsprosedyren for at en skal oppnå konvergens. Derimot oppdaget vi flere tilfeller der denne prosedyren ikke konvergerte. Det er viktig å påpeke at dette ikke er et unikt fenomen for denne masteroppgaven, men også er beskrevet for andre numeriske metoder for mindre generelle former av MFG-likningene av andre forfattere.

Etter et spenn av numeriske tester utvikler vi en intuisjon for årsakene bak mangel på konvergens for løsningsprosedyren. I tillegg introduserer vi alternative løsningsprosedyrer som har litt bedre konvergensegenskaper. Vi er også istand til å produsere konvergente numeriske løsninger for enkelte tilfeller som ikke dekkes av publiserte resultater, såvidt vi er kjent. Disse resultatene forblir spekulative.

Basert på våre erfaringer foreslår vi emner for fremtidig arbeid på numerisk beregning av MFG-likningene. We foreslår også noen ideer for utbedringer av våre egne metoder.

Preface

This thesis concludes my Master of Science education in Industrial Mathematics at The Norwegian University of Science and Technology (NTNU) in Trondheim. The thesis was produced throughout my 10th semester in the spring of 2015, at the Department of Mathematical Sciences (IME). During my years at NTNU, I have matured and developed my intellect to its peak; thus far. It is not a journey I have walked alone.

First of all, I would like to thank my supervisor, professor Espen Robstad Jacobsen, for all his time, comments, remarks and feedback throughout the work on this thesis and its preparatory project. Our weekly meetings were events where mathematical illusions were broken and laughter was shared. Without his casual remark about "mean field games" one day in September 2014, I would never have written this particular thesis.

I will forever be grateful for all instructors I have had the last five years. Without their influences, passions, styles of teaching and humour, I might have ended up an engineer instead. I will single out a special thanks to professor Elena Celledoni at NTNU for the writing recommendation letters that got me accepted for my exchange year at UC Berkeley in California.

I would never have been where I am now without my parents. I would like to thank my mother for asking me if I *really* were satisfied with a grade 4 out of 6 on my first test in junior high school. Without this remark, the standards to which I hold myself would be considerably lower. I would also like to thank my father for the time he encouraged me to quit the Army and get a degree while I was still young... and to go back into the Army afterwards if I still wanted to drive a tank.

All classmates, past and present, have provided me with friends and company for all my years at university. I would in particular like to thank my regular lunch buddies: Geir Amund Svan Hasle, Trygve Bærland and Audun Reigstad. Without them I would surely have spent less time drinking coffee. Furthermore, I would like to thank Trygve Reinertsen Sørgård and Liv Monica Trondrud for their proofreading efforts.

Lastly, I would like to thank all those who have ever doubted in my capabilities. May I never stop proving them wrong.

Nomenclature

Abbreviations

MFG	Mean Field Game
HJB	Hamilton-Jacobi-Bellman
FP	Fokker-Planck
ND	N-dimensional *or* N dimensions
FMM	Fast-Marching Method

Notation

$a^+ = \max(a, 0)$ for any number a.
$a^- = \min(a, 0)$ for any number a.

Symbols

α	Strategy *or* control
u	Cost *or* potential
m	Distribution (of agents)
f	(Agent) velocity
$\mathbb{D}$	Diffusion tensor
σ	Diffusion (function)
L	Running cost or cost function

Contents

Chapter 1

Introduction

In this chapter we introduce the mean field game equations in the general form that will be dealt with. We describe how the quantities in the equations are to be interpreted. The chapter concludes with an overview of the thesis, and allude to our primary findings.

1.1 The mean field game equations

The mean field game (MFG) equations describe the movement of a future-anticipating, rational continuum of agents. MFG is a new, relatively unexplored problem with many real-world applications. The MFG equations take the form of two partial differential equations, one Hamilton-Jacobi-Bellman (HJB) equation with a terminal condition and one Fokker-Planck (FP) equation with an initial condition.

$$u_t + \inf_{\alpha \in \mathcal{A}} \left(L + \sum_{i=1}^{N} \frac{\partial u}{\partial x_i} f_i + \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathbb{D}_{ij} \frac{\partial^2 u}{\partial x_i x_j} \right) = 0, \tag{1.1a}$$

$$m_t + \sum_{i=1}^{N} \frac{\partial}{\partial x_i}(f_i m) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\partial^2}{\partial x_i \partial x_j} \left(\mathbb{D}_{ij} m \right), \tag{1.1b}$$

$$\begin{aligned} u(T, x) &= u_T(x), \\ m(0, x) &= m_0(x). \end{aligned} \tag{1.1c}$$

The quantities m, u denote respectively the **distribution** of agents and the **potential** under which they move. The infimum-term in (1.1a) is referred to as the **Hamiltonian**. From the Hamiltonian, the agents' optimal **control** or strategy α may be obtained. The other functions and expressions are

formally defined as follows,

$$\begin{aligned}
&\mathbb{D}_{ij} = (\sigma\sigma^T)_{ij}, \\
&f = f(t,x,\alpha) : [0,T] \times \mathbb{R}^N \times \mathcal{A} \mapsto \mathbb{R}^N, \\
&L = L(t,x,\alpha,m) : [0,T] \times \mathbb{R}^N \times \mathcal{A} \times \mathcal{P} \mapsto \mathbb{R}, \\
&\sigma = \sigma(t,x,\alpha) : [0,T] \times \mathbb{R}^N \times \mathcal{A} \mapsto \mathbb{R}^M, \\
&\mathcal{A} \subset \mathbb{R}^N.
\end{aligned} \tag{1.2}$$

The interpretation of these functions in the context of MFG will become more obvious over the next chapters. We will refer to L as the *running cost*, f as the **velocity** and $\mathbb{D}$ as the **diffusion tensor**. Note especially the terminal-initial conditions (1.1c); this implies that (1.1a) goes backwards in time and that (1.1b) goes forwards in time. This backwards-forwards structure introduces difficulties when solving (1.1a)(1.1b) numerically. We will return to this later.

Forms of the MFG equations can heuristically be derived from (at least): game theory (see [7]), optimal control theory (see appendix A) and statistical physics (as argued in [13]). In regards to game theory, for example, MFG consider Nash-equilibria[a] in N-player games as $N \to \infty$, where individual players become indistinguishable. Players thus form their strategies based on their own state and the statistics of the overall community of other players, *as well as* their anticipation of the future.

1.2 An overview of the thesis

The author does not know of any published work that prove the existence of solutions for the general forms (1.1a)(1.1b), which makes the numerical computation of these both speculative and interesting. The existing material on the MFG equations apply for variations of (1.1). We present the main results in for these variations in chapter 2 and speculate on how these results may be interpreted in the context of (1.1). We also describe existing numerical methods for these variations of MFG.

We present monotone discretisations of the HJB equation (1.1a) in one and two dimensions in chapter 3. In chapter 4 we discretise the one- and two-dimensional FP equations. We present several discretisations for FP in 1D, and present several ways to discretise with the potentially problematic

[a] A group of players are in a Nash equilibrium if each one is making the best decision possible, taking into account the decisions of all others in the game as long the others' decisions remains unchanged.

diffusion tensor $\mathbb{D}$. Brute force, generic optimisation methods for computing the optimal control α are developed in chapter 5.

All threads converge to the numerical tests in chapter 6. Here, the discretisations from chapters 3 to 4 are tested individually in verification tests. We also apply the MFG equations (1.1) to several interesting modelling scenarios in one and two dimensions, and present numerical solutions of these. We evaluate the convergence of the schemes as the mesh is refined. Several interesting results surface here, which shed light on some intriguing difficulties that apply for the numerical solution of the MFG equations.

We summarise and discuss our findings from the numerical tests in chapter 7. Chapter 8 concludes the thesis. Auxiliary results and implementation details are located in the appendices.

Chapter 2

Theoretical background and objectives

Before we proceed towards the core of this thesis, we present an overview of the most important results for MFG within the context of this thesis. This includes known conditions for the existence and uniqueness of solutions, as well as an overview of previous work on numerical methods for MFG. This is useful for gauging the state of the field, and we will echo back to this chapter when evaluating our numerical results. In light of this previous work, we also present the objectives of our own discretisations.

2.1 Existence and uniqueness of solutions

Mean field games were introduced in lectures[19] and papers[17][18] by Pierre-Louis Lions in 2006, and has been developed further by the likes of Pierre Cardaliaguet. As such, the analytic theory of MFG is in its relative infancy. The main results may be found in [15][7][10].

We present theorems on the existence and uniqueness of solutions for three forms of the MFG equations. In common for all these forms is that the optimal control α is assumed known as a function of the Hamiltonian.

2.1.1 MFG with known control and convex Hamiltonian

We start by presenting the first known non-stationary formulation of the MFG equations from [18]. Relative to (1.1), the agent velocity is set as $f(\alpha) = \alpha$ and the diffusion is constant $\mathbb{D} = \sigma^2 I$. The Hamiltonian is then split into two terms, a reduced form $H(x, p)$ and a generic cost term $F(x, m)$. with the optimal control assumed known as $\alpha = -\frac{\partial H}{\partial p}$. This gives the simpler form,

$$u_t - H(x, Du) + \frac{\sigma^2}{2} \sum_{i=1}^{N} \frac{\partial^2 u}{\partial x_i^2} = -F(x, m), \tag{2.1a}$$

$$m_t - \sum_{i=1}^{N} \frac{\partial}{\partial x_i}\left(\frac{\partial H}{\partial p} m\right) = \frac{\sigma^2}{2} \sum_{i=1}^{N} \frac{\partial^2 m}{\partial x_i^2}, \tag{2.1b}$$

$$\begin{aligned} u(T,x) &= G(x, m(T)), \\ m(0,x) &= m_0(x). \end{aligned}$$

To the author's knowledge, no derivation of this form exists in published papers. We assume that the functions F, G are local and present the following theorem from [17][18] without proof[a]:

Theorem 1. *Assume that the following conditions hold:*

- *The Hamiltonian $H(x,p)$ in* (2.1a) *is strictly convex and C^1 with respect to p.*
- *The Hamiltonian $H(x,p)$ in* (2.1a) *is Lipschitz continuous with respect to x, uniformly from bounded p.*
- *The functions F, G satisfy suitable growth conditions*[b].

Then the coupled problem (2.1a)(2.1b) *has at least one solution on $\Omega = [0,1]^d$ with periodic boundary conditions*[c].

Let $\mathcal{P}_1$ be the set of first order Borel probability measures. If in addition F, G are monotone over m in the following sense,

$$\begin{aligned} &\int_{\mathbb{R}^d} \left(F(x,m_1) - F(x,m_2)\right) d(m_1 - m_2)(x)dx > 0, \\ &\int_{\mathbb{R}^d} \left(G(x,m_1) - G(x,m_2)\right) d(m_1 - m_2)(x)dx \geq 0, \\ &\qquad\qquad\qquad\qquad \forall m_1, m_2 \in \mathcal{P}_1, m_1 \neq m_2 \end{aligned} \tag{2.2}$$

where $d(\cdot)$ is the Wasserstein metric, then the coupled problem (2.1a)(2.1b) *has a unique solution.*

[a] The theorem is essentially presented without proof in [17][18]. It is possible that the proof appears in the original series of lectures, see [19].

[b] The theorem does not specify which growth conditions.

[c] In [17] is is reported that the proof will hold for other choices of domains, such as $\mathbb{R}^d$, as long as the growth conditions are suitably modified.

2.1.2 MFG with known control and quadratic Hamiltonian

This is a special case we will refer to as the **canonical MFG** equations. The equations (1.1a)(1.1b) are modified by setting:

$$\begin{aligned} L(t,x,\alpha,m) &= \frac{1}{2}|\alpha|^2 + F(t,x,m), \\ f(t,x,\alpha) &= \alpha, \\ \mathbb{D} &= \sigma^2 I. \end{aligned} \tag{2.3}$$

In this case, the Hamiltonian in (1.1a) is analytically resolvable and one finds that $\alpha_i = -u_{x_i}$. The canonical MFG equations become

$$u_t - \frac{1}{2}\sum_{i=1}^{N}\left(\frac{\partial u}{\partial x_i}\right)^2 + \frac{\sigma^2}{2}\sum_{i=1}^{N}\frac{\partial^2 u}{\partial x_i^2} = -F(t,x,m), \tag{2.4a}$$

$$m_t - \sum_{i=1}^{N}\frac{\partial}{\partial x_i}\left(\frac{\partial u}{\partial x_i}m\right) = \frac{\sigma^2}{2}\sum_{i=1}^{N}\frac{\partial^2 m}{\partial x_i^2}. \tag{2.4b}$$

This theorem from [7] applies for the canonical MFG equations (2.4):

Theorem 2. *Assume the following conditions hold:*

- *The functions F, G are uniformly bounded.*
- *The functions F, G are Lipschitz continuous over x and m.*
- *The probability measure m_0 is absolutely continuous with respect to the Lebesgue measure and has a Hölder continuous density m_0^* that satisfies $\int_{\mathbb{R}^d}|x|^2 m_0^*(x)dx < \infty$. Then there exist at least one solution to (2.4). If in addition (2.2) holds, the solution is unique.*

There is also an important subcase of the canonical equations where $\sigma = 0$. We refer to this case as the **deterministic-canonical** MFG equations:

$$u_t - \frac{1}{2}\sum_{i=1}^{N}\left(\frac{\partial u}{\partial x_i}\right)^2 = -F(t,x,m), \tag{2.5a}$$

$$m_t - \sum_{i=1}^{N}\frac{\partial}{\partial x_i}\left(\frac{\partial u}{\partial x_i}m\right) = 0. \tag{2.5b}$$

We present this theorem from [7] without proof:

Theorem 3. *Assume that $m_0(x)$ is absolutely continuous, bounded and has compact support. Assume also that F, G are continuous over x, m and that F, G are bounded in the C^2-norm;*

$$\|f(x)\|_{C^2} = \sup_{x \in \mathbb{R}^d} \left(|f(x)| + |D_x f(x)| + |D_{xx} f(x)| \right). \tag{2.6}$$

Then (2.5) *has at least one solution. If in addition* (2.2) *holds, the solution is unique.*

2.1.3 Relationship to the form (1.1)

As pointed out earlier, the forms (2.1)(2.4)(2.5) have all eliminated any dependence on the control α by assuming it known as a function of other functions. As such, the equations for u, m are directly coupled. In our case, they are only implicitly coupled through the optimal control α.

The MFG equations (1.1) we negotiate in this thesis are in general not of the exact forms the theorems are made. As such, it must be mentioned that this makes some of our numerical results speculative. We will attempt to interpret how the conditions of theorems 1-3 relate to our form (1.1).

First we must give an interpretation of the variable p in $H(x, p)$. In the canonical MFGs (2.4)(2.5) we have that $p = -Du = \alpha$, hence p is simply the optimal control. In the form (2.1), we see that $p = Du$. We also have the fact that $\alpha = -\frac{\partial H}{\partial p}$ in (2.1b). This implies that p in this context is at least proportional to the optimal control α.

In theorem 1, $H(x, p)$ and $F(x, m)$ are effectively the terms that make up our Hamiltonian in (1.1a). The conditions on the cost $F(x, m)$ vary from the obscure requirement of "suitable growth conditions", to boundedness, continuity and monotonicity over m (2.2). We interpret this as such that any terms of the running cost $L(t, x, \alpha, m)$ involving m must satisfy analogues of these conditions. In addition, the conditions on $H(x, p)$ imply (at least) that L should be strictly convex and C^1 over α.

We will echo the speculation in this section later when we interpret the numerical results of chapter 6 in chapter 7.

2.2 Previous work on numerical methods for canonical MFG

Let us first explain the procedure for numerically solving the easier forms of the MFG equations presented in this chapter. Assume that we have some initial guess on the entire solution of the distribution $m(t, x)$. Then the following procedure is used, for iteration k:

1. Solve for u^k using m^{k-1}.

2. Solve for m^k using u^{k-1}.

3. Stop if $\|m^k - m^{k-1}\| < \epsilon$ for some tolerance ϵ.

We will in this section refer to procedure as the **solution procedure**. We will later in the text refer to two different forms of convergence: the convergence of this solution procedure, and convergence of the solutions of (u, α, m) as the mesh is refined.

The work surveyed show that the existing work is concerned with finite difference methods primarily, with some work on semi-Lagrangian methods. The most interesting work on finite difference methods is presented by Achdou et al in [3]. Here, a fully implicit scheme is proposed for solving (2.1a)(2.1b). The coupled form of these equations allow for the interpretation of their discretisation as a nonlinear system of equations. A Newton method is used to solve this system for all times $t \in [0, T]$ simultaneously. It is reported in [3] that the choice of initial guess of $m(t, x)$ influences the convergence properties of . We will return to this observation later.

Some novel reformulations of the canonical MFG equations (2.4) are presented in [11][12] by Guéant, using a change of variables with exponential functions. These give either two coupled heat equations or two coupled convection-diffusion equations. In earlier work by the author of this thesis, an explicit finite difference scheme suggested in [12] for one of these reformulations was implemented. While well-behaved in terms of convergence of the solution procedure when $\sigma > 0.1$, the scheme was neither monotone or conservative. In addition, the change of variable could potentially lead to numerical overflow when translating the solution back to (u, m), particularly for choices of $\sigma < 0.1$.

Lastly, semi-Lagrangian schemes are formulated in [6][9][8] by Carlini et al for the canonical and the deterministic-canonical MFG. The schemes are relatively straight-forward to implement, and conservative. However, the schemes involve an intermediary step that requires the computation of an optimal control problem. Recall that in the canonical MFG equations, the control is known. In light of this, the need to compute an optimal control problem regardless is daunting. In earlier work by the author of this thesis, the schemes were implemented and experimented with. We experienced that the schemes struggle or even fail to converge to solutions when the coupling between the MFG equations is strong; specifically when the cost function F in (2.4a) depends upon m in a non-negligible way. We will see later that this is not necessarily a fault of these schemes themselves.

See also [20] for a method that, instead of solving the both MFG equations, only seeks to solve the a variant of the Fokker-Planck equation (1.1b), with

$f(\alpha) = \alpha$ and $\mathbb{D} = \sigma^2 I$. This is achieved by iteratively approximating and refining the optimal control α with tricks derived from algebraic manipulation.

2.3 Objectives of our discretisations

The goals of our discretisation is to find stable, robust methods for (1.1). These discretisations should be suitable for, in principle, any choice of functions (1.2). To find the agents' strategies α via the Hamiltonian in (1.1a), we will use computational optimisation methods to find the numerical values. In this way, we will avoid potentially ill-behaved nonlinear terms at the expense of time-consuming computations. Positivity-preserving or monotone methods will be prioritised. This will preserve the physicality of the distribution m, and will prevent oscillations. For m in particular we will look for discretisations that are also conservative (mass-preservative). The latter point is especially important in order for any cost terms that depend upon m to influence the solution correctly.

Chapter 3

Discretisation of the Hamilton-Jacobi-Bellman equation

We will use computational optimisation to find the optimal control α, and so assume it known for the purpose of this discretisation. This resolves the Hamiltonian in (1.1a), and so we will discretise:

$$u_t + L + \sum_{i=1}^{N} \frac{\partial u}{\partial x_i} f_i + \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathbb{D}_{ij} \frac{\partial^2 u}{\partial x_i x_j} = 0. \tag{3.1}$$

3.1 Finite differences in one dimension

In one dimension, (3.1) is simplified as

$$u_t + L + f u_x + \frac{\sigma^2}{2} u_{xx} = 0.$$

We place $N + 1$ equidistant nodes on the domain $\Omega = [a, b]$ such that $x_0 = a, x_N = b$ and $\Delta x = (b - a)/N$.

3.1.1 Explicit scheme

We use upwind for $f u_x \approx \frac{1}{\Delta x} \left(f_i^+ (u_{i+1} - u_i) + f_i^- (u_i - u_{i-1}) \right)$ and central differences for the diffusion term $u_{xx} = \frac{1}{(\Delta x)^2} (u_{i+1} + u_{i-1} - 2u_i)$. This yields

$$
\begin{aligned}
u_i^{n-1} = u_i^n \left(1 - \frac{\Delta t}{\Delta x}\left(\frac{(\sigma_i^n)^2}{\Delta x} + |f_i^n|\right)\right) \\
+ u_{i+1}^n \frac{\Delta t}{\Delta x}\left(\frac{(\sigma_i^n)^2}{2\Delta x} + (f_i^n)^+\right) \\
+ u_{i-1}^n \frac{\Delta t}{\Delta x}\left(\frac{(\sigma_i^n)^2}{2\Delta x} - (f_i^n)^-\right) \\
+ \Delta t L(t^n, x_i, \alpha_i^n, m_i^n)
\end{aligned}
\quad . \tag{3.2}
$$

Theorem 4. *Assume that σ, f, L are bounded functions and that the inequality*

$$\Delta t < \frac{(\Delta x)^2}{\|\sigma\|_\infty^2 + \Delta x \|f\|_\infty}$$

holds. Then (3.2) *is a positive, stable and consistent scheme of order* $\mathcal{O}(\Delta t + \Delta x)$.

Proof. Positivity of the scheme is ensured if the coefficients for $u_i^{n+1}, u_{i+1}^{n+1}, u_{i-1}^{n+1}$ are positive. These are always positive save for the diagonal terms, which is positive if

$$\Delta t < \frac{(\Delta x)^2}{\sigma_i^2 + \Delta x |f_i|}, \tag{3.3}$$

which trivially gives the inequality above.

Consistency is a straight-forward computation which shows that the scheme has a local truncation error of

$$\tau_i^n = \frac{\Delta x}{2}|f_i^n| u_{xx} + \frac{(\Delta x)^2}{6} f_i^n u_{xxx}. \tag{3.4}$$

Stability follows from positivity and the convenient fact that the coefficients in (3.2) sum to 1. In the following, let $a_i, a_{i\pm1} \geq 0$ be the coefficients for $u_i, u_{i\pm1}$:

$$\|u^n\|_\infty \leq (a_i + a_{i+1} + a_{i-1}) \|u^{n+1}\|_\infty + \Delta t \|L\|_\infty = \|u^{n+1}\|_\infty + \Delta t \|L\|_\infty.$$

We can generalise this over the entire spacetime grid,

$$\|u^0\|_\infty \leq \|u_T\|_\infty + T\|L\|_\infty,$$

where u_T is the terminal condition and u^0 is the approximation of $u(\cdot, x)$. □

3.1.2 Semi-implicit scheme

We will use a semi-implicit formulation for this scheme, in which the diffusion terms are treated implicitly. This will give a nicer condition on Δt, of form $\Delta t = O(\Delta x)$. This semi-implicit formulation can be written as the linear system

$$Au^n = Bu^{n+1} \tag{3.5}$$

for a diffusion matrix A and a convection matrix B. By (3.2) we see that elements $(A)_{i,j}$ of the matrix A will be of the form

$$(A)_{i,i} = 1 + \frac{\Delta t}{(\Delta x)^2}\sigma_i^2 \quad \forall i,$$
$$(A)_{i,i\pm 1} = -\frac{\Delta t}{2(\Delta x)^2}\sigma_i^2 \quad \forall i.$$

As A is diagonally dominant, with a positive diagonal and negative off-diagonals, A is an M-matrix[a] and thus A^{-1} will be nonnegative. If the conditions of theorem 4 hold (with the condition of form $\Delta t = O(\Delta x)$ instead), B is a also nonnegative matrix. This ensures that the solution u^n of (3.5) will be nonnegative, assuming $u^{n+1} \geq 0$.

3.2 Finite differences in two dimensions

Let us first set

$$\mathbb{D} = \begin{bmatrix} D^{11} & D^{12} \\ D^{12} & D^{22} \end{bmatrix}.$$

Then we may state (3.1) in two dimensions as

$$u_t + L + f_1 u_x + f_2 u_y + \frac{1}{2}D_{11}u_{xx} + \frac{1}{2}D_{22}u_{yy} + D_{12}u_{xy} = 0. \tag{3.6}$$

We choose the following discretisations;

$$f_1 u_x \approx \frac{1}{\Delta x}\left((f_{i,j})_1^+(u_{i+1,j} - u_{i,j}) + (f_{i,j})_1^-(u_{i,j} - u_{i-1,j})\right)$$
$$f_2 u_y \approx \frac{1}{\Delta y}\left((f_{i,j})_2^+(u_{i,j+1} - u_{i,j}) + (f_{i,j})_2^-(u_{i,j} - u_{i,j-1})\right)$$
$$u_{xx} \approx \frac{1}{(\Delta x)^2}\left(u_{i+1,j} + u_{i-1,j} - 2u_{i,j}\right)$$
$$u_{yy} \approx \frac{1}{(\Delta y)^2}\left(u_{i,j+1} + u_{i,j-1} - 2u_{i,j}\right)$$

[a] We present the relevant theory on M-matrices in appendix B

$$
\begin{aligned}
D_{12}u_{xy} \approx \frac{D_{12}^{+}}{2\Delta x\Delta y}\Big(&u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} \\
&- 2u_{i,j} - u_{i+1,j-1} - u_{i-1,j+1}\Big) \\
+\frac{D_{12}^{-}}{2\Delta x\Delta y}\Big(&2u_{i,j} + u_{i+1,j+1} + u_{i-1,j-1} \\
&- u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1}\Big),
\end{aligned}
$$

where the discretisation of the "cross-diffusion" is due to Kushner (see for instance [4]).

For a Cartesian grid we have the fully explicit scheme:

$$
\begin{aligned}
u_{i,j}^{n} =& u_{i,j}^{n+1}\left(1 - \frac{\Delta t}{\Delta x}|(f_{i,j}^{n})_1| - \frac{\Delta t}{\Delta y}|(f_{i,j}^{n})_2| + \frac{\Delta t}{\Delta x\Delta y}|(D_{12})_{i,j}|\right) \\
&+u_{i,j}^{n+1}\left(-\frac{\Delta t}{(\Delta x)^2}(D_{11})_{i,j} - \frac{\Delta t}{(\Delta y)^2}(D_{22})_{i,j}\right) \\
&+u_{i+1,j}^{n+1}\frac{\Delta t}{2(\Delta x)^2}\left(2\Delta x(f_{i,j}^{n})_1^{+} + (D_{11})_{i,j} - \frac{\Delta x}{\Delta y}|(D_{12})_{i,j}|\right) \\
&+u_{i-1,j}^{n+1}\frac{\Delta t}{2(\Delta x)^2}\left(-2\Delta x(f_{i,j}^{n})_1^{-} + (D_{11})_{i,j} - \frac{\Delta x}{\Delta y}|(D_{12})_{i,j}|\right) \\
&+u_{i,j+1}^{n+1}\frac{\Delta t}{2(\Delta y)^2}\left(2\Delta y(f_{i,j}^{n})_2^{+} + (D_{22})_{i,j} - \frac{\Delta y}{\Delta x}|(D_{12})_{i,j}|\right) \\
&+u_{i,j-1}^{n+1}\frac{\Delta t}{2(\Delta y)^2}\left(-2\Delta y(f_{i,j}^{n})_2^{-} + (D_{22})_{i,j} - \frac{\Delta y}{\Delta x}|(D_{12})_{i,j}|\right) \\
&+u_{i+1,j+1}^{n+1}\frac{\Delta t(D_{12})_{i,j}^{+}}{2\Delta x\Delta y} + u_{i-1,j-1}^{n+1}\frac{\Delta t(D_{12})_{i,j}^{+}}{2\Delta x\Delta y} \\
&+u_{i+1,j-1}^{n+1}\frac{-\Delta t(D_{12})_{i,j}^{-}}{2\Delta x\Delta y} + u_{i-1,j+1}^{n+1}\frac{-\Delta t(D_{12})_{i,j}^{-}}{2\Delta x\Delta y} + \Delta t L_{i,j}^{n}.
\end{aligned}
\tag{3.7}
$$

This is equivalent to a matrix-vector formulation,

$$\bar{u}^{n} = A^{*}\bar{u}^{n+1}$$

for a banded and nonnegative matrix A^{*}.

Theorem 5. *Assume that $\Delta x = \Delta y$, $\mathbb{D} \geq 0$, $\mathbb{D}$ is diagonally dominant and that the inequality*

$$\Delta t \leq \frac{(\Delta x)^2}{2\left(\|D\|_\infty + \|f\|_\infty\right)} \tag{3.8}$$

holds. Then (3.7) *is a positive, stable and consistent scheme of order $\mathcal{O}(\Delta t + h)$, where $h = \max(\Delta x, \Delta y)$.*

Proof. Consistency follows from standard Taylor expansions and gives a truncation error of

$$\begin{aligned}\tau_{i,j}^n \leq \frac{\Delta t}{2} u_{tt} + \frac{h}{2}\left(|(f_{i,j}^n)_1| u_{xx} + |(f_{i,j}^n)_2| u_{yy}\right) \\ + \frac{h^2}{3}\Big(D^{12} u_{xyyy} + u_{xxxy} + \frac{(f_{i,j}^n)_1}{2} u_{xxx} \\ + \frac{(f_{i,j}^n)_2}{2} u_{yyy} + \frac{D_{i,j}^{11}}{8} u_{xxxx} + \frac{D_{i,j}^{22}}{8} u_{yyyy}\Big).\end{aligned} \tag{3.9}$$

Positivity is immediate for all coefficients except the one of $u_{i,j}^{n+1}$. Observe that

$$\begin{aligned}D^{11} + D^{22} - |D^{12}| \leq 2\max\left(D^{11}, D^{12}\right) \leq 2\|D\|_\infty, \\ |f_1| + |f_2| \leq 2\max\left(|f_1|, |f_2|\right) \leq 2\|\hat{f}\|_\infty.\end{aligned}$$

Applying these inequalities to the coefficient for $u_{i,j}^{n+1}$, we simply rearrange the terms to yield inequality (3.8).

Stability in the infinity-norm is a consequence of positivity, and the fact that the coefficients sum to 1. Let the coefficients in the scheme be given by $a_{i+j} \geq 0, j \in \mathbb{Z}$, and write

$$\|u^n\|_\infty \leq \sum_j a_{i+j} \|u^{n+1}\|_\infty + \Delta t \|L\|_\infty = \|u^{n+1}\|_\infty + \Delta t \|L\|_\infty.$$

Once more we may generalise over the spacetime grid and write

$$\|u^0\|_\infty \leq \|u_T\|_\infty + T\|L\|_\infty,$$

where u_T is the terminal condition and u^0 is the approximation of $u(0, x, y)$. □

3.2.1 Semi-implicit formulation

As in section 3.1.2, we choose to treat the diffusion terms implicitly to yield a system

$$Au^n = Bu^{n+1}.$$

Again, if the conditions of theorem 5 hold (with the nicer $\Delta t = O(\Delta x)$), B is a nonnegative matrix, A is an M-matrix, and $u^n \geq 0$ if $u^{n+1} \geq 0$.

Chapter 4

Discretisation of the Fokker-Planck equation

4.1 Discretisations of the Fokker-Planck equation in one dimension

In one dimension, (1.1b) takes the simpler form

$$m_t + (fm)_x = \frac{1}{2}(\sigma^2 m)_{xx}. \tag{4.1}$$

In this section we consider a few different discretisations of the Fokker-Planck equation in one dimension. Due to the nature of m as representing a distribution ($\int m dx = 1$), we shall be keen on schemes that maintain positivity and that preserve mass.

4.1.1 Finite differences

4.1.1.1 Naive approach

We use centred differences directly upon each spatial derivative in (4.1),

$$\begin{aligned}(fm)_x &= \frac{1}{2\Delta x}\left(f_{i+1}m_{i+1} - f_{i-1}m_{i-1}\right)\\ \left(\sigma^2 m\right)_{xx} &= \frac{1}{(\Delta x)^2}\left(\sigma^2_{i+1}m_{i+1} + \sigma^2_{i-1}m_{i-1} - 2\sigma^2_i m_i\right)\end{aligned}$$

This gives the explicit scheme

$$\begin{aligned}m_i^{n+1} = m_i^n &\left(1 - \Delta t\left(\frac{\sigma_i^n}{\Delta x}\right)^2\right)\\ &+ m_{i+1}^n \frac{\Delta t}{2\Delta x}\left(\frac{(\sigma_{i+1}^n)^2}{\Delta x} - f_{i+1}^n\right)\\ &+ m_{i-1}^n \frac{\Delta t}{2\Delta x}\left(\frac{(\sigma_{i-1}^n)^2}{\Delta x} + f_{i-1}^n\right).\end{aligned} \tag{4.2}$$

While simply formulated and of second order, (4.2) is not monotone without a condition on Δx if $\sigma > 0$, and not at all if $\sigma = 0$.

Theorem 6. *Assume that $\sigma > 0$ and that the inequalities*

$$\begin{aligned} \Delta t &< \left(\frac{\Delta x}{\|\sigma\|_\infty}\right)^2, \\ \Delta x &< \frac{\inf \sigma^2}{\|f\|_\infty}, \end{aligned} \tag{4.3}$$

hold. Then (4.2) is a (conditionally) monotone, consistent scheme of order $\mathcal{O}(\Delta t + (\Delta x)^2)$. It is also conservative.

Proof. Consistency is shown by a straight-forward, tedious computation. The local truncation error becomes (with $\Delta x = h$)

$$\begin{aligned} \tau_i^n =& h^2 m_i^n \left(\frac{f_{xxx}}{3} + \frac{\sigma_{xx}^2}{4} + \frac{\sigma_x \sigma_{xxx}}{3}\right) + h^2 m_x \left(\frac{f_{xx}}{2} + \frac{\sigma\sigma_{xxx}}{2} + \sigma_x \sigma_{xx}\right) \\ &+ h^2 m_{xx} \left(\frac{f_x}{2} + \frac{\sigma\sigma_{xx}}{2} + \frac{\sigma_x^2}{2}\right) + h^2 m_{xxx} \left(\frac{f_i^n}{6} + 2\sigma\sigma_x\right) + \mathcal{O}(h^3). \end{aligned} \tag{4.4}$$

Monotonicity: We check that coefficients are positive as we did for the HJB equation. The coefficient for m_i^n gives the first condition upon the timestep,

$$\Delta t < \left(\frac{\Delta x}{\sup \sigma}\right)^2. \tag{4.5}$$

The following inequalities upon Δx follow from the other two coefficients,

$$\begin{aligned} \Delta x &< \frac{\sigma^2}{f}, \\ \Delta x &< -\frac{\sigma^2}{f}, \end{aligned}$$

which comes out as

$$\Delta x < \frac{\inf \sigma^2}{\|f\|_\infty}. \tag{4.6}$$

Conservativeness is seen from

$$\begin{aligned} \sum_i m_i^{n+1} &= \sum_i \sum_j a_{i,j} m_j^n = \sum_i \sum_{j=-1}^{1} a_{i,j} m_j^n = \sum_i \left(a_{i-1,i} + a_{i,i} + a_{i+1,i}\right) m_i^n \\ &= \sum_i m_i^n = 1, \end{aligned}$$

where we've used that

$$a_{i-1,i} + a_{i,i} + a_{i+1,i} = 1 - \frac{k}{h^2}\sigma_i^2 + \frac{k}{2h^2}\sigma_i^2 - f_i + \frac{k}{2h^2}\sigma_i^2 + f_i = 1.$$

□

Note that this scheme is only conditionally monotone. We will see later that this scheme is not preferable despite being of second order.

4.1.1.2 A monotone scheme

This time we split the convection term in two:

$$(fm)_x = f_x m + f m_x \approx \frac{f_{i+1} - f_{i-1}}{2\Delta x} m_i + \frac{f_i^+}{\Delta x}(m_i - m_{i-1}) + \frac{f_i^-}{\Delta x}(m_{i+1} - m_i).$$

Note that we upwind differently than we did for the HJB equation[a]. This gives the scheme

$$\begin{aligned} m_i^{n+1} = m_i^n &\left(1 - \Delta t \left(\frac{\sigma_i^n}{\Delta x}\right)^2 - \frac{\Delta t}{2\Delta x}\left(f_{i+1} - f_{i-1} + 2|f_i|\right)\right) \\ &+ m_{i+1}^n \frac{\Delta t}{2(\Delta x)^2}\left((\sigma_{i+1}^n)^2 - 2\Delta x f_i^-\right) \\ &+ m_{i-1}^n \frac{\Delta t}{2(\Delta x)^2}\left((\sigma_{i-1}^n)^2 + 2\Delta x f_i^+\right). \end{aligned} \tag{4.7}$$

Theorem 7. *The scheme* (4.7) *is monotone and consistent of order* $\mathcal{O}(\Delta t + \Delta x)$*, assuming*

$$\Delta t < \frac{(\Delta x)^2}{\|\sigma\|^2 + \Delta x \|f\| + (\Delta x)^2 \|f_x\|} \tag{4.8}$$

holds.

Proof. Monotonicity: We check the coefficients as before. The coefficients for $m_{i\pm1}^n$ are unconditionally positive. The coefficient for m_i^n follows by reasserting that $f_x \approx \frac{1}{2\Delta x}(f_{i+1} - f_{i-1})$, and taking the suprema of all the functions therein.

[a] Both approaches are consistent with a different first order local truncation error.

Consistency is trivial and becomes (with $h = \Delta x$)

$$\begin{aligned}\tau_i^n = &- h|f_i^n| m_{xx} + h^2 m_i^n \left(\frac{f_{xxx}}{6} + \frac{\sigma_{xx}^2}{4} + \frac{\sigma_x \sigma_{xxx}}{3} \right) \\ &+ h^2 m_x \left(\frac{\sigma \sigma_{xxx}}{2} + \sigma_x \sigma_{xx} \right) + h^2 m_{xx} \left(\frac{\sigma \sigma_{xx}}{2} + \frac{\sigma_x^2}{2} \right) \\ &+ h^2 m_{xxx} \left(-\frac{|f_i^n|}{6} + 2\sigma\sigma_x \right) + \mathcal{O}(h^3).\end{aligned} \tag{4.9}$$

□

We no longer have an extra condition on Δx for this monotone scheme. However, we may expect the condition upon Δt to be stricter than for the previous scheme. Note also that this scheme is not conservative.

4.1.2 Finite volumes

Alternatively, we can use a finite volume method. As we will eventually pursue a finite volume method for the 2D Fokker-Planck, we will proceed more carefully. Let us begin by rewriting (4.1) into divergence form:

$$\begin{aligned} m_t + (fm)_x &= \frac{1}{2} \left(\sigma^2 m \right)_{xx} = (\sigma \sigma_x m)_x + \frac{1}{2} \left(\sigma^2 m_x \right)_x \\ m_t + ((f - \sigma\sigma_x)m)_x &= \frac{1}{2} \left(\sigma^2 m_x \right)_x . \end{aligned}$$

4.1.2.1 Definitions and discretisation

The idea of finite volume schemes is to compute the average value of a solution over each *control volume*:

Definition 1. ***Control volume (1D)****: Assume that we solve the problem on the domain $\Omega = [a, b]$. We place $N + 1$ equidistant nodes in Ω such that $\Delta x = (b - a)/N$, $x_0 = a$ and $x_N = b$. Associate a subdomain with each node such that $\Omega_i = [x_i - \Delta x/2, x_i + \Delta x/2]$, $\Omega_0 = [x_0, x_0 + \Delta x/2]$ and $\Omega_N = [x_N - \Delta x/2, x_N]$. These subdomains are referred to as control volumes or cells.*

Note that indices of the form $i \pm \frac{1}{2}$ will refer to the *cell faces* between cells. This will be used to handle the *fluxes* between the control volumes. Fluxes are associated with cell faces, and express the quantity that leaves or enters the cells by the cell face. An important attribute when defining fluxes is *flux consistency*:

Definition 2. ***Flux consistency:*** *Consider two neighbouring cells, Ω_k and Ω_{k+1}, and let $\phi^k_{k+\frac{1}{2}}, \phi^{k+1}_{k+\frac{1}{2}}$ be the fluxes associated with the cells over their common cell face. Flux consistency is the requirement that*

$$\phi^k_{k+\frac{1}{2}} = -\phi^{k+1}_{k+\frac{1}{2}}, \tag{4.10}$$

which simply means that the flux must be locally conservative and no new energy is created in the cell face.

In this section from here on, we set

$$m^n_i = \frac{1}{|\Omega_i|}\int_{\Omega_i} m(t^n, x)dx. \tag{4.11}$$

We set $F = f - \sigma\sigma_x$ and proceed to compute the cell averages of a given control volume. We use the divergence theorem on the second and third integrals:

$$\frac{1}{|\Omega_i|}\int_{\Omega_i} m_t(t^n, s)dx + \frac{1}{|\Omega_i|}\int_{\Omega_i} (Fm)_x(t^n, s)dx = \frac{1}{2|\Omega_i|}\int_{\Omega_i} (\sigma^2 m_x)_x(t^n, s)dx,$$

$$\frac{\partial}{\partial t}m^n_i + \frac{(Fm)^n_{i+\frac{1}{2}} - (Fm)^n_{i-\frac{1}{2}}}{\Delta x} = \frac{\left(\sigma^2 m_x\right)^n_{i+\frac{1}{2}} - \left(\sigma^2 m_x\right)^n_{i-\frac{1}{2}}}{2\Delta x}.$$

We need to determine two fluxes, the convective and the diffusive. We will show how to attain monotone approximations to these in the next sections.

4.1.2.2 Convection flux

In the associated literature, there are numerous ways to define this flux (see for instance [16]). Common for many of them is that the convective function F is assumed to be known over the entire domain. In our case, we only have the values of F_i at the nodes x_i, so that we will have to interpolate in order to get the value at the interface. We will avoid this by using a different flux suggested in [16].

$$(Fm)_{i+\frac{1}{2}} = F^+_i m_i + F^-_{i+1} m_{i+1}. \tag{4.12}$$

This flux is monotone, consistent of first order, and flux-consistent.

4.1.2.3 Diffusion flux

The terms

$$D^E_i = (\sigma^2 m_x)_{i+\frac{1}{2}},$$
$$D^W_i = (\sigma^2 m_x)_{i-\frac{1}{2}},$$

require more attention. There are a few ways to approach these terms. First we could simply use centered differences on the interface gradient and get,

$$D_i^E = \sigma^2_{i+\frac{1}{2}}(m_{i+1} - m_i),$$
$$D_i^W = \sigma^2_{i-\frac{1}{2}}(m_i - m_{i-1}).$$

We will in general not know the value of σ at the interface, however, if σ is a function of the control α. We could solve this by simply averaging as we did for the convective flux. However, it is remarked in [25][b] that this may be unfavourable in the case where σ is discontinuous. As such, we proceed differently and introduce the reader to ghost nodes for the first time below.

We now introduce the unknown ghost node $m_{i+\frac{1}{2}}$, which we will use to approximate the derivative in D_i^E. We will in turn remove it by enforcing flux consistency at the interface at $x = x_{i+\frac{1}{2}}$. We start by approximating the gradient by backwards/forward differences, and get

$$D_i^E = (\sigma^2 m_x)_{i+\frac{1}{2}} = 2\sigma_i^2(m_{i+\frac{1}{2}} - m_i)/\Delta x,$$
$$-D_{i+1}^W = (\sigma^2 m_x)_{i+\frac{1}{2}} = -2\sigma_{i+1}^2(m_{i+\frac{1}{2}} - m_{i+1})/\Delta x.$$

To get rid of it $m_{i+\frac{1}{2}}$, we enforce flux consistency (4.10) such that $D_i^E = -D_{i+1}^W$. Solving for $m_{i+\frac{1}{2}}$, we get

$$m_{i+\frac{1}{2}} = \frac{\sigma_i^2 m_i + \sigma_{i+1}^2 m_{i+1}}{\sigma_i^2 + \sigma_{i+1}^2}.$$

Substituting this back into D_i^E yields:

$$D_i^E = \frac{2\sigma_i^2\sigma_{i+1}^2}{\Delta x\left(\sigma_i^2 + \sigma_{i+1}^2\right)}(m_{i+1} - m_i).$$

We proceed similarly to find D_i^W to get

$$\begin{aligned} D_i^E - D_i^W &= \frac{2\sigma_i^2\sigma_{i+1}^2}{\Delta x\left(\sigma_i^2 + \sigma_{i+1}^2\right)}(m_{i+1} - m_i) - \frac{2\sigma_i^2\sigma_{i-1}^2}{\Delta x\left(\sigma_i^2 + \sigma_{i-1}^2\right)}(m_i - m_{i-1}) \\ &= \frac{1}{\Delta x}\left(\Sigma_i^E(m_{i+1} - m_i) - \Sigma_i^W(m_i - m_{i-1})\right) \end{aligned} \tag{4.13}$$

[b] Specifically, in section 2.3.1.

where we've set

$$\Sigma_i^E = \frac{2\sigma_i^2\sigma_{i+1}^2}{\sigma_i^2+\sigma_{i+1}^2},$$
$$\Sigma_i^W = \frac{2\sigma_i^2\sigma_{i-1}^2}{\sigma_i^2+\sigma_{i-1}^2}.$$

These terms of the form $\frac{2ab}{a+b}$ are called *harmonic means* of the numbers a, b.

4.1.2.4 Explicit scheme

The fully explicit scheme is

$$\begin{aligned} m_i^{n+1} = m_i^n &\left(1 - \frac{\Delta t}{2(\Delta x)^2}\left(\Sigma_i^E + \Sigma_i^W + 2\Delta x|F_i|\right)\right) \\ &+ m_{i+1}^n \frac{\Delta t}{2(\Delta x)^2}\left(\Sigma_i^E - 2\Delta x F_{i+1}^-\right) \\ &+ m_{i-1}^n \frac{\Delta t}{2(\Delta x)^2}\left(\Sigma_i^W + 2\Delta x F_{i-1}^+\right). \end{aligned} \tag{4.14}$$

Theorem 8. *Assume that*

$$\Delta t < \frac{(\Delta x)^2}{\|\sigma\| + \Delta x\|F\|} \tag{4.15}$$

holds. Then the scheme (4.14) *is monotone, consistent of order* $\mathcal{O}(\Delta t + \Delta h)$ *and stable in the 1-norm. It is also conservative.*

Proof. Consistency is straight-forward for the convective terms, which give the first order truncation error (with $h = \Delta x$)

$$\tau_i^n = -h\left(\frac{F_{xx}}{2}m + |F_x|m_x\right) + \frac{h^2}{2}\left(\frac{1}{3}F_{xx}m + F_{xx}m_x + F_x m_{xx}\right). \tag{4.16}$$

The diffusion flux terms are also of first order, see [25][c].

Positivity is immediate as long as the coefficient for m_i^n is also positive. Note that $\Sigma_E + \Sigma_W \leq \sigma_i^2$. Then we have the condition (for $F = f - \sigma\sigma_x$)

$$\Delta t < \frac{(\Delta x)^2}{\|\sigma\| + \Delta x\|F\|}.$$

[c] Specifically, theorem 2.3.

Stability is a consequence of positivity and the fact that the coefficients in (4.14) sum to 1:

$$\begin{aligned}\|m^{n+1}\|_1 &\leq \Delta x|\sum_i \sum_{j=-1}^{1} a_{i,j} m^n_{i+j}| \\ &\leq \sum_i \left(\sum_j |a_{i,j}|\right)|m^n_i| = \sum_i |m^n_i| = \|m^n\|_1.\end{aligned}$$

Conservativeness is seen from

$$\begin{aligned}\sum_i m^{n+1}_i &= \sum_i \sum_j a_{i,j} m^n_j = \sum_i \sum_{j=-1}^{1} a_{i,j} m^n_j = \sum_i \left(a_{i-1,i} + a_{i,i} + a_{i+1,i}\right) m^n_i \\ &= \sum_i m^n_i = 1,\end{aligned}$$

where we've used that (recall flux consistency)

$$\begin{aligned}&a_{i-1,i} + a_{i,i} + a_{i+1,i} \\ &= 1 - \frac{k}{h^2}\left(\Sigma^i_E + \Sigma^i_W\right) - \frac{k}{h}|F_i| + \frac{k}{h}\left(F^+_i - F^-_i\right) + \frac{k}{h^2}\left(\Sigma^{i+1}_W + \Sigma^{i-1}_E\right) = 1.\end{aligned}$$

□

4.1.2.5 Semi-implicit scheme

We use a semi-implicit formulation as we did for the HJB equation; see sections 3.1.2 and 3.2.1. We apply similar arguments as in these sections to argue that the diffusion matrix A in

$$Am^{n+1} = Bm^n$$

is an M-matrix when the conditions of theorem 8 hold. In this case B is also a nonnegative matrix, and the solution m^{n+1} will be nonnegative assuming $m^n \geq 0$.

4.2 Finite volume discretisation of the Fokker-Planck equation in 2D

Once more we rewrite the model equation

$$m_t + (f_1 m)_x + (f_2 m)_y = \frac{1}{2}(\mathbb{D}_{11} m)_{xx} + \frac{1}{2}(\mathbb{D}_{22} m)_{yy} + (\mathbb{D}_{12} m)_{xy} \tag{4.17}$$

to divergence form. We can rearrange the terms into the following, where $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)^T$:

$$m_t + \frac{1}{2}\nabla \cdot \left(\left(2f - \nabla^T \mathbb{D}\right) m\right) = \frac{1}{2}\nabla \cdot (\mathbb{D}\nabla m). \tag{4.18}$$

The form of (4.18) is well-suited for a finite volume discretisation. Recall the definitions in section 4.1.2.1; we only alter the definition of the control volumes.

Definition 3. ***Control volume in 2D****: Assume that we solve* (4.18) *on the domain* $\Omega = [a, b] \times [c, d]$*. We place* $(N + 1)^2$ *equidistant nodes in* Ω *such that* $\Delta x = (b - a)/N$*,* $x_0 = a$*,* $x_N = b$ *and* $\Delta y = (d - c)/N$*,* $y_0 = a$*,* $y_N = b$*. We associate a subdomain with each node such that* $\Omega_{i,j} = [x_i - \Delta x/2, x_i + \Delta x/2] \times [y_j - \Delta y/2, y_j + \Delta y/2]$*. Since* $\Omega_{i,j} \subset \Omega$*, no such subdomains go outside* Ω*; the subdomains along the boundary are hence smaller. These subdomains are referred to as control volumes or cells.*

Definition 4. ***Neighbourhood and cell interfaces:*** *For each cell* Ω_i*, we associate the set of nodes that* Ω_i *shares interfaces with as the neighbourhood* N_i *of* Ω_i*. In addition, for each* $j \in N_i$ *(where we use* j *as short-hand for* Ω_j*, we will use* σ_{ij} *to refer to the interface between cells* Ω_i *and* $\Omega_j \in N_i$*. The vector* $\bar{n}_{i,j}$ *is the normal vector out of the cell of* Ω_i *on the interface* σ_{ij}*.*

Note that the points x_i, y_j must be the same nodes on which we compute the optimal control (see section 5). By calculating the mean values of the cells $\Omega_{i,j}$ and applying the divergence theorem, we get for a single cell

$$\begin{aligned} \frac{\partial}{\partial t} m_{i,j} + &\frac{1}{2m(\Omega_{i,j})} \int_{\partial\Omega_{i,j}} \left(\left(2f - \nabla^T \mathbb{D}\right) m\right) \cdot \bar{n} dS(x) \\ = &\frac{1}{2m(\Omega_{i,j})} \int_{\partial\Omega_{i,j}} (\mathbb{D}\nabla m) \cdot \bar{n} dS(x), \end{aligned} \tag{4.19}$$

where $m(\Omega_{i,j})$ is the area of the cell. The first term is a regular convection term, while the second is a diffusion term. We shall address these similarly as for the 1D case.

4.2.1 Convection flux

Since we are on a rectangular, cell-centered grid we can rewrite the convection term of (4.19) into integration over strictly x or y. Let

$$F = f - \frac{1}{2}\nabla^T \mathbb{D}.$$

Then we get that,

$$\begin{aligned}
&\int_{\partial\Omega_{i,j}} ((F)\, m) \cdot \bar{n} dS(x) \\
&\quad = \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} F(x, y_{j+\frac{1}{2}}) m(x, y_{j+\frac{1}{2}}) dx + \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} F(x_{i+\frac{1}{2}}, y) m(x_{i+\frac{1}{2}}, y) dy \\
&\quad\quad - \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} F(x, y_{j-\frac{1}{2}}) m(x, y_{j-\frac{1}{2}}) dx - \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} F(x_{i-\frac{1}{2}}, y) m(x_{i-\frac{1}{2}}, y) dy \\
&\quad \approx \Delta x \left((F_2^+)_{i,j} m_{i,j} + (F_2^-)_{i,j+1} m_{i,j+1} \right) + \Delta y \left((F_1^+)_{i,j} m_{i,j} + (F_1^-) i+1, j m_{i+1,j} \right) \\
&\quad\quad - \Delta x \left((F_2^-)_{i,j} m_{i,j} + (F_2^+)_{i,j-1} m_{i,j-1} \right) - \Delta y \left((F_1^-)_{i,j} m_{i,j} + (F_2^+) i-1, j m_{i-1,j} \right).
\end{aligned}$$

Here we have used the midpoint rule[d] to approximate the integrals. We have then used upwinding in the same manner as we did for the one-dimensional case in section 4.1.2.2, which means this flux approximation is only of first order. Using a semi-implicit formulation for this method will give the coefficient for $m_{i,j}^n$ in the explicit convective scheme:

$$m_{i,j}^n \left(1 - \frac{\Delta t}{\Delta x} |(F_1)_{i,j}| - \frac{\Delta t}{\Delta y} |(F_2)_{i,j}| \right),$$

which immediately gives us the condition on Δt,

$$\begin{aligned}
\Delta &\leq \frac{\Delta x \Delta y}{\Delta x \|F_1\|_\infty + \Delta y \|F_2\|_\infty} \\
&\leq \frac{h}{2\|F\|_\infty} \quad \text{when} \quad \Delta x = \Delta y
\end{aligned} \tag{4.20}$$

This flux is conservative by construction, and monotone.

[d] We use the midpoint rule with one subinterval so that $\int_{x-\Delta x/2}^{x+\Delta x/2} f(x)dx \approx \Delta x f(x) + O((\Delta x)^3)$.

4.2.2 Diffusion fluxes for the diffusion tensor $\mathbb{D}$

We restate the diffusion flux,

$$\int_{\partial\Omega_{i,j}} (\mathbb{D}\nabla m) \cdot \bar{n} dS(x). \tag{4.21}$$

The term (4.21) is generally nontrivial to discretise due to the fact that $\mathbb{D}$ is a tensor. While we have found no "canonically correct" method to do this, we will examine three methods. They are a naive method, the so-called O-method and a nonlinear method.

4.2.2.1 The naive method

This method is derived by simply evaluating (4.21) directly on our Cartesian grid:

$$\begin{aligned}
&\int_{\partial\Omega_{i,j}} (\mathbb{D}\nabla m) \cdot \bar{n} dS(x) \\
&\quad = \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \left(D^{11}(x_{i+\frac{1}{2}}, y) m_x(x_{i+\frac{1}{2}}, y) + D^{12}(x_{i+\frac{1}{2}}, y) m_y(x_{i+\frac{1}{2}}, y) \right) dy \\
&\quad + \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \left(D^{12}(x, y_{j+\frac{1}{2}}) m_x(x, y_{j+\frac{1}{2}}) + D^{22}(x, y_{j+\frac{1}{2}}) m_y(x, y_{j+\frac{1}{2}}) \right) dx \\
&\quad - \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \left(D^{11}(x_{i-\frac{1}{2}}, y) m_x(x_{i-\frac{1}{2}}, y) + D^{12}(x_{i-\frac{1}{2}}, y) m_y(x_{i-\frac{1}{2}}, y) \right) dy \\
&\quad - \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \left(D^{12}(x, y_{j-\frac{1}{2}}) m_x(x, y_{j-\frac{1}{2}}) + D^{22}(x, y_{j-\frac{1}{2}}) m_y(x, y_{j-\frac{1}{2}}) \right) dx
\end{aligned}$$

We apply the midpoint rule to resolve the integrals. The flux-terms involving D^{11} and D^{22} all work *through* their respective cell interfaces. We approach this the same way we did as for the 1D-case in section 4.1.2.3, which gives us harmonic means of the form

$$D^{11}_{i+\frac{1}{2},j} = \frac{2D^{11}_{i,j} D^{11}_{i+1,j}}{D^{11}_{i,j} + D^{11}_{i+1,j}}.$$

The cross-terms involving D^{12} require some special attention, as they are all gradients that go *along* the cell interfaces. To resolve these integrals, we set them as the average of the two gradients that go through the two cells that share this interface. This is more easily explained visually in figure 4.1. This

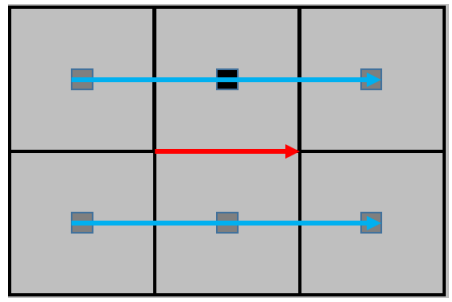

FIGURE 4.1: Gradient approximation for the naive diffusion flux method. The black square shows the current node in the current cell center. The red arrow indicates the gradient to be evaluated, while the blue arrows indicate the two gradients we approximate this by, using the nodal values of the grey squares the arrows begin and end in.

give us terms of the form:

$$\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} (D^{12} m_y)_{i+\frac{1}{2}} dx \approx \frac{D^{12}_{i,j}}{4} (m_{i,j+1} - m_{i,j-1}) + \frac{D^{12}_{i+1,j}}{4} (m_{i+1,j+1} - m_{i+1,j-1})$$

That is, each interface gives a sum of two expressions, where one term $D^{12}_{i,j}$ is for the current cell and the other term $D^{12}_{k,l}$ is for the cell on the other side of the interface. By summing across the four interfaces, all the D^{12}-terms for the current cell cancel each other out. All in all, we are left with the naive discretisation of (4.21) as:

$$\begin{aligned}
\int_{\partial\Omega_{i,j}} & (\mathbb{D}\nabla m) \cdot \bar{n} dS(x) \\
& \approx D^{11}_{i+\frac{1}{2},j} (m_{i+1,j} - m_{i,j}) + D^{22}_{i,j+\frac{1}{2}} (m_{i,j+1} - m_{i,j}) \\
& \quad + D^{11}_{i-\frac{1}{2},j} (m_{i-1,j} - m_{i,j}) + D^{22}_{i,j-\frac{1}{2}} (m_{i,j-1} - m_{i,j}) \\
& \quad + \frac{1}{4} m_{i+1,j+1} \left(D^{12}_{i+1,j} + D^{12}_{i,j+1}\right) + \frac{1}{4} m_{i-1,j-1} \left(D^{12}_{i,j-1} + D^{12}_{i-1,j}\right) \\
& \quad - \frac{1}{4} m_{i+1,j-1} \left(D^{12}_{i+1,j} + D^{12}_{i,j-1}\right) - \frac{1}{4} m_{i-1,j+1} \left(D^{12}_{i,j+1} + D^{12}_{i-1,j}\right).
\end{aligned}$$

It is straight-forward to show that this is flux-consistent, and thus also conservative. This scheme is also straight-forward to implement[e]. However, we can immediately see that this scheme is not monotone when $D^{12} \neq 0$. As such, we risk that this method will not preserve positivity of the solution of the distribution m. This is what motivates the investigation into the other methods.

4.2.2.2 The O-method

The O-method is a conditionally monotone nine-point flux approximation scheme introduced by Aavatsmark (see for instance [2]) for use in tensor diffusion problems in oil reservoir computations. The O-method is used on quadrilateral grids, which our Cartesian grid is. The idea of the O-method is to set up an *interaction volume* between each set of nodes that share cell corners; see figure 4.2. All diffusive fluxes are resolved within each interaction volume in the grid. We see in figure 4.2 that each cell interface is effectively split in two parts that operate on different interaction volumes. We will refer to these halves as half-edges. As with the 1D diffusion flux (section 4.1.2.3),

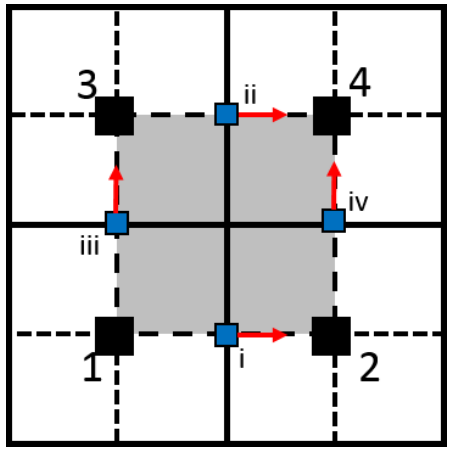

FIGURE 4.2: Visual representation of the O-method. The interaction volume is coloured grey, and covers one quadrant of each of the cell volumes that share corners. The dashed lines separate the other interaction volumes each cell is involved in. The blue squares indicate the placement of the ghost nodes, with the Roman numerals showing their numbering. The red arrows show the direction of the half-edge fluxes.

we use intermediary/ghost nodes on the interfaces between cells. When

[e]Do not forget to this expression by $\frac{1}{2m(\Delta_{i,j}}$ if implementing this method.

enforcing flux consistency, expressions for the values at these ghost nodes appear and allow us to eliminate them from the flux expressions. By varying the placement of the ghost nodes on the half-edges, a family of methods can be explored (see [2][1]). For the O-method, the ghost nodes are placed exactly between the cell volume nodes, as in figure 4.2.

In the following derivation, we set $\hat{m} = (m_i, m_{ii}, m_{iii}, m_{iv})$ to be the half-edge ghost node values, and set $\bar{m} = (m_1, m_2, m_3, m_4)$ to be the cell-center nodal values. We also set $f_k^{(n)}$ to be the half-edge flux of half-edge k from cell n.

We approximate the half-edge fluxes by using a two-point flux approximation. These take the form

$$\begin{aligned}(f_i^{(1)}, f_{iii}^{(1)})^T &= G_{(1)}(m_i - m_1, m_{iii} - m_1)^T, \\ (f_{ii}^{(4)}, f_{iv}^{(4)})^T &= G_{(4)}(m_4 - m_{ii}, m_4 - m_{iv})^T,\end{aligned} \tag{4.22}$$

where $G_{(k)}$ is the local average of the diffusion tensor for node k,

$$G_{(k)} = \frac{1}{m(\Omega_k)} \mathbb{D}_k.$$

To eliminate the ghost node value of m_i in (4.22), we first set up the two-point flux approximation of the same half-edge flux from the opposite side, from node 2:

$$(f_i^{(2)}, f_{iv}^{(2)})^T = G_{(2)}(m_2 - m_i, m_{iv} - m_2)^T.$$

Then we simply set these expressions equal to one another, $f_i^{(1)} = -f_i^{(2)}$, and solve for m_i. Instead of doing this individually, we set up a linear system for all half-edges in the interaction volume,

$$A\hat{m} = B\bar{m}. \tag{4.23}$$

Observe now that we may set up a linear system equivalent to (4.22) as well,

$$\bar{f} = C\hat{m} + F\bar{m}. \tag{4.24}$$

Note that A, B, C, F are all four-by-four matrices that depend on the values of $G_{(k)}$ for the four nodes that make up the vertices of the interaction volume. There are several ways in which to define these matrices, but all ways lead to the same solution.

We solve (4.23) for $\hat{m}$ to get $\hat{m} = A^{-1}B\bar{m}$. We insert this into (4.24) to yield

$$\bar{f} = T\bar{m} = \left(CA^{-1}B + F\right)\bar{m}, \tag{4.25}$$

where T is referred to in [2] as the transmissibility matrix. To find the total influence on one node, one thus has to compute $\bar{f}$ for all four interaction volumes that has the node as one of its vertices, as proposed in [2]. Thusly, four interaction volumes will be treated to get the full information on all fluxes in and out of a single cell. We have used a slightly different approach.

Recall that for finite element methods, each element contributes a local stiffness matrix to the global stiffness matrix. We can uncover such a local matrix for any given interaction volume. Let F_1, F_2, F_3, F_4 denote the fluxes going out of each of the cells in the interaction volume. In figure 4.2 we see that

$$\begin{aligned} F_1 &= f_1 + f_3, \\ F_2 &= f_4 - f_1, \\ F_3 &= f_2 - f_3, \\ F_4 &= -f_2 - f_4. \end{aligned}$$

We can represent in matrix form for interaction volume i as

$$\bar{F}_i = R\bar{f} = RT_i\bar{m}_i, \tag{4.26}$$

where

$$R = \begin{bmatrix} 1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 0 & -1 \end{bmatrix},$$

and $\bar{m}_i$ refer to the nodal values influenced by the interaction volume. By considering each interaction volume i in turn, we may use an assembly process to insert the local diffusion fluxes $\bar{F}_i$ into a global diffusion matrix.

One disadvantage of the O-method is that it is not possible in general to derive a nice expression for A^{-1}, meaning that the transmissibility matrix T has to be computed for every interaction volume in the mesh. This is no longer the case if $\mathbb{D}$ is diagonal ($D^{12} = 0$). In this case, the O-method becomes equivalent to the naive method.

A rigorous investigation into the monotonicity properties of the O-method is difficult for the reasons noted above. Conditions for monotonicity are reported[2][1] only for the case of a constant $\mathbb{D}$. These take the form of the inequalities

$$|D^{12}|\left(2 - \frac{(D^{12})^2}{D^{11}D^{22}}\right) < \frac{2D^{11}D^{22}}{D^{11}+D^{22}},$$

$$2D^{11}D^{22}\left(1 - \frac{1}{2|D^{12}|}\frac{2D^{11}D^{22}}{D^{11}+D^{22}}\right) < \frac{(D^{12})^2}{D^{11}D^{22}} < \frac{2D^{11}D^{22}}{D^{11}+D^{22}}\min\left(D^{11}, D^{22}\right).$$

It is reported in [1] that all linear nine-point methods like the O-method are at best only conditionally monotone. The monotonicity regions of several such methods are compared in [1], and the O-method strikes a happy balance between ease of implementation and its monotonicity conditions. However, in order to have unconditional monotonicity we have to leave linearity behind.

4.2.2.3 A monotone, nonlinear diffusion flux

An unconditionally monotone, but nonlinear diffusion flux of up to second order accuracy is introduced in [24][f]. We will present a concise derivation of the scheme here, and direct the interested reader to the full derivation in [24]. We must also point out that this scheme behaves slightly different if the off-diagonal terms in $\mathbb{D}$ are greater than the diagonal terms. We present the scheme assuming that $D^{12} \leq \min(D^{11}, D^{22})$ such that positivity for the 2D HJB scheme (3.7) is satisfied. The method as presented will thus apply for all interfaces.

The derivation starts by rewriting (4.21) slightly:

$$\int_\sigma (\mathbb{D}\nabla m) \cdot \bar{n} dS(x) = F_{K,\sigma} = \int_\sigma \nabla m \cdot (\mathbb{D}^T \bar{n}_{K,\sigma}) dS(x). \tag{4.27}$$

Here the notation $F_{K,\sigma}, \bar{n}_{K,\sigma}$ refers to the flux going out of cell K through the interface σ, and the respective normal vector. Now consider figure 4.3, in particular the lines KP_1, KP_2[g]. We split the last term of (4.27) into two vectors, $\bar{t}_{KP_1}, \bar{t}_{KP_2}$, that correspond to the lines KP_1, KP_2. Then we identify the elementary relation

$$\frac{\mathbb{D}^T \bar{n}_{K,\sigma}}{|\mathbb{D}^T \bar{n}_{K,\sigma}|} = \frac{\sin\theta_{K_2}}{\sin\theta_K} \bar{t}_{KP_1} + \frac{\sin\theta_{K_1}}{\sin\theta_K} \bar{t}_{KP_2}. \tag{4.28}$$

By combining (4.27) and (4.28) we get that

$$\begin{aligned} F_{K,\sigma} &= \int_\sigma (\mathbb{D}\nabla m) \cdot \bar{n} dS(x) \\ &= \int_\sigma |\mathbb{D}^T \bar{n}_{K,\sigma}| \left(\frac{\sin\theta_{K_2}}{\sin\theta_K} (\nabla m) \bar{t}_{KP_1} + \frac{\sin\theta_{K_1}}{\sin\theta_K} (\nabla m) \bar{t}_{KP_2} \right) dS(x) \\ &\approx |\mathbb{D}^T \bar{n}_{K,\sigma}||\sigma| \left(\frac{\sin\theta_{K_2}}{\sin\theta_K} \frac{m_{P_1} - m_K}{|KP_1|} + \frac{\sin\theta_{K_1}}{\sin\theta_K} \frac{m_{P_2} - m_K}{|KP_2|} \right). \end{aligned} \tag{4.29}$$

[f] See the sequel [23] for the same ideas applied to slightly more general grids. This is not necessary for the scope of this thesis.

[g] The points P_1, P_2 are determined by which interface the vector $\mathbb{D}n_{\bar{K},\sigma}$ intersects. Since we assume a diagonally dominant $\mathbb{D}$, the vector will always have a larger component in the x-direction for east-/west-facing interfaces, and a larger component in the y-direction for north-/south-facing interfaces.

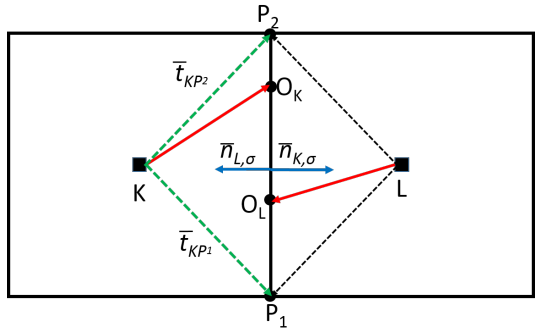

FIGURE 4.3: Illustration of the idea behind the nonlinear diffusion flux in section 4.2.2.3. The red arrows indicate the vectors $\mathbb{D}n_{\bar{K},\sigma}, \mathbb{D}n_{\bar{L},\sigma}$. The points P_1, P_2 indicate the vertices of the interfaces intersected by $\mathbb{D}n_{\bar{K},\sigma}, \mathbb{D}n_{\bar{L},\sigma}$.

In the last equality, we've used the first order gradient and integral approximations.

In order to proceed with the derivation, a similar expression is derived for $F_{L,\sigma}$ in the same manner. Next, flux consistency is enforced by setting $F_{K,\sigma} = -F_{L,\sigma}$. The resulting calculations are long-winded and technical, and fully detailed in [24]. We summarise the prevailing expression for $F_{K,\sigma}$ in (4.30):

$$
\begin{aligned}
F_{K,\sigma} &= A_{K,\sigma} m_K - A_{L,\sigma} m_L, \\
A_{K,\sigma} &= \mu_1 \frac{|\sigma| |\mathbb{D}_K^T \bar{n}_{K,\sigma}|}{|KO_K|}, \\
A_{L,\sigma} &= \mu_2 \frac{|\sigma| |\mathbb{D}_L^T \bar{n}_{L,\sigma}|}{|LO_L|}, \\
(\mu_1, \mu_2) &= \begin{cases} \left(\frac{1}{2}, \frac{1}{2}\right) & \text{if } a_1 + a_2 = 0 \\ \left(\frac{a_2}{a_1+a_2}, \frac{a_1}{a_1+a_2}\right) & \text{otherwise} \end{cases}, \\
a_1 &= \frac{|\sigma| |\mathbb{D}_K^T \bar{n}_{K,\sigma}|}{\sin\theta_K} \left(\frac{\sin\theta_{K_2}}{|KP_1|} m_{P_1} + \frac{\sin\theta_{K_1}}{|KP_2|} m_{P_2} \right), \\
a_2 &= \frac{|\sigma| |\mathbb{D}_L^T \bar{n}_{L,\sigma}|}{\sin\theta_L} \left(\frac{\sin\theta_{L_2}}{|LP_4|} m_{P_3} + \frac{\sin\theta_{L_1}}{|LP_3|} m_{P_4} \right),
\end{aligned}
\tag{4.30}
$$

where $|\sigma|$ is the length of the interface, m_{P_i} are node values at the interface vertices, and $\theta_{K_1} + \theta_{K_2} = \theta_K$ are the angles between KO_K and P_1, P_2. The use of the values m_{P_i} makes the scheme nonlinear. With our Cartesian grid

with $\Delta x = \Delta y = h$, we simply use the average of the four closest nodes to interpolate these values[h]. Let us note that all quantities in (4.30) are nonnegative, assuming $m \geq 0$.

The expressions for a_1, a_2 in (4.30) are simplified for the Cartesian grid we employ:

$$\begin{aligned} a_1 &= \frac{|\mathbb{D}_K^T \bar{n}_{K,\sigma}|}{|KO_K|} \left(|O_K P_2| m_{P_1} + |O_K P_1| m_{P_2} \right), \\ a_2 &= \frac{|\mathbb{D}_L^T \bar{n}_{L,\sigma}|}{|LO_L|} \left(|O_L P_2| m_{P_1} + |O_L P_1| m_{P_2} \right). \end{aligned} \tag{4.31}$$

Here, the lengths of the lines in (4.31) are for the case in figure 4.3 may be found as (for $\Delta x = \Delta y = h$):

$$\begin{aligned} |KO_K| &= \frac{h}{2} \sqrt{1 + \left(\frac{D_K^{12}}{D_K^{11}} \right)}, \\ |LO_L| &= \frac{h}{2} \sqrt{1 + \left(\frac{D_L^{12}}{D_L^{11}} \right)^2}, \\ |O_K P_1| &= \frac{h}{2} |1 + \frac{D_K^{12}}{D_K^{11}}|, \\ |O_K P_2| &= \frac{h}{2} |1 - \frac{D_K^{12}}{D_K^{11}}|, \\ |O_L P_1| &= \frac{h}{2} |1 - \frac{D_L^{12}}{D_L^{11}}|, \\ |O_L P_2| &= \frac{h}{2} |1 + \frac{D_L^{12}}{D_L^{11}}|. \end{aligned} \tag{4.32}$$

In general, an assembly process is necessary to generate a global diffusion matrix for this method, by iterating over each interface once. Again, the Cartesian grid simplifies this significantly. In particular, this is done by making one-dimensional arrays out of all the auxiliary quantities in (4.31)(4.32), where each element is associated with a node. This, combined with the techniques in section E.3, allows a quick generation of the matrix.

Assuming we evaluate the tensor diffusion implicitly, we are left with a nonlinear system of the form

$$A(m^{n+1}) m^{n+1} = B m^n. \tag{4.33}$$

[h] It is reported in [24] that the choice of interpolant for these vertex unknowns is critical for general grids. However, for our grid our choice is as good as any.

The matrix A is an M-matrix[24], and will preserve the positivity of m. We solve (4.33) using Picard iterations, but a Newton method is also an option.

4.2.3 Summary of the 2D finite volume scheme

We have presented a monotone convection flux, and three different ways to discretise the diffusion flux. Each of the diffusion fluxes are suitable depending on the properties of the diffusion tensor $\mathbb{D}$. If $\mathbb{D}$ is diagonal, the naive method and the O-method are equivalent and well suitable. In the case of a non-diagonal but diagonally dominant $\mathbb{D}$, the O-method may be suitable depending on the magnitude of the off-diagonal terms. The nonlinear method should be suitable for all choices of $\mathbb{D}$, but comes at the cost of having to solve the nonlinear system (4.33) for each time step.

We close this section by noting that any of the diffusion flux methods may be used for the semi-implicit formulation,

$$Am^{n+1} = Bm^n.$$

The conditions for the diffusion matrix A to be an M-matrix are the same as for the diffusion flux to be monotone.

Chapter 5

Computing the optimal control

In this section we will describe how we have chosen to discretise the Hamiltonian, and present a selection of strategies in computing the optimal control based upon this discretisation. Unless the input functions $f, L, \sigma/\mathbb{D}$ are chosen so that the optimal control α can be found analytically, a necessary step to solving the MFG equations is to numerically compute the control for each node for each time in the grid. This is time-consuming, and it is essential to find fast ways to do this. While there are numerous built-in libraries in Matlab and Python to handle optimisation problems like this one directly, the methods presented here have all been found to be faster.

We will assume that the existence of a unique optimal control is a consequence of the functions f, L and $\mathbb{D}$ or σ; in a sense that the model is well-posed, or at least that the admissible set $\mathcal{A}$ is bounded.

All methods presented have been implemented (see notes in section E) and tested (see section 6.2).

5.1 Discretisations

The problem is special in the sense that the objective is to optimise a function that in itself is an approximation of another function. Our discretisations of the Hamiltonian echo those in section 3.

5.1.0.1 In one dimension

In one dimension, the Hamiltonian is formally stated as

$$\operatorname{argmin}_{\alpha\in\mathcal{A}} \left(f(t,x,\alpha)u_x + L(t,x,\alpha,m) + \frac{1}{2}\sigma(t,x,\alpha)^2 u_{xx} \right). \tag{5.1}$$

We use the following notation for this section:

$$\begin{aligned} a^+ + a^- &= \max(a,0) + \min(a,0) = a, \\ D^+u &= u_{i+1} - u_i, \\ D^-u &= u_i - u_{i-1}, \\ g_i^n(\alpha) &= g(t^n, x_i, \alpha) \text{ for any function.} \end{aligned}$$

The discrete one-dimensional Hamiltonian becomes:

$$\begin{aligned} J_i^n(\alpha) &= \left(\frac{f_i^n(\alpha)}{\Delta x}\right)^+ D^-u + \left(\frac{f_i^n(\alpha)}{\Delta x}\right)^- D^+u \\ &\quad + \frac{1}{2}\left(\frac{\sigma_i^n(\alpha)}{\Delta x}\right)^2 \left(D^+u - D^-u\right) \\ &= D^+u\left(\frac{1}{2}\left(\frac{\sigma_i^n(\alpha)}{\Delta x}\right)^2 + \left(\frac{f_i^n}{\Delta x}\right)^-\right) \\ &\quad + D^-u\left(-\frac{1}{2}\left(\frac{\sigma_i^n(\alpha)}{\Delta x}\right)^2 + \left(\frac{f_i^n}{\Delta x}\right)^+\right). \end{aligned} \tag{5.2}$$

And thus

$$\alpha_i^n = \text{argmin}_{\alpha\in\mathcal{A}} J_i^n(\alpha)$$

5.1.1 In two dimensions

We set

$$\begin{aligned} a^+ + a^- &= \max(a,0) + \min(a,0) = a, \\ D_x^+u &= u_{i+1,j} - u_{i,j}, \\ D_x^-u &= u_{i,j} - u_{i-1,j}, \\ D_y^+u &= u_{i,j+1} - u_{i,j}, \\ D_x^-u &= u_{i,j} - u_{i,j-1}, \\ g(\alpha) &= g(t^n, x_i, y_j, \alpha) \text{ for any function .} \end{aligned}$$

Similarly, the two-dimensional Hamiltonian becomes

$$\begin{aligned} J^n_{i,j}(\alpha) = {} & D^+_x u \left(\frac{\mathbb{D}_{11}(\alpha)}{2(\Delta x)^2} + \left(\frac{f_1(\alpha)}{\Delta x} \right)^- \right) + D^-_x u \left(-\frac{\mathbb{D}_{11}(\alpha)}{2(\Delta x)^2} + \left(\frac{f_1(\alpha)}{\Delta x} \right)^+ \right) \\ & + D^+_y u \left(\frac{\mathbb{D}_{22}(\alpha)}{2(\Delta y)^2} + \left(\frac{f_2(\alpha)}{\Delta y} \right)^- \right) + D^-_y u \left(-\frac{\mathbb{D}_{22}(\alpha)}{2(\Delta y)^2} + \left(\frac{f_2(\alpha)}{\Delta y} \right)^+ \right) \\ & + \frac{\mathbb{D}^+_{12}(\alpha)}{2\Delta x \Delta y} \Big(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} \\ & \qquad - 2u_{i,j} - u_{i+1,j-1} - u_{i-1,j+1} \Big) \\ & + \frac{\mathbb{D}^-_{12}(\alpha)}{2\Delta x \Delta y} \Big(2u_{i,j} + u_{i+1,j+1} + u_{i-1,j-1} \\ & \qquad - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1} \Big) + L(\alpha). \end{aligned} \tag{5.3}$$

5.2 Brute force scatter search method

This is a brute force method that only assumes that the admissible set $\mathcal{A}$ is bounded, and relies upon the use vectorised function calls (see appendix E). We present the method in its 1D formulation, but the algorithm is easily extended to two dimensions.

Let

$$\alpha \in \mathcal{A} = \{\mathbf{x} | \mathbf{x} \in [\mathbf{x}_{\min}, \mathbf{x}_{\max}]\} \subset \mathbb{R} \tag{5.4}$$

and

$$R = |\mathbf{x}_{\max} - \mathbf{x}_{\min}|. \tag{5.5}$$

1. Find an initial guess.
   - Evaluate $J(\alpha)$ at N equidistant sample points α_j in $\mathcal{A}$. Note that the distance between the points in α_j is $\Delta\alpha_0 = \frac{R}{N}$.
   - Set $\alpha^0 = \alpha^* = \text{argmin}_j J(\alpha_j)$.
2. Until $\Delta\alpha_k \leq \epsilon$, do:
   - Set $\Delta\alpha^k = \frac{2\Delta\alpha^{k-1}}{n}$.
   - Sample $J(\alpha)$ in N sample points in $[\alpha^k - \Delta\alpha^k, \alpha^k + \Delta\alpha^k]$.
   - Set $\alpha^{k+1} = \alpha^* = \text{argmin}_j J(\alpha_j)$.

Assuming that N is chosen high enough to saturate $\mathcal{A}$ sufficiently, this method will find the minimiser. This method can be interpreted as a generalisation of the bisection method; an N-section method.

5.2.1 Error tolerance and number of iterations

It is possible to precompute the number of iterations required to attain the desired accuracy ϵ by starting with the definition $\Delta\alpha_k \leq \epsilon$ and realising that

$$\Delta\alpha_k = 2\Delta\alpha_{k-1}/N \rightarrow \Delta\alpha_k = 2^k \frac{R_i}{N^{k+1}}$$

such that one can see that

$$k \geq \frac{\log\left(\frac{R}{\epsilon N}\right)}{\log\left(\frac{N}{2}\right)} \tag{5.6}$$

This value should be precomputed. The total number of evaluations of (5.2)(5.3) will be

$$k^N N_t \Pi_{i=1}^N N_{x_i} \tag{5.7}$$

where N_{x_i} refer to the number of gridpoints in the x_i-direction.

As an example to how costly this is, consider $\Omega \in [0,1]^2$ with $\Delta x = \Delta y = 0.1$ and $\Delta t = 0.1\Delta x$. Assume that $\mathcal{A} = [-1,1]^2$ and that we set $N = 40$ with tolerance $\epsilon = 10^{-6}$. This yields that $k = 4$, for a grand total of $1.6 \cdot 10^5$ evaluations. This motivates a search for better methods.

5.3 Hybrid method

We wish to improve the scatter search method, particularly by limiting its memory usage and how many sample points it evaluates. We begin by taking cues from point-wise line search methods, in particular the idea of search directions. Recall that line search methods typically resemble

$$\alpha_{n+1} = \alpha_n + \gamma_n p_n,$$

for step size γ_n and search direction p_n. After a sufficiently dense initial scatter, we have an initial guess of the minimiser α_0, and we know that the minimiser α^* is in the domain $K = (\alpha_0 - \Delta\alpha, \alpha_0 + \Delta\alpha)$. If we assume that we can compute the gradient of $J(\alpha_0)$ and that J is sufficiently smooth in K, we know which half of the domain K to find x^*. This gives us the method below.

- Find a good initial guess α_0 such that $\alpha^* \in (\alpha_0 - \Delta\alpha_0, \alpha_0 + \Delta\alpha_0)$.

- For $k = 0, 1, 2...$:
  - Compute $J'(\alpha_k)$:
    - If $J'(\alpha_k) < 0$, set $\alpha_k^j \in [\alpha_k, \alpha_k + \Delta\alpha_k]$ with $\frac{N}{2}$ equidistant points.
    - If $J'(\alpha_k) > 0$, set $\alpha_k^j \in [\alpha_k - \Delta\alpha_k, \alpha_k]$ with $\frac{N}{2}$ equidistant points.
    - If $J'(\alpha_k) = 0$, set $\alpha_k^j \in [\alpha_k - \Delta\alpha_k/2, \alpha_k + \Delta\alpha_k/2]$ with $\frac{N}{2}$ equidistant points.
  - Set $\alpha^{k+1} = \alpha^* = \text{argmin}_j J(\alpha_j)$ and $\Delta\alpha_{k+1} = \Delta\alpha_k/N$.
  - If $|\alpha_{k+1} - \alpha_k| < \epsilon$ for some tolerance ϵ, return α_{k+1}. Otherwise, repeat.

The same idea is trivially extended to 2D. The gradient of $J(\alpha)$ may be computed, or it may be implemented and called directly. We test both approaches in section 6.2.

5.3.1 Error tolerance and number of iterations

Let us assume the search domain is divided into N_0 points, and each subsequent subdomain is divided into n points. Let I be the size of the initial search domain. Bearing in mind that each subdomain is simply half (in 1D) or a quarter (in 2D) the size of the scatter search domain, we have

$$\Delta x_k = \begin{cases} \frac{I}{n^k N_0} \text{ (in 1D)} \\ \frac{I}{(2n)^k N_0} \text{ (in 2D)} \end{cases}$$

This gives us to the number of iterations to satisfy an error tolerance of ϵ as

$$k \geq \begin{cases} \frac{\log\left(\frac{I}{\epsilon N_0}\right)}{\log(n)} \text{ (in 1D)} \\ \frac{\log\left(\frac{I}{\epsilon N_0}\right)}{\log(2n)} \text{ (in 2D)} \end{cases} \tag{5.8}$$

5.4 Error propagation from computed control

In this section we investigate the effect of computing the optimal control to within a tolerance ϵ_α such that, in effect, we solve the MFG equations with $\alpha = \alpha^* + \epsilon_\alpha$. By using Taylor expansion on the Hamiltonian in (1.1a), we get

$$u_t + H(\alpha^* + \epsilon_\alpha) = u_t + H(\alpha^*) + \epsilon_\alpha H'(\alpha^*) + \mathcal{O}(\epsilon_\alpha^2) = 0.$$

If we assume that $\alpha^* \notin \partial\mathcal{A}$, viz the optimal control is contained within $\mathcal{A}$, then we expect that $H'(\alpha^*) = \mathcal{O}(\Delta x)$ (from our discretisation of the Hamiltonian). As such, the error from computing the control will be of second order as long as $\epsilon_\alpha \leq \Delta x$.

The Fokker-Planck equation yields a comparatively similar result,

$$m_t + \sum_{i=1}^{N} \left(f_i(\alpha^*)m\right)_{x_i} - \sum_{i=1}^{N}\sum_{j=1}^{N} \left(\mathbb{D}_{ij}(\alpha^*)m\right)_{x_i x_j} = \epsilon_\alpha \left(\sum_{i=1}^{N}\sum_{j=1}^{N} \left(\mathbb{D}'_{ij}(\alpha^*)m\right)_{x_i x_j} - \sum_{i=1}^{N} \left(f_i'(\alpha^*)m\right)_{x_i} \right) + \mathcal{O}(\epsilon_\alpha^2),$$

but the error will be of first order in ϵ_α regardless. As such we expect that the error will be slightly worse for the FP equation. However, the error from the computed control will be no better than the order of the discretisation, in our case if $\epsilon_\alpha \leq \Delta x$. To avoid significant influence on the error, a precision should simply be chosen such that ϵ_α is sufficiently smaller than the discretisation error, without leading to too many evaluations in the optimisation methods.

5.5 Vectorised versions

There are two different ways of implementing the schemes that we have considered; sequentially, or vectorised. Recall that we have a problem that cannot be done for all times, but may in principle be done for all spatial points at the same time. In the 1D-case, the sequential approach solves the optimisation problem for a single spatial node completely before proceeding to the next. This is done in our methods by making a one-dimensional array of sample values α_j for each spatial node.

The vectorised approach for this is to create a two-dimensional array of sample values for all spatial nodes over the $x\alpha$-plane. This means that each row/column of this array contains the sample values for one spatial point. This will occupy more computer memory, but may be faster. We will see the quantified differences between sequential and vectorised scatter search in section 6.2.

5.6 A note on brute force computations for the 2D problem

We have a few options in the control problem for MFG in 2D. The sequential option in terms of implementation is to freeze x, y and thus simply use scatter search to find the optimal controls for each grid point in turn. It

is however fully possible to vectorise this as well, as we described above. The optimisation problem then becomes a four-dimensional problem over the $xy\alpha_1\alpha_2$-space. This approach may not be favourable due to the sheer memory demand.

An alternative is to use a fixed-point approach by freezing one of the controls (say α_2) at some initial guess and consider the three-dimensional space over the other control (say $xy\alpha_1$). This will have to be repeated for the opposite control, and so on, until a fixed point (α_1^*, α_2^*) is reached. As such this procedure will likely use several iterations to converge. However, the memory usage will not be as staggering as solving the full four-dimensional problem.

Thus we may implement the 2D problem as a sequential manner, a 4D manner or a 3D manner. We compare the running times of the three approaches in section 6.2.2. Some related implementation details and learned lessons are covered in section E in the appendix.

Chapter 6

Numerical tests

In this section we present a range of numerical tests, from verification methods of the individual schemes to speculative mean field games applied to modelling scenarios.

6.1 Verification tests

In this section we present results that verify that our discretisations are sound. We cover the one-dimensional (1D) and two-dimensional (2D) problems.

6.1.1 Verification test for the HJB-equation (1D)

For this verification test we wished to verify our discretisation of the HJB equation (1.1a). We chose the functions in (1.1a) to resemble the canonical variant (2.4a):

$$\begin{aligned} f(t,x,\alpha) &= \alpha, \\ L(t,x,\alpha) &= \frac{1}{2}\alpha^2 + F(t,x), \\ \sigma(t,x,\alpha) &= \sigma(t,x). \end{aligned}$$

We then chose the solution as $u(t,x) = e^{-t}\sin x$, and sought to resolve the equation and balance equality by selecting $F(t,x)$ appropriately. For this choice of functions, the Hamiltonian is resolved to yield the optimal control $\alpha = -u_x$. For the numerical computation itself, we used our optimisation methods to compute α. However, for the sake of selecting an appropriate F, we resolved the Hamiltonian as if it was indeed $\alpha = -u_x$:

$$u_t - \frac{1}{2}u_x^2 + \frac{\sigma(t,x)^2}{2}u_{xx} = -F(t,x,y)$$

We insert this into our solution of u, to yield F as:

$$F(t,x) = \left(\tau + \frac{\omega^2}{2}\sigma(t,x)^2\right)e^{-\tau t}\sin(\omega x) + \frac{\omega^2}{2}e^{-2\tau t}\cos^2(\omega x)\,. \tag{6.1}$$

For the test we chose $\Delta x = 0.1\Delta t$ and $T = 1$, and computed the solution for progressively finer grids. To keep from solving on the entire real line, we chose boundaries that matched reflective Neumann boundary conditions. The scheme performed as expected with a first order convergence in all the norms; see figure 6.1.

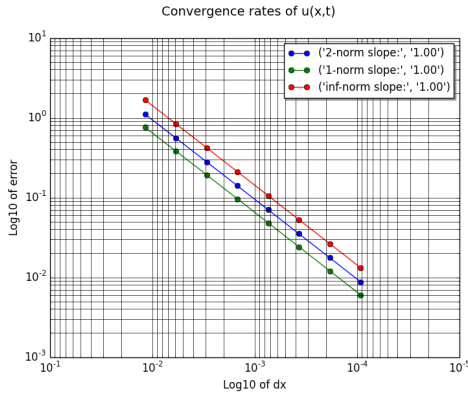

FIGURE 6.1: Convergence rate of the one-dimensional finite difference scheme for the HJB-equation. The scheme performed as expected with a first order convergence rate.

6.1.2 Verification test for the HJB-equation (2D)

For this test we proceeded in a similar way as in the previous section. We chose the functions in (1.1a) as

$$\begin{aligned} f(t,x,\alpha) &= (\alpha_1, \alpha_2)^T, \\ L(t,x,\alpha) &= \frac{1}{2}\left(\alpha_1^2 + \alpha_2^2\right) + F(t,x), \\ u(t,x,y) &= e^{-t}\cos x \cos y. \end{aligned}$$

We find from the Hamiltonian that this gives the optimal control

$$(\alpha_1, \alpha_2) = (-u_x, -u_y)\,. \tag{6.2}$$

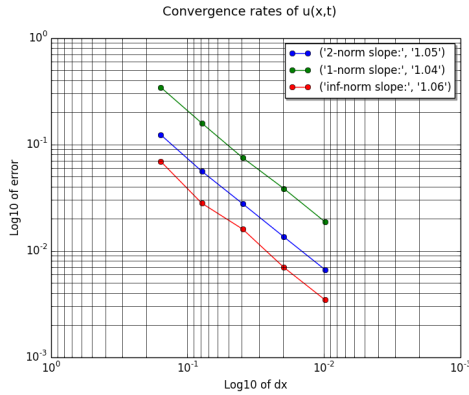

FIGURE 6.2: Convergence rate of the finite difference scheme for the two-dimensional HJB-equation. The scheme performs as expected with first order convergence.

By rewriting (1.1a) and inserting the above functions, we get that the source term F is:

$$F(t,x) = \left(1 + \frac{D_{11}}{2} + \frac{D_{22}}{2}\right) e^{-t} \cos x \cos y + \frac{e^{-2t}}{2} \left(\sin^2 x \cos^2 y + \cos^2 x \sin^2 y\right) - D^{12} e^{-t} \sin x \sin y.$$

We ran the test using $\Delta x = 0.1\Delta t$ and $T = 1$, for progressively finer resolutions. The scheme performed as expected with first order convergence; see figure 6.2.

6.1.3 Verification tests for the FP-equation (1D)

We demonstrate our schemes for the one-dimensional Fokker-Planck equation on cases for which exact solutions exist.

6.1.3.1 Constant coefficients

For this case we chooe a scalar the Fokker-Planck equation that is simply a scalar convection-diffusion equation

$$m_t + a m_x = D m_{xx}, \tag{6.3}$$

for coefficients a, D. The exact solution is

$$m(x,t) = h_F(x-at,t) * m_0(x) = \frac{1}{\sqrt{4\pi Dt}} \int_{-\infty}^{\infty} \exp\left(-\frac{(y-at)^2}{4Dt}\right) m_0(x-y)dy \tag{6.4}$$

where h_F is the fundamental solution of the heat equation. For $a = D = 0.5$ we got the results in figure 6.3. The naive scheme (4.2) outperformed the

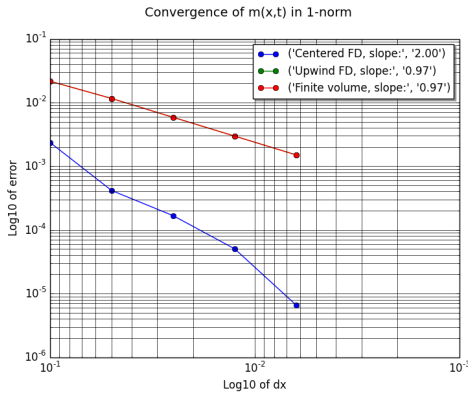

FIGURE 6.3: Convergence rate in the L_1-norm of the Fokker-Planck equation with constant coefficients. The rates were computed without the first data point. The rate for the monotone finite difference scheme (4.7) and the finite volume scheme (4.14) were identical because the schemes became equivalent for this particular case.

other schemes with a second order convergence, but all other schemes were still first-order convergent.

6.1.3.2 Variable velocity

Consider now the variant

$$m_t - \gamma(xm)_x = Dm_{xx}, \tag{6.5}$$

for $\gamma \in \mathbb{R}, D > 0$. This is the Fokker-Planck equation for the Ornstein-Uhlenbeck process. The fundamental solution to this is (see appendix C for details),

$$m_F(x,t) = \sqrt{\frac{\gamma}{2\pi D\left(1-e^{-2\gamma t}\right)}} \exp\left(-\frac{\gamma x^2}{2D(1-e^{-2\gamma t})}\right), \tag{6.6}$$

so that the solution is

$$m(x,t) = m_F(x,t) * m_0(x). \tag{6.7}$$

The test was run with $v = -10D = -1$ on $x \in [0, 10]$. The large grid and the strict condition on Δt (4.3) for the naive scheme forced us to use $\frac{\Delta t}{\Delta x} = 0.05$. Convergence results in the infinity-norm can be seen in figure 6.4. The naive finite difference scheme was only of first order for this case, contrary to its second order behaviour in the constant coefficient case. This is likely due to the non-constant velocity, $f(x) = -\gamma x$. The other schemes achieved first order convergence. The finite volume scheme (4.14) had similar performance as the other schemes, and has the lowest condition on the time step Δt. Hence we prefer this scheme over the others.

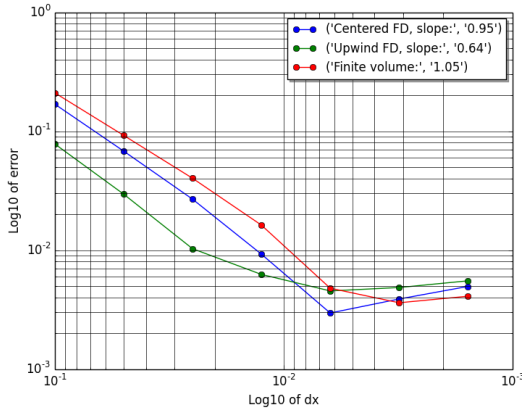

FIGURE 6.4: Convergence rate in the infinity-norm of the Fokker-Planck equation with variable velocity. The convergence rate deteriorated for the finest resolutions due to boundary condition treatment near $x = 0$.

6.1.4 Verification tests for the FP-equation (2D)

In this test we were mainly interested in how the diffusion flux methods in section 4.2.2 behaved for different choices of diffusion tensor $\mathbb{D}$, specifically in regards to preserving positivity in the solution. We used $\Omega = [0, 1] \times [0, 1]$ throughout this test, with the initial distribution

$$m_0(x, y) = \begin{cases} C \text{ if } (x, y) \in [0.1, 0.3] \times [0.1, 0.3] \\ 0 \text{ otherwise} \end{cases},$$

where C is such that $\int_\Omega m_0 dx = 1$. We also used the diagonal diffusion and velocity as given below:

$$f(x,y) = \frac{1}{2}\begin{bmatrix} x, & y \end{bmatrix}^T$$
$$D_{11}(x) = \begin{cases} 0.005 \text{ if } x \leq \frac{1}{2} \\ 0.01 \text{ if } x > \frac{1}{2} \end{cases},$$
$$D_{22}(y) = \begin{cases} 0.005 \text{ if } y \leq \frac{1}{2} \\ 0.01 \text{ if } y > \frac{1}{2} \end{cases}.$$

For the first test we simply set $D_{12} = 0$. Note that for this test, the naive and O-method become identical. We found that both the linear methods and the nonlinear method had the same first order rate of convergence for this case, and preserved positivity in the solution; see figure 6.5. Now, let

$$D_{12}(x,y) = 0.00125(2 + x + y). \tag{6.8}$$

Recalling the discussion of the naive method, we expected that that it would not conserve positivity. However, the naive method and the O-method perform similarly in this case as well. In addition, all methods have about the same first order convergence rate; see figure 6.6. This behaviour by the naive method is likely the rare exception rather than the rule. Wepresent our findings for the last case, in which

$$D_{12} = \begin{cases} .005 \text{ if } x, y \leq \frac{1}{2} \\ 0.01 \text{ if } x, y > \frac{1}{2} \end{cases}. \tag{6.9}$$

In this case, both the linear methods introduced oscillations that grew rapidly worse as the mesh was refined, and only the nonlinear scheme converged as expected while maintaining the positivity of the solution; compare the naive and nonlinear schemes in figure 6.7.

We should also point out that the nonlinear scheme required more and more Picard iterations to resolve the nonlinear system (4.33) as $\mathbb{D}$ grew less diagonally dominant, from 2 in the diagonal case, to 6 in case (6.8) and up to 9 for case (6.9).

6.1.5 No-game error propagation test

In this test we wished to inspect any differences in convergence rate or error depending on what precision we chose for computing the optimal control.

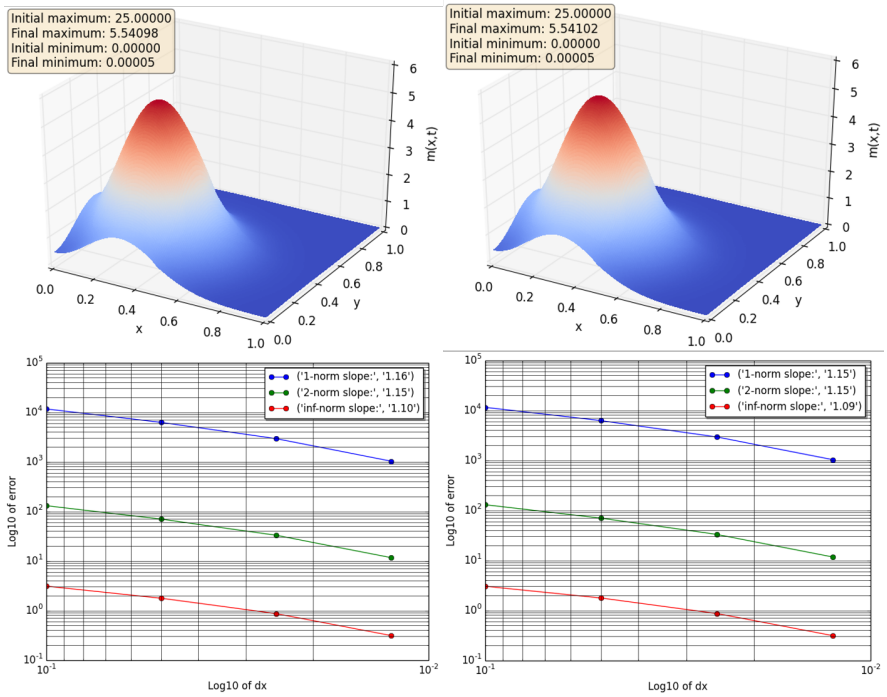

FIGURE 6.5: Convergence rates (top row) and solutions (bottom row) of the 2D Fokker-Planck equation for $D_{12} = 0$. The naive method was used on the left, the nonlinear diffusion scheme on the right. Both methods produced similar solutions and have the same convergence rates.

Consider the following set of input functions;

$$
\begin{aligned}
f(\alpha) &= \alpha \\
L(x,\alpha) &= \frac{1}{2}\alpha^2 + F(x)m_0(x) \quad = C\left(\exp\left(-\frac{(x-0.3)^2}{0.01}\right) + \exp\left(-\frac{(x-0.6)^2}{0.01}\right)\right), \\
u_T(x) &= 0, \\
F(x) &= (x-0.5)^2, \\
\sigma &= 0, \\
x \in [0,1] \quad t &\in [0,1].
\end{aligned}
$$

Despite setting $\sigma = 0$, we will have no issues with convergence in this case since F does not depend upon m. As such, the solution procedure converged

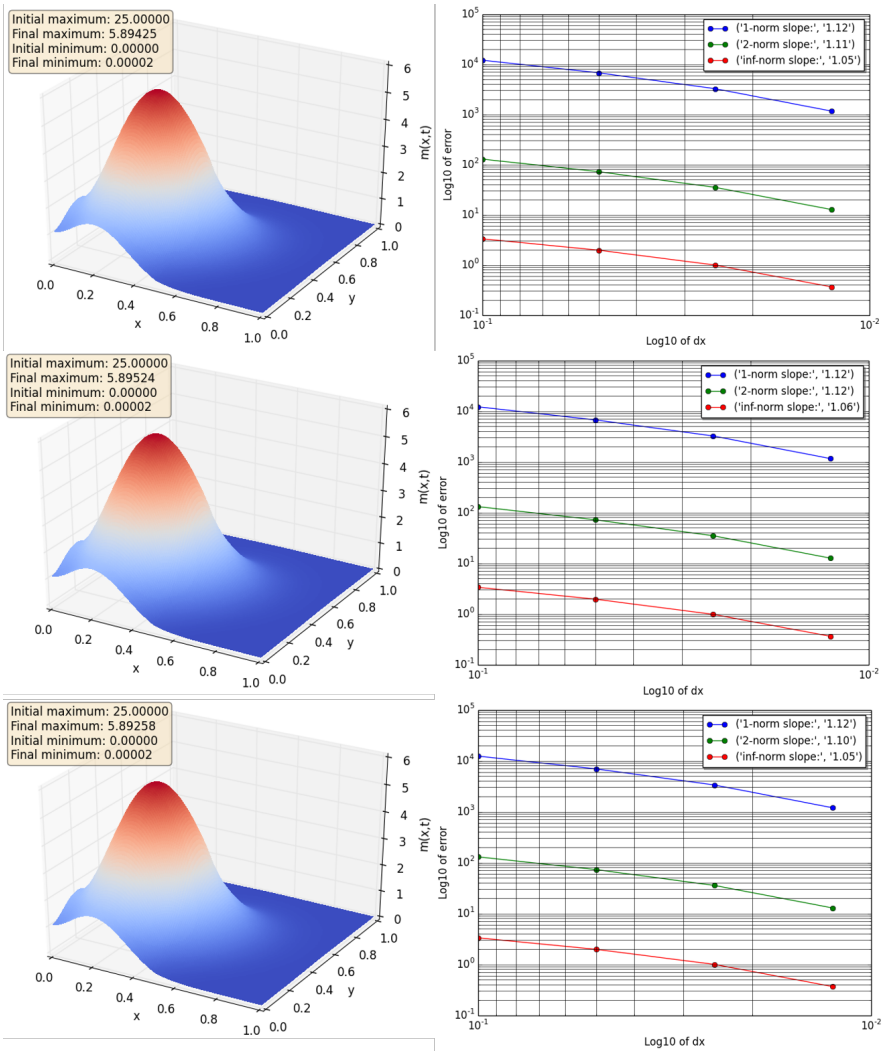

FIGURE 6.6: Convergence rates (first column) and solutions (second column) of the 2D Fokker-Planck equation, with (6.8). From top to bottom: naive method, O-method and the nonlinear method. All methods produced similar solutions and had the same convergence rates.

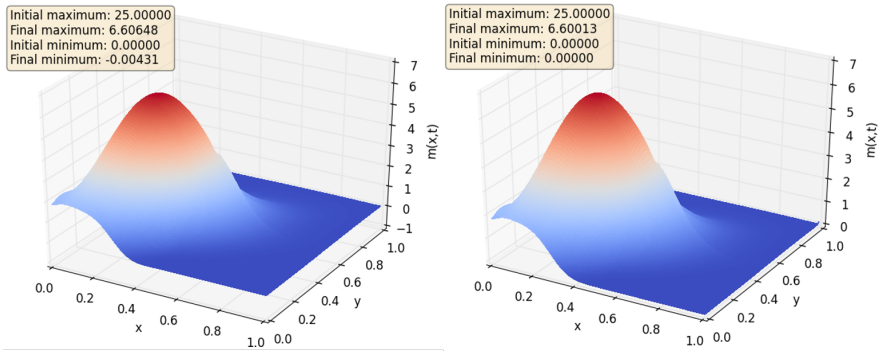

FIGURE 6.7: Solutions of the 2D Fokker-Planck for a case when the diffusion tensor $\mathbb{D}$ is not diagonally dominant; the naive method on the left, and the nonlinear method on the right. Note that oscillations have introduced negativity into the solution for the naive method.

after a single iteration. We chose two different precisions,

$$\epsilon_\alpha^1 = \Delta x$$
$$\epsilon_\alpha^2 = 10^{-6}(\Delta x)^3$$

When running the tests, the difference in convergence rates and in the errors match our expectations from section 5.4; negligible difference for the HJB equation, and observable for the FP equation. See figures 6.8 and 6.9 for a summary of this test. Note that the convergence rates for m are lower than expected, while the ones for u are as expected. This turned out to be due to the choice of $\sigma = 0$; when the same test was run for $\sigma = 0.1$, the convergence rates were all 1 for ϵ_α^2 and about 0.9 for ϵ_α^1. Also in figure 6.8 is a compilation of all the solutions (with increasing precision) plotted in the same figure. Spikes and rapid changes in the graph like this are quite common for MFG with $\sigma = 0$, and may have contributed lower convergence rate of m. Since more diffusion seemed to weigh up for the worse precision, it may be that more diffusion may in some way decrease the importance of exact α.

6.2 Optimisation method performance tests

In this section we present test results for the optimisation methods in section 5 on test cases in 1D and 2D. We measure their performance as the relative running times in order to draw conclusions about how the methods scale with the problem size, both in terms of grid size and the size of $\mathcal{A}$. All the

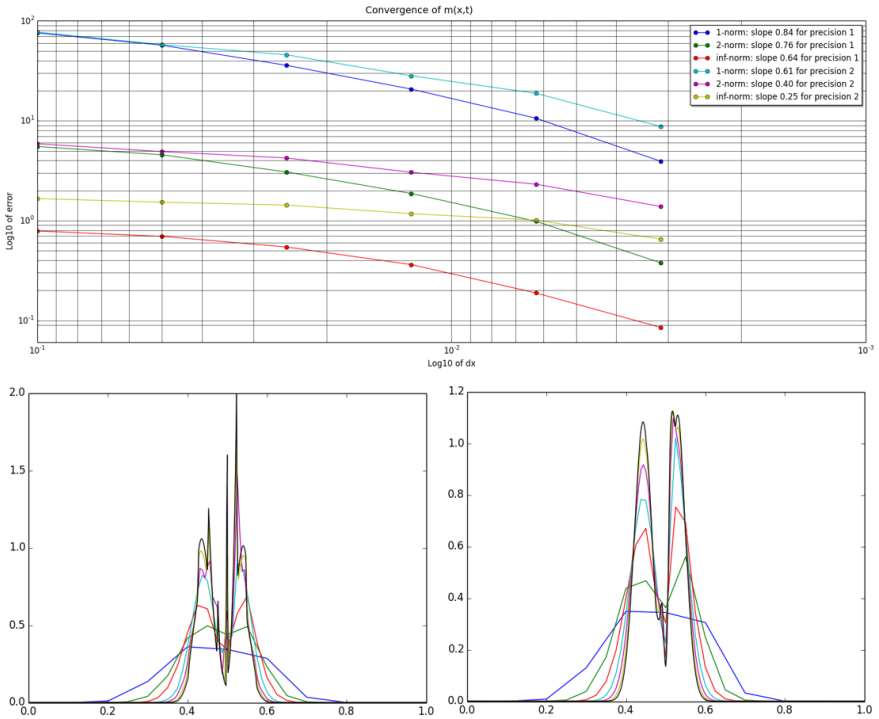

FIGURE 6.8: Results for the error propagation test. On the top are the convergence rates for m. On the lower row are the solutions of $m(T, \cdot)$ (for ϵ_α^1 on the left, and ϵ_α^2 on the right) as the grid is refined.

methods in section 5 iteratively refine their search space a predetermined number of times. In all these tests, all methods used four refinements.

All tests in this section were run on a 64-bit computer with an Intel Core i7-4500U 1.8-3.0GHz processor with 4MB of cache, and 12GB of RAM memory.

6.2.1 Running times for optimisation in 1D

We present test results for the 1D versions of the optimisation routines in tables 6.1 and 6.2. We used the spatial domain $\Omega = [0, 1]$, using equidistant nodes; the number of nodes is shown in the leftmost columns of the tables. For the search space $\mathcal{A}$ we used the same grid resolution, $\Delta x = \frac{1}{N+1} = \Delta\alpha$. As noted in the captions, the difference between the two tables is the size of

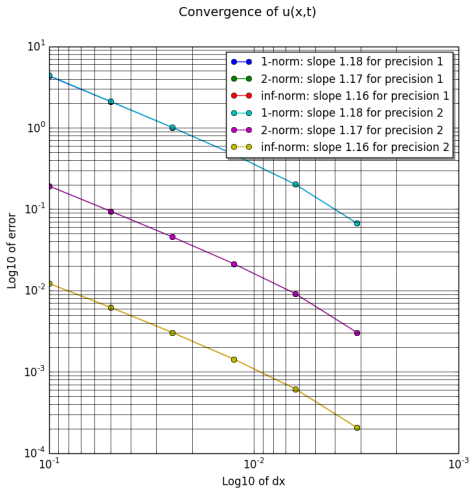

FIGURE 6.9: Convergence rates for u for the error propagation test. Note that there is only negligible difference between the rates and errors.

the control domain $\mathcal{A}$, which has a significant impact on the results.

TABLE 6.1: Average running times (in seconds) for the 1D optimisation routines for increasing grid resolutions. (S) indicates sequential, non-vectorised code, and (V) indicates vectorised versions. For the hybrid method, (C) indicates a computed gradient. This is for $\mathcal{A} = [-3, 3]$.

Nodes	(S) SciPy	(S) Scatter	(V) Scatter	(V) Hybrid(C)	(V) Hybrid
10	0.014505	0.005033	0.001895	0.002038	0.001699
20	0.079262	0.012139	0.005351	0.004648	0.004200
40	0.186722	0.029055	0.016912	0.013010	0.011752
80	0.425403	0.084447	0.079713	0.051565	0.045810
160	0.926268	0.264815	0.313623	0.226958	0.196543

Across the board, we see that the built-in optimisation of SciPy[a] was consistently slower and scales unfavourably compared to even the sequential version of scatter search method. By comparing the sequential and vectorised versions of scatter search method, we see a speedup that declined as the grid is refined (faster so for the larger $\mathcal{A}$). The (vectorised) hybrid method in

[a]We have found no obvious way to vectorise SciPy's optimisation library. As we see from tables 6.1 and 6.2, there is also no obvious reason to do so.

TABLE 6.2: The same details apply for this table as for table 6.1. This is for $\mathcal{A} = [-1, 1]$.

Nodes	(S) SciPy	(S) Scatter	(V) Scatter	(V) Hybrid(C)	(V) Hybrid
10	0.014187	0.004365	0.001186	0.001454	0.001257
20	0.028983	0.009251	0.002500	0.002579	0.002343
40	0.122514	0.020000	0.006721	0.005915	0.005322
80	0.478464	0.047714	0.022791	0.017839	0.016335
160	0.983324	0.131891	0.106902	0.069402	0.062312

both its forms performed both slower and faster than scatter search method, being slower for coarse grids and faster for finer grids. Note that the hybrid method that does not compute its gradient was consistently faster than its computed variant. In addition, it had a bigger relative speedup when $\mathcal{A}$ is bigger.

More or less all of this is explained by memory usage and cache misses. A cache miss is a failed attempt to read or write a piece in the cache memory, meaning that the data must be loaded from a deeper, slower layer of memory. The scatter search uses twice as much memory than the hybrid method for its search space, but the hybrid methods also use slightly more evaluations. At the finest resolution in table 6.1, we see that the sequential scatter search was faster than the vectorised one. This likely represents the break-even point between the gain in evaluating the entire grid and the increased memory usage of doing so. This is supported by the fact that the vectorised version was faster for the finest resolution in table 6.2, for a smaller $\mathcal{A}$.

All in all the hybrid methods had the best running times for all but the coarsest grid resolutions. The fact that there is a break-even point for the vectorised versus sequential scatter search implies that the same applies for the hybrid methods. As such, for resolutions finer than those used here, it may be ideal to use a sequential version of the hybrid methods.

6.2.2 Running times for 2D optimisation

We present test results for 2D versions of the optimisation routines in tables 6.3 and 6.4. On the basis of the lackluster performance of the pointwise/sequential scatter search below, we chose to only test vectorised versions of the methods. All tests were run on the domain $\Omega = [0, 1] \times [0, 1]$, using equidistant nodes. The same number of nodes were used in each dimension, and so the total number of spatial nodes used is shown in the leftmost columns of the tables. For the search space $\mathcal{A}$ we used the same grid resolution, $\Delta x = \Delta y = \frac{1}{N+1} = \Delta\alpha$. The first thing to note is that the full 4D-search was very slow when

TABLE 6.3: Average running times (in seconds) for the 2D optimisation routines for increasing grid resolutions. These are the results for $\mathcal{A} = [-3, 3] \times [-3, 3]$.

Nodes	Pointwise	4D	3D	3D Hybrid (C)	3D Hybrid
10^2	0.720513	0.606407	0.050000	0.042128	0.039748
20^2	4.709110	7.798255	0.210484	0.164122	0.153054
40^2	38.791792	Failed	2.152870	1.537187	1.429983

TABLE 6.4: Average running times for the 2D optimisation routines for increasing grid resolutions. These are the results for $\mathcal{A} = [-1, 1] \times [-1, 1]$.

Nodes	Pointwise	4D	3D	3D Hybrid (C)	3D Hybrid
10^2	0.438032	0.076285	0.025698	0.025872	0.024610
20^2	2.472344	0.987909	0.100107	0.090193	0.086085
40^2	11.879513	Failed	0.763849	0.606633	0.576157

compared to the 3D fixed point methods, and even failed for a fine grid resolution due to running out of memory. For the 3D methods we see that the hybrid methods were generally faster than the scatter method, with a speedup that became more substantial the larger $\mathcal{A}$ was. This is explained in the same way as the results for the 1D case. While we for the 1D case argued that sequential search methods may be beneficial as the grid becomes extremely fine, there is no indication from the results in the 2D case that this may also apply here. It may also be that we simply have not tested the methods on a fine enough grid.

Some lessons learned when implementing the optimisation methods in NumPy are covered in section E in the appendix.

6.3 On solution procedures on MFG

In order to solve the MFG equations (1.1), we used a two-step iteration procedure:

1. Compute α^k, u^k using m^{k-1}.
2. Compute m^k using α^k.

This procedure was repeated until $\|m^k - m^{k-1}\| < \epsilon$ for some tolerance value ϵ[b].

[b]We chose the solution of m as a measure of convergence, as this is typically the quantity one will be the most interested in.

While doing preliminary experiments on some MFG scenarios, we ran into cases in which this procedure simply failed to converge. One of two conditions typically applied to such cases; the first of which was that the diffusion was set too low, the second that m-dependent cost terms in L were too dominating. We give some speculation on this later, but refer to similar experiences reported in [3]. We must also point out that this applied even for our tests on canonical MFG.

To cope with this difficulty, we conceived two supplements to the iteration procedure: **viscosity** and **damping** sequences. The former applies artificial diffusion to (1.1a)(1.1b), while the latter dampens m-dependent cost terms. Both methods were able to produce convergent solutions to (1.1a)(1.1b) in cases where the regular procedure failed.

A **viscosity sequence** is a decreasing sequence $(\delta_n^v) \to 0$. Starting with some δ_0, the MFG equations are solved with this extra diffusion. Once a convergent solution is attained, this solution is used as the initial guess of $m(t,x)$ for the next number in the sequence. A **damping sequence** is an increasing sequence $0 \leq (\delta_n^d) \to 1$, and is applied in the way demonstrated below:

$$L(x,\alpha,m) = \alpha^2 + \frac{x}{1+m} \Rightarrow L_{\delta_n^d}(x,\alpha,m) = \alpha^2 + \frac{x}{1+\delta_n^d m}.$$

Note that both variations and combinations of both viscosity and damping sequences is possible. One such we have employed is to let the coefficients be functions of time, for example $\delta_k(t) = \delta_k^*(1-t)$.

Due to some rather challenging cases, we implemented a program that stored the last convergent intermediate solution of m. In this way, we could attempt to load the solution after one failed sequence and try to continue the computation for another sequence.

6.4 Application: Economic modelling

For this set of cases in one dimension, we took inspiration from the economic modelling scenario described in [14]. In this model, the state $x \in [0,1]$ refers to the level of isolation in the agents' home, and the aim is for the agents to minimise their heating cost over the winter $t \in [0,T]$, with $T = 1$. The magnitude of the control α thus represents the amount of funds each agent spends; a negative control means they spend funds to decrease their isolation, and vice versa. We introduced a cost term

$$F(t,x,m) = 2p(t)(1-\beta x) + \frac{0.4x}{(1+m(t,x))}, \tag{6.10}$$

where $p(t)$ is the cost of electricity, and $\beta = 0.95$ is a discount associated with having more isolation. The second term reflects maintenance costs associated with more isolation, which becomes discounted the more agents have the same isolation level. We set the cost of electricity as

$$p(t) = \sin(\pi t)$$

and let the initial distribution be Pareto-like,

$$m_0(x) = \frac{32}{5\,(1+3x)^3}.$$

Over the next subsections we will look at this case in different ways, to yield different results. We continued with the solution iteration procedure until $\|m^k(T,\cdot) - m^{k-1}(T,\cdot)\|_1 < 10^{-4}$. We used reflective boundary conditions for u and natural boundary conditions for m.

6.4.1 Canonical MFG, with diffusion

For this case we set the terminal cost as $u_T = 0.4x^2(1.5 - x)$, to emulate some "isolation tax" at the end of winter. In this case we also used a bit of diffusion. The functions used in this case are summarised as

$$\begin{aligned} f(\alpha) &= \alpha, \\ L(t,x,\alpha,m) &= \frac{1}{2}\alpha^2 + F(t,x,m), \\ \sigma &= 0.1, \end{aligned} \tag{6.11}$$

where $F(t,x,m)$ is given by (6.10). The diffusion means that the agents' isolation may degrade or upgrade randomly over the course of time, which we interpret as damages and repairs done to the isolation as time goes by. We will in section 6.4.3 look at what happens when σ depends on the control α.

We used $\frac{\Delta t}{\Delta x} = \frac{1}{4}$ for these tests, and measured convergence by comparing the interpolated solutions of $m(x,T)$ for coarser grids to the finest solution with 2560 spatial nodes. The convergence rate appears to be approximately of order 1; see figure 6.10 for convergence rates and figure 6.11 for a comparison of the solutions $m(x,T)$ as the grid is refined.

Consider the solutions of (u,α,m) presented in figure 6.12. We see that agents gather into two separate distributions (the "poor" and "rich"), and that which of these they end up in depend on what their initial level of isolation was. We also see that agents with a decent level of isolation ($x \in [0.4, 0.5]$) chose to invest in upgrades at a higher rate (higher α) than those of a lesser level of isolation ($x < 0.4$), in order to "catch up" with the "rich" agents.

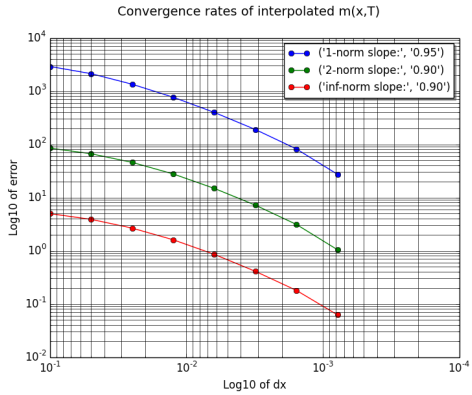

FIGURE 6.10: Convergence rates of $m(x,T)$ for the canonical mean field game in section 6.4.1. The rates appear to be of first order.

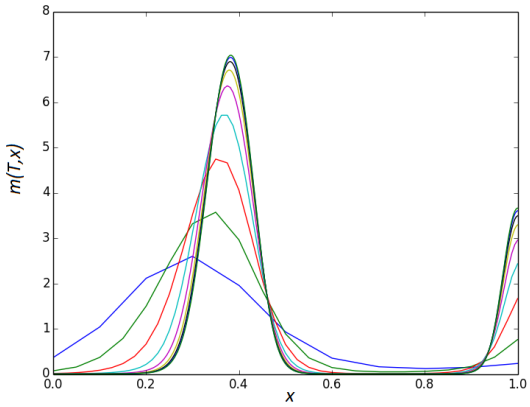

FIGURE 6.11: Comparisons of solutions of $m(x,T)$ as the mesh is refined in section 6.4.1.

Note that the cost of electricity in (6.10) peaks at $t = 0.5$, and compare this to when the agents playing catch up reach very high isolation levels. The downgrades the "poor" distribution does as $t = T$ approaches is both due to the terminal cost u_T, in which they are punished for their level of isolation, and the maintenance cost outweighing the electricity discount. The "rich" distribution only slightly downgrade when $t > 0.9$.

We should mention that this game very closely resembles canonical MFG (2.4). As such, this numerical convergence is not necessarily as speculative as

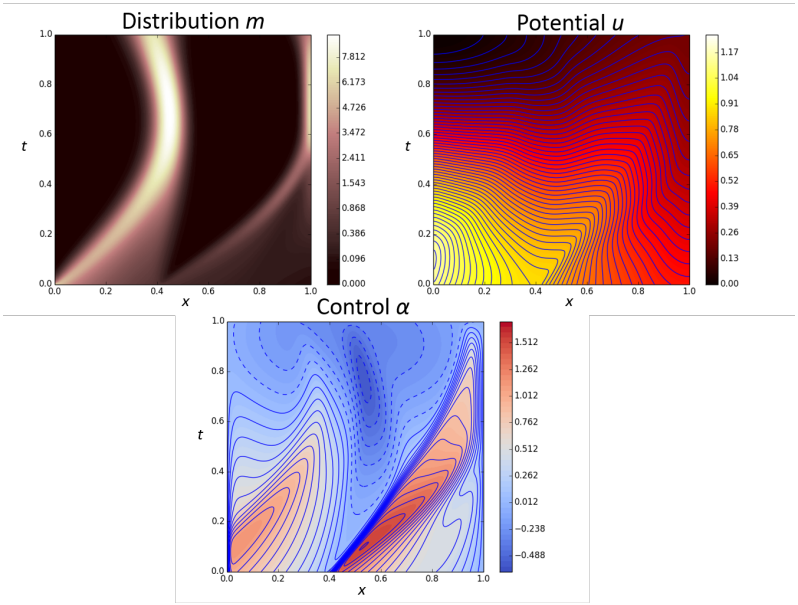

FIGURE 6.12: Contours of the solutions of (u, α, m) for 2560 spatial nodes, for all times t and states x. These are the solutions of the economic modelling case in section 6.4.1.

other cases.

6.4.2 Canonical MFG, without diffusion

In this case we chose the same input functions (6.11) as the last section, but instead chose to set

$$\sigma = 0.$$

As such, there was no random damage or improvement to the isolation.

This greatly affected the convergence properties of the iteration procedure, and in many cases caused the iteration to simply diverge in the sense that $\|m^k(T,\cdot) - m^{k-1}(T,\cdot)\|_1$ never gets past a certain value; see figure 6.13. While attempting to tune the coefficients in (6.10) to induce better behaviour, it was discovered that the choice of coefficients in (6.10) that affect m-terms *also* greatly influenced whether the iteration procedure is able to converge at all. In particular, the greater the role of m in the cost, the tougher it is to achieve convergence, if it is possible at all. In addition, refining the grid

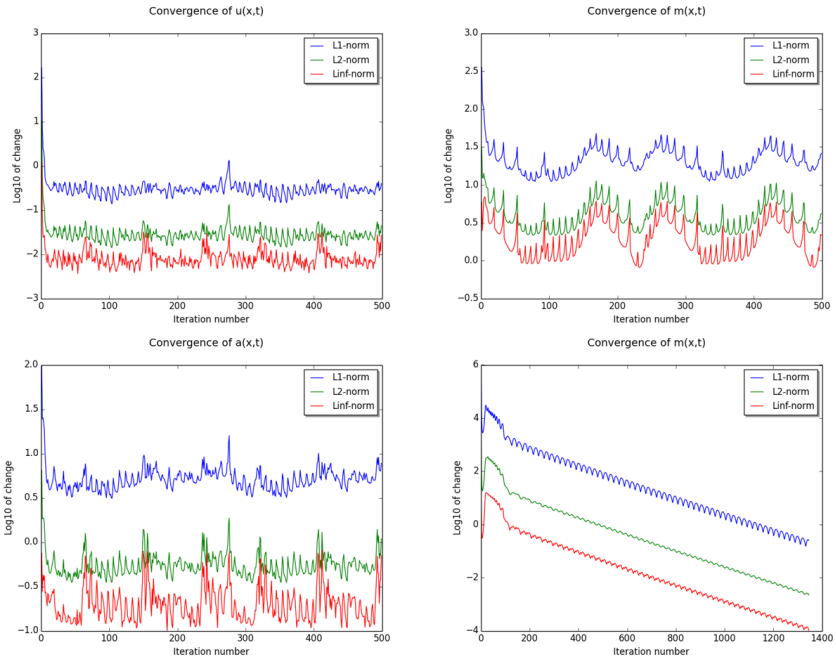

FIGURE 6.13: Norms of the change $\|x^k - x^{k-1}\|$ for the quantities (α, u, m) in a case where convergence was not achieved. In the bottom right corner is the successful convergence of another case; the oscillatory shape is typical of solving the MFG equations.

seems to hinder convergence in the sense that exponentially more iterations are required to make the fixed point iteration converge (if it converges at all). This inspired the alternative approaches detailed in section 6.3; both were able to achieve convergence in some cases in which the straight-forward approach failed. While other cases were not successful to converge to the true solution (with no damping or viscosity), the experiments done with both viscosity and damping sequences give slightly different results that shed light on a probable cause of the lack of convergence when using these methods.

In figure 6.14 we compare results for using both damping and viscosity sequences. We see that there is less change in the solution for the damping sequences, whereas the viscosity sequence has a more obvious transition from smeared out solution, to the spikes showing as the viscosity decreases. In the two bottom rows, note also the qualitative difference between the results; the viscosity sequence has only in a single case produced the second peak that

all the damping sequence solutions have produced.

We were able to use a viscosity sequence to achieve convergence for 60 spatial nodes. See figure 6.16 for the solutions, and figure 6.15 for a comparison of the progress of the viscosity sequence versus a damping sequence[c]. Compare this solution to those in figure 6.14, and note how, typically, the viscosity has influenced the computation in such a manner such that the second peak never appears. Note also that there is no congregation of agents at the rightmost boundary.

We have plotted the solution of (α, u, m) for 60 spatial nodes in figure 6.16. Note the distinct separation of agents that occur around $x = 0.2$, due to the sudden change in the control α at the same location. It is our current understanding that since any deviation (due to damping and/or viscosity sequences) in the location of this jump in the control will have a profound downstream effect on the solution of m, as we can observe that agents flock into two separate distributions. A change in the location of this control jump will greatly change the maximum value in these distributions, particularly as the mesh is refined. This may also explain the results in figures 6.14 and 6.15, in particular the differences between the intermediate solutions of the damping versus viscosity sequences.

It should be noted that we experienced a slightly better results by decreasing the ratio $\frac{\Delta t}{\Delta x}$. This allowed in some cases to reach lower levels of artificial viscosity and/or damping. It could be that a higher order time integration method is prudent to use.

The lessons learned here may be helpful for the development of schemes that combine or innovate the ideas presented in section 6.3 in order to solve low-to-no-diffusion MFG equations, or MFG with a cost function L with a highly influential m-term. We would also like to remark that this case seems to satisfy conditions in theorem 3.

6.4.3 Mean field game: constant vs. controlled diffusion

We altered (6.11) by changing the velocity and cost functions. The new problem becomes

$$L(t, x, \alpha, m) = \frac{0.5\alpha^2 + 0.4x}{1 + m(t, x)} + 2\sin(\pi t)(1 - 0.95x),$$

[c]This same case had failed to converge when using a viscosity sequence where coefficients were only slightly different, roughly $|\delta_n - \delta_{n-1}| < 10^{-4}$. The successful sequence used a greater difference between the coefficients, which may be seen in figure 6.15. This implies which implies that the choice of viscosity sequence matters as well.

$$f(t,x,\alpha) = \begin{cases} (1 - 0.5\sin(\pi t) - 0.25x)\alpha \text{ if } \alpha \geq 0 \\ 0.1\alpha \text{ if } \alpha < 0 \end{cases},$$
$$u_T(x) = 0.$$

In addition to the maintenance cost discount, agents get a similar discount for upgrading/downgrading their isolation; see L. The velocity f is radically changed. We may attach additional expenses to any upgrade the agents do by penalising f[d]. First of all, downgrading ($\alpha < 0$) is more expensive than upgrading. As the price of electricity becomes more expensive, it also becomes more expensive to upgrade. This is reflected in the sin-term in f. The x-term in f reflects the fact that upgrading becomes more expensive the more isolation agents already have.

We ran these tests for two different choices of σ:

$$\begin{aligned} \sigma_1(t,x,\alpha) &= 0.1 \\ \sigma_2(t,x,\alpha) &= 0.1\,(1 + |\alpha|(1-x)) \end{aligned} \tag{6.12}$$

We introduced a control-dependence in σ_2, which deserves a comment. We may interpret this as an inherent uncertainty of the end-product when agents upgrade/downgrade their isolation. This uncertainty decreases the better isolation agents have, which can reflect the increased skill of the labourers who do the work.

We used $\frac{\Delta t}{\Delta x} = \frac{1}{4}$ for these tests and measure convergence as we have done for the other experiments in this case. We get similar first-order convergence as for the canonical MFG; see figure 6.17. The solutions for the finest meshes used for both cases are shown in figure 6.18. The solution for the controlled diffusion were similar to the solution for constant diffusion, with the only remarkable difference being a smoother solution for the control α.

There are obvious differences between this case and the earlier case from section 6.4.1, seen by comparing figures 6.12 and 6.18. In the case of this section, almost no agents at any time are downgrading their isolation due to the penalty of doing so. In fact, the solution of α for the same times t remain quite homogeneous over all isolation levels x, even more so in the controlled diffusion case. As a results, agents only accumulate into one obvious distribution. A similarity in both cases is that agents seem to decrease their spending at around $t = 0.5$. This makes sense in the context of the price of electricity $p(t)$ begins to decrease at this time. In the case of this section, the time $t = 0.5$ is also the lowest point of $f(t,\cdot,\cdot)$.

[d]For example, with $f(t,x_1,\alpha) = 0.5\alpha$ and $f(t,x_2,\alpha) = \alpha$, we see that agents at state x_1 will have to use a higher α than agents at state x_2 in order to attain the same velocity.

In exploring the effect of controlled diffusion, we attempted to use choices of smaller coefficients for σ_2 in (6.12) while still producing convergent solutions. No such coefficients were found.

This case is completely unlike the less general forms of the MFG equations, and the theorems that apply for these are hard to put in context of this case, particularly due to σ_2. The running cost L is however strictly convex and C^1 over α, which we argued earlier is related to the variable p in the MFG form (2.1). As such, closer inspection of this case compared to theorem 1 may result in the revelation that this case satisfy the conditions of this theorem.

6.5 Application: Evacuation

For the tests in this section we used the MFG equations to model an evacuation scenario in which a distribution of agents wish to exit an area via one or more exits, while avoiding congestion. In these tests we were primarily interested in seeing change in the solutions depending on what inputs are used, so that no convergence analysis is attempted.

6.5.1 On our treatment of obstacles and target locations

In crowd dynamics it is prudent to operate with goal locations that agents will actively pursue. This is a natural way to handle the modelling of anything between panicked evacuation and fevered pursuit of teenage idols. The goal location(s) can be implemented into the MFG equation by adding a *distance cost* G to the cost function L. This can be done with an ordinary metric like

$$G(x, y) = (x - x_0)^2 + (y - y_0)^2.$$

Barring other costs, agents will likely follow a straight line to reach (x_0, y_0). This becomes problematic when impassable obstacles are introduced. Unless the distance cost is in some way accommodated to deal with the obstacles, agents are likely to smash into walls or get stuck in corners while pursuing the straightest path. This makes no physical sense in the context of rational agents, which the MFG equations model.

In order to accommodate obstacles for the distance cost, we opted to use the solution of the Eikonal equation

$$|\nabla G| = C \tag{6.13}$$

where $C = C(x, y)$ is the cost associated with moving into the point (x, y). We set C as

$$C(x, y) = \begin{cases} \Delta x \text{ when } (x, y) \text{ not in obstacle,} \\ 100\Delta x \text{ otherwise} \end{cases} \tag{6.14}$$

for all our tests. By setting $G(x_0, y_0) = 0$ for any preferred locations, G will have its minima at any favoured locations, its maxima at obstacles, and will be increasing away from the favoured location. See figure 6.19 for a visual example. We used the fast marching method[e] to solve (6.13).

6.5.2 Numerical experiment

For this test we considered evacuation from a room with two doorways, one which is narrow and one which is wider. The narrow exit is closest to the agents, while the wide one is further away; see figure 6.20. The obstacles are placed at $\Omega_1 = [0.0, 0.2] \times [0.4, 0.5], \Omega_2 = [0.5, 1.6] \times [0.4, 0.5]$ and $\Omega_3 = [1.7, 2.0] \times [0.4, 0.5]$. We use the functions

$$\begin{aligned} f(\alpha_1, \alpha_2) &= (\alpha_1, \alpha_2)^T, \\ L(t, x, y, \alpha_1, \alpha_2) &= \frac{1}{2}\left(\alpha_1^2 + \alpha_2^2\right) + G(x, y) + \frac{1}{\epsilon} m(x, y), \\ \mathcal{A} &= [-2, 2] \times [-2, 2], \end{aligned}$$

where G is the solution of (6.13) for the obstacles shown in figure 6.5.2. With this cost function, agents wish to reach the target area while avoiding concentrations of other agents. We will be interested in seeing qualitatively how many agents decide to take the longest path by varying the parameter ϵ in (6.5.2), which dictates the level of aversion. We use a reasonable amount of diffusion to help with convergence and to keep with the theme of a panicked evacuation.

To measure convergence for the fixed point iterations, we used the norm

$$e_n = \sup_{t \in [0,T]} \|m_n(t, \cdot, \cdot) - m_{n-1}(t, \cdot, \cdot)\|_1. \tag{6.15}$$

and stop when $e_n \leq 10^{-3}$. In our computations we used $\Delta x = \Delta y = 1/20 = 5\Delta t$. We set the diffusion tensor as $\mathbb{D} = 0.05I$, and keep the control bounded in $\mathcal{A} = [-2, 2] \times [-2, 2]$. Part of our experiment was also to take a closer look at the role of the terminal time T for models like this.

6.5.2.1 The role of T

We set $\epsilon = 50$ and chose two different terminal times, $T_1 = 1.5$ and $T_2 = 4$. We compare the solutions at approximately the same time t in figure 6.21. It is obvious that figure 6.21 does not show a snapshot of the same solution, even if all other parameters are equal. In fact, agents for the T_2-case quickly

[e] We describe fast marching method in section D in the appendix.

TABLE 6.5: Results for different choices of the congestion parameter ϵ from the test in section 6.5.2. The percentage of agents in the left half of the domain at $t = 1.03$ is shown in the rightmost column. For $\epsilon = 5$, the fixed point computation did not converge.

ϵ	Iterations	Detour agents
50	9	13.70%
25	13	14.67%
10	35	17.03%
7.5	70	18.07%
6	374	18.97
5	Failed	N/A

evacuate (total evacuation around $t = 1.5$) and spend the rest of the time slowly accumulating into the goal location. For the T_1-case, the solution for $t = 1.4$ in figure 6.21 was qualitatively the same at $t = T_1$.

We may explain the behaviour of the T_2-case by recalling that the agents aim to minimize the cost over time, where the cost is influenced by how fast they move and *where they are located*. For the longer T, it is thus the most profitable to move quickly (α high) to near the preferred location, and spending the rest of the running time moving very little ($\alpha \approx 0$) around this area. In order to create the same effect for the shorter time frame, one would have to subsidise moving quickly by increasing the influence of G in (6.5.2).

6.5.2.2 The role of the congestion parameter ϵ

We chose $T = 2.5$ for these computations, using (6.5.2) for a selection of different congestion parameters ϵ. We expected both that the number of iterations required grew for decreasing ϵ, and that the number of agents who take the detour through the wide exit increase as the narrow exit becomes congested. In particular, we will compare the total percentage of agents inside the left half of the domain shown in figure 6.20 (in $x \in [0, 1]$) at $t = 1.03$. We summarise our findings on this in table 6.5. Note that the iterations required grew exponentially as ϵ decreased, while the difference on the percentage of detouring agents is not very significant. In figure 6.22 we compare the solution of m at $t = 1.03$ for the highest and lowest ϵ-values used. The solutions are qualitatively similar, but the lower ϵ has caused the solution to be more diffused; that is, agents spread out more in order to avoid congestion. Indeed, it seems that the increased amount of agents who detour through the wider exit (see table 6.5) is primarily due to this increased diffusive behaviour. Referring to the terrain laid out in figure 6.20, we see that agents will tend

to take the detour once they move into an area where $x < 1.1$ (the shortest path for this region is through the narrow exit). More diffusive behaviour will naturally mean that agents tend to drift into this region and take the detour. We would have hoped to find a case in which agents would have chosen to take the detour in a more obvious fashion, but for any smaller choices of ϵ the fixed point iteration failed to converge. In order to model a proper evacuation scenario, there are several minor changes possible to what we have used here. While the choice of T has a big influence, this could also be counterweighted with a higher penalty on G, and possibly also a terminal cost associated with staying in the evacuation zone.

6.6 Application: Pursuit of moving object

In this case we considered a scenario in which agents wish to be close, but not too close, to a moving object. Agents have limited movement when congested, but have no aversion to simply being in a crowded area. Besides an interesting application, these tests also served the purpose of attempting convergence analysis of the solutions of (1.1). For convergence of the solution procedure we again use the norm (6.15).

Distance to the object is indicated by the function $d(t, x, y) = d_1(t, x) + d_2(t, y)$. We summarise the choice of functions for (1.1a)(1.1b) below,

$$\begin{aligned} f(\alpha_1, \alpha_2) &= (\alpha_1, \alpha_2)^T, \\ L(t, x, y, \alpha_1, \alpha_2) &= \frac{1}{2}\left(\alpha_1^2 + \alpha_2^2\right)\left(1 + \frac{1}{20}m(t, x, y)\right) + 2d(t, x, y) \\ &\quad + 200\exp\left(-\frac{d_1(t, x)^2 + d_2(t, y)^2}{0.01^2}\right), \\ \mathcal{A} &= [-3, 3] \times [-3, 3], \end{aligned} \tag{6.16}$$

where

$$\begin{aligned} d(t, x, y) &= d_1(t, x) + d_2(t, y), \\ d_1(t, x) &= |x - \cos(t)|, \\ d_2(t, y) &= |y + \sin(t) - 1|, \end{aligned} \tag{6.17}$$

are distance functions. We used $T = 1$ and the domain $\Omega = [0, 1] \times [0, 1]$. The agents' initial distribution was given as

$$m_0(x, y) = \begin{cases} C \text{ if } (x, y) \in [0.1, 0.6] \times [0, 1] \\ 0 \text{ otherwise} \end{cases},$$

where C is such that $\int_\Omega m_0 dx = 1$. Observe that $d(t, x, y)$ tells us that the object moves in a semi-circle from $(x_0, y_0) = (1, 1)$ to $(x_T, y_T) = (0, 0)$. We will be interested in seeing the effect of the increased cost in L as agents get *too* close to the object. We did tests on this case for two choices of $\mathbb{D}$:

$$\mathbb{D}_1 = 0.05I,$$
$$\mathbb{D}_2 = \begin{bmatrix} D_{(d)} & D_{(o)} \\ D_{(o)} & D_{(d)} \end{bmatrix}$$

$$\begin{cases} D_{(d)} &= 0.03125 \left(1 + 0.1 \ln \left(1 + \sqrt{d_1(t,x)^2 + d_2(t,y)^2}\right)\right) \\ D_{(o)} &= 0.0125 \end{cases}.$$

In the case for $\mathbb{D}_2$, the agents experience more diffusion the closer they get to the object, in addition to cross-diffusion. The coefficients in $\mathbb{D}_2$ are chosen to attain convergence even as the grid was refined. This choice of $\mathbb{D}_2$ required the use of the nonlinear diffusion flux from section 4.2.2.3. For both tests we used $\frac{\Delta t}{\Delta x} = \frac{\Delta t}{\Delta y} = 0.2$.

We present a comparison of solutions for both choices of $\mathbb{D}$ in figure 6.23. This is a case that highlights the future-anticipating nature of the MFG equations. As we have a strong cost associated with being too near the object, agents pave the way for the object to pass by them. Due to the sheer computational cost of computing these cases, only a handful of solutions were produced. As the case with $\mathbb{D}_2$ did not converge for the coarsest resolution, its convergence rate plot has only three data points; see figure 6.24. For finer resolutions than those used to compare for numerical convergence in figure 6.24, the solution procedure did not converge for any of the cases. These computations took literally hours to do, and we have thus not attempted to use viscosity or damping sequences to induce better behaviour. An obvious solution would be to use more diffusion in $\mathbb{D}$, but we realised that this would smear out the interesting "paving-of-way" we see in figure 6.23. The results in figure 6.24 could imply that for our earlier cases with convergence for the MFG equations may equally refuse to converge in the solution procedure as the grid is refined. However, we argue that the finest resolution attempted, $\Delta x = \Delta y = 1/60$, does not rid the scheme sufficiently of artificial viscosity terms introduced by using upwinding. We will return to this in the next chapter.

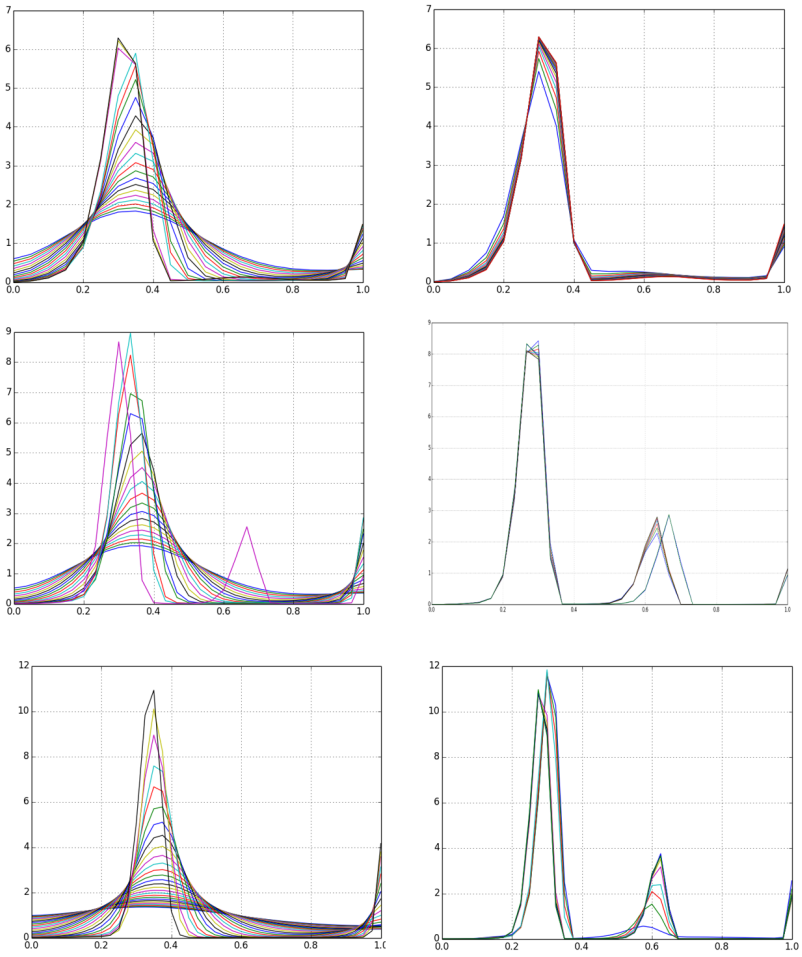

FIGURE 6.14: Comparison of results using the viscosity and damping sequence procedures (left and right columns, respectively) for the numerical test in section 6.4.2. The solutions for $m(x, T)$ are plotted for decreasing coefficients: 20, 30 and 40 spatial nodes on the top, middle and bottom row respectively. Note that the solutions become increasingly qualitatively dissimilar as the mesh is refined.

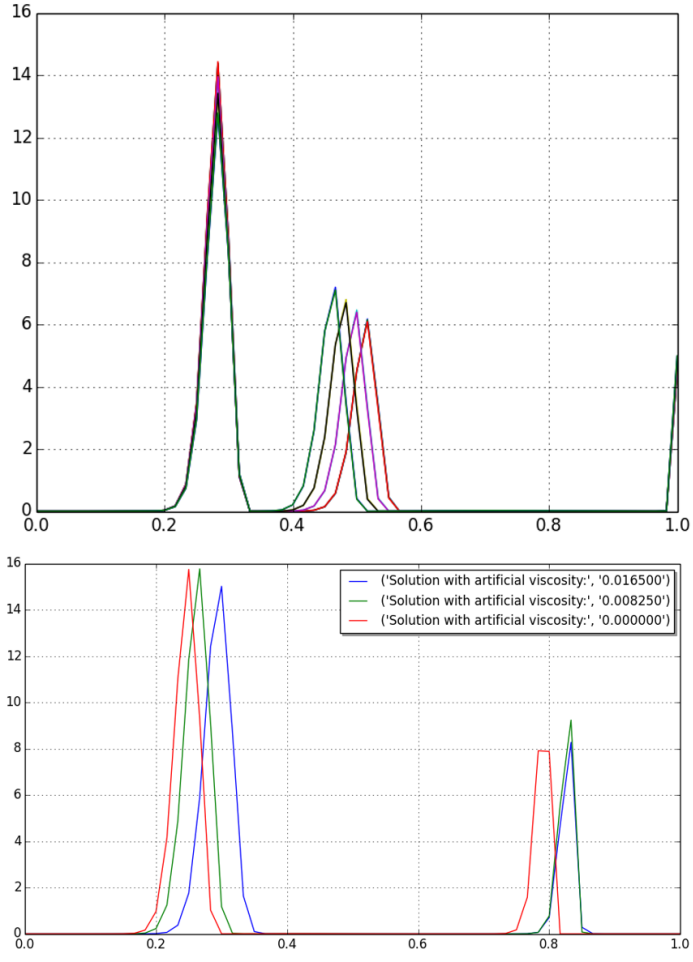

FIGURE 6.15: Comparison of results using the viscosity and damping sequence procedures (bottom and top, respectively) for the numerical test in section 6.4.2. Shown is the solution for $m(\cdot, T)$ plotted for decreasing coefficients, for 60 spatial nodes. Convergence has been reached for the viscosity sequence, whereas the damping sequence had its final successful damping coefficient at 0,265. The difference in damping coefficient value for the other solutions is $5 \cdot 10^{-3}$.

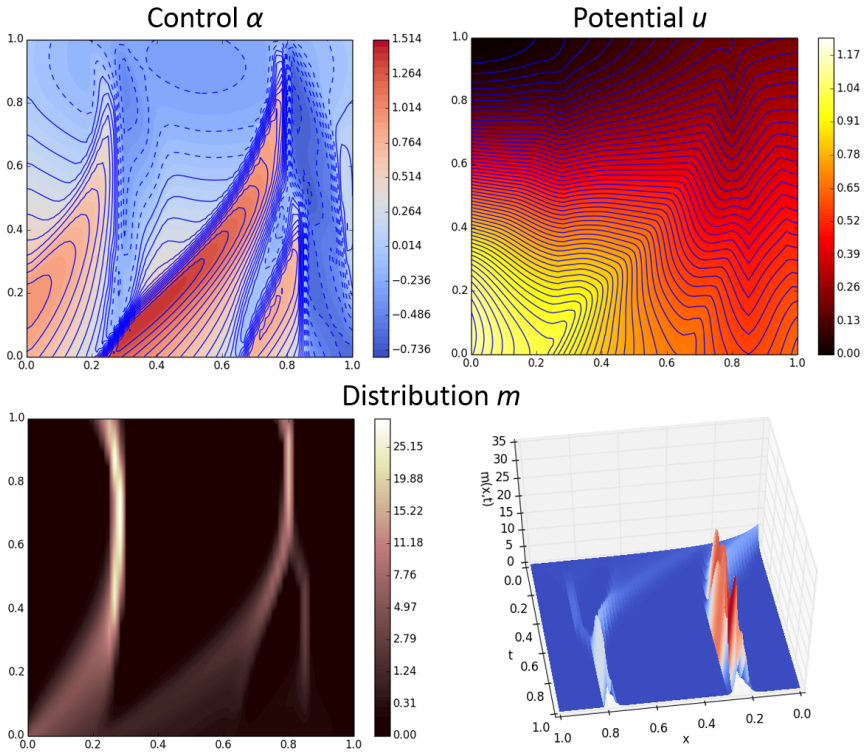

FIGURE 6.16: Solutions of (α, u, m) for the numerical test in section 6.4.2 for $\Delta x = 1/60, \Delta t = 0.1\Delta x$.

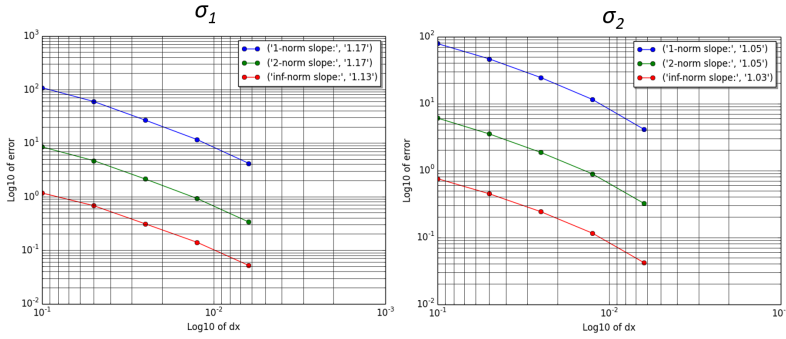

FIGURE 6.17: Convergence rates for $m(T, x)$ for the MFG tests in section 6.4.3.

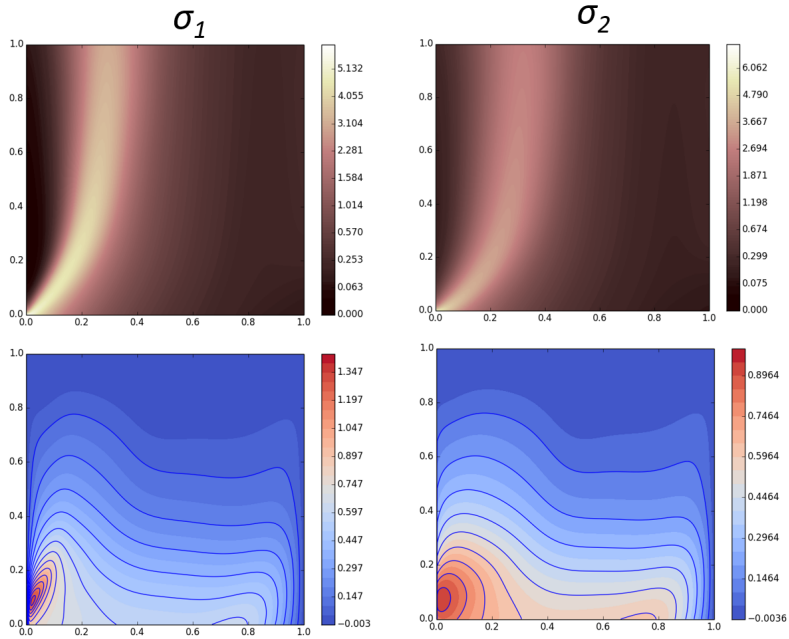

FIGURE 6.18: Comparison of the solutions of $m(t,x), \alpha(t,x)$ for the MFG tests in section 6.4.3.

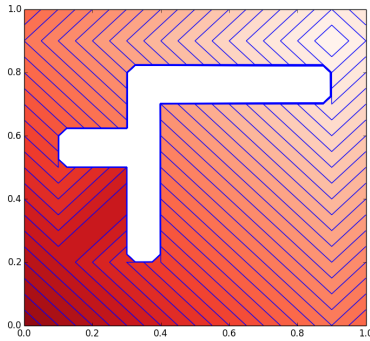

FIGURE 6.19: Visual representation of the location preference function G for 2D mean field games. The white area indicate the obstacles, while the light patch in the upper right corner indicate the preferred location. The value of G increases as the colors grow more red. The shortest path from any point may be found by following the normal direction of the contour lines in a direction of lighter colour.

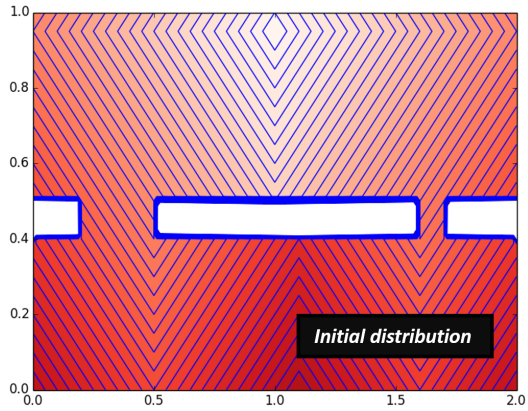

FIGURE 6.20: Obstacles and initial distribution of the case in section 6.5.2. The agents are evenly distributed within $\Omega = [1.1, 1.9] \times [0.1, 0.2]$.

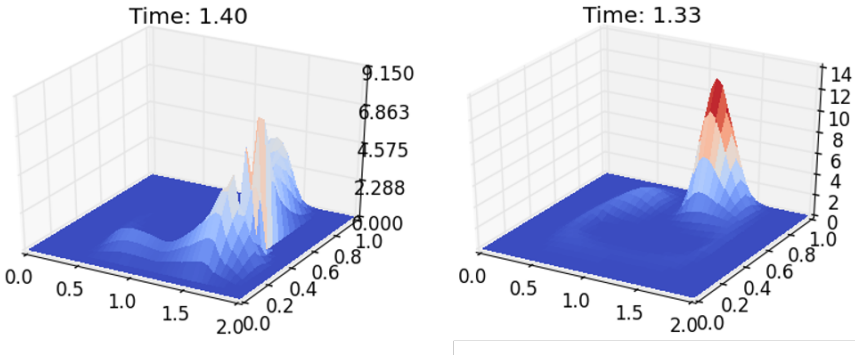

FIGURE 6.21: Comparison of the solution at time $t \approx 1.35$ for two different choices of T. We have used $T = 1.5$ on the left, and $T = 4$ on the right.

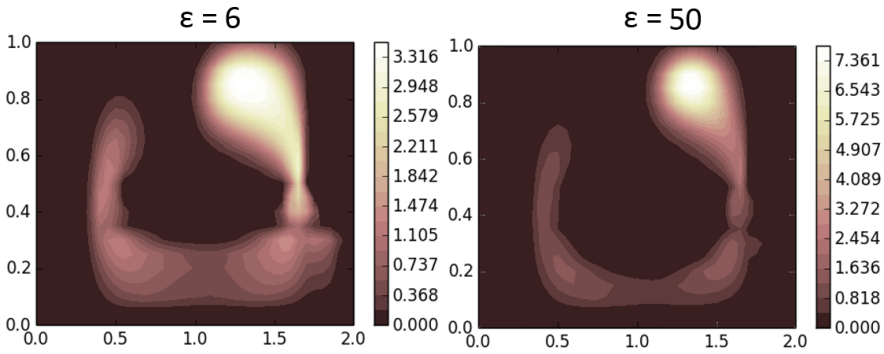

FIGURE 6.22: Comparison of the solution at time $t = 1.03$ for two different choices of ϵ. While qualitatively similar, the lower ϵ seems to have a more diffused solution. Note the scale difference.

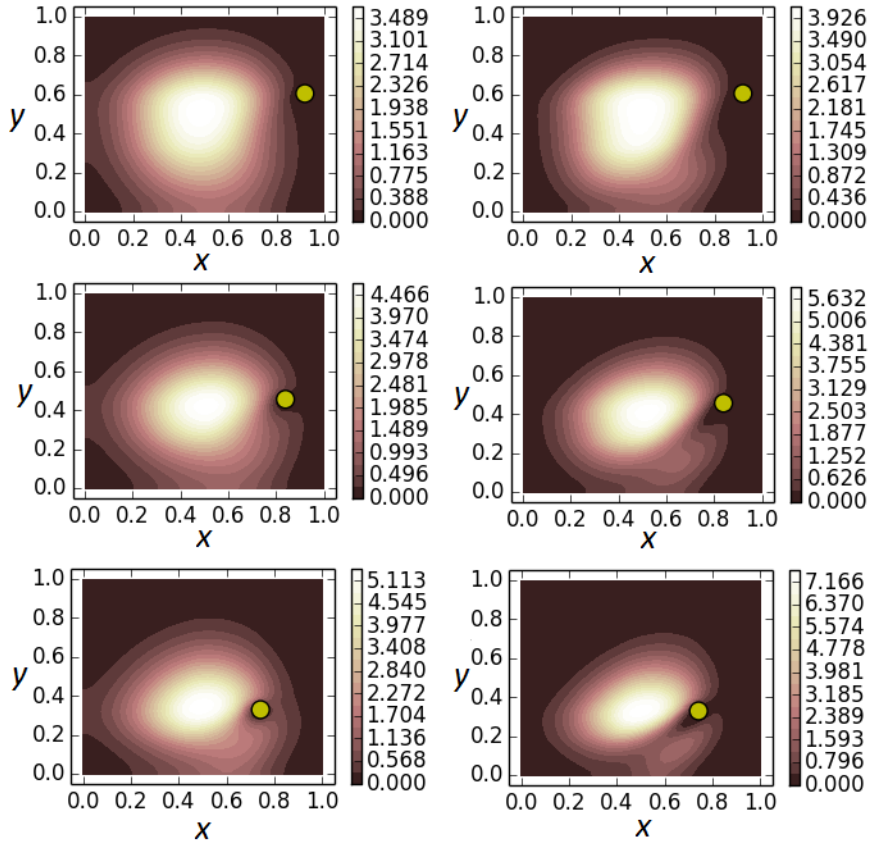

FIGURE 6.23: Comparison of the pursuit case in section 6.6. The left column shows results for $\mathbb{D}_1$, the right for $\mathbb{D}_2$. The time values are $(t_1, t_2, t_3) = (x, y, z)$, from top to bottom. The yellow ball is the target object. Observe that the agents literally pave the way for the object in order to avoid getting too close to it.

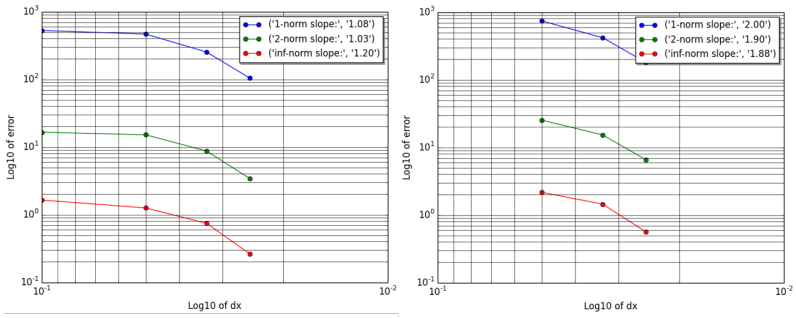

FIGURE 6.24: Convergence plots for the cases $\mathbb{D}_1$ (left) and $\mathbb{D}_2$ (right). These results lose their value when considering that neither converged for the highest resolution of 60 nodes in each spatial direction. The last convergent solutions for both are for 50 nodes in each spatial direction.

Chapter 7

Discussion of findings

This section serves to both revisit concerns raised throughout the text, and to discuss other topics not necessarily covered in the text.

7.1 Discretisations and optimisation

In this section we will reflect on the apparent strengths and shortcomings of our methods, compared to the goals laid out at the end of chapter 2.

7.1.1 The HJB discretisations

We presented numerical schemes for the HJB equation in one and two dimensions in chapter 3. We have not experienced any cases of unwanted behaviour for these methods which could not be contributed to any other cases. Indeed, the solutions for the potential u we see in the tests in section 6.4 appear smooth, especially compared the corresponding solutions of the distribution m and the control α. The schemes are also positivity-preserving in the semi-implicit formulation for similar conditions to those of the FP schemes.

One downside of the HJB schemes is that they are of first order. This is especially due to the discretisation of the Hamiltonian, which is critical for computing the optimal control. The brute force methods use the discretised Hamiltonian to compute the optimal control, and will have an error of the same order as the discretisation used for the HJB equation.

In addition, the schemes' nature as finite difference methods may be a disadvantage if the methods are applied to more general grids.

7.1.2 The FP discretisations

We presented finite volume schemes for the FP equation in chapter 4. These schemes are conservative, positivity-preserving[a] and can easily be modified to apply for more general grids. Conservativity ensures that the running cost L will not experience any artificial damping or amplification due to mass

[a] When (4.20) is satisfied.

being produced or removed by the numerical scheme. We have made choices between various ways to define the convective and diffusive fluxes. It could have been interesting to do an investigation to see whether we could produce different results for the MFG equations when using different fluxes.

The discretisation of the two-dimensional FP equation (4.17) calls for discretising the diffusion tensor $\mathbb{D}$. This is trivial on Cartesian grids when $\mathbb{D}$ is diagonal, but is considerably more challenging when it is not. While we have presented an unconditionally monotone scheme to approach this in section 4.2.2.3, this scheme is nonlinear and hence computationally intense[b]. It bears repeating that the use of a Cartesian grid simplified the generation of the nonlinear diffusion matrix, in particular for the condition that $\mathbb{D}$ is diagonally dominant (this condition stems from the scheme for the 2D HJB equation). Alternate ways of dealing with the diffusion tensor involves remeshing the grid such that the cell edges are orthogonal to the direction of $\mathbb{D}$. In this way, no special treatment of the off-diagonal terms of $\mathbb{D}$ is necessary. However, this puts the cell centres for the solution of m away from the nodes for the HJB scheme, and away from the solutions of u, α. This will require interpolation to evaluate. In addition, for a case such as $\mathbb{D}_2$ in the pursuit model in section 6.6, this remeshing would be required for each time step.

In conclusion, the discretisation we have presented is sound. Despite the potential use of a nonlinear method to evaluate $\mathbb{D}$ correctly, the Cartesian grid simplifies the generation of the matrix for this method. Experiments with more advanced convection fluxes could have been done.

7.1.3 Optimisation methods

Computing the optimal control quickly is a crucial step for the use of our numerical method, in order to produce results within a practical amount of time. As such, considerable time has been spent implementing, testing and optimising the running time the optimisation methods. Several lessons learned during implementation of these methods in NumPy are found in appendix E. However, the use of native NumPy functions may itself be crippling to the performance of the functions. NumPy is built upon Python, which is renowned for its simplicity of syntax, but is generally very slow compared to the likes of Fortran. Within the constraints of using Python with NumPy, however, the methods perform reasonably well. We have presented several different approaches that each have their own strengths and weaknesses.

There are two more technical ways to speed up the methods; decreasing the rate of cache misses and using less memory. The methods suffer slowdowns from cache misses, and the obvious solution for this is to increase the cache

[b]It must be said that the Cartesian

size. Graphical processing units (GPUs) offer orders of magnitude more cache memory than the processor we have used for our computations[c]. GPU programming is becoming more accessible and common, and is the most obvious next step for improving our methods. We should note that there is no readily available GPU-support for NumPy.

The other way is to decrease memory usage by using smaller data quantities to store number. For example (using C-terminology), instead of using 64-bit doubles, 32-bit floats serve the purpose just as well[d] while taking only half the memory.

7.2 Convergence of the solution procedure

In this section we revisit factors that have been observed to influence the convergence of the solution procedure for the mean field game equations. The factors are presented in descending order of their perceived influence.

7.2.1 Influence of distribution costs on convergence

We have with the tests in section 6 seen that the influence of distribution m in the running cost L has a profound effect upon the fixed-point convergence properties of the MFG equations. In particular we refer to table 6.5 from section 6.5.2. Here we demonstrated that increased dependence of the distribution m in the cost L leads to more iterations being used to attain convergence, if at all. We should point out that L increased with m in this case. However, we also experienced increased iteration count when doing experiments on the no-diffusion case in section 6.4.2[e]. Here, L was a decreasing function of m. This essentially means that the more coupled the MFG equations (1.1) are, the more difficult convergence is to attain.

We also experienced that this phenomenon seems related to grid resolution. In particular, given the same choice of functions for (1.1a)(1.1b), we could experience convergence to a solution for a course resolution, only to have it fail for a finer resolution. We argue that this is related to m-terms in the cost in two ways. The first relates to the terms proportional to the second spatial derivatives of u, m in the local truncation errors of our methods[f]. These introduce artificial diffusion that dissipates with decreasing spatial

[c]The processor used in this thesis has 4MB of cache memory, while commercial GPUs have cache sizes ranging into Gigabytes.

[d]Except maybe for vanishingly small precisions for the computations

[e]When viscosity sequences were used in these experiments, the experiments with a lower dependence on m were able to converge faster and to lower viscosity than those with a higher dependence on m.

[f]These are typically introduced from our use of upwinding.

resolution at rate $\mathcal{O}\left((\Delta x)^2\right)$. It is well-known that diffusion in general helps convergence[g]. It is conceivable that the artificial diffusion introduced by the truncation error has an influence. In any case, solutions tend to be more smeared out for lower grid resolutions; see in particular figure 6.11 for solutions of $m(T, x)$ for increasing grid resolution. Here, the coarsest solution barely resembles the finest solution.

We may rationalise all these accusations by going back to the theorems on existence and uniqueness in section 2. In theorem 1, a condition upon the cost function $F(x, m)$ from (2.1a) is that F satisfies suitable growth conditions over m. Increased dependence on m means that the cost L will increase/decrease faster with m. This applies especially for terms like

$$L(t, x, \alpha, m) = \frac{1}{2}\alpha^2 \left(1 + \beta m(t, x)\right),$$

where m, in addition to a coefficient β, scales with the control α. We would expect even worse behaviour for this case.

7.2.2 Choice of initial guess

We have experienced that the choice of initial guess of the solution of m can influence the convergence of the solution procedure. The straight-forward approach we generally have used is to let the initial guess be that $m(t, x) = m_0(x)$. We have presented two ideas that have proved somewhat effective at provoking convergence in cases where this straight-forward approach failed. It was discovered during our work that similar experiences are reported by Achdou in [3]. Achdou approached this by essentially using viscosity sequences as we described in section 6.3.

In section 6.4.2 we examined the behaviour of both viscosity sequences and damping sequences on a case in which there was no diffusion. The results of that section imply that while viscosity sequences may suitable in some cases, damping sequences seem to produce intermediate solutions that more closely resemble the true solution qualitatively. We also experienced that in one case, the choice of viscosity sequence determined its success[h]. We have also attempted to interpolate the solution of the same case from a coarser resolution, and use this as initial guess. This was in general not as successful as viscosity sequences.

[g] See in particular the only difference between the cases in section 6.4.1 and 6.4.2; added diffusion.

[h] In particular in terms of the difference between two coefficients, $\delta_n - \delta_{n+1}$. Where an increasingly smaller difference had failed to yield convergence of the solution procedure, a larger suddenly difference was used to yield convergence to the true solution.

We postulate that if m is competing with other cost terms in L, it is likely that the true solution of the MFG equations (if it exists) is so highly unlike that solution which is produced in the first iteration. This may lead to that the fixed-point iteration requires many more iterations, or is simply doomed.

7.2.3 Diffusion

The role of diffusion is probably intimately connected to m-terms in the running cost L. We have seen the effect diffusion has by considering the same model equation with or without diffusion in sections 6.4.1 and 6.4.2. With diffusion, convergence occurs, without it is extremely challenging. This is unsurprising, but it provides hints as to what may cause the lack of fixed-point convergence in other cases. The greater the effect of m, the greater the influence on α from changes in m. If there is not sufficient diffusion to dissipate and smear out the solutions, thus decreasing the concentration of agents (and the amplitude of m), we experience a lack of convergence.

It appears that the case of controlled diffusion is not wholly relatable to the adage of the more diffusion the better. In the one-dimensional test in section 6.4.3, we presented a convergent case for a choice of diffusion function σ that was proportional to the optimal control α. Specifically, we ran experiments with the two diffusion functions,

$$\begin{aligned}\sigma_1 &= 0.1,\\ \sigma_2 &= 0.1\left(1 + |\alpha|(1-x)\right).\end{aligned}$$

Note that $\sigma_1 \leq \sigma_2$. Any lesser choices we made of coefficient for σ_2 yielded solutions that would not converge for finer resolutions. It should be noted that the solutions for the MFG equations with controlled diffusion is highly speculative, and as we have experienced, likely to not converge as the mesh is refined.

7.2.4 Time step refinement

To a certain extent, we experienced during our various tests and experiments that refining the timestep tends to decrease the number of iterations required for convergence. When using viscosity and damping sequences, we were also able to reach convergence for slightly lower influences of these sequences when the time step was refined. One thing that happens when the timestep is refined, is that the truncation error is decreased. This warrants the idea of using higher order time integration methods. These methods generally require the storage of solutions for several times, which may be unwanted. However, the solution procedure for the MFG equations requires that all

solutions for all timesteps are stored. This makes such methods feasible for solving the MFG equations.

There may be another, maybe more likely explanation to this behaviour. The control α may be rapidly changing, and small variations in its solution may have profound downstream effects on the solution of the distribution m (as we argued in section 6.4.2). Refining the timestep means that the control will be evaluated at more points. This again leads to any rapid changes in the control to manifest themselves and influence the downstream (in time) solution of m.

7.2.5 Precision of optimisation methods

In a few cases we experienced that using insufficient precision in the optimisation methods could slow down convergence in terms of iteration count, or even hinder it altogether. This was easily avoided by simply refining the control search space, and in general seem to be a tag-along effect of any of the other factors. We must point out that any precision attained using the optimisation methods is only precise within the discretisation error of the Hamiltonian in (1.1a).

7.3 Modelling with the MFG equations

In this case we present an overview of our subjective opinion about how to pick functions to model real world scenarios.

In some of the numerical tests of chapter 6 we choose functions for (1.1) that at least heuristically modelled several real-world scenarios. A bit of care should be used when choosing functions, as a "too difficult" configuration might not converge. This seems to apply particularly to introducing control dependence in σ or $\mathbb{D}$.

If the agent velocity f is dependent upon the control, it would be prudent to also limit the size of the control domain $\mathcal{A}$. First of all, this will help keep the HJB and FP schemes within the conditions of their positivity-preservation. Secondly, the magnitude of α is intuitively how much effort or money or energy agents spend. In its context as a physical size, it makes sense to keep it bounded. In some cases we have also penalised the agent velocity in order to model the increased cost to take an action.

The only way to penalise or reward association with the distribution m is to introduce m-terms into the running cost L. As we know, a too-high dependence causes problems with convergence. Hence it is prudent to spend some time scaling these terms, either by doing several experiments and picking

one that produces convergence, or by scaling the m-terms with the physical context of the other quantities, like the control α.

For the evacuation scenario in section 6.5, we introduced the use of the Eikonal equation (6.13) as a way to deal with obstacles. While we believe the arguments for using this approach make it the better modelling option over a naive distance function, we have not attempted the other approach for this scenario. Using the Eikonal equation may be unsuitable for cases in which any desired locations move, like the object in section 6.6.

Chapter 8

Conclusion

In this thesis we have developed robust and general numerical methods for solving the mean field game equations, as well as brute-force optimisation methods for solving the optimal control. The schemes have been verified and used on challenging MFG test scenarios. The coupled nature of the MFG equations lead to the necessity of a solution procedure for solving the equations in an alternating fashion. We experienced, like Achdou reports[3], that there are cases for which the straight-forward application of this procedure fails. We present building blocks for more complex and potentially successful solution procedures.

An intuition for the causes of this lack of convergence has been developed and reported on, after a range of numerical tests. In addition, we introduce alternative solution procedures with slightly better convergence properties. Some of the results for our convergent cases remain speculative, as they are in general unsupported by existing existence theorems.

Based on our experiences, we propose topics of future work to deal with the numerical solution of the MFG equations. We also present some ideas for improvements to our solution methods.

8.1 Future work

The primary concern we have about this work is the lack of convergence for the solution procedure. We have presented the factors we suspect influence this in section 7.2. Of particular interest to us is the influence of the initial guess for $m(t, x)$, which is also reported by Achdou[3] to in some cases hinder convergence altogether. To the author of this thesis, this appears to be an issue worthy of theoretical analysis in order to develop a greater intuition to the behaviour of the solution procedure convergence. We also believe there to be potential for developing computational algorithms that use variations and developments of damping and viscosity sequences to enforce convergence. These algorithms may also use higher order time integration.

The discretisations we have used are only of first order, though they satisfy positivity-preservation and preserve the mass of the distribution. Recall however that the discretisation order of the HJB equation dictates

the order of the approximation of the Hamiltonian. This in turn influences the precision of the computed control α. The optimisation methods we have developed only find a solution that is still influenced by discretisation errors. Increasing the order of this discretisation could lead to higher precision in the control α, which again has a great influence on m. It is possible that such high order methods will require more complicated methods to deal with the diffusion tensor $\mathbb{D}$ than we currently have used for the 2D Fokker-Planck.

Appendix A

Heuristic derivation of the MFG equations

In this section we derive the general forms of the MFG equations (1.1a)(1.1b), from the perspective of stochastic optimal control. For the sake of presentation we neglect any technical details required in the arguments. We point the interested reader to [17][18][7][15] for more viewpoints on how the MFG equations may be derived. We remind the reader of the two primary sizes in MFG; the potential or cost $u(t,x)$ and the distribution of agents in the field $m(t,x)$.

A.1 Derivation of HJB equation

Suppose the position of an agent in the state space is given by

$$X_t = x + \int_t^T f(X_t^x, \alpha_t)dt + \int_t^T \sigma(X_t^x, \alpha_t)dB_t, \tag{A.1}$$

where $f(x,\alpha)$ is the velocity, $\sigma(x,\alpha)$ is the amount of noise or uncertainty. Recall that for Brownian noise, $\mathbb{E}(\int_0^t F(t,\omega)dB_t) = 0$ for a process F with suitable conditions. Suppose also that the agent wishes to minimise the cost

$$u(t,x) := \inf_{\alpha\in\mathcal{A}} \mathbb{E}\left(\int_t^T L(s, X_s, \alpha_s, m_s)ds + G(X_T)\right) \tag{A.2}$$

for some running cost L and terminal cost G. We will need the differential of (A.2), which is provided by Ito's lemma:

$$du = \left(u_t + fDu + \frac{\sigma^2}{2}D^2u\right)dt + \sigma Du dB_t. \tag{A.3}$$

We apply the principle of dynamic programming to (A.2) by first splitting the integral,

$$u(t,x) = \inf_{\alpha\in\mathcal{A}} \mathbb{E}\Big(\int_t^{t+\Delta t} L(s, X_s, \alpha_s, m_s)ds + \int_{t+\Delta t}^{T} L(s, X_s, \alpha_s, m_s)ds + G(X_T)\Big)$$

and then applying theorem 4.3.3 in [22] to yield

$$u(t,x) = \inf_{\alpha\in\mathcal{A}} \mathbb{E}\left(\int_t^{t+\Delta t} L(s, X_s, \alpha_s, m_s)ds + u(t+\Delta t, X_{t+\Delta t})\right). \quad \text{(A.4)}$$

We integrate (A.3) to find an expression for $u(t+\Delta t, X_{t+\Delta t})$:

$$u(t+\Delta t, X_{t+\Delta t}) - u(t, X_t) = \int_t^{t+\Delta t} u_t dt + \int_t^{t+\Delta t} \left(fDu + \frac{\sigma^2}{2} D^2 u\right) dt + \int_t^{t+\Delta t} \sigma Du dB_t$$

We insert this into (A.4) and approximate all the integrals as $\int_t^{t+\Delta t} x(s)ds = \Delta t x(t)$. Note that the last integral vanishes from the expectation of a Brownian integral.

$$\begin{aligned} u(t,x) &= \inf_{\alpha\in\mathcal{A}} \mathbb{E}\Big(\int_t^{t+\Delta t} \left(L(s, X_s, \alpha_s, m_s) + u_t + fDu + \frac{\sigma^2}{2} D^2 u\right) ds \\ &\quad + u(t,x) + \int_t^{t+\Delta t} \sigma Du dB_t\Big) \\ &\approx \inf_{\alpha\in\mathcal{A}} \mathbb{E}\Big(\Delta t \left(u_t + L + fDu + \frac{\sigma^2}{2} D^2 u\right) + u(t,x) \\ &\quad + \sigma Du(B_{t+\Delta t} - B_t)\Big) \\ &= \inf_{\alpha\in\mathcal{A}} \mathbb{E}\left(\Delta t \left(L + fDu + \frac{\sigma^2}{2} D^2 u\right)\right) + u(t,x) + u_t(t,x)\Delta t \end{aligned}$$

Dividing by Δt gives us the Hamilton-Jacobi-Bellman equation:

$$u_t + \inf_{\alpha} \left(L + \sum_{i=1}^{N} f_i \frac{\partial u}{\partial x_i} + \frac{1}{2} \sum_{i=1}^{N}\sum_{j=1}^{N} \left(\sigma_i \sigma_j \frac{\partial^2 u}{\partial x_i \partial x_j}\right)\right) = 0. \quad \text{(A.5)}$$

A.2 Derivation of the Fokker-Planck equation

Let us define the distribution m in the context of this derivation:

$$\int_D m(t,x|s,y)dx = \Pr\left(X_{t+s} \in D | X_s = y\right) = \Pr\left(X_t \in D | X_0 = y\right). \quad \text{(A.6)}$$

Keeping in mind (A.1), we introduce an arbitrary function $\phi = \phi(X_t)$ with boundary conditions $\phi(X_0) = \phi(X_T) = 0$. Following Ito's lemma, we integrate over $t \in [0,T]$:

$$\begin{aligned} 0 = \phi(X_T) - \phi(X_0) &= \int_0^T d\phi \\ &= \int_0^T \left(\phi_t + f\phi_x + \frac{\sigma^2}{2}\phi_{xx}\right) dt + \int_0^T \left(\sigma\phi_x\right) dB_t. \end{aligned}$$

Next, we take the conditional expectation $\mathbb{E}\left(\cdot | X_0 = y\right) = \mathbb{E}(\cdot)$ of the above expression. Note also that the second integral vanishes from the expectation of a Brownian integral.

$$\begin{aligned} 0 &= \mathbb{E}\left(\int_0^T \left(\phi_t + f\phi_x + \frac{\sigma^2}{2}\phi_{xx}\right) dt + \int_0^T \left(\sigma\phi_x\right) dB_t\right) \\ &= \mathbb{E}\left(\int_0^T \left(\phi_t + f\phi_x + \frac{\sigma^2}{2}\phi_{xx}\right) dt\right) \\ &= \int_{\mathbb{R}} \left(\int_0^T \left(\phi_t + f\phi_x + \frac{\sigma^2}{2}\phi_{xx}\right) dt\right) m(t,x|s,y)dx \\ &= \int_{\mathbb{R}}\int_0^T \left(\phi_t m\right) dtdx + \int_{\mathbb{R}}\int_0^T \left((fm)\phi_x\right) dtdx + \int_{\mathbb{R}}\int_0^T \left(\frac{\sigma^2}{2}\phi_{xx}\right) dtdx. \end{aligned}$$

Note that we have set $m = m(t,x|s,y)$. We will now isolate ϕ by integrating by parts:

$$\begin{aligned} \int_{\mathbb{R}}\int_0^T \left(\phi_t m\right) dtdx &= \int_{\mathbb{R}} \left((m\phi)\big|_0^T - \int_0^T \phi m_t dt\right) dx \\ \int_{\mathbb{R}}\int_0^T \left((fm)\phi_x\right) dtdx &= \int_0^T \int_{\mathbb{R}} \left((fm)\phi_x\right) dxdt \\ &= \int_0^T \left((fm\phi)\big|_{-\infty}^{\infty} - \int_{\mathbb{R}} \left((fm)_x\phi\right) dx\right) dt \\ \int_{\mathbb{R}}\int_0^T \left(\frac{\sigma^2}{2}\phi_{xx}\right) dtdx &= \int_0^T \left(\left(\frac{\sigma^2 m\phi_x}{2}\right)\Big|_{-\infty}^{\infty} dt - \int_{\mathbb{R}} \left(\frac{\sigma^2 m}{2}\right)_x \phi_x dx\right) dt \end{aligned}$$

$$= \int_0^T \left(\frac{\sigma^2 m \phi_x}{2} - \frac{(\sigma^2 m)_x \phi_x}{2} \right) \Big|_{-\infty}^{\infty} dt$$
$$+ \int_{\mathbb{R}} \int_0^T \left(\frac{\sigma^2 m}{2} \right)_{xx} \phi dt dx$$

We may neglect the boundary terms by choosing suitable boundary conditions upon ϕ, and are left with

$$\int_{\mathbb{R}} \int_0^T \left(-m_t - (fm)_x + \frac{1}{2} \left(\sigma^2 m \right)_{xx} \right) \phi dt dx = 0.$$

From the arbitrariness of ϕ, we end up with the Fokker-Planck equation,

$$m_t + (fm)_x = \frac{1}{2} \left(\sigma^2 m \right)_{xx}.$$

for the probability distribution of the agents' location.

A.3 A closing word on the derivations

Strictly speaking, these derivations only apply to the actions of a single agent, and considerable work is put into other derivations to making the equations apply for a *continuum* of agents. As such, the Fokker-Planck equation becomes not the probability density of the state of a single agent, but the distribution of the field of agents. We turn the interested reader to a derivation of the canonical MFG equations based on game theory in [7].

Appendix B

M-matrices

M-matrices are matrices of a special but common form, for which no all-encompassing definition exist. Numerous conditions that define a matrix to be an M-matrix are found in [5]. Among their primary properties that make them relevant to our work is their diagonal dominance and inverse-positivity.

Definition 5. *A matrix A is said to be* ***inverse-positive*** *iff its inverse $A^{-1} = (a_{ij}^{-1})_{n\times n}$ satisfies*

$$(a_{ij}^{-1})_{n\times n} \geq 0 \forall i, j \in [1, n]. \tag{B.1}$$

The condition (B.1) *is equivalent to A^{-1} being* ***nonnegative****.*

From [5] we have the following useful lemma:

Lemma 1. *For an irreducible matrix $A = (a_{ij})_{n\times n}$ satisfying $a_{i,i} > 0 \forall i \in [1, n]$ and $a_{i,j} \leq 0 \forall i, j \in [1, n]$, if A is weak diagonal dominant in rows, that is*

$$\sum_{j=1}^{n} a_{i,j} \geq 0 \forall i \in [1, n] \tag{B.2}$$

with strict inequality for at least one term in (B.2)*. Then the matrix A is an M-matrix, and its inverse is nonnegative.*

Appendix C

Ornstein-Uhlenbeck process solution

In this section we find the solution of (6.5),

$$\begin{aligned} m_t - \gamma(xm)_x &= Dm_{xx} \\ m_0(x) &= \delta(x - x_0) \end{aligned}$$

Multiply both sides by e^{-ikx} and integrate over $\mathbb{R}$ to yield

$$\begin{aligned} \tilde{m}_t + \gamma k \tilde{m}_k &= -Dk^2 \tilde{m} \\ \tilde{m}_0(k) &= e^{-ikx_0} \end{aligned} \tag{C.1}$$

for $\tilde{m} = \tilde{m}(t, k)$. We then use the method of characteristics to get

$$\frac{d\tilde{m}(t(s), k(s))}{ds} = \frac{\partial \tilde{m}}{\partial t}\frac{\partial t}{\partial s} + \frac{\partial \tilde{m}}{\partial k}\frac{\partial k}{\partial s} = -Dk^2\tilde{m}$$

which applies for

$$\begin{aligned} \frac{\partial t}{\partial s} = 1 \Rightarrow t(s) = s \\ \frac{\partial k}{\partial s} = \frac{\partial k}{\partial t} = \gamma k \Rightarrow k(t) = k_0 e^{\gamma t} \end{aligned}$$

We solve the differential equation,

$$\begin{aligned} \frac{d\tilde{m}}{dt} = -Dk^2\tilde{m} = -Dk_0^2 e^{2\gamma t}\tilde{m} \\ \tilde{m} = Ce^{-\frac{Dk_0^2}{2\gamma}e^{2\gamma t}} \Rightarrow \tilde{m} = \exp\left(-ik_0x_0 - \frac{Dk_0^2}{2\gamma}\left(e^{2\gamma t} - 1\right)\right) \\ \tilde{m}(t, k) = \exp\left(-ikx_0e^{-\gamma t} - \frac{Dk^2}{2\gamma}\left(1 - e^{-2\gamma t}\right)\right) \end{aligned}$$

The last expression is the Fourier transform of a Gaussian with mean $\mu(t) = x_0e^{-\gamma t}$ and standard deviation $\sigma(t) = \sqrt{\frac{D}{\gamma}\left(1 - e^{-2\gamma t}\right)}$. We take the inverse

Fourier transform and get the fundamental solution

$$m_F(t,x) = \sqrt{\frac{\gamma}{2\pi D\left(1-e^{-2\gamma t}\right)}}\exp\left(-\frac{\gamma x^2}{2D(1-e^{-2\gamma t})}\right).$$

Appendix D

Fast marching methods for obstacle handling

The fast marching method (FMM) is a numerical method introduced by Sethian (see for instance [21]) for solving Eikonal equations. In our work it was used to compute the location preference function G for MFG in 2D in section 6.5.2. We will provide a short description of FMM in the context of our work.

We will divide the nodes in the mesh into three disjoint sets, K, T, U to indicate the **k**nown, **t**rial and **u**nvisited nodes. The set K holds all nodes whose values of $G(x, y)$ we know, T holds all nodes whose values we have some estimate or guess on, and U holds all sets we have not considered yet. It will also be necessary to do bookkeeping on the estimated values of the nodes in T. This will be clearer as we briefly explain the algorithm.

We initialise the method by setting nodes corresponding to the preferred location(s) into K, and setting the neighbours[a] of these nodes into T. We then proceed as follows:

1. Select the node n_0 in T that has the smallest estimated value, and put this node into K.
2. Put the neighbours of n_0 that are in U into T.
3. For each of the neighbours of n_0, n_i, set their new estimated value as

$$\text{val}(n_i) = \min\left(\text{val}(n_0) + C(x_i, y_i), \text{val}^*(n_i)\right),$$

   where $\text{val}^*(n_i)$ refers to any previously computed estimate.

This algorithm is run until all nodes are in the set K.

[a] For our finite volume method, we let the cells that share edges be neighbours.

Appendix E

Implementation details and lessons learned

In this section we present some tips and tricks that proved necessary for implementing the methods detailed in this thesis, as well as to speed up certain subroutines. All our work has been done using the Python libraries SciPy and NumPy. NumPy, not unlike Matlab, is optimised for handling vector broadcasting and N-dimensional arrays, and has linear algebra routines based on BLAS. As such, the most time-consuming parts of running the code has been generating matrices, evaluating functions and running the optimisation routines detailed in this section. As work progressed and familiarity with NumPy increased, these routines were all vectorised to yield substantial code speedup. This section will provide some code examples for the most substantial learned lessons. The tricks used should be easily extended to similar languages and tools that is optimised to handle arrays, such as Matlab.

E.1 Handling N-dimensional arrays for optimisation methods

A challenge when implementing the optimisation methods in 2D is the need to handle three- or four-dimensional arrays. In particular, to find and refer to the minimum values in these arrays along a certain axis, and to update the search domains along certain axes. This case is curious, as the NumPy documentation has no examples on how to use the library for such cases. Thankfully, experiments have yielded a fast way to do this. Critical here are the NumPy functions **numpy.argmin**[a] and **numpy.indices**[b].

We present a snippet from the 3D scatter method, showing only explicitly the part of the code that finds the best values for α_1 given some guess on α_2.

```
# ScatterSearchVectorised.py
import numpy as NP
import InputFunctions as IF
```

[a] In our context, **numpy.argmin** returns the indices of the minimum values along an axis in a NumPy array.

[b] This function returns arrays corresponding to indices that allow access to all values in an array.

```
def ScatterSearchVectorised(SearchX,SearchY,x,y,...):
  ####
  #inputs: vector x, vector y, vector SearchX, vector SearchY,
    other arguments
  #output: vector BestX, vector BestY
  ...
  #make 3D function call
  ValueArray = IF.Hamiltonian(SearchX,BestGuessY,x,y,...)
  #find indices of minima along alpha_1's axis
  ind1 = NP.argmin(ValueArray,axis=1)
  #generate indices required to access minimum values in
    ValueArray
  ind0,ind2 = NP.indices(ind1.shape)
  #store minima
  BestGuessX = SearchX[ind0,ind1,ind2]
  ...
  return BestGuessX,BestGuessY
```

The indices in **ind1** allow access to the minimum values of **ValueArray** along the relevant axis, provided we have all the indices to the other axes, which is provided by the call to **numpy.indices**. Note that we have assumed that the search values over α_1 is stored in the second axis of the array **ValueArray**. The code above is easily altered to deal with cases where this is the wrong axis. A detail that caused some confusion when implementing is that the line

```
BestGuessX = SearchX[ind0,ind1,ind2]
```

returns a 2D array[c].

E.2 Lessons learned on updating the search space

This is a continuation of the last section. Now that we have the current best guesses, we want to update the search domain with these values. The naive method for this is to simply use nested for-loops to iterate over each of the best guesses and generate a new search space around these. As for-loops generally slow down Python code when arrays become large, we spent time researching the use of Python **generators** to replace the nested for-loops that seem obvious to use. With some timing experiments it was discovered that this actually caused the code to run *slower*, and is a rare case in the implementation in which for-loops were the fastest option.

[c]For the 4D optimisation method, the corresponding line will return a 3D array, and the procedure in the code example will have to be repeated. This is messier, and given the lackluster performance of this method, we will spare the reader of the details of this inferior method.

We present and explain the "fancy generator" method below. We will use standard Python functions, as well as the NumPy function **numpy.linspace**[d].

```
# ScatterSearchVectorised.py
import numpy as NP
import InputFunctions as IF
def ScatterSearchVectorised(SearchX,SearchY,x,y,...):
  ####
  #inputs: vector x, vector y, vector SearchX, vector SearchY,
    other arguments
  #output: vector BestX, vector BestY
  ...
  #1st step: get list of indices for accessing values in the
    vectors x,y
  x_ind = range(SearchX.shape[0])
  y_ind = range(SearchX.shape[2])
  #2nd step: use generators to create lists of indices
  mylist = [(j,k) for j in x_ind for k in y_ind]
  mylistx = [x[0] for x in mylist] #all x-indices
  mylisty = [x[1] for x in mylist] #all y-indices
  #3rd step: update the search space
  SearchX[mylistx,:,mylisty] = NP.array([np.linspace(BestGuessX[
    j,k]-ax,BestGuessX[j,k]+ax,N[0]) for j in x_ind for k in
    y_ind])
  SearchY[myxlist,:,myylist] = NP.array([np.linspace(BestGuessY[
    j,k]-ay,BestGuessY[j,k]+ay,N[1]) for j in x_ind for k in
    y_ind])d
  ...
  return BestGuessX,BestGuessY
```

Note that **ax** is the current "search width" for the scatter search, and **Nx** is the number of points we evaluate for in each scatter evaluation. The list **mylist** contains all permutations of the indices in **x_ind** and **y_ind**. When running time profiling is used on the code above, the second step in particular is extremely fast. It is the third step, the use of the generated lists to update the search space, that is much slower than simply using nested for-loops.

There may be some break-even point using a very fine grid resolution where the above code is faster than using nested for-loops, but this has not been tested.

E.3 Quick matrix generation

For all computations on the HJB and FP equations in this thesis, it will be necessary to generate new computation matrices for each time step. Recall

[d]The function call **numpy.linspace(start,stop,N)**, returns **N** points evenly distributed between the values **start** and **stop**.

that our methods yield linear systems of the form

$$Ax = By,$$

where A represents the diffusion terms and B represents the convective terms. While cases in which the diffusion is constant allow us to only generate the diffusion matrix A only once, B will always be necessary to generate for each timestep. If one uses nonlinear methods like the one in section 4.2.2.3, quick matrix generation becomes especially important.

A naive implementation would involve a double for-loop in which elements in some empty matrix is altered, value by value. For high-level languages like Python, this is exceedingly slow. For speeding up this, we note that due to our Cartesian grid, the matrices will be sparse, banded matrices built by diagonal vectors. With some light work, these vectors can be generated, naively by using a single for-loop to iterate over each element and insert/add the respective value, and faster by simply generating them as the sums of vectors from vectorised function calls. To insert these vectors along the diagonals of a sparse matrix, one should use the SciPy function **scipy.sparse.diags**. See an example below:

```
# GenerateConvection.py
import scipy as SP
import numpy as NP
import InputFunctions as IF
def GenerateConvection(time,x,a,dt,dx):
  ####
  #inputs: scalar time, vector x, vector a (control), scalar dt,
     scalar dx
  #output: sparse convection matrix
  ####
  #make vectorised function calls
  sigma = IF.Sigma(time,x,a)
  vel = IF.Velocity(time,x,a)
  #generate the flux vectors
  FluxEast,FluxWest = IF.GenerateConvectionFlux(sigma,vel)
  #generate diagonal vectors
  zero = NP.zeros(x.size)
  east = -dt/dx*NP.minimum(FluxEast[1:],zero[1:])
  west = dt/dx*NP.maximum(FluxWest[:-1],zero[:-1])
  here = 1-dt/dx*(NP.maximum(FluxWest,zero) - NP.minimum(
    FluxEast,zero))
  #return sparse, diagonal matrix
  return SP.sparse.diags([here, east, west],[0, 1, -1])
```

The function **scipy.sparse.diags** takes two arguments, a list of vectors that will make up the diagonals, and another list that hold the offsets from the main diagonal on which to place the vectors. It is important that the lengths

of the vectors match the length of the diagonal the offset implies they are to be placed in.

Bibliography

[1] I. Aavatsmark. "Comparison of Monotonicity for some Multipoint Flux Approximation Methods".
In: *Finite Volumes for Complex Applications V*. 2008.

[2] I. Aavatsmark.
"Multipoint flux approximation methods for quadrilateral grids".
In: *9th International Forum on Reservoir Simulation, Abu Dhabi, 9-13 December 2007*. 2007.

[3] Y. Achdou. "Finite Difference Methods for Mean Field Games".
In: *Hamilton-Jacobi Equations: Approximations, Numerical Analysis and Applications*. Lecture Notes in Mathematics.
Springer Berlin Heidelberg, 2013, pp. 1–45.
URL: http://dx.doi.org/10.1007/978-3-642-36433-4_1.

[4] G. Barles and E. R. Jacobsen. "On the convergence rate of approximation schemes for Hamilton-Jacobi-Bellman equations".
In: *Mathematical Modelling and Numerical Analysis* 36.1 (2002), pp. 33–54.

[5] A. Berman and R. J. Plemmons.
Nonnegative Matrices in the Mathematical Sciences.
Academic Press, Inc., 1979.

[6] F. Camilli and F. J. Silva. "A semi-discrete in time approximation for a first order-finite mean field game problem".
In: *Networks and Heterogeneous Media* 7.2 (2012).

[7] P. Cardaliaguet. *Notes on Mean Field Games*. Available online at https://www.ceremade.dauphine.fr/c̃ardalia/ as of 12th July 2015. 2013.

[8] E. Carlini and F. J. Silva. "A Fully Discrete Semi-Lagrangian Scheme for a First Order Mean Field Game Problem".
In: *SIAM Journal on Numerical Analysis* 52.1 (2014), pp. 45–67.

[9] E. Carlini and F. J. Silva. "A Semi-Lagrangian scheme for a degenerate second order Mean Field Game system".
In: *Preprint* (2014). arXiv: 1404.5932v1 [hep-th].

[10] D. A. Gomes and J. Saúde.
"Mean Field Games Models - A Brief Survey".
In: *Dynamic Games and Applications* 4 (2 2014), pp. 110–154.

[11] O. Guéant. “Mean Field Games with a Quadratic Hamiltonian: A Specific Approach”.
In: *Mathematical Models and Methods in Applied Sciences* 22.9 (2012).

[12] O. Guéant. “New Numerical Methods for Mean Field Games with Quadratic Costs”. In: *Networks and Heterogeneous Media* 7.2 (2012).

[13] O. Guéant, J.-M. Lasry, and P.-L. Lions.
“Mean Field Games and Applications”.
In: *Paris-Princeton Lectures on Mathematical Finance 2010*.
Lecture Notes in Mathematics. Springer Berlin Heidelberg, 2011, pp. 205–266.
URL: http://dx.doi.org/10.1007/978-3-642-14660-2_3.

[14] A. Lachapelle, J. Salomon, and G. Turinici.
“Computation of mean field equilibria in economics”.
In: *Mathematical Models and Methods in Applied Sciences* 20.4 (2010).

[15] J.-M. Lasry and P.-L. Lions. “Mean field games”.
In: *Japan Journal of Mathematics* 2 (2007), pp. 229–260.

[16] R. J. LeVeque. *Finite-Volume Methods for Hyperbolic Problems*.
Cambridge University Press, 2002.

[17] P.-L. Lions. “Jeux à champ moyen. I. Le cas stationnaire.”
In: *Comptes Rendus Mathematique* 343 (9 2006), pp. 619–625.

[18] P.-L. Lions.
“Jeux à champ moyen. II. Horizon fini et contrôle optimal.”
In: *Comptes Rendus Mathematique* 343 (10 2006), 679–684.

[19] P.-L. Lions. *Lectures on mean field games*. Online lecture series. 2011.
URL: http://www.college-de-france.fr/site/en-pierre-louis-lions/course-2011-2012.htm.

[20] J. Salomon and G. Turinici.
“A monotonic method for solving nonlinear optimal control problems with concave dependence on the state”.
In: *International Journal of Control* 84.3 (2011), pp. 551–562.
URL: http://dx.doi.org/10.1080/00207179.2011.562548.

[21] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. 2nd ed.
Cambridge University Press, 1999.

[22] J. Yong and X. Y. Zhou.
Stochastic Controls: Hamiltonian Systems and HJB Equations.
Springer New York, 1999.

[23] G. Yuan and Z. Sheng. “An improved monotone finite volume schemes for diffusion equation on polygonal meshes”.
In: *Journal of Computational Physics* 231 (2012), pp. 3739–3754.

[24] G. Yuan and Z. Sheng. “Monotone finite volume schemes for diffusion equations on polygonal meshes”.
In: *Journal of Computational Physics* 227 (2008), pp. 6288–6312.

[25] R. E. et al. *Finite Volume Methods*. Online notes. Available online at http://www.cmi.univ-mrs.fr/ herbin/PUBLI/bookevol.pdf as of 12th July 2015. 2003.
URL: `http://www.cmi.univ-mrs.fr/~herbin/PUBLI/bookevol.pdf`.