NETWORKS AND HETEROGENEOUS MEDIA
©American Institute of Mathematical Sciences
Volume **7**, Number **2**, June **2012**
doi:10.3934/nhm.2012.7.197
pp. **197–217**

ITERATIVE STRATEGIES FOR SOLVING LINEARIZED DISCRETE MEAN FIELD GAMES SYSTEMS

Yves Achdou and Victor Perez

Université Paris Diderot, UMR 7598, Laboratoire Jacques-Louis Lions
Paris, France

Abstract. Mean fields games (MFG) describe the asymptotic behavior of stochastic differential games in which the number of players tends to $+\infty$. Under suitable assumptions, they lead to a new kind of system of two partial differential equations: a forward Bellman equation coupled with a backward Fokker-Planck equation. In earlier articles, finite difference schemes preserving the structure of the system have been proposed and studied. They lead to large systems of nonlinear equations in finite dimension. A possible way of numerically solving the latter is to use inexact Newton methods: a Newton step consists of solving a linearized discrete MFG system. The forward-backward character of the MFG system makes it impossible to use time marching methods. In the present work, we propose three families of iterative strategies for solving the linearized discrete MFG systems, most of which involve suitable multigrid solvers or preconditioners.

1. **Introduction.** Mean field type models describing the asymptotic behavior of stochastic differential games (Nash equilibria) as the number of players tends to $+\infty$ have recently been introduced by J-M. Lasry and P-L. Lions [12, 13, 14]. In the periodic setting, a typical such model comprises the following system of evolution partial differential equations for the unknown scalar functions $u = u(t,x)$ and $m = m(t,x)$

$$\frac{\partial u}{\partial t}(t,x) - \nu\Delta u(t,x) + H(x,\nabla u(t,x)) = \Phi(m(t,x)), \qquad \text{in } (0,T)\times\mathbb{T}^d, \quad (1)$$

$$\frac{\partial m}{\partial t}(t,x) + \nu\Delta m(t,x) + \operatorname{div}\left(m\frac{\partial H}{\partial p}(x,\nabla u(t,x))\right) = 0, \qquad \text{in } (0,T)\times\mathbb{T}^d, \quad (2)$$

with the initial and terminal conditions

$$u(0,x) = u_0(x), \quad m(T,x) = m_T(x), \quad \text{in } \mathbb{T}^d, \quad (3)$$

given a cost function u_0 and a probability density m_T. The system (1)-(3) implies that

$$\int_{\mathbb{T}^d} m(t,x)dx = 1, \quad m > 0\,. \quad (4)$$

We denote by $\mathbb{T}^d = [0,1]^d$ the $d-$dimensional unit torus, by ν a nonnegative constant and by Δ, ∇ and div, respectively, the Laplace, the gradient and the divergence operator acting on the x variable. The other operators involved in the system

2000 *Mathematics Subject Classification.* Primary: 91-08, 91A23, 65N22; Secondary: 65M06, 65F10.

Key words and phrases. Mean field games, numerical methods, iterative solvers, multigrid methods.

are the scalar Hamiltonian $H(x,p)$, $\mathcal{C}^1$ regular w.r.t. x and p and convex in the second variable, and Φ is a $\mathcal{C}^1$ function from $\mathbb{R}_+$ to $\mathbb{R}$. The notation $\frac{\partial H}{\partial p}(x,q)$ is used for the gradient of $p \mapsto H(x,p)$ at $p=q$.

System (1)- (2) consists of a forward Bellman equation coupled with a backward Fokker-Planck equation. The forward-backward structure is an important feature of this system, which makes it necessary to design new strategies for its mathematical analysis (see [13, 14]) and for numerical approximation.

Remark 1. In some relevant situations, and for the justification of the model and the mathematical analysis of the system (1)- (4), it is possible to replace the right hand side of (1) by $V[m(\cdot,t)](x)$ where the operator V maps a probability density μ on $\mathbb{T}^d$ to a real valued function $V[\mu]$ defined on $\mathbb{T}^d$. It is often helpful to assume that V is a nonlocal regularizing operator. Although it is important, this case will not be discussed in detail in what follows, and we will limit ourselves to saying if the methods proposed below can be used or not.

Similarly, we have chosen to focus on the case when the cost $u_{|t=0}$ depends directly on x. In some realistic situations, $u_{|t=0} = V_0[m_{|t=0}](x)$, where V_0 is an operator acting on probability densities. This case can be treated by the methods presented below, at least when V_0 is a local operator, i.e. $V_0[\mu](x) = \Phi_0(\mu(x))$, but we will not discuss it in the present work.

Examples of MFG models with applications in economics and social sciences are proposed in [9]. Many important aspects of the mathematical theory developed by J-M. Lasry and P-L. Lions on MFG are not published in journals or books, but can be found in the videos of the lectures of P-L. Lions at Collège de France: see the web site of Collège de France, [15].

An important research activity is currently going on about approximation procedures of different types of mean field games models, see [11] for a numerical method based on the reformulation of the model as an optimal control problem for the Fokker-Planck equation with an application in economics and [7] for a work on discrete time, finite state space mean field games. We also refer to [8] for a specific constructive approach when the Hamiltonian is quadratic.

In [3] and [2], the authors have proposed and studied finite difference methods basically relying on monotone approximations of the Hamiltonian and on a suitable weak formulation of the Fokker-Planck equation, both for infinite horizon mean field games (which lead to a stationary system of PDEs) and for finite horizon mean field games (leading to system (1)-(4)).

Inexact Newton iterations may be used for solving the system of nonlinear equations arising from the discretization of (1)-(4). They require solving the linearized discrete MFG system. The present work is devoted to preconditioned iterative methods for solving the discrete linearized MFG systems of equations. We propose and compare three different strategies:

1. Eliminate u by solving a Cauchy problem with the linearized discrete HJB equation: the resulting system of linear equations (whose unknowns correspond to m only and whose matrix corresponds to a nonlocal operator) is solved by a preconditioned iterative method (BiCGstab in our implementation). The elimination of u and the preconditioning can be either done exactly with a direct solver (Algorithm A below) or iteratively by means of multigrid schemes (Algorithm B below).

2. Eliminate m by using the linearized discrete HJB equation: the resulting system (whose unknowns correspond to u only) is solved by a preconditioned iterative method (BiCGstab in our implementation). It corresponds to a boundary value problem in the domain $[0,T]\times\mathbb{T}^d$ involving a degenerate elliptic operator which is fourth order w.r.t. x and second order w.r.t. t. Applying the preconditioner consists of performing one step of a carefully chosen multigrid method. We will see that the choice of the multigrid scheme is delicate because of the anisotropy (and degeneracy) of the above mentioned operator, but that a good method can indeed be obtained. This method is termed Algorithm C below.
3. Apply a preconditioned iterative method (BiCGstab in our implementation) directly to the full system. Applying the preconditioner consists of performing one step of a multigrid method, whose design requires a good understanding of the system of PDEs, as for Algorithm C. This method is termed Algorithm D below.

We will see that the three strategies lead to rather efficient algorithms at least when ν is not too small. Of course, the complexity of the system of PDEs (1)-(4) makes it difficult to carry out a rigorous analysis of the multigrid schemes. Hence, we will not try to fully analyze the multigrid methods, but we will rather justify our choices and explain the observed behaviors by drawing relations with simpler problems already studied in the literature. The work is organized as follows: the finite difference schemes and the Newton method are presented in § 2. The first strategy and the related algorithms are discussed in § 3. Section 4 is devoted to the second strategy. The third strategy is addressed in § 5. Comparison of computing times is the topic of Section 6 and some conclusions are drawn in Section 7.

2. **Finite difference schemes.**

2.1. **Schemes preserving the structure of (1)-(4).**

For simplicity of notations, we will always consider the case $d=2$ although our approach and results hold for general d (in practice $d\leq 4$ because of computer memory limitations).

Let $\mathbb{T}^2_h$ be a uniform grid on the two-dimensional torus with mesh step h (assuming that $1/h$ is an integer N) and denote by x_{ij} a typical point in $\mathbb{T}^2_h$. Let N_T be a positive integer and $\Delta t = T/N_T$, $t_n = n\Delta t$, $n=0,\dots,N_T$. The values of u and m at $(x_{i,j},t_n)$ are approximated, respectively by $U^n_{i,j}$ and $M^n_{i,j}$. Note that the choice of a single grid step h for the two spatial directions is made only for simplicity.

We first discuss the approximations of the nonlinear operators in (1).
We introduce the finite difference operators

$$(D_1^+U)_{i,j}=\frac{U_{i+1,j}-U_{i,j}}{h}\quad\text{and}\quad(D_2^+U)_{i,j}=\frac{U_{i,j+1}-U_{i,j}}{h},\tag{5}$$

and define

$$[D_hU]_{i,j}=\big((D_1^+U)_{i,j},(D_1^+U)_{i-1,j},(D_2^+U)_{i,j},(D_2^+U)_{i,j-1}\big)^T\in\mathbb{R}^4,\tag{6}$$

$$(\Delta_hU)_{i,j}=-\frac{1}{h^2}(4U_{i,j}-U_{i+1,j}-U_{i-1,j}-U_{i,j+1}-U_{i,j-1}).\tag{7}$$

In order to approximate the Hamiltonian H in equation (1), we consider a numerical Hamiltonian $g:\mathbb{T}^2\times\mathbb{R}^4\to\mathbb{R}$, $(x,q_1,q_2,q_3,q_4)\mapsto g(x,q_1,q_2,q_3,q_4)$ satisfying the following conditions:

($\mathbf{G_1}$) *monotonicity*: g is nonincreasing with respect to q_1 and q_3 and nondecreasing with respect to q_2 and q_4.

($\mathbf{G_2}$) *consistency:* $g\,(x, q_1, q_1, q_2, q_2) = H(x, q), \quad \forall x \in \mathbb{T}^2, \forall q = (q_1, q_2) \in \mathbb{R}^2$.

($\mathbf{G_3}$) *differentiability*: g is of class $\mathcal{C}^1$.

($\mathbf{G_4}$) *convexity* : $(q_1, q_2, q_3, q_4) \mapsto g\,(x, q_1, q_2, q_3, q_4)$ is convex.

We will often make the following assumption on Φ:

($\mathbf{\Phi_1}$) The function Φ is monotone, strictly increasing.

Standard examples of numerical Hamiltonians fulfilling these requirements are provided by Lax-Friedrichs or Godunov type schemes, see [3].

The approximation of equation (1) is given by the semi-implicit scheme:

$$\frac{U_{i,j}^{n+1} - U_{i,j}^{n}}{\Delta t} - \nu(\Delta_h U^{n+1})_{i,j} + g(x_{i,j}, [D_h U^{n+1}]_{i,j}) = \Phi(M_{i,j}^{n}). \tag{8}$$

In order to approximate equation (2), it is convenient to consider its weak formulation which involves in particular the term

$$\int_{\mathbb{T}^2} \operatorname{div}\left(m \frac{\partial H}{\partial p}(x, \nabla u)\right) w\, dx$$

which by periodicity turns out be equal to

$$-\int_{\mathbb{T}^2} m \frac{\partial H}{\partial p}(x, \nabla u) \cdot \nabla w\, dx$$

for any test function w. This term will be approximated by

$$-h^2 \sum_{i,j} M_{i,j} \nabla_q g(x_{i,j}, [D_h U]_{i,j}) \cdot [D_h W]_{i,j} = h^2 \sum_{i,j} \mathcal{T}_{i,j}(U, M) W_{i,j},$$

where the transport operator $\mathcal{T}$ is defined as follows:

$$\begin{aligned} h\mathcal{T}_{i,j}(U, M) = \; & M_{i,j} \frac{\partial g}{\partial q_1}(x_{i,j}, [D_h U]_{i,j}) - M_{i-1,j} \frac{\partial g}{\partial q_1}(x_{i-1,j}, [D_h U]_{i-1,j}) \\ & + M_{i+1,j} \frac{\partial g}{\partial q_2}(x_{i+1,j}, [D_h U]_{i+1,j}) - M_{i,j} \frac{\partial g}{\partial q_2}(x_{i,j}, [D_h U]_{i,j}) \\ & + M_{i,j} \frac{\partial g}{\partial q_3}(x_{i,j}, [D_h U]_{i,j}) - M_{i,j-1} \frac{\partial g}{\partial q_3}(x_{i,j-1}, [D_h U]_{i,j-1}) \\ & + M_{i,j+1} \frac{\partial g}{\partial q_4}(x_{i,j+1}, [D_h U]_{i,j+1}) - M_{i,j} \frac{\partial g}{\partial q_4}(x_{i,j}, [D_h U]_{i,j}). \end{aligned} \tag{9}$$

The discrete version of equation (2) is chosen as the following scheme:

$$\frac{M_{i,j}^{n+1} - M_{i,j}^{n}}{\Delta t} + \nu(\Delta_h M^n)_{i,j} + \mathcal{T}_{i,j}(U^{n+1}, M^n) = 0. \tag{10}$$

Remark 2. It is important to realize that the operator $M \mapsto -\nu(\Delta_h M)_{i,j} - \mathcal{T}_{i,j}(U, M)$ is the adjoint of the linearization of the operator $U \mapsto -\nu(\Delta_h U)_{i,j} + g(x_{i,j}, [D_h U]_{i,j})$.

Finally, we introduce the compact and convex set

$$\mathcal{K} = \{(M_{i,j})_{0 \le i,j < N} : h^2 \sum_{i,j} M_{i,j} = 1, M_{i,j} \ge 0\} \tag{11}$$

which can be viewed as the set of the discrete probability measures.

The fully discrete scheme for system (1),(2),(3),(4) is therefore the following:

$$\begin{cases} \frac{U_{i,j}^{n+1}-U_{i,j}^{n}}{\Delta t} - \nu(\Delta_h U^{n+1})_{i,j} + g(x_{i,j}, [D_h U^{n+1}]_{i,j}) = & \Phi(M_{i,j}^n), \\ \frac{M_{i,j}^{n+1}-M_{i,j}^{n}}{\Delta t} + \nu(\Delta_h M^{n})_{i,j} + \mathcal{T}_{i,j}(U^{n+1}, M^n) = 0, & 0 \le n < N_T, \\ & 0 \le i,j < N, \\ M^n \in \mathcal{K}, & n = 0, \dots N_T, \\ M_{i,j}^{N_T} = (m_T)_{i,j}, \qquad u_{i,j}^0 = u_0(x_{i,j}), & 0 \le i,j < N, \end{cases} \tag{12}$$

where $(m_T)_{i,j} = \frac{1}{h^2}\int_{|x-x_{i,j}|_\infty \le h/2} m_T$.

Under assumptions ($\mathbf{G_1}$)-($\mathbf{G_4}$) existence for (12) has been proved in [3], using Brouwer fixed point theorem. Note that this existence proof is not constructive, i.e. it does not rely on an algorithm for solving (12).

Under the additional assumption ($\mathbf{\Phi_1}$), uniqueness holds for (12). This can be checked by using the fact that the discrete problem has exactly the same structure as the continuous one, so the proofs proposed in [13, 14] can be used.

Remark 3. Note that the fact that $M^n \in \mathcal{K}$, $n = 0, \dots N_T - 1$, is a direct consequence of (10) and $M^{N_T} \in \mathcal{K}$, so this constraint can be removed from (12).

2.2. **Newton methods for solving (12).** Hereafter, we assume that $p \mapsto H(x,p)$ and $q \mapsto g(x,q)$ are C^2 regular. This will allow us to use Newton like algorithms for (9).

System (12) can be seen as a forward discrete HJB equation for U with a Cauchy condition at $t = 0$ coupled with a backward discrete Fokker-Planck equation for M with a Cauchy condition at final time. This structure prohibits the use of a straightforward time-marching solution procedure.

Call $\mathcal{U}$ and $\mathcal{M}$ the vectors of $\mathbb{R}^{N^T N^2}$ such that $\mathcal{U}_{kN^2+iN+j} = U_{i,j}^k$ and $\mathcal{M}_{kN^2+iN+j} = M_{i,j}^{k-1}$. (recall that U^0 and M^{N_T} are given). The system of nonlinear equations can be written

$$\mathcal{F}_U(\mathcal{U}, \mathcal{M}) = 0, \quad \text{and} \quad \mathcal{F}_M(\mathcal{U}, \mathcal{M}) = 0, \tag{13}$$

with

- $\mathcal{F}_U(\mathcal{U}, \mathcal{M}) = 0 \Leftrightarrow$ (8) $\quad \forall n,\ 0 \le n < N_T,\ \forall i,j.$
- $\mathcal{F}_M(\mathcal{U}, \mathcal{M}) = 0 \Leftrightarrow$ (10) $\quad \forall n,\ 0 \le n < N_T,\ \forall i,j.$

In order to discuss the Newton method for solving (13), we use the following notation

$$\begin{array}{ll} A_{U,U}(\mathcal{U}, \mathcal{M}) = D_{\mathcal{U}}\mathcal{F}_{\mathcal{U}}(\mathcal{U}, \mathcal{M}), & A_{U,M}(\mathcal{U}, \mathcal{M}) = D_{\mathcal{M}}\mathcal{F}_{\mathcal{U}}(\mathcal{U}, \mathcal{M}), \\ A_{M,U}(\mathcal{U}, \mathcal{M}) = D_{\mathcal{U}}\mathcal{F}_{\mathcal{M}}(\mathcal{U}, \mathcal{M}), & A_{M,M}(\mathcal{U}, \mathcal{M}) = D_{\mathcal{M}}\mathcal{F}_{\mathcal{M}}(\mathcal{U}, \mathcal{M}). \end{array} \tag{14}$$

The matrices $A_{UU}(\mathcal{U}, \mathcal{M})$ and $A_{UM}(\mathcal{U}, \mathcal{M})$ have the form

$$A_{UU} = \begin{pmatrix} D_1 & 0 & \dots & \dots & 0 \\ -\frac{1}{\Delta t} I & D_2 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -\frac{1}{\Delta t} I & D_{N_T} \end{pmatrix}, A_{UM} = \begin{pmatrix} E_1 & 0 & \dots & \dots & 0 \\ 0 & E_2 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & & 0 & E_{N_T} \end{pmatrix}. \tag{15}$$

The blocks of $A_{UU}(\mathcal{U}, \mathcal{M})$ are sparse. The block D_n corresponds to the discrete operator $(Z_{i,j}) \mapsto \left(\frac{1}{\Delta t} Z_{i,j} - \nu(\Delta_h Z)_{i,j} + [D_h Z]_{i,j} \cdot \nabla g(x_{i,j}, [D_h U^n]_{i,j})\right)$ coming from the linearization of the discrete Bellman equation. From the assumptions ($\mathbf{G_1}$) and ($\mathbf{G_3}$) on g, D_n is a M-matrix, thus A_{UU} is invertible.

The blocks of $A_{UM}(\mathcal{U},\mathcal{M})$ are diagonal matrices, (note that they would be dense matrices if $\Phi(m(t,x))$ was replaced by $V[m(t,\cdot)](x)$ for V a nonlocal operator). Under Assumption $(\mathbf{\Phi_1})$, $\Phi' > 0$ so the diagonal entries of $A_{UM}(\mathcal{U},\mathcal{M})$ are negative.

From Remark 2, the matrices $A_{MM}(\mathcal{U},\mathcal{M})$ and $A_{MU}(\mathcal{U},\mathcal{M})$ have the form

$$A_{MM} = A_{UU}^T, \qquad A_{MU} = \begin{pmatrix} \widetilde{E}_1 & 0 & \dots & \dots & 0 \\ 0 & \widetilde{E}_2 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \widetilde{E}_{N_T-1} & 0 \\ 0 & \dots & \dots & 0 & \widetilde{E}_{N_T} \end{pmatrix}. \tag{16}$$

The block A_{MM} corresponds to a discrete linear transport equation. Note that

$$\mathcal{V}^T \widetilde{E}_n \mathcal{W} = \sum_{i,j} M_{i,j}^{n-1} [D_h V]_{i,j} \cdot D^2_{q,q} g(x_{i,j}, [D_h U^n]_{i,j}) [D_h W]_{i,j}.$$

From the convexity of g, we see that the block $\widetilde{E}_n$ is symmetric and positive semi-definite if M^{n-1} is a nonnegative grid function.

In [2], it is proved that under Assumptions $(\mathbf{G_1})$-$(\mathbf{G_4})$ and $(\mathbf{\Phi_1})$, and if the iterate produced by the Newton method satisfies $\mathcal{M} \geq 0$, then the Jacobian matrix

$$\begin{pmatrix} A_{U,U} & A_{U,M} \\ A_{M,U} & A_{M,M} \end{pmatrix}$$

is invertible. The proof is similar to that used for the uniqueness of the solution of (9). The positivity of $\mathcal{M}$ is not guaranteed though, but if the initial guess is close enough to a solution $(\widehat{\mathcal{U}},\widehat{\mathcal{M}})$ with $\widehat{\mathcal{M}} > 0$, then the iterates $\mathcal{M}$ will stay positive.

Assuming the invertibility of the matrix, the most time consuming part of the procedure lies in solving the system of linear equations

$$\begin{pmatrix} A_{U,U} & A_{U,M} \\ A_{M,U} & A_{M,M} \end{pmatrix} \begin{pmatrix} \mathcal{U} \\ \mathcal{M} \end{pmatrix} = \begin{pmatrix} G_U \\ G_M \end{pmatrix}. \tag{17}$$

In the sequel, we propose iterative strategies for solving (17).

As explained above, the Newton method described above may break down if in the Newton loop, the approximation of m_h takes negative values. A similar phenomenon was observed by Benamou, Brenier and Guittet [4, 5] when they studied a somewhat similar but simpler penalty method (using conjugate gradient iterations instead of Newton) for computing a mixed L^2-Wasserstein distance between two probability densities. This is of course a drawback of the method. However, breakdown does not happen if the initial guess is close enough to a solution. Therefore, it is important to find good initial guesses for the Newton method.

A possible way of avoiding breakdowns is to start solving (12) with a rather high value of the parameter ν (of the order of 1), then gradually decrease ν down to the desired value, the solution of (12) found by the Newton method for a given value of ν being used as an initial guess for the next and smaller value of ν. Doing so in our tests, we have avoided breakdowns of the Newton method. We have generally taken ν between 1 and 0.1: the number of iterations of the Newton method to achieve that the ℓ_2 norm of the residual be smaller than 10^{-5} was found to be less than 10 and to increase as ν decreases. In our experiments, it has not been necessary to use very high values of ν, for which an asymptotic analysis remains to be done (except

for t close to 0 and T, the functions $x \mapsto u(t,x)$ and $x \mapsto m(t,x)$ should be close to constant as $\nu \to \infty$).

Although this is not the topic of the present work, we think that it would be interesting to better understand nonlinear iterative algorithms for (12).

2.3. **Our numerical tests.** Hereafter, all the proposed methods will be tested for different values of ν, in the following case:

$$T = 1, \tag{18}$$

$$H(x,p) = \sin(2\pi x_1) + \sin(2\pi x_2) + \cos(4\pi x_1) + |p|^3, \tag{19}$$

$$\Phi(m) = m, \tag{20}$$

$$u_0(x) = 0, \tag{21}$$

$$m_T(x) = 1, \tag{22}$$

and g corresponds to a classical Godunov scheme, see [3].

For what follows it interesting to plot the contours of m at time $t = T/2 = 0.5$: on Figure 1, we display the contours of m for $\nu = 0.6$ and $\nu = 0.08$. Note that for $\nu = 0.08$, i.e. rather close to the deterministic case $\nu = 0$, $m(x)$ is small (smaller than 0.01) in a large region.

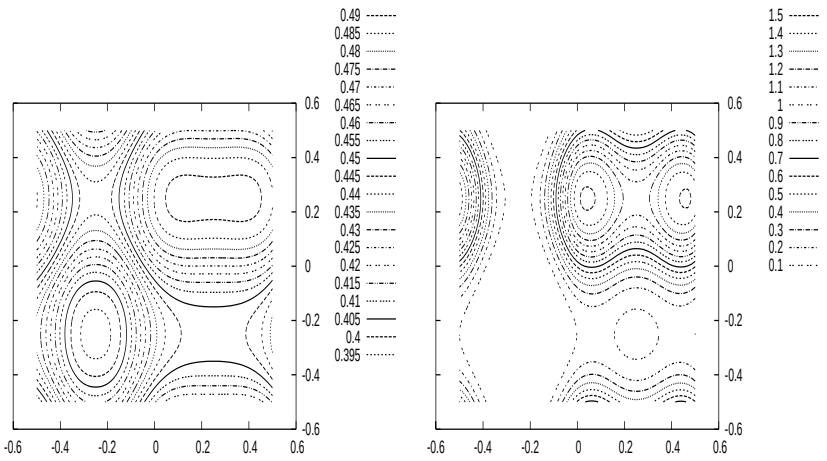

FIGURE 1. Contours of m for $\nu = 0.6$ (left) and $\nu = 0.08$ (right) at time $t = 0.5$

3. Iterative strategies for solving (17) based on eliminating $\mathcal{U}$.

3.1. The most basic iterative method.

The principle of the method is as follows:

1. first solve
$$A_{U,U}\widetilde{\mathcal{U}} = G_U. \tag{23}$$
This is done by sequentially solving
$$D_1\widetilde{U}^1 = G_U^1, \tag{24}$$
then
$$D_k\widetilde{U}^k = \frac{1}{\Delta t}\widetilde{U}^{k-1} + G_U^k, \quad \text{for } k > 1, \tag{25}$$
i.e. marching in time in the forward direction. We know that (24) and (25) have a unique solution if g satisfies assumptions ($\mathbf{G_1}$) and and ($\mathbf{G_3}$).
2. Introducing $\overline{\mathcal{U}} = \mathcal{U} - \widetilde{\mathcal{U}}$, the vector $(\overline{\mathcal{U}}, \mathcal{M})^T$ satisfies
$$\begin{pmatrix} A_{U,U} & A_{U,M} \\ A_{M,U} & A_{M,M} \end{pmatrix} \begin{pmatrix} \overline{\mathcal{U}} \\ \mathcal{M} \end{pmatrix} = \begin{pmatrix} 0 \\ G_M - A_{M,U}\widetilde{\mathcal{U}} \end{pmatrix}, \tag{26}$$
which implies
$$\left(A_{M,M} - A_{M,U}A_{U,U}^{-1}A_{U,M}\right)\mathcal{M} = G_M - A_{M,U}\widetilde{\mathcal{U}}. \tag{27}$$
3. Once (27) is solved, $\overline{\mathcal{U}}$ is obtained by solving the discrete forward linearized HJB equation
$$A_{U,U}\overline{\mathcal{U}} = -A_{U,M}\mathcal{M} \tag{28}$$
by the same method as for (24), (23).

The system (27) is solved by means of an iterative method, for example, the BiCGstab algorithm [17] in all what follows; it only requires an implementation of the matrix-vector product with the matrix $A_{M,M} - A_{M,U}A_{U,U}^{-1}A_{U,M}$. Of course, this matrix is not assembled: the matrix-vector product involves matrix-vector products with the matrices $A_{M,M}$, $A_{M,U}$ and $A_{U,M}$ and solving a linear system of the form (23), similar to that appearing in the first step.

Numerical tests not reported here show that, with the previously described iterative method, the number of iterations to reduce the error by a fixed factor increases as the size of the mesh grows; this can also be foreseen by using arguments similar to those in § 3.2.1 below: hence it is desirable to modify this basic method by using a suitable preconditioner.

3.2. Preconditioned iterative methods.

We propose to use A_{MM} as a preconditioner for (27): it amounts to applying an iterative algorithm (i.e. the BiCGstab algorithm) to
$$\left(I - A_{M,M}^{-1}A_{M,U}A_{U,U}^{-1}A_{U,M}\right)\mathcal{M} = A_{M,M}^{-1}(G_M - A_{M,U}\widetilde{\mathcal{U}}) \tag{29}$$
rather than to (27).

3.2.1. *A heuristic interpretation in terms of partial differential operators.* A heuristic explanation for this preconditioner choice is as follows: calling v and n two functions on $\mathbb{T}^2$,

- A_{UU} is the matrix counterpart of the linearized Bellman operator (advection-diffusion operator):
$$v \mapsto \text{ Lin-HJB}(v) := \tfrac{\partial v}{\partial t} - \nu\Delta v + \tfrac{\partial H}{\partial p}(x, \nabla u)\cdot\nabla v$$
- $A_{U,M}$ is the matrix counterpart of the operator: $n \mapsto -\Phi'(m)n$
- A_{MM} is the matrix counterpart of the Fokker-Planck operator (transport-diffusion operator):
$$n \mapsto \text{ FP}(n) := -\tfrac{\partial n}{\partial t} - \nu\Delta n - \operatorname{div}(n\tfrac{\partial H}{\partial p}(x, \nabla u))$$
- $A_{M,U}$ is the matrix counterpart of the operator: $v \mapsto -\operatorname{div}\left(mH_{pp}(x, \nabla u)\nabla v\right)$, where $H_{pp}(x, q)$ stands for the Hessian of $p \mapsto H(x, p)$ at $p = q$.

Let us define $\text{ Lin-HJB}^{-1}(w)$ as the unique solution v of the Cauchy problem involving the linearized Bellman equation:

$$\begin{aligned} \frac{\partial v}{\partial t} - \nu\Delta v + \frac{\partial H}{\partial p}(x, \nabla u)\cdot\nabla v = w \quad \text{in } (0,T]\times\mathbb{T}^2, \\ v|_{t=0} = 0 \quad \text{in } \mathbb{T}^2, \end{aligned} \tag{30}$$

and $\text{FP}^{-1}(r)$ as the unique solution n of the backward Cauchy problem involving the Fokker-Planck equation:

$$\begin{aligned} \frac{\partial n}{\partial t} + \nu\Delta n + \operatorname{div}(n\frac{\partial H}{\partial p}(x, \nabla u)) = -r \quad \text{in } (0,T]\times\mathbb{T}^2, \\ n|_{t=T} = 0 \quad \text{in } \mathbb{T}^2, \end{aligned} \tag{31}$$

The matrix $-A_{M,M}^{-1}A_{M,U}A_{U,U}^{-1}A_{U,M}$ is the counterpart of the nonlocal operator:

$$n \mapsto \Big(\text{ FP}^{-1} \circ \big(v \mapsto -\operatorname{div}\left(mH_{pp}(x, \nabla u)\nabla v\right)\big) \circ \text{ Lin-HJB}^{-1}\Big)\big(\Phi'(m)n\big).$$

Now, assuming that u and m belong to $C^{1+\alpha/2,2+\alpha}([0,T]\times\mathbb{T}^2)$, it can be shown that the latter operator maps continuously $C^{\alpha/2,\alpha}([0,T]\times\mathbb{T}^2)$ to $C^{1+\alpha/2,2+\alpha}([0,T]\times\mathbb{T}^2)$, so it is a compact operator on $C^{\alpha/2,\alpha}([0,T]\times\mathbb{T}^2)$. Compactness in $L^2((0,T)\times\mathbb{T}^2)$ is also true. Hence $I - A_{M,M}^{-1}A_{M,U}A_{U,U}^{-1}A_{U,M}$ is the discrete version of the perturbation of the identity by a compact operator. Therefore, the convergence of the BiCGstab algorithm should not depend on the size the grid. This will be confirmed by the numerical experiments below.

The PDE interpretation of the preconditioner also leads us to predict that the number of iterations needed by the iterative solver should increase as ν decreases to zero, which will indeed appear clearly in the tests.

3.2.2. *Algorithm A.* The matrix $A_{M,M}^{-1}A_{M,U}A_{U,U}^{-1}A_{U,M}$ is not assembled. The proposed method only requires an implementation of the matrix-vector product with the matrix $A_{M,M} - A_{M,U}A_{U,U}^{-1}A_{U,M}$ as discussed above (it does not need the matrix $A_{U,U}^{-1}$), and solving systems of linear equations of the form

$$A_{M,M}\widetilde{\mathcal{M}} = G_M. \tag{32}$$

This is done by sequentially solving

$$D_{N_T}^T\widetilde{M}^{N_T-1} = G_M^{N_T}, \tag{33}$$

then

$$D_k^T \widetilde{M}^{k-1} = -\frac{1}{\Delta t}\widetilde{M}^k + G_M^k, \quad \text{for } 1 \leq k < N_T, \tag{34}$$

i.e. marching in time in the backward direction. It has already been seen that the blocks D_k are invertible, and so are the blocks D_k^T.

Note that an iteration of the preconditioned BiCGstab method involves two solves of systems of the type (23) and two solves of systems of the type (32).
Solving the systems (24), (25), (33), (34) (two dimensional problems) can be done with fast direct solvers: in our implementation, we have used the open source library UMFPACK [1] which contains an Unsymmetric MultiFrontal method for solving linear systems.
Hereafter, we shall refer to the above method as Algorithm A.

In Table 1, we show the number of iterations needed to decrease the residual norm by a factor 10^{-3} or 10^{-7} with the preconditioned BiCGstab method. In our tests, choosing an error reduction of 10^{-3} instead of 10^{-7} had no effect on the convergence of the inexact Newton method. For this reason, we shall hereafter make all our tests with an error reduction threshold of 10^{-3} in the preconditioned BiCGstab methods.

We see that, as expected, the number of BiCGstab iterations is small and does not depend on the size of the grid, and that it increases as ν decreases.

TABLE 1. Algorithm A for solving the linearized MFG system: average (on the Newton loop) number of iterations of BiCGstab to decrease the residual by a factor 10^{-3} or 10^{-7}

grid	$32 \times 32 \times 32$		$64 \times 64 \times 64$		$128 \times 128 \times 64$	
rel. accur.	10^{-3}	10^{-7}	10^{-3}	10^{-7}	10^{-3}	10^{-7}
$\nu = 0.6$	1	2	1	2	1	2
$\nu = 0.36$	1.75	2	1.75	2	1.8	2
$\nu = 0.2$	2	3.5	2	3.5	2	4
$\nu = 0.12$	3	6	3	6	3	6.1
$\nu = 0.046$	4.9	10	5.1	10	5.1	10

Remark 4. It is possible to use this family of algorithms when the right hand side of (1) is of the form $V[m(t,\cdot)](x)$, at least when V is a monotone operator in the following sense: $\langle \mu_1 - \mu_2, V[\mu_1] - V[\mu_2] \rangle \geq 0$, for all probability measures μ_1 and μ_2 on $\mathbb{T}^2$.

Remark 5. We have used this family of algorithms with success in some cases when the monotonicity assumption on Φ is not fulfilled, for example $\Phi(m) = -\log(m)$.

3.2.3. *Algorithm B.* Algorithm A works quite well but it relies on the use of a very efficient direct solver: this has two consequences:

- the complexity of Algorithm A is superlinear in the number of degrees of freedom, even with the best direct solvers
- More important, it seems difficult to use Algorithm A for $d = 3$.

The method described in this paragraph differs from Algorithm A in the fact that the systems (24), (25), (33), (34) are now solved by means of suitable multigrid methods. Multigrid methods are iterative, so the previously mentioned systems are solved approximately only (the multigrid iterations stop when the norm of the

residual is below a given threshold ϵ). Therefore, the algorithm for solving the linear problem (17) now involves nested iterations, and a small value of ϵ is needed for the leading iterative method (BiCGstab) to converge.

Let us briefly discuss the multigrid treatment of the system

$$BU = F, \tag{35}$$

where B is any of the matrices D_k or D_k^T (more generally, (35) may be a linear system arising from the discretization of a partial differential equation by a finite difference method). Note that the matrices D_k are diagonal dominant.
We refer to [6], [16] and references therein for thorough introductions to multigrid computational methods.

The basic idea of multigrid is that standard stationary iterative methods such as Jacobi or Gauss-Seidel are often successful in damping oscillatory harmonics of the initial residual, but they are inefficient in removing the smooth modes. Such methods take a small number of iterations to make the residual smooth. For this reason, stationary iterative techniques are called *smoothers*.

Multigrid methods combine the smoothing property of the stationary iterative techniques with a suitable coarsening technique: a two grids algorithm consists of first making a few smoothing iterations (η_1 smoothing iterations), then performing a coarse grid correction (by solving the discrete PDE on the coarser mesh, the right hand side being the residual approximated on the coarser mesh, and interpolating the solution back onto the fine grid), then applying again a few smoothing iterations (η_2 smoothing iterations).

If there are more than two nested grids, one can design recursive algorithms by realizing that smooth grid functions on a given grid generally look oscillatory on a coarsened grid. The principle is to replace the coarse grid correction by one or several recursive calls of the algorithm at the coarser level.

Basic multigrid scheme are the V-cycle and W-cycle in which the coarse grid correction consists of one or two recursive applications of the multigrid scheme (i.e. the V-cycle (resp W-cycle) corresponds to the choice $\gamma = 1$ (resp. $\gamma = 2$) in the following algorithm, whereas $\gamma = -1$ yields the so-called F-cycle, which is intermediate between the V-cycle and the W-cycle):

function Multigrid-Scheme$(\ell, U_\ell, F_\ell, \gamma)$
  if $\ell = 0$ **then**
    $U_0 \leftarrow B_0^{-1} F_0$
  else
    for $g = 1, \ldots, |\gamma|$ **do**
      $U_\ell \leftarrow S_\ell(U_\ell, F_\ell, \eta_1)$ (η_1 presmoothing iterations)
      $U_{\ell-1} \leftarrow 0$
      if $\gamma = -1$ **then**
        $u_{\ell-1} \leftarrow$ **Multigrid-Scheme**$(\ell - 1, U_{\ell-1}, I_\ell^{\ell-1}(F_\ell - B_\ell U_\ell), \gamma)$
      end if
      Multigrid-Scheme$(\ell - 1, U_{\ell-1}, I_\ell^{\ell-1}(F_\ell - B_\ell U_\ell), |\gamma|)$
      $U_\ell \leftarrow S_\ell(U_\ell + I_{\ell-1}^{\ell} U_\ell - 1, F_\ell, \eta_2)$ (η_2 postsmoothing iterations)
    end for
  end if
end function

The multigrid solver used in Algorithm B has the following components.

- **Hierarchy of grids:** For simplicity, we focus on the case when the mesh step is of the form $h = 2^{-L}H$, where L is a positive integer. The hierarchy of grids is $(\mathbb{T}^2_{2^{-\ell}H})$, for $\ell = 0, \dots, L$.
- **Cycle:** We use the V-cycle or the F-cycle.
- **Restriction operator:** In any multigrid cycle, there are two types of intergrid communication. The fine-to coarse transfer (restriction) $I_\ell^{\ell-1}$ is the intergrid transfer from the grid $\mathbb{T}^2_{2^{-\ell}H}$ to the next coarser grid $\mathbb{T}^2_{2^{-\ell+1}H}$, which produces an approximation of a grid function defined on $\mathbb{T}^2_{2^{-\ell}H}$ by a grid function on the coarser grid $\mathbb{T}^2_{2^{-\ell+1}H}$.

  We shall use the second order full weighting operator $I_\ell^{\ell-1}U_\ell = RU_\ell$ where

$$(RW)_{i,j} = \frac{1}{16}\left(\begin{array}{l} 4W_{2i,2j} + 2\left(W_{2i+1,2j} + W_{2i-1,2j} + W_{2i,2j+1} + W_{2i,2j-1}\right) \\ +W_{2i+1,2j+1} + W_{2i-1,2j+1} + W_{2i+1,2j-1} + W_{2i-1,2j-1} \end{array}\right). \tag{36}$$

- **Interpolation operator:** The coarse-to-fine transfer (prolongation) $I_{\ell-1}^\ell$ interpolates a grid function on the grid $\mathbb{T}^2_{2^{-\ell+1}H}$ to the next finer grid. In our implementation, we have chosen for $I_{\ell-1}^\ell$ the standard bilinear interpolation which is second order accurate. Convergence analysis of multigrid methods shows that it is necessary that the sum of the accuracy orders of the two intergrid transfer operators be not smaller than the order of the partial differential operator. Here the sum of the orders of the transfer operators is 4 which is greater than the order of the PDE. In this case, multigrid theory states that convergence holds even with a single smoothing step: $\eta_1 + \eta_2 = 1$.
- **Approximation of the operator on coarser grids:** In general, the sequence of matrices B_ℓ can either be obtained by applying the finite difference scheme to the partial differential operator on $\mathbb{T}^2_{2^{-\ell}H}$ (in particular $B_L = B$), or from the backward induction $B_{\ell-1} = I_\ell^{\ell-1} B_\ell I_{\ell-1}^\ell$ (Galerkin approximation) with $B_L = B$. In our case, since B comes from a linearization process, we have chosen the second way.
- **Smoother:** The last component is the smoother S_ℓ. We have used the Gauss-Seidel method with lexicographic ordering of the unknowns, and we have taken $\eta_1 = \eta_2 = 1$.

  Alternatively, it is possible to construct the smoothing step by a multiplicative combination of four Gauss-Seidel sweeps corresponding to the four following lexicographic orderings of the unknowns:

  Ordering 1: the unknown corresponding to $x_{i,j}$ is in position $i \times N + j$.

  Ordering 2: the unknown corresponding to $x_{i,j}$ is in position $j \times N + N - i$.

  Ordering 3: the unknown corresponding to $x_{i,j}$ is in position $(N - i) \times N + N - j$.

  Ordering 4: the unknown corresponding to $x_{i,j}$ is in position $(N-j) \times N + i$.

  This particular choice of the smoother aims at correctly treating the advection in the advection-diffusion equation (30) and the transport in (31) when ν gets very small, because the velocities may take any direction a priori.

Remark 6. Note that with Algorithm B, any iteration of BiCGstab involves $2 \times 2N_T$ multigrid solves of bidimensional periodic problems with $2 \times 2N_T$ sparse matrices of order N^2, two by two distinct. Therefore, the initialization of the multigrid methods (i.e. the construction of the sequence of the matrices B_ℓ) has to be done $2 \times 2N_T$ times. Initialization takes an important part of the computing time, even though the initialization step of a single multigrid method has a complexity of $O(N^2)$.

With Algorithm B, the number of iterations needed to decrease the residual norm by a given factor with the preconditioned BiCGstab method does not differ from Algorithm A, see Table 1. In Table 2, we plot the average (on the Newton loop) number of multigrid cycles needed to solve the systems of the form (25) and (34) with an absolute accuracy of 10^{-7} in the normalized ℓ^2 norm. The number of cycle is fairly low, so the use of the more sophisticated smoother described above was not necessary.

TABLE 2. Algorithm B: average (on the Newton loop) number of multigrid cycles in order to solve the systems of the form (25) and (34) with an absolute accuracy of 10^{-7} in the normalized ℓ^2 norm

$\nu\backslash$ grid	$32\times32\times32$	$64\times64\times32$	$128\times128\times32$
0.6	2.65	2.85	2.83
0.36	2.82	3.07	3.15
0.2	2.82	3.06	3.15
0.12	2.78	2.88	2.80
0.046	2.94	3.15	3.18

4. Iterative strategies for solving (17) based on eliminating $\mathcal{M}$.

4.1. **Elimination of $\mathcal{M}$ and PDE interpretation.** This second family of iterative strategies can be used only in the case when the right hand side of (1) is $\Phi(m(t,x))$ and Φ is C^1 strictly monotonous function. In this case, A_{UM} is a diagonal invertible matrix, and it possible to eliminate $\mathcal{M}$ from the linearized discrete Bellman equation. The principle of the method is as follows:

1. solve first
$$A_{U,M}\widetilde{\mathcal{M}} = G_U, \tag{37}$$
which is very easy since $A_{U,M}$ is diagonal.
2. Introducing $\overline{\mathcal{M}} = \mathcal{M} - \widetilde{\mathcal{M}}$, the vector $(\overline{\mathcal{U}}, \mathcal{M})^T$ satisfies
$$\begin{pmatrix} A_{U,U} & A_{U,M} \\ A_{M,U} & A_{M,M} \end{pmatrix} \begin{pmatrix} \overline{\mathcal{U}} \\ \mathcal{M} \end{pmatrix} = \begin{pmatrix} 0 \\ G_M - A_{M,M}\widetilde{\mathcal{M}} \end{pmatrix}, \tag{38}$$
which implies
$$\Big(A_{M,U} - A_{M,M}A_{U,M}^{-1}A_{U,U}\Big)\,\mathcal{U} = G_M - A_{M,M}\widetilde{\mathcal{M}}. \tag{39}$$
We propose to solve (39) by means of a suitably preconditioned iterative method, (we have used a preconditioned BiCGstab method).

Remark 7. Note that the idea of eliminating m from (1)-(4) has been used by P-L. Lions [15] e.g. for the mathematical analysis of the planning problem with mean field games: under some suitable assumptions, P-L. Lions focused on the deterministic case $\nu = 0$ and showed that eliminating m yields a boundary value problem with a nonlinear elliptic PDE for which existence can be proved.

With the notations introduced in § 3.2.1, we see that the matrix $A_{M,U} - A_{M,M}A_{U,M}^{-1}A_{U,U}$ corresponds to the discretization of a three dimensional boundary value problem in $(0,T)\times\mathbb{T}^2$ of the form

$$-\operatorname{div}(mH_{pp}(x,\nabla u)\nabla v) + \text{ FP}\circ\left(n\mapsto(\Phi'(m))^{-1}n\right)\circ \text{ Lin-HJB}(v) = f \quad (40)$$
$$\text{in } (0,T)\times\mathbb{T}^2,$$
$$v_{|t=0} = 0, \quad (41)$$
$$(\text{ Lin-HJB}(v))_{|t=T} = g. \quad (42)$$

Note that (40) involves a partial differential operator of order 4 with respect to x and 2 with respect to t. This is in contrast with the Schur complement $A_{M,M} - A_{M,U}A_{U,U}^{-1}A_{U,M}$ used in § 3 whose counterpart is a nonlocal operator. It is easily seen that the principal part of the partial differential operator in (40) is

$$v\mapsto(\Phi'(m))^{-1}(-\frac{\partial^2 v}{\partial t^2}+\Delta^2 v),$$

i.e. a degenerate fourth order elliptic operator in $(0,T)\times\mathbb{T}^2$ for which a theory of weak solutions can be used.

The local character of $A_{M,U} - A_{M,M}A_{U,M}^{-1}A_{U,U}$ is a nice feature, since a matrix-vector multiplication has a complexity which is linear in the size of the vector. The bad news are that since the related partial differential operator is fourth order, the condition number of $A_{M,U} - A_{M,M}A_{U,M}^{-1}A_{U,U}$ grows like that $O(h^{-4})$ when h tends to 0. This fast growth of the condition number makes it mandatory to use a preconditioner; for example, in our experiments with the unpreconditioned BiCGstab method for (39) and $h = \Delta t = 1/32$, we were not able to reduce the initial residual even by a (large) factor 1/10.

The next paragraphs will be devoted to the construction of a suitable preconditioner for $A_{M,U} - A_{M,M}A_{U,M}^{-1}A_{U,U}$. The idea is to precondition the matrix by performing one step of a suitable multigrid algorithm. By contrast with Algorithm B, multigrid schemes will be used for constructing preconditioners rather than as stand-alone solvers.

4.2. **Non optimality of standard full coarsening multigrid preconditioners.** It is tempting to try a standard multigrid method for the three dimensional problem (39). Coarsening of the grid is performed in both the spatial and the time variables. This is termed *full coarsening* in the multigrid literature. Assuming that $\Delta t = h = 2^{-L}H$, one can try to use a method similar to the one described in § 3.2.3. Let us specify the different components of the multigrid scheme:

- **Restriction operator:** The sum of the orders of the restriction and interpolation operators must be at least 4 w.r.t x and 2 w.r.t. t: hence, we choose the second order full weighting restriction
$$(I_\ell^{\ell-1}U)_{i,j}^n = \frac{1}{4}\left(R\left(U^{2n}+U^{2n+1}+U^{2n-1}\right)\right)_{i,j}$$
where R is given by (36).
- **Interpolation operator:** We choose the trilinear interpolation operator (same as in § 3.2.3 with a three dimensional grid).
- **Approximation of the operator on coarser grids:** Galerkin approximation.
- **Smoother:** We have tried combinations of (damped) Gauss-Seidel or (damped) Jacobi iterations with several lexicographic orderings.

Remark 8. A difficulty that we do not have in the present context of periodic problems is the treatment of boundary conditions by the multigrid strategy: some other components have generally to be added to the smoothers to cope with the boundary conditions; we refer to [10] for an article on multigrids for fourth order diffusion equations with boundary conditions and to [6] for a theoretical article on multigrid methods including the discussion of the boundary conditions.

Unfortunately, the multigrid strategy described above does not yield good preconditioners for $A_{M,U} - A_{M,M}A_{U,M}^{-1}A_{U,U}$: in Table 3, we show the number of iterations needed to decrease the residual norm by a factor 10^{-2}, with the preconditioned BiCGstab method. It shows that the number of iterations grows as the grid size is increased. In the last column, the prescribed maximal number of iterations (200) was reached before convergence.

TABLE 3. Full coarsening multigrid preconditioner with 4 levels: average (on the Newton loop) number of preconditioned BiCGstab iterations to decrease the residual by a factor 0.01

$\nu\backslash$ grid	$32\times32\times32$	$64\times64\times64$	$128\times128\times64$
0.6	40	92	fail
0.36	24	61	not done
0.2	21	45	not done

The reason for this bad behavior can be understood by realizing that the matrix is strongly anisotropic: indeed, looking at the corresponding partial differential operator, we have seen that it is fourth order in x and second order in t. Therefore, the entries of $A_{M,U} - A_{M,M}A_{U,M}^{-1}A_{U,U}$ connecting the unknowns sharing the same time index, for example $V_{i,j}^n$ and $V_{i\pm1,j\pm1}^n$, are of the order of h^{-4}, whereas the entries connecting the unknowns with different time indices, for example $V_{i,j}^n$ and $V_{i,j}^{n\pm1}$ are of the order of $\Delta t^{-2} = h^{-2}$. This implies that the Gauss-Seidel or Jacobi iterations will smooth the residual in the planar cross-sections $t = n\Delta t$, but will fail to smooth the residual along the lines $x = x_{i,j}$.

Note that in Table 3, the number of iterations decreases as ν decreases: indeed the anisotropy of the matrix decreases, so the full coarsening strategy becomes more adapted.

4.3. **Algorithm C: A semi-coarsening multigrid preconditioner.** Having understood the reason of the bad behavior of standard multigrid preconditioners, we are led to propose another method, where the hierarchy of nested grids is obtained by coarsening the grids in the x directions and not in the t direction. Indeed, since the smoother does not damp the fast oscillations of the residual in the time variable, there is no point in coarsening the mesh in the time direction. Therefore, the grid of $[0,T]\times\mathbb{T}^2$ corresponding to level ℓ has $\sim N_T \times \frac{\ell^2}{H^2}$ nodes. Semi-coarsening multigrid methods for anisotropic second order diffusion equations are dealt with in [16].

Let us specify the different components of the multigrid scheme:

- **Hierarchy of grids:** Obtained by semi coarsening.
- **Cycle:** We use either the V-cycle or the F-cycle.
- **Restriction operator:** We use the second order restriction operator:

$$(I_\ell^{\ell-1}U)_{i,j}^n = (RU^n)_{i,j},$$

where R is defined in (36)

- **Interpolation operator:** We choose the bilinear interpolation operator in the planar cross-sections $t = n\Delta t$, as in § 3.2.3.
- **Approximation of the operator on coarser grids:** Galerkin approximation.
- **Smoother:** We use (damped) Gauss-Seidel relaxation with a lexicographic ordering. We take $\eta_1 = 1$ and $\eta_2 = 3$.
- **Coarse problem:** A direct solver is used at the coarsest level: compared with the full coarsening multigrid, the coarser problem is larger here.

In Table 4, we show the number of iterations needed to decrease the residual norm by a factor 10^{-3}, with the preconditioned BiCGstab method (Algorithm C with Gauss-Seidel smoother). We see that the number of iterations is much smaller than with full coarsening but still depends on the size of the grid. It also grows as ν decreases. When using damped Gauss-Seidel smoother with damping factor $\omega = 0.8$, see Table 5, the number of iterations still depends on the size of the grid but it less sensitive to the variations of the parameter ν.

TABLE 4. Algorithm C with Gauss-Seidel smoother: average (on the Newton loop) number of iterations of the BiCGstab method to decrease the residual by a factor 0.001

$\nu\backslash$ grid	$32 \times 32 \times 32$	$64 \times 64 \times 64$	$128 \times 128 \times 64$
0.6	4	5	7
0.36	4	5	7
0.2	4	5.5	7
0.12	6	9	12

TABLE 5. Algorithm C with the damped Gauss-Seidel smoother, ($\omega = 0.8$): average (on the Newton loop) number of iterations of the BiCGstab method to decrease the residual by a factor 0.001

$\nu\backslash$ grid	$32 \times 32 \times 32$	$64 \times 64 \times 64$	$128 \times 128 \times 128$
0.6	3.75	4.5	6
0.36	3.4	4	5.4
0.2	3	4.15	5.5
0.12	4	4.5	6.2

Remark 9. Since the multigrid method is used as a preconditioner only, it does not need be updated at each Newton step. In particular, close to the convergence of the Newton method, it is possible to avoid reconstructing the multigrid components. This saves computing time.

To summarize, Algorithm C is based on a multigrid method for a degenerate elliptic PDE which is fourth order in the space variable and second order only in time. Its results are satisfactory. The key feature is the semi-coarsening of the grids. Note that the sum of the orders of the grid transfer operators matches exactly the order of the operator, although the theory of A. Brandt[6] tells us that it is desirable that the former be (strictly) larger than the latter for the multigrid

method to have good convergence properties (independent of (η_1, η_2): this explains why in Tables 4 and 5, the number of iterations depends on the size of the grid. A possible improvement of Algorithm C would be to use bicubic interpolation instead of bilinear and change the Galerkin approximation of the operator on coarser grids accordingly, but we have not tested it.

For the same reason, we know that the best multigrid methods for solving boundary value problems with the bilaplacian (see e.g. [16] and [10]) do not address the PDE directly, but are rather based on decomposing the fourth order PDE into a system of two second order PDEs and using efficient multigrid methods for the second order operators. From this argument, we expect that when ν is not too small, it may be better not to eliminate the variable $\mathcal{M}$ and to directly address the full system of PDEs: this is the topic of the next section.

5. **Multigrid preconditioners for the full system of PDEs.** In § 3 (resp. §4), we considered methods based on the elimination of $\mathcal{U}$, (resp. $\mathcal{M}$). Here, we are going to consider preconditioned iterative methods for solving (17) without eliminating any of the unknown grid functions ($\mathcal{U}$ or $\mathcal{M}$). We make the same assumptions as in §4, in particular ($\mathbf{\Phi_1}$). The preconditioner will be constructed by means of a multigrid method for the following boundary value problem posed on a three dimensional domain:

$$\begin{aligned} \frac{\partial v}{\partial t} - \nu\Delta v + \frac{\partial H}{\partial p}(x, \nabla u)\cdot\nabla v - \Phi'(m)n &= w \quad \text{in } (0,T]\times\mathbb{T}^2, \\ \frac{\partial n}{\partial t} + \nu\Delta n + \operatorname{div}(n\frac{\partial H}{\partial p}(x,\nabla u)) + \operatorname{div}(mH_{pp}(x,\nabla u)\nabla v) &= r \quad \text{in } (0,T]\times\mathbb{T}^2, \\ v|_{t=0} &= 0 \quad \text{in } \mathbb{T}^2, \\ n|_{t=T} &= 0 \quad \text{in } \mathbb{T}^2. \end{aligned} \tag{43}$$

We refer to [16], chapter 8, for a survey on multigrid for systems of PDEs, with a particular emphasis for systems arising in fluid mechanics. The main difference with scalar PDEs lies in the choice of the smoothing schemes: quoting [16], a natural extension of smoothing by relaxation (for example, damped Gauss-Seidel or damped Jacobi) is smoothing by *collective* relaxation. That is all the unknowns at each single grid point are relaxed simultaneously.

5.1. **Algorithm D: A multigrid preconditioner for the full linearized MFG system.** Algorithm D consists of applying a preconditioned BiCGstab method to (17). Applying the preconditioner consists of performing one step of a multigrid method either to the full matrix

$$\begin{pmatrix} A_{UU} & A_{UM} \\ A_{MU} & A_{MM} \end{pmatrix}$$

or to the simpler one

$$\begin{pmatrix} A_{UU} & A_{UM} \\ 0 & A_{MM} \end{pmatrix}.$$

Using the latter is equivalent to dropping the second order differential operator $v \mapsto -\operatorname{div}(mH_{pp}(x,\nabla u)\nabla v)$ in (40).

The components of the multigrid scheme are as follows:

Hierarchy of grids: From the conclusions of §4, we obtain the hierarchy of grids by semi coarsening the finest one: coarsening is performed only for the spatial variable.

- **Cycle:** We use either the V-cycle or the F-cycle.

- **Restriction operator:** We use the same second order restriction operator as in Algorithm C, except that we apply it to both $\mathcal{U}$ and $\mathcal{M}$.
- **Interpolation operator:** We use the same second order interpolation operator as in Algorithm C, except that we apply it to both $\mathcal{U}$ and $\mathcal{M}$.
- **Approximation of the operator on coarser grids:** Galerkin approximation.
- **Smoother:** Since the smoother is the same on all grids, let us describe it on the finest grid: for $0 < n < N_T$, the unknowns $U^n_{i,j}$ and $M^{n-1}_{i,j}$ are grouped together and considered as a single vectorial one: hence, the unknowns are ordered as follows

$$\underbrace{U^0_{0,0}, \dots U^0_{N,N}}, \quad \underbrace{\begin{pmatrix} U^1_{0,0} \\ M^0_{0,0} \end{pmatrix}, \dots, \begin{pmatrix} U^1_{N,N} \\ M^0_{N,N} \end{pmatrix}}, \dots, \underbrace{\begin{pmatrix} U^{N_T}_{0,0} \\ M^{N_T-1}_{0,0} \end{pmatrix}, \dots, \begin{pmatrix} U^{N_T}_{N,N} \\ M^{N_T-1}_{N,N} \end{pmatrix}}, \quad \underbrace{M^{N_T}_{0,0}, \dots M^{N_T}_{N,N}},$$

  where a lexicographic ordering with respect to the (i, j) indices is used for the under-braced collections of scalar/vectorial unknowns. The smoother is a collective Gauss-Seidel relaxation (or block Gauss-Seidel, with blocks of order one/two): for $0 < n < N_T$, the unknowns $U^n_{i,j}$ and $M^{n-1}_{i,j}$ are updated simultaneously by solving a 2 by 2 systems of linear equations. In our tests, we choose $\eta_1 = 1$ and $\eta_2 = 3$.
- **Coarse problem:** The coarse problem is solved by a direct method: its size is twice the size of the corresponding coarse problem in Algorithm C, but its matrix is sparser.

In Table 6, we show the number of iterations needed to decrease the residual norm by a factor 10^{-3}, with the preconditioned BiCGstab method, see (19) for a description of the PDEs in this test. For the smoother, we have used a collective damped Gauss-Seidel method, with damping factor $\omega = 0.8$. Compared to Algorithm C, the performances of Algorithm D do not deteriorate as the number of grid points grows, which confirms the arguments given at the end of § 4: for ν large enough, the multigrid method for the full system of PDEs has a better behavior than the multigrid method for (39). Therefore, Algorithm D is extremely efficient when ν is large enough, i.e. when the system has a clear elliptic behavior. On the other hand, the performances of Algorithm D deteriorate rapidly as ν decreases; this may be explained as follows: in the deterministic limit ($\nu = 0$), m takes the value 0 in a region with non zero measure (see the right side of Figure 1 where we see that m at $t = 0.5$ is very small in a large region); in that region, (43) is a system of two weakly coupled hyperbolic equations, for which the smoother described above ceases to be efficient. It may be possible to improve the method by proposing other smoothers, but we have not done it.

6. **Computing times.** It is a bit delicate to compare computing times, since they depend much on implementation. In particular, for Algorithms B, C and D, the C++ STL *map* data structure was used for storing the sparse matrices: this makes coding much easier, but the complexity is not optimal. Moreover, the BLAS library was not used for coding Algorithms B, C and D whereas it was used in Algorithm A. Hence, one should not draw too many conclusions from the following data, except

TABLE 6. Algorithm D with the collective damped Gauss-Seidel smoother ($\omega = 0.8$) : average (on the Newton loop) number of iterations of the BiCGstab method to decrease the residual by a factor 10^{-3}

$\nu \backslash$ grid	$32 \times 32 \times 32$	$64 \times 64 \times 64$	$128 \times 128 \times 128$
0.6	1.75	1.5	1.25
0.36	2.2	2	2
0.2	4.9	3.5	2.9
0.12	14.4	11.4	6.8

for tendencies and qualitative behaviors. Although our program is sequential, the times have been obtained on a Dell server with Six-core 2.93GHz Intel(R) Xeon(R) X5670 processors.

Table 7 contains the average computing times for solving the linear systems of the form (17) for different grids, in the case $\nu = 0.6$, more precisely for constructing the preconditioner (possibly the multigrid data structure) and for reducing the residual by a factor 10^{-3}) with the preconditioned BiCGstab methods proposed above. We see at first glance that the computing times are of the same orders with Algorithms A,B and D, and that Algorithm C takes more time: this is mainly because the number of iterations is higher than with the other algorithms and increases as the grid step decreases, see Table 5. Although it does not appear on the table, the time spent for constructing the multigrid data structure with Algorithms B and D represents an important part of the overall computing time (close to one third), since the number of iterations is small for $\nu = 0.6$

In Figure 2, we plot the computing times versus the number of unknowns for Algorithm A, B, and D: with Algorithms B and D, the computing times grow close to linearly with the number of unknowns; thus we may say that these methods are close to optimal.

Finally, table 8 contains the average computing times for solving the linear systems of the form (17) with the preconditioned BiCGstab methods proposed above, in the case $\nu = 0.12$. The computing times are longer than in the previous case, because the number of iterations is larger. Here too, the computing times with Algorithms B and D grow close to linearly with the number of unknowns, but the time spent with Algorithm D is longer than with Algorithms A and B. Indeed, Algorithm D is more sensitive to the variations of ν than Algorithms A and B, (compare Tables 1 and 5). Algorithm D becomes less attractive when ν becomes small.

TABLE 7. $\nu = 0.6$: average computing time for solving the linearized problem

Algorithm \ grid	$32 \times 32 \times 32$	$64 \times 64 \times 64$	$128 \times 128 \times 128$
A	2.06	19.9	234.7
B	2.64	27.20	256.03
C	7.24	79.05	874.16
D	3.14	27.20	239.25

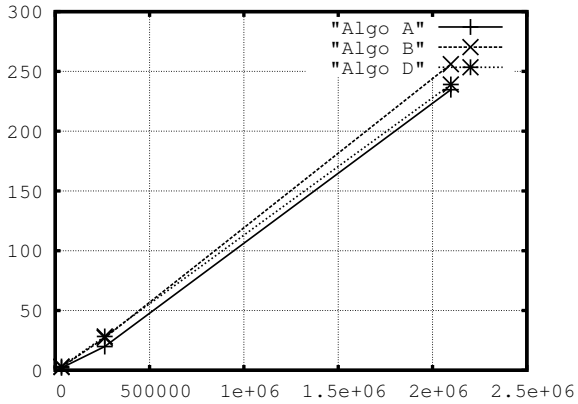

FIGURE 2. $\nu = 0.6$: computing times for Algorithms A, B and D versus the number of grid points $N_T \times N^2$

TABLE 8. $\nu = 0.12$: average computing time for solving the linearized problem

Algorithm \ grid	$32 \times 32 \times 32$	$64 \times 64 \times 64$	$128 \times 128 \times 128$
A	5.02	50.03	577.25
B	6.64	65.7	581.05
C	12	145,46	1921
D	23.99	244.37	1181

7. **Conclusions.** We have presented three families of iterative strategies for solving the linearized MFG systems. They have been tailored from the structure of the PDEs system.

Algorithms A and B belong to the first family. They consist in eliminating $\mathcal{U}$, which leads to a problem involving a nonlocal operator. Preconditioning yields a problem with a compact perturbation of the identity; hence Algorithms A and B have the nice feature that the number of iterations does not depend on the grid size and increases moderately as ν becomes small, even very small (we tried $\nu = 0.01$ for which the minimum of m was found smaller than 10^{-6}). Algorithm A requires a very efficient direct solver for sparse matrices arising from bidimensional PDEs, but this kind of solvers are available as free-wares. The computing time grows slightly superlinearly with the number of unknowns (due to the use of a direct solver). Algorithm B is a variant of Algorithm A consisting of solving the numerous 2-dimensional problems with multigrid iterations. With our implementation, using the C++ STL *map* data structure, and not using the BLAS library, the computing times with Algorithm B were in general slightly longer than those with Algorithm A: this confirms the fact that the direct solvers in [1] are very efficient. However, in dimension $d = 3$, Algorithm B should definitely be preferred to Algorithm A.

Algorithm C consists of eliminating $\mathcal{M}$ (Assumption $(\mathbf{\Phi_1})$ is needed), which leads to a problem involving a partial differential operator, second order with respect to t and fourth order with respect to x, and applying a suitable multigrid preconditioner. The important feature is that the hierarchy of grids is obtained by semicoarsening.

The number of iterations, which is higher than with Algorithm B, depends on the grid size.

Algorithm D consists of using a suitable multigrid preconditioner for the full system: it uses semicoarsening and collective smoothing. The computing time grows close to linearly with the number of unknowns but it is sensitive to ν. Thus, Algorithm D is efficient for large values of ν but becomes less attractive than Algorithms A or B when ν gets small and the density m gets close to zero in some regions. Finally, note that the ingredients of Algorithm D could be reused to design a multigrid *full approximation scheme* for (12), i.e. to apply directly the multigrid strategy to (12) and not to its linearized version.

REFERENCES

[1] UMFPACK. Available from: http://www.cise.ufl.edu/research/sparse/umfpack/current/.

[2] Y. Achdou, F. Camilli, and I. Capuzzo Dolcetta, *Mean field games: numerical methods for the planning problem*, SIAM J. Control Optim., **50** (2012), 77–109.

[3] Y. Achdou and I. Capuzzo-Dolcetta, *Mean field games: Numerical methods*, SIAM J. Numer. Anal., **48** (2010), 1136–1162.

[4] J.-D. Benamou and Y. Brenier, *Mixed L^2-Wasserstein optimal mapping between prescribed density functions*, J. Optim. Theory Appl., **111** (2001), 255–271.

[5] J.-D. Benamou, Y. Brenier and K. Guittet, *The Monge-Kantorovitch mass transfer and its computational fluid mechanics formulation*, ICFD Conference on Numerical Methods for Fluid Dynamics (Oxford, 2001), Internat. J. Numer. Methods Fluids, **40** (2002), 21–30.

[6] A. Brandt, *Rigorous quantitative analysis of multigrid. I. Constant coefficients two-level cycle with L_2-norm*, SIAM J. Numer. Anal., **31** (1994), 1695–1730.

[7] D. A. Gomes, J. Mohr and R. R. Souza, *Discrete time, finite state space mean field games*, J. Math. Pures Appl. (9), **93** (2010), 308–328.

[8] O. Guéant, *Mean field games equations with quadratic hamiltonian: A specific approach*, arXiv:1106.3269, 2011.

[9] 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 Math., **2003**, Springer, Berlin, (2011), 205–266.

[10] S. Henn, *A multigrid method for a fourth-order diffusion equation with application to image processing*, SIAM J. Sci. Comput., **27** (2005), 831–849 (electronic).

[11] A. Lachapelle, J. Salomon and G. Turinici, *Computation of mean field equilibria in economics*, Math. Models Methods Appl. Sci., **20** (2010), 567–588.

[12] J.-M. Lasry and P.-L. Lions, *Jeux à champ moyen. I. Le cas stationnaire*, C. R. Math. Acad. Sci. Paris, **343** (2006), 619–625.

[13] J.-M. Lasry and P.-L. Lions, *Jeux à champ moyen. II. Horizon fini et contrôle optimal*, C. R. Math. Acad. Sci. Paris, **343** (2006), 679–684.

[14] J.-M. Lasry and P.-L. Lions, *Mean field games*, Jpn. J. Math., **2** (2007), 229–260.

[15] P.-L. Lions, Cours du Collège de France, 2007–2011. Available from: http://www.college-de-france.fr/default/EN/all/equ$_-$der/.

[16] U. Trottenberg, C. W. Oosterlee and A. Schüller, "Multigrid," With contributions by A. Brandt, P. Oswald and K. Stüben, Academic Press, Inc., San Diego, CA, 2001.

[17] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., **13** (1992), 631–644.

Received November 2011; revised March 2012.

E-mail address: achdou@ljll.univ-paris-diderot.fr
E-mail address: v.j.f.perez@gmail.com