Production Capacity issues and "Time to Build" in Oil Production

Directeur de la Recherche :

Pierre-Louis Lions, Professeur au Collège de France

Juillet 2011

Production capacity issues and "Time to Build" in Oil Production

Contents

Preface

Starting with the seminal article by Kydland and Prescott [6], the issue of the time needed to build capacity of production before being able to produce has been studied in the economic literature. This work is a new attempt to tackle this important question linked to the evolution of supply and we provide new mathematical tools[1] to model competition between economic agents facing *time-to-build* issues. These methods allow to study in a very general framework, the dynamics of the entire production capacity, be it productive now or expected in the future. More precisely, our modeling framework permits us to take account of both the rate of progress in the construction of new production capacity units and the ageing of currently available production structures. In other words, the state variable of our problem is the number of units in each possible state, be they under construction, production or obsolete.

In our framework, decisions are taken by rational individual entrepreneurs in charge of their own production structure. The global production structure is therefore the outcome of a large number of decisions, taken by individual atomized agents that are too small to be individually able to influence the global dynamics. In spite of this absence of influence, strategies and decisions are complex. The time-to-build issue and the (rational) expectations indeed generate complex reactions in response to demand fluctuations and global production capacity evolution (the global stock of production units under construction having a lot of influence on agents' decision process). The time-to-build issue hence generates behaviors that are not only based on observed prices. The expectation on future prices can only be made through the knowledge of the entire production structure.

The important result we provide – which is not limited to the economic problem under scrutiny – is that, for a large class of situations, the resulting dynamics of the entire production structure, although agents optimize selfishly without any cooperation, is identical to the dynamics a benevolent planner would end up with for a certain global optimization criterion. This optimization criterion is more than the average of individuals' objective function and takes into account the interactions between the underlying entrepreneurs.

This mathematical result is new and its general proof is out of the scope of this text although we provide specific proofs when needed. However, this result will certainly be the subject of a paper in an economic journal.

This text focuses on the presentation of our new modeling methodology and on its applications. To better understand the framework, we start with simplified models. For instance, we

[1] These methods are rooted to the mean field game theory introduced in 2006 by J.-M. Lasry and P.-L. Lions. Mean field game theory allows to find equilibria of games with a lot of players and is now widely studied both to extend the theoretical results and to use it in applications.

focus first on the decision process of production capacity construction in several deterministic contexts where a demand shock is expected to happen and perfectly anticipated. These simplified models are in fact necessary steps toward the presentation of the general model with time to build and randomness both at the agent level and on total demand. Then, once the general framework has been presented along with the equivalence with a planning problem, we focus on the numerical tools to solve the partial differential equations arising when modeling the decision process of new capacity construction.

The models we develop describe some aspects of industries such as the oil industry. In addition to the outcomes of these models, we think that the reader will catch the main innovations and the power of our new approach. This text should pave the way for future applications, especially as far as the supply of raw materials is concerned.

Introduction

A substantial part of macroeconomic fluctuations is linked to the behavior of commodity prices such as oil prices (see for instance [5], [13]). In particular, it is widely known that important recessions are correlated with turmoil on oil prices.
The high volatility of oil prices is therefore a matter of important preoccupation and has been studied with different perspectives. The most common perspective is the financial one. A large economic and financial literature is indeed devoted to the study of oil prices and to the time structure of these prices in future markets. Aside from this literature, but somehow linked to it, economists have tried to explain the determinants of such a high volatility and some common features of oil prices have also been studied such as the important skewness of price distribution, inducing short periods with very high prices. In particular, non-linear models have been developed (see for instance [2], [3], [14]) that incorporate the important issue of reserves or storage in the dynamics of oil prices.

In this text we consider the time-to-build dimension of the oil industry and we model oil companies' decision making in the development and the exploitation of new production capacities. Similar to the hypotheses on investment in the time-to-build literature (see [6], [10]), we develop models in which capacity can only evolve progressively. This aspect of industrial processes is often neglected in the economic modeling of most industries but it is an important determinant as far as the oil industry is concerned. Oil companies indeed decide on exploration and then on production with respect to the economic environment and the delay between a decision and the actual subsequent increase in production capacity may be very long. Hence, decision making not only has to take account of current and anticipated dynamics for oil demand but also depends on the increase in production capacities planned by competitors.
The models we develop are devoted to a description (with microeconomic foundations) of the progressive adaptation of suppliers' production capacity to shocks on demand. Thus, since slow capacity evolution implies absorption of demand shocks through prices and then through change in production capacity, the models also allow us to study the price reaction function that transforms exogenous shocks on demand into price fluctuations. This modeling is a new attempt to provide determinants to oil prices volatility and we see that, contrary to what happens in other industries, exogenous demand shocks are hardly attenuated. Low price elasticity of oil demand due to the lack of a good substitute, important delay in capacity adaptation and volatility of demand (and supply[2]) are indeed the three ingredients for high volatility in prices.

[2]although this is not modeled *per se* in this text.

The very nature of the problem we model, that is the backward dimension of optimal control based on anticipations, coupled with the forward nature of the industrial process, imposes the use of the tools developed in the mean field game theory [7, 8, 9].
Part I is devoted to a simple model of oil production with slow (endogenous) capacity evolution. This model developed in a deterministic framework is well summed up by a forward/backward dynamical system. When it comes to adapting this model to a new framework in which demand is stochastic, then, the fact that noise was common to all producers made it necessary to develop a new framework to deal with common noise and thus, with Hamilton-Jacobi-Bellman equations in infinite dimension spaces. This new framework for mean field games, developed in 2010 and 2011 by J.-M. Lasry and P.-L. Lions has been used here in the case of a finite state space that reduces the HJB equation in infinite dimension to a system of coupled HJB equations. This new method is exemplified at the end of Part I and Part II is devoted to the derivation of important results in this new framework. In particular, it is shown in Part II that, for a certain class of games called potential games, a solution to the problem can be found by solving only one HJB equation. Part III presents an application of this new framework to the modeling of the time-to-build dimension of oil industry: before a capacity production is available, it has to go through a given number of steps and the transition from one step to the next can be partially controlled by the oil producers.

Part I

Production capacity choices

In this first part we will present a model of production capacity choices applied to oil industry. The goal of this first part is twofold. First we aim at understanding the implications of production capacity choice in terms of oil prices. The problem exhibits indeed a complex forward/backward structure since production capacity choices depend on expectations and will in turn govern the dynamics of oil supply for some time. Our first goal is then to model the impact of imperfect adjustment of supply to demand fluctuations, especially in terms of prices. Second, we will use this model to present a new approach allowing us to solve a certain class of mean field games. This new approach will then be developed theoretically and applied to a model dealing with "time-to-build" issues in oil industry.

1 Setup of the model

We regard oil industry as an industry with infinitely many participants. Each participant in the oil industry has a certain production capacity, these production capacities being chosen on entering the market. Hence we denote q_i the production capacity of agent i and the total quantity available for production (equal to supply in our model) m.
To model oil prices we introduce a time-varying demand function $D(t,p)$ and the associated inverse demand function $P(t,\cdot)$.
An agent with production capacity q makes a profit between t and $t+dt$ that is therefore:

$$\pi(t,q) = qP(t,m(t))$$

Each participant produces oil for a certain period of time τ that is random and modeled by an underlying Poisson process of intensity λ.
Hence, an agent entering the market at date t with production capacity q has an expected profit of the form:

$$\begin{aligned}\Pi(t,q) &= \mathbb{E}\left[\int_t^{t+\tau} qP(s,m(s))e^{-r(s-t)}ds\right] \\ &= q\int_t^{\infty} P(s,m(s))e^{-(r+\lambda)(s-t)}ds\end{aligned}$$

where r is the interest rate used to discount profits.
For further analysis we denote $u(t)$ the profit associated to a unitary production capacity *i.e.*

$$u(t) = \int_t^\infty P(s, m(s))e^{-(r+\lambda)(s-t)}ds$$

Now, at each period of time a new participant enters the market and must choose its production capacity q. It is rather natural to think that this production capacity will be an increasing function of u and we assume that an agent entering the market at date t chooses $q = ku(t)$ where k is a constant.
Another way to introduce this is to say that a producer willing to have a production capacity equal to q incurs a cost equal to $\frac{q^2}{2k}$.

2 A forward-backward dynamical system

To solve this model, we need to deal with the interaction between supply and profit: if profit is expected to increase, then more production capacities will be built. In turn, if m increases, the price will drop, inducing a decrease in expected profit. The system is then characterized by two equations.

Starting with the dynamics of supply, m satisfies the following ordinary differential equation:

$$m'(t) = ku(t) - \lambda m(t)$$

the initial condition being $m(0) = m_0$ a given constant.
Then, we can differentiate the equation that defines u and we get:

$$u'(t) = (r + \lambda)u(t) - P(t, m(t))$$

To guarantee that the solution of the preceding equation is well defined and indeed equals the intertemporal expected profit u, we impose the following transversality condition:

$$\lim_{t\to+\infty} e^{-(r+\lambda)t}u(t) = 0$$

Hence, we have to solve the following dynamical system:

$$\begin{aligned} u'(t) &= (r+\lambda)u(t) - P(t,m(t)) \qquad \lim_{t\to+\infty} e^{-(r+\lambda)t}u(t) = 0 \\ m'(t) &= ku(t) - \lambda m(t) \qquad\qquad\qquad m(0) = m_0 \end{aligned}$$

The first equation is backward since it characterizes a value function obtained by backward induction. The second equation states the evolution of the system and is therefore forward. For this reason, the above system is thereafter referred to as (FBDS) – for forward/backward dynamical system.

3 A benchmark case

As a reference case, we will consider an inverse demand function that does not depend on time: $P(t,m) = P(m)$ – this function being continuous and decreasing in m.

3.1 Solution trajectory

We start with the stationary equilibrium which is easily found.

Proposition 1. *There exists a unique stationary equilibrium point (u^*, m^*) to (FBDS).*
m^* is the unique solution of $P(m^*) = \frac{1}{k}\lambda(r+\lambda)m^*$ and $u^* = \frac{\lambda}{k}$.

Now, although it is a forward-backward dynamical system we can draw a phase diagram (Figure 1) and we see that (FBDS) is an hyperbolic dynamical system with a saddle point. Hence for a given starting point m_0, we need to know the value of u_0 to know whether or not we are on the stable manifold.

In fact because of the very definition of u (or equivalently because of the transversality condition), we know that if $t \in [\hat{t}, +\infty[\mapsto m(t)$ were increasing for some $\hat{t}$ then $t \in [\hat{t}, +\infty[\mapsto u(t)$ must be a decreasing function. Indeed, $\forall t_2 > t_1 > \hat{t}$:

$$\begin{aligned} u(t_2) &= \int_{t_2}^{\infty} P(m(s))e^{-(r+\lambda)(s-t_2)}ds \\ &\leq \int_{t_2}^{\infty} P(m(s+t_1-t_2))e^{-(r+\lambda)(s-t_2)}ds \\ &\leq \int_{t_1}^{\infty} P(m(s))e^{-(r+\lambda)(s-t_1)}ds \\ &\leq u(t_1) \end{aligned}$$

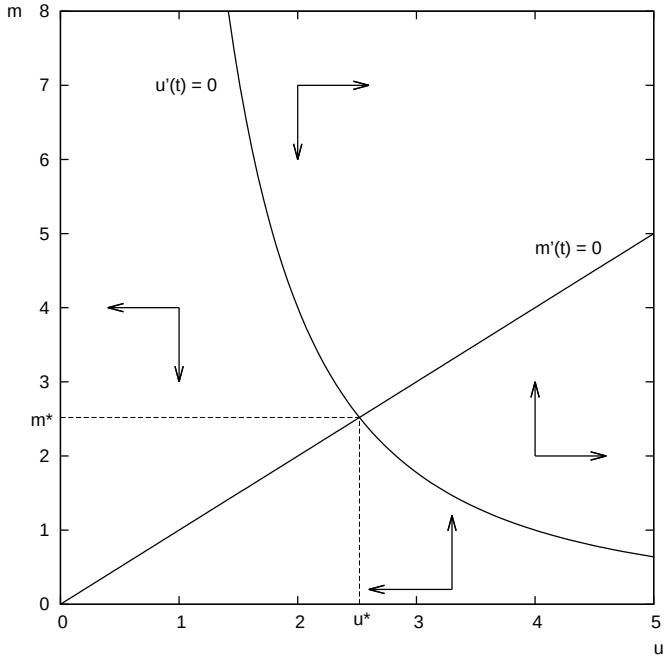


Figure 1: Phase diagram of (FBDS). Parameters are: $k = 0.2$, $\lambda = 0.2$, $r = 0.05$ and $P(m) = m^{-\frac{1}{2}}$.

Hence, the trajectory cannot go in the east quadrant of the above phase diagram since it would stay in this quadrant and both u and m would be increasing after some time.

Since the same reasoning applies symmetrically for the west quadrant, the trajectory must lie in the north or in the south quadrant. The only possibility is then that the trajectory lies on the stable manifold of the dynamical system.

Proposition 2. *The solution (u, m) of (FBDS) lies on the stable manifold of the dynamical system and hence converges toward (u^*, m^*).*

3.2 Numerical aspects

For the above benchmark as well as for the models to come, we will need to find numerically the trajectory followed by (u, m). We propose two techniques to approximate the trajectory, the first being based on the eductive strategy used to solved mean field games and the second being based on the resolution of a nonlinear ordinary differential equation. Other methods may be constructed like shooting methods.

3.2.1 The eductive strategy

Let us consider a finite horizon analog of the equations (FBDS). u and m are replaced by $t \mapsto u(t, T)$ and $t \mapsto m(t, T)$ where T is the horizon of the problem, chosen large enough in what

follows to ensure a good approximation. Since we know the asymptotic value for u, a natural choice to define a finite horizon approximation is:

$$\begin{aligned} u'(t,T) &= (r+\lambda)u(t,T) - P(t,m(t,T)) \qquad u(T,T) = u^* \\ m'(t,T) &= ku(t,T) - \lambda m(t,T) \qquad\qquad\quad m(0,T) = m_0 \end{aligned}$$

We know from the same analysis as above that there exists a unique solution to this system and we easily see that:

$$\lim_{T\to\infty} \sup_{t\in[0,T]} |u(t,T) - u(t)| = 0$$

and

$$\lim_{T\to\infty} \sup_{t\in[0,T]} |m(t,T) - m(t)| = 0$$

Hence, a way to approximate the solution (u,m) is first to approximate u^* and then to use a method that approximates $(u(t,T), m(t,T))$.

Now, if we integrate the above system we have:

$$u(t,T) = u^* e^{-(r+\lambda)(T-t)} + \int_t^T P(s,m(s,T))e^{-(r+\lambda)(s-t)}ds$$

and

$$m(t,T) = m_0 + k\int_0^t u(s,T)e^{-\lambda s}ds$$

The eductive approach then consists in starting with $u^0(t,T) = u^*$ and $m^0(t,T) = m_0$, $\forall t \in [0,T]$, and then computing recursively (u^{n+1}, m^{n+1}) from (u^n, m^n) by:

$$m^{n+1}(t,T) = (1-\theta)m^n(t,T) + \theta\left[m_0 + k\int_0^t u^n(s,T)e^{-\lambda s}ds\right]$$

and

$$u^{n+1}(t,T) = (1-\theta)u^n(t,T) + \theta\left[u^* e^{-(r+\lambda)(T-t)} + \int_t^T P(s,m^n(s,T))e^{-(r+\lambda)(s-t)}ds\right]$$

for some small parameter θ.

Now, depending on the specification for P and the values of the parameters, the sequence (u^n, m^n) may converge and in that case it converges towards $(u(\cdot,T), m(\cdot,T))$ that is a rather good approximation of (u,m) when T is sufficiently large.

Two instances of such a resolution are carried out below.
First, we consider a case where $m_0 < m^*$ (Figure 2) and we see that, as expected, the value of $u(0)$ is such that $(u(0), m_0)$ lies on the stable manifold. This value $u(0)$ is endogenously determined by the algorithm and corresponds to the expected intertemporal profit of an agent at time 0.

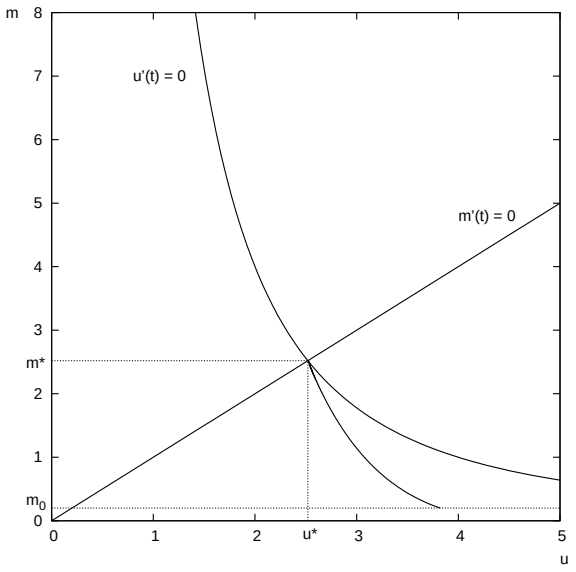


Figure 2: Solution of (FBDS) for $m_0 = 0.2$. Parameters are: $k = 0.2$, $\lambda = 0.2$, $r = 0.05$ and $P(m) = m^{-\frac{1}{2}}$.

Second, we consider a case where $m_0 > m^*$ (Figure 3) and the trajectory lies on the stable manifold as above.

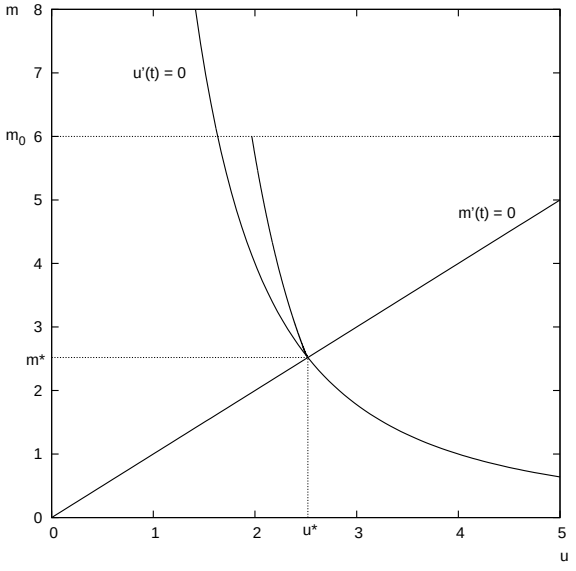


Figure 3: Solution of (FBDS) for $m_0 = 6$. Parameters are: $k = 0.2$, $\lambda = 0.2$, $r = 0.05$ and $P(m) = m^{-\frac{1}{2}}$.

3.2.2 A nonlinear second order equation

Another way to find the trajectory is to transform (FBDS) in a second order (nonlinear) ordinary differential equation. Rewriting the system

$$\begin{aligned} u'(t) &= (r+\lambda)u(t) - P(m(t)) \\ m'(t) &= ku(t) - \lambda m(t) \end{aligned}$$

we see that

$$\begin{aligned} u'(t) &= (r+\lambda)u(t) - P(m(t)) \\ u'(t) &= \frac{1}{k}(r+\lambda)(m'(t) + \lambda m(t)) - P(m(t)) \end{aligned}$$

Hence,

$$\begin{aligned} m''(t) &= ku'(t) - \lambda m'(t) \\ m''(t) &= (r+\lambda)(m'(t) + \lambda m(t)) - kP(m(t)) - \lambda m'(t) \end{aligned}$$

Hence, the second order equation for m is:

$$-m''(t) + rm'(t) + (r+\lambda)\lambda m(t) - kP(m(t))$$

The initial condition that derived from (FBDS) is $m(0) = m_0$ and the transversality condition is $\lim_{t\to+\infty} e^{-(r+\lambda)t}(m'(t) + \lambda m(t)) = 0$

This transversality condition is not practical and since we know that $\lim_{t\to\infty} m(t) = m^*$, we may replace the transversality by this condition, especially in approximations.

As before, we can consider for T sufficiently large the unique solution $m(\cdot, T)$ of[3]

$$\Rightarrow -m''(t,T)+rm'(t,T)+(r+\lambda)\lambda m(t,T)-kP(m(t,T)) = 0, \qquad m(0,T) = m_0, \qquad m(T,T) = m^*$$

[3]the notation $m(\cdot, T)$ is the same but this function is different from the one introduced in the above paragraphs.

Hence, the second method consists in solving a nonlinear[4] second order differential equation with Dirichlet limit conditions in 0 and T. We will see in what follows that this method is well suited for generalizations of the model.

4 The impact of a demand shock

Now, we do not suppose anymore that the inverse demand function is independent of time and rather study the impact of a shock on demand. Our interest will be both on permanent and temporary shocks. The goal is to understand the decisions made by the agents in terms of production capacity building with respect to their (perfect) anticipations of future demand and to study the impact of their individual decisions on price dynamics.

4.1 Permanent shocks

4.1.1 Framework

Let us start with the case of a permanent shock on demand. This shock is modeled through a change in the inverse demand function and we suppose that, at equilibrium between supply and demand, $(t, m) \mapsto P(t, m)$ is of the form $p(c(t), m)$ with

$$p(c, m) = cm^{-\eta}, \qquad 0 < \eta < 1$$

where

$$c(t) = \begin{cases} c_1, & \text{if } t \leq t_1 \\ \frac{t_2 - t}{t_2 - t_1} c_1 + \frac{t - t_1}{t_2 - t_1} c_2, & \text{if } t \in]t_1, t_2[\\ c_2 & \text{if } t \geq t_2 \end{cases}$$

Hence, the system to be solved is:

$$m'(t) = ku(t) - \lambda m(t), \qquad m(0) = m_0$$

$$u'(t) = (r + \lambda)u(t) - p(c(t), m(t)), \qquad \lim_{t \to +\infty} e^{-(r+\lambda)t} u(t) = 0$$

Because $c(t) = c_2$ after time t_2 we know from the preceding section that the stationary point is unique and that $(u(t), m(t))$ will converge toward this stationary point (u^*, m^*) given by Proposition 1 for $P(m) = c_2 m^{-\eta}$. The only difficulty, due to the forward-backward structure is to compute the trajectory toward equilibrium.

[4] unless the demand function is itself linear

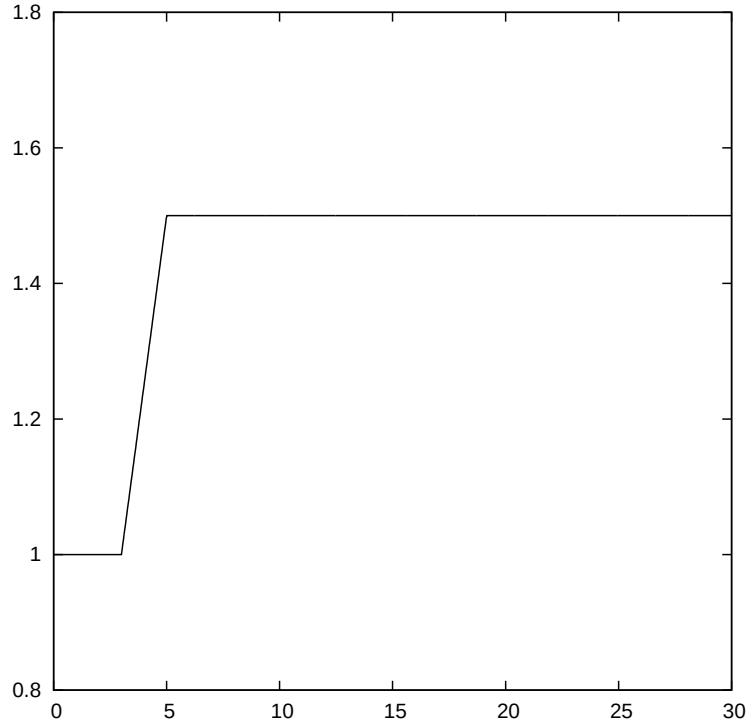


Figure 4: Form of the shock on c with $c_1 = 1$, $c_2 = 1.5$, $t_1 = 3$ and $t_2 = 5$.

In fact, since we can compute the limit (u^*, m^*) either numerically or analytically we can use the same eductive strategy as above.

This eductive strategy consists here in starting with $u^0(t,T) = u^*$ and $m^0(t,T) = m_0$, $\forall t \in [0,T]$ and then computing recursively (u^{n+1}, m^{n+1}) from (u^n, m^n) by:

$$m^{n+1}(t,T) = (1-\theta)m^n(t,T) + \theta\left[m_0 + k\int_0^t u^n(s,T)e^{-\lambda s}ds\right]$$

and

$$u^{n+1}(t,T) = (1-\theta)u^n(t,T) + \theta\left[u^* e^{-(r+\lambda)(T-t)} + \int_t^T p(c(s), m^n(s,T))e^{-(r+\lambda)(s-t)}ds\right]$$

for some small parameter θ and a large horizon of time T.

Another strategy may be used that relies on the second order elliptic equation introduced above. Similar derivations indeed lead to approximate the trajectory followed by m by the unique solution of the following equation:

$$-m''(t,T)+rm'(t,T)+(r+\lambda)\lambda m(t,T)-kp(c(t),m(t,T)) = 0, \qquad m(0,T) = m_0, \qquad m(T,T) = m^*$$

4.1.2 Economic analysis and simulations

To better understand the effect of the shock, we carried out our computations with a starting point m_0 that equals the stationary value the system would attain in the absence of a shock. In our case, if we consider $\eta = \frac{1}{2}$ and the function $c(t)$ as described on Figure 4 with $c_1 = 1$ and $c_2 = 1.5$, this starting point m_0 is the limit value of $m(t)$ as $t \to +\infty$ for a demand function $P(m) = m^{-\frac{1}{2}}$ as in the preceding section (see Figure 2 and Figure 3 for the value of this limit value).

This hypothesis on m_0 allows us to better understand the very nature of the shock. We can indeed consider that before time $t = 0$ the system is at its stationary point corresponding to $c_1 = 1$ and that at time $t = 0$ a shock is announced to be expected at time $t_1 = 3$ – the other parameters being $t_2 = 5$ and $c_2 = 1.5$. Hence at time $t = 0$, the trajectory will jump from the former stationary equilibrium point of Figure 2 and Figure 3 to a new point in the phase diagram with the same value for m but a new (greater) value for u and then follows a continuous trajectory (see Figure 5) toward the limit value (u^*, m^*).

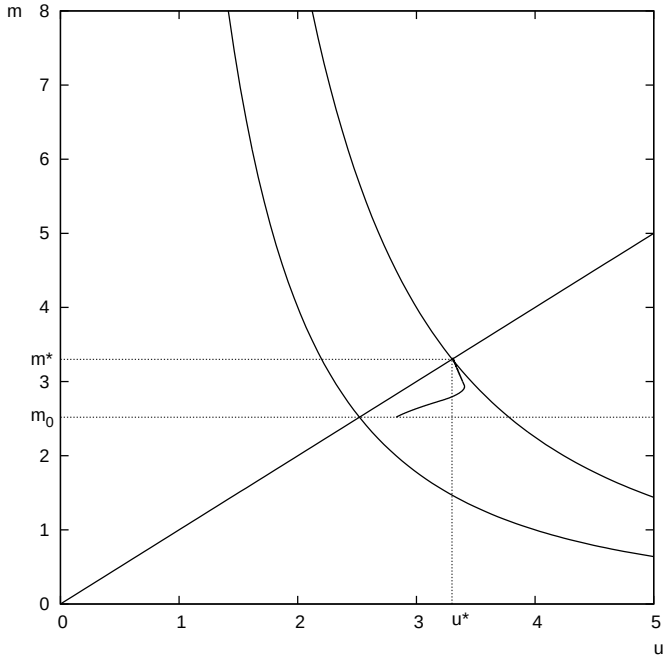


Figure 5: Trajectory in the phase diagram for a permanent shock with c as above, $k = 0.2$, $\lambda = 0.2$, $r = 0.05$ and $\eta = 0.5$. m_0 is chosen as the stationary value when $P(m) = m^{-\frac{1}{2}}$.

We can also look at the trajectory followed by m as a function of time (Figure 6) and we see that once the shock is announced at time 0 the participants in the market start to build production capacity to be ready when the shock is happening. This anticipation induces a decrease in price (Figure 7) before the shock occurs. However, even after $t = t_2$, the equilibrium point is not

attained and the stock of production capacity still rises and induces a decrease in price toward its asymptotic value.

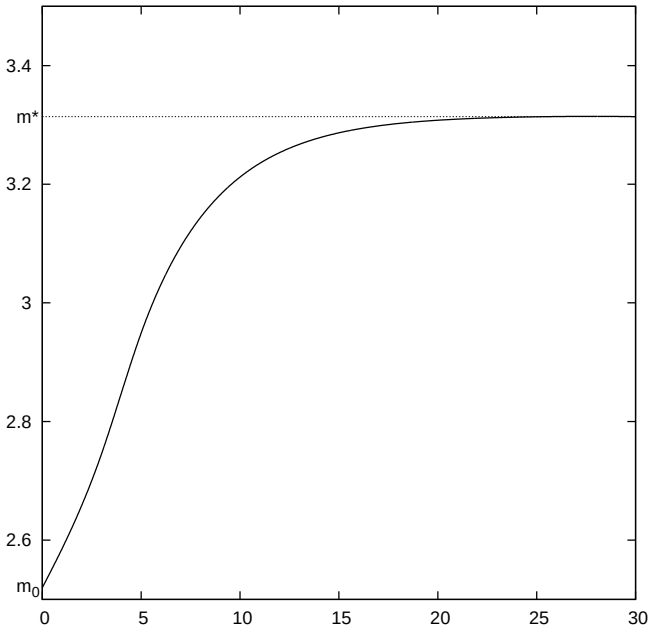


Figure 6: Trajectory of m for a permanent shock with c as above, $k = 0.2$, $\lambda = 0.2$, $r = 0.05$ and $\eta = 0.5$. m_0 is chosen as the stationary value when $P(m) = m^{-\frac{1}{2}}$.

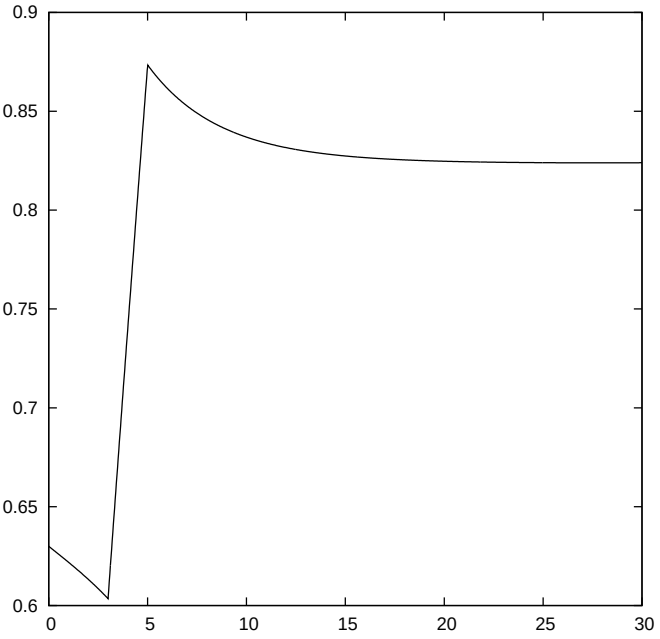


Figure 7: Trajectory of the price for a permanent shock with c as above, $k = 0.2$, $\lambda = 0.2$, $r = 0.05$ and $\eta = 0.5$. m_0 is chosen as the stationary value when $P(m) = m^{-\frac{1}{2}}$.

4.2 Transitory shocks

Let us turn now to the case of a transitory shock. As above, we chose to carry out our computations with a starting point m_0 that equals the stationary value the system would attain in the absence of the shock.

4.2.1 Framework

The transitory shock on demand is modeled through a change in the inverse demand function and we suppose that, at equilibrium between supply and demand, $(t, m) \mapsto P(t, m)$ is of the form $p(c(t), m)$ with

$$p(c, m) = cm^{-\eta}, \qquad 0 < \eta < 1$$

where

$$c(t) = \begin{cases} c_1, & \text{if } t \leq t_1 \\ c_2 - \frac{c_2 - c_1}{t_2 - t_1}|2t - (t_1 + t_2)|, & \text{if } t \in]t_1, t_2[\\ c_1 & \text{if } t \geq t_2 \end{cases}$$

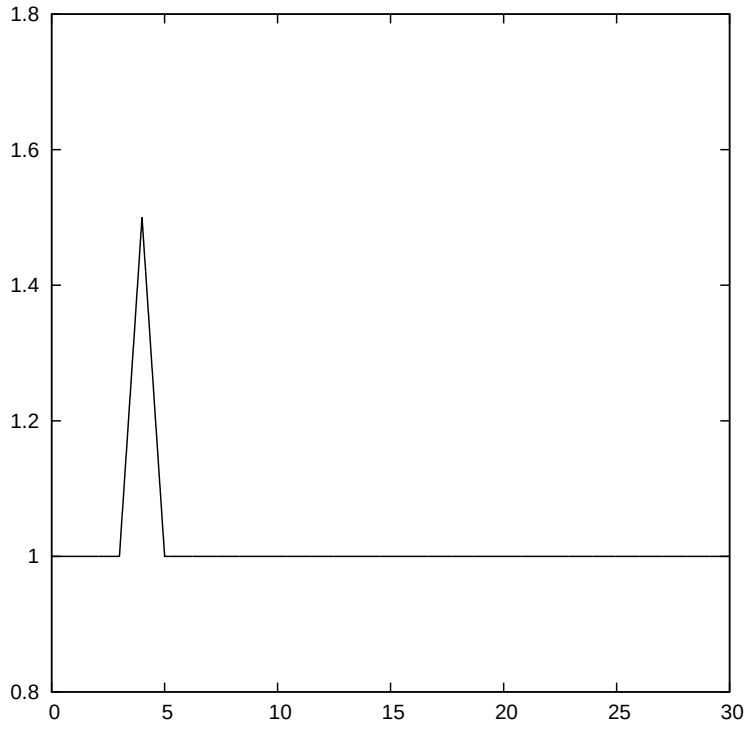


Figure 8: Form of the shock c with $c_1 = 1$, $c_2 = 1.5$, $t_1 = 3$ and $t_2 = 5$.

Because $c(t) = c_1$ after time t_2 we know that the stationary point is unique and that $(u(t), m(t))$ will converge toward this stationary point (u^*, m^*) which is given by Proposition 1 for $P(m) = c_1 m^{-\eta}$. Hence, the same numerical methods as above can be used.

4.2.2 Economic analysis and simulation

The interpretation of the shock and its consequences can be done using the following scenario. Before time $t = 0$, the equilibrium is at its stationary point corresponding to $c_1 = 1$. Then, at time $t = 0$, a shock is announced to be expected at time $t_1 = 3$, lasting until $t_2 = 5$ (the value of c_2 being 1.5). Hence at time $t = 0$, the trajectory will jump from the former stationary equilibrium point of Figure 2 and Figure 3 to a new point in the phase diagram with the same value for m but a new (greater) value for u and then follows a continuous trajectory (see Figure 9) toward the limit value (u^*, m^*) which is nothing else but the initial point before time $t = 0$.

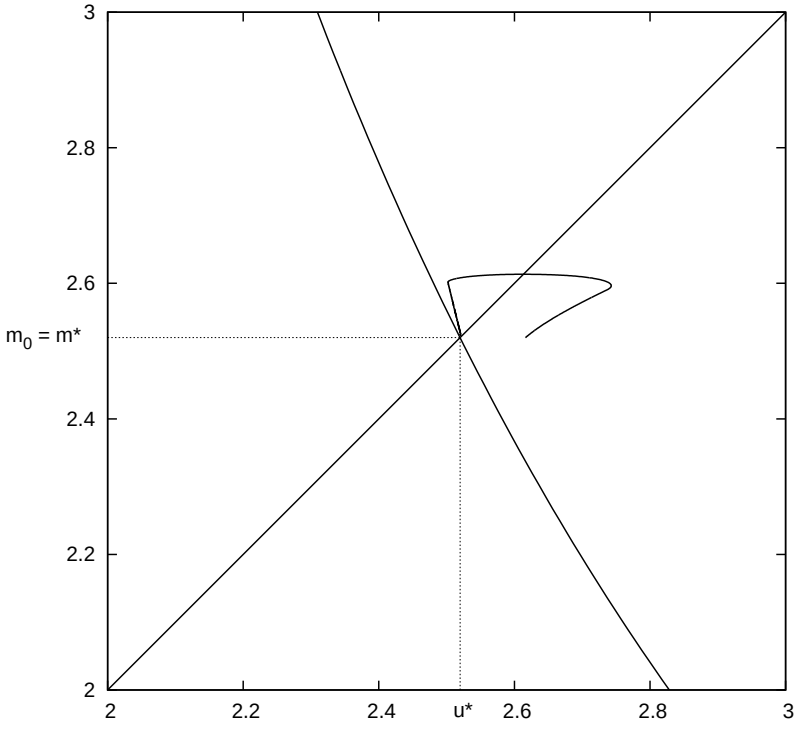


Figure 9: Trajectory in the phase diagram for a transitory shock with c as above, $k = 0.2$, $\lambda = 0.2$, $r = 0.05$ and $\eta = 0.5$. m_0 is chosen as the stationary value when $P(m) = m^{-\frac{1}{2}}$ – a zoom in the area of interest has been done.

We can also look at the trajectory followed by m as a function of time (Figure 10) and we see that once the shock is announced at time 0, the participants in the market start to build production capacity to be ready when the shock actually starts. This anticipation induces a decrease in price (Figure 11) before the shock occurs. Then, after the shock, once the demand level is back to its initial value, there is a production capacity surplus due to the preceding increase in production capacity and the price decreases below its asymptotic value before increasing toward it. Hence, the anticipation of this transitory shock induces a decrease in price both before and after the shock. This is typical of the forward-backward structure that leads to an impact of the

shock both before and after the shock.

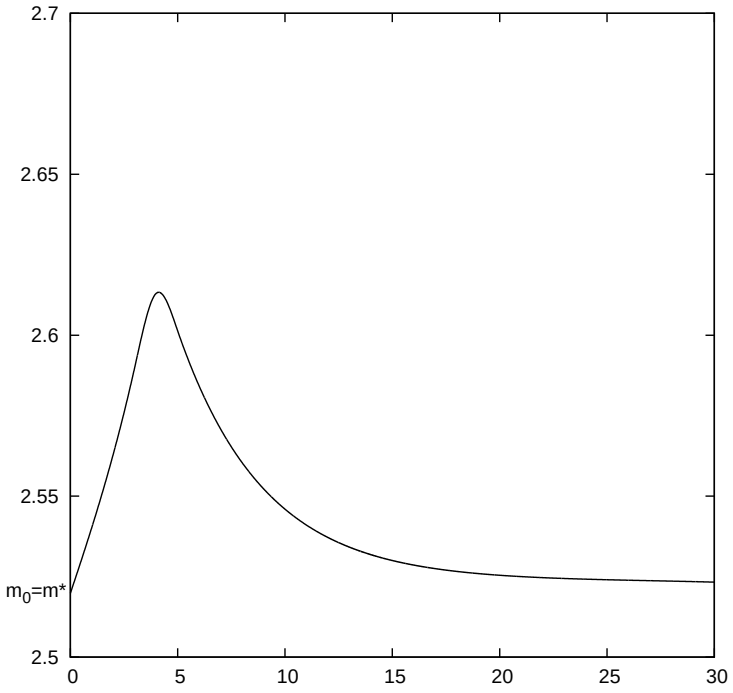


Figure 10: Trajectory of m for a transitory shock with c as above, $k = 0.2$, $\lambda = 0.2$, $r = 0.05$ and $\eta = 0.5$. m_0 is chosen as the stationary value when $P(m) = m^{-\frac{1}{2}}$.

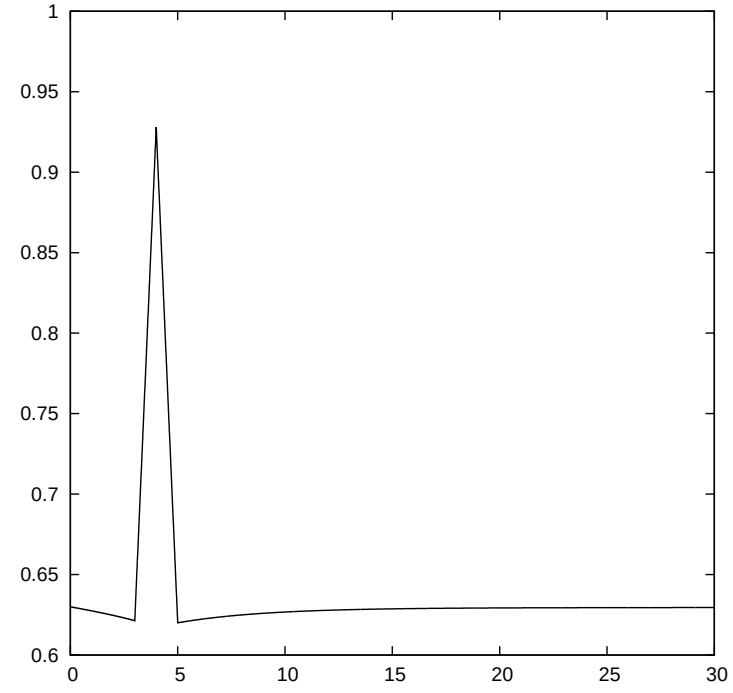


Figure 11: Trajectory of the price for a transitory shock with c as above, $k = 0.2$, $\lambda = 0.2$, $r = 0.05$ and $\eta = 0.5$. m_0 is chosen as the stationary value when $P(m) = m^{-\frac{1}{2}}$.

4.3 The case of a general demand function

In the preceding paragraphs, we studied the model in the case of a demand function with a shock either permanent or transitory. In all these situations the demand function was supposed to be independent of time after a certain period. However, it is important to understand the behavior of the production capacities for a more general demand function that may evolve in a deterministic or random way. In this subsection we will focus on the case of a demand function that evolves in a deterministic fashion and leave the random case to the next section that focuses on a new approach consisting of writing the problem as a macroscopic problem for a benevolent planner.

The inverse demand function we consider is still of the form $p(c(t), m)$ with $(c, m) \in \mathbb{R}_+ \times \mathbb{R}_+^* \mapsto p(c, m) = cm^{-\eta}$ and where $t \to c(t)$ is a given (deterministic) continuous function that is positive[5] and bounded.

In this general framework no phase diagram can be built and the very existence of a solution (u, m) to (FBDS) is at stake.
To deal with this issue, we recall that a solution (u, m) of (FBDS) must satisfy

$$\begin{aligned} u'(t) &= (r+\lambda)u(t) - p(c(t), m(t)) \\ m'(t) &= ku(t) - \lambda m(t) \qquad m(0) = m_0 \end{aligned}$$

and

$$u(0) = \int_0^\infty p(c(t), m(t)) e^{-(r+\lambda)t} dt$$

Now, up to some technical restrictions to avoid negative values for u and m, we can look for a value u_0 of $u(0)$ in coherence with the constraint $u(0) = \int_0^\infty p(c(t), m(t)) e^{-(r+\lambda)t} dt$.

If we consider indeed the forward/backward system

$$\begin{aligned} u'(t) &= (r+\lambda)u(t) - p(c(t), m(t)) \qquad u(0) = u_0 \\ m'(t) &= ku(t) - \lambda m(t) \qquad m(0) = m_0 \end{aligned}$$

we can easily prove that $\forall t, m(t)$ is an increasing and continuous function of u_0.

[5] We have in fact to impose a lower bound strictly greater than 0

Hence, $\int_0^\infty p(c(t), m(t))e^{-(r+\lambda)t}dt$ is a decreasing and continuous function of u_0. As a consequence, there is only one value for u_0 such that the solution of the above forward/backward system verifies $u_0 = u(0) = \int_0^\infty p(c(t), m(t))e^{-(r+\lambda)t}dt$.

Now, to solve the problem numerically in such a general situation, we can use the same ideas as before, although they are not best suited to case for which we do not have information on asymptotic values. Rather, we will use another method that will be useful to solve the problem even when c has a random component. This new method will be exemplified on this model and then presented in a very general case in the next part.

5 A first planning problem

In this section we go back to the initial problem and wonder whether the decentralized problem in which each agent chooses his production capacity is equivalent to a problem faced by a benevolent planner or monopoly that would own all the potential production capacities and make a decision for all producers. We will in fact state the circumstances under which a macroscopic control problem does exist from which we can deduce the solution of the decentralized problem.

5.1 The approach

If we consider the problem of a planner that faces a deterministic inverse demand function $\tilde{P}(t, \cdot)$ and owns all the potential production capacities, this problem is[6]:

$$\sup_{q(t)} \int_0^\infty \left(m(t)\tilde{P}(t, m(t)) - \frac{q(t)^2}{2k} \right) e^{-rt}dt$$
$$\text{s.t.} \qquad m'(t) = q(t) - \lambda m(t), \qquad m(0) = m_0$$

This problem can be solved using a Bellman approach and we introduce the value function $(t, m) \mapsto \Phi(t, m)$ defined by:

$$\Phi(t, m) = \sup_{q} \int_t^\infty \left(m(s)\tilde{P}(s, m(s)) - \frac{q(s)^2}{2k} \right) e^{-r(s-t)}ds$$
$$\text{s.t.} \qquad m'(s) = q(s) - \lambda m(s), \qquad m(t) = m$$

This function Φ verifies the following Hamilton-Jacobi equation:

[6]We recall that to introduce a new production capacity of size qdt between t and $t+dt$, the cost to pay is $\frac{q^2}{2k}dt$.

$$\partial_t \Phi(t,m) - r\Phi(t,m) + m\tilde{P}(t,m) + \sup_q \left(\partial_m \Phi(t,m)(q - \lambda m) - \frac{q^2}{2k} \right) = 0$$

i.e.

$$\partial_t \Phi(t,m) - r\Phi(t,m) + m\tilde{P}(t,m) - \lambda m \partial_m \Phi(t,m) + \frac{k}{2} \left(\partial_m \Phi(t,m) \right)^2 = 0$$

and the optimal control is $q(t,m) = k\partial_m \Phi(t,m)$.

Now, the trajectory followed by m is:

$$m(0) = m_0, \qquad m'(t) = k\partial_m \Phi(t, m(t)) - \lambda m(t)$$

By analogy with the dynamics in the decentralized problem, let us introduce a function v defined by:

$$v(t) = \partial_m \Phi(t, m(t))$$

If we differentiate the Hamilton-Jacobi equation verified by Φ with respect to the variable m we get:

$$\partial_t \partial_m \Phi(t,m) - r\partial_m \Phi(t,m) + \partial_m \left(m\tilde{P}(t,m) \right) - \lambda \partial_m \Phi(t,m)$$
$$-\lambda m \partial^2_{mm} \Phi(t,m) + k\partial^2_{mm} \Phi(t,m) \partial_m \Phi(t,m) = 0$$

Hence,

$$\begin{aligned} v'(t) &= \partial_t \partial_m \Phi(t, m(t)) + m'(t) \partial^2_{mm} \Phi(t, m(t)) \\ &= \partial_t \partial_m \Phi(t, m(t)) + k\partial_m \Phi(t, m(t)) \partial^2_{mm} \Phi(t, m(t)) - \lambda m(t) \partial^2_{mm} \Phi(t, m(t)) \\ &= (r + \lambda) \partial_m \Phi(t, m(t)) - \partial_m \left(m(t) \tilde{P}(t, m(t)) \right) \\ &= (r + \lambda) v(t) - \partial_m \left(m(t) \tilde{P}(t, m(t)) \right) \end{aligned}$$

Since $m'(t) = kv(t) - \lambda m(t)$, we see that if we consider a function $\tilde{P}$ such that $\partial_m \left(m\tilde{P}(t,m) \right) = P(t,m)$ then (v,m) verifies the same equation as (u,m) and hence[7] both u and m can be derived from Φ.

This approach may seem weird at first sight because the demand function has been changed. We argue that to solve the decentralized problem, we can consider the problem faced by a plan-

[7] We will show in the next section a very general result corresponding to this approach, though in finite horizon, and the additional difficulty here is the transversality condition, a difficulty that is not tackled in this text.

ner with a different pay-off. We shall see with a lot more generality, that, if the gradient of this new payoff with respect to m is the payoff of the agents in the initial decentralized problem, then the solution of the decentralized problem can be derived from the solution of the planning problem.

In our case, if $P(t,m) = p(c(t),m) = c(t)m^{-\eta}$, then we need to solve the following Hamilton-Jacobi equation for Φ:

$$\partial_t \Phi(t,m) - r\Phi(t,m) + \frac{1}{1-\eta} c(t) m^{1-\eta} - \lambda m \partial_m \Phi(t,m) + \frac{k}{2} \left(\partial_m \Phi(t,m)\right)^2 = 0$$

with the appropriate transversality condition[8].

5.2 Application to a random demand shifter

In the last paragraphs of this first part, we turn to the case of a random demand shifter $c(t)$. Now $c(t)$ will stand for a continuous stochastic process and we consider for the sake of simplicity – and because it is the most relevant hypothesis in terms of modeling – the case of an Ornstein-Uhlenbeck process:

$$dc(t) = -\alpha(c(t) - \bar{c})dt + \sigma dW_t$$

where W stands for a standard brownian motion independent of the Poisson process introduced above to rule the life cycle of a production capacity.

In this context, there is nothing like a deterministic function $u(t)$ and we need to write the problem faced by each potential participant in the market as depending on $U(c,m)$. The expected intertemporal profit (per unit of production capacity) of an agent entering the market when total supply is m and when the value of the demand shifter is c:

$$U(c,m) = \mathbb{E}\left[\int_0^\infty p(c(t),m(t)) e^{-(r+\lambda)t} dt \mid c(0) = c, m(0) = m\right]$$

with

$$dc(t) = -\alpha(c(t) - \bar{c})dt + \sigma dW_t, \qquad m'(t) = kU(c(t),m(t)) - \lambda m(t)$$

From this definition, U satisfies a partial differential equation that is:

$$p(c,m) - (r+\lambda)U(c,m) + (kU(c,m) - \lambda m)\partial_m U(c,m)$$
$$-\alpha(c-\bar{c})\partial_c U(c,m) + \frac{\sigma^2}{2}\partial^2_{cc} U(c,m) = 0$$

[8] In practice we solve the same equation with a finite horizon T and the final condition $\Phi(T,m) = 0$. Il corresponds to imposing $u(T) = 0$ in the decentralized problem so that the solution is well approximated for $t \ll T$ (and T large enough)

Now, if we use the planning problem approach with the modified demand function associated to $p(c,m) = cm^{-\eta}$, we will see that this equation can be obtained through the differentiation of an Hamilton-Jacobi equation.

Let us indeed, as above, introduce the following planning problem in which the payoff is an antiderivative of $p(c,m)$ with respect to m:

$$\sup_{q(t)} \mathbb{E}\left[\int_0^\infty \left(\frac{1}{1-\eta}c(t)m(t)^{1-\eta} - \frac{q(t)^2}{2k}\right)e^{-rt}dt\right]$$

$$\text{s.t.} \qquad m'(t) = q(t) - \lambda m(t), \quad m(0) = m_0$$

$$\text{and} \qquad dc(t) = -\alpha(c(t) - \bar{c})dt + \sigma dW_t, \quad c(0) = c_0$$

Let us introduce the value function $(c,m) \mapsto \Phi(c,m)$ associated to this problem.

By definition:

$$\Phi(c,m) = \sup_{q(t)} \mathbb{E}\left[\int_0^\infty \left(\frac{1}{1-\eta}c(t)m(t)^{1-\eta} - \frac{q(t)^2}{2k}\right)e^{-rt}dt \mid c(0) = c, m(0) = m\right]$$

$$\text{s.t.} \qquad m'(t) = q(t) - \lambda m(t), \qquad dc(t) = -\alpha(c(t) - \bar{c})dt + \sigma dW_t$$

Φ verifies the following Hamilton-Jacobi-Bellman equation:

$$-r\Phi(c,m) + \frac{1}{1-\eta}cm^{1-\eta} + \sup_q \left((q - \lambda m)\partial_m\Phi(c,m) - \frac{q^2}{2k}\right)$$

$$-\alpha(c - \bar{c})\partial_c\Phi(c,m) + \frac{\sigma^2}{2}\partial^2_{cc}\Phi(c,m) = 0$$

i.e.

$$-r\Phi(c,m) + \frac{1}{1-\eta}cm^{1-\eta} - \lambda m\partial_m\Phi(c,m) + \frac{k}{2}\left(\partial_m\Phi(c,m)\right)^2$$

$$-\alpha(c - \bar{c})\partial_c\Phi(c,m) + \frac{\sigma^2}{2}\partial^2_{cc}\Phi(c,m) = 0$$

Now, if we differentiate this equation with respect to m, we obtain:

$$-r\partial_m\Phi(c,m) + cm^{-\eta} - \lambda\partial_m\Phi(c,m) - \lambda m\partial^2_{mm}\Phi(c,m) + k\partial_m\Phi(c,m)\partial^2_{mm}\Phi(c,m)$$

$$-\alpha(c - \bar{c})\partial_c\partial_m\Phi(c,m) + \frac{\sigma^2}{2}\partial^2_{cc}\partial_m\Phi(c,m) = 0$$

i.e.

$$-(r+\lambda)\partial_m\Phi(c,m) + p(c,m) + \left(k\partial_m\Phi(c,m) - \lambda m\right)\partial^2_{mm}\Phi(c,m)$$

$$-\alpha(c - \bar{c})\partial_c\partial_m\Phi(c,m) + \frac{\sigma^2}{2}\partial^2_{cc}\partial_m\Phi(c,m) = 0$$

and we see that, as announced, $\partial_m \Phi$ verifies the same equation as U.

5.3 Empirical results

The preceding method will be used extensively for the remainder of this paper. To exemplify this, let us start with a particular specification of the above model with a random demand shifter.

First we start with the dynamics for m:

$$m'(t) = k\partial_m \Phi(t, m(t)) - \lambda m(t)$$

with $k = 0.2$ and $\lambda = 0.2$.

Profits are discounted at rate $r = 5\%$.

The inverse demand function is $P(t, m) = c(t)m^{-\eta}$ with $\eta = 0.5$ and c an Ornstein-Uhlenbeck process:

$$dc(t) = -\alpha(c(t) - \bar{c})dt + \sigma dW_t$$

where $\alpha = 0.2$, $\bar{c} = 1$ and $\sigma = 0.15$.

A trajectory for c is drawn for 150 years on Figure 12.

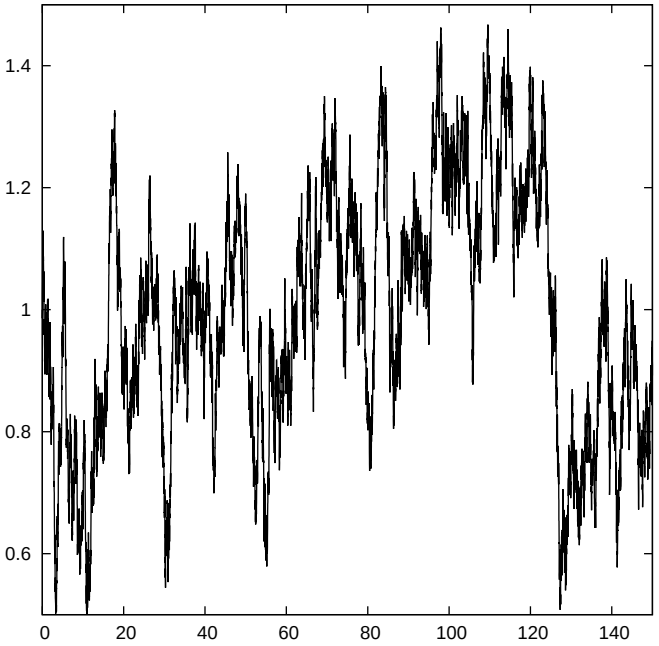


Figure 12: Trajectory of c with $\alpha = 0.2$, $\bar{c} = 1$ and $\sigma = 0.15$. Reflections have been introduced at $c = 0.5$ and $c = 1.5$.

Then Φ is solved using classical methods for stationary solutions of Hamilton-Jacobi-Bellman equations (Figure 13).

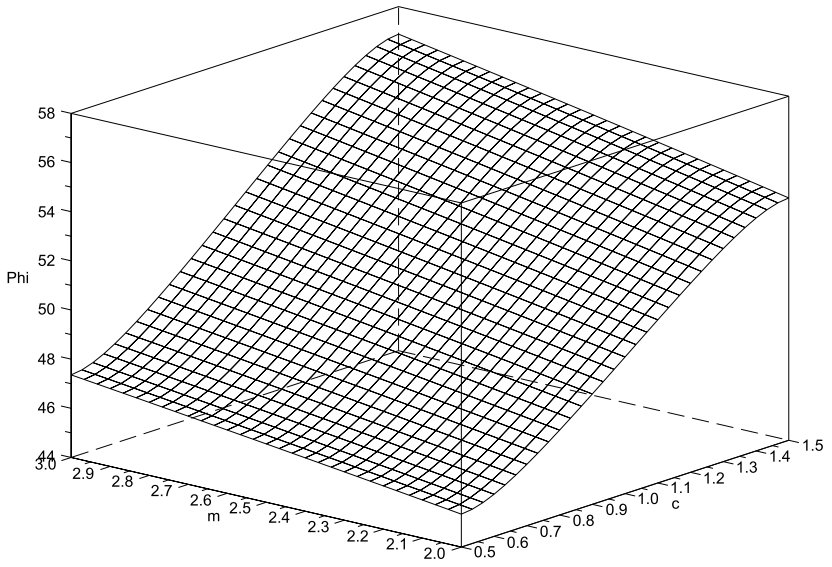


Figure 13: Resolution of the equation for Φ with $c \in [0.5, 1.5]$ (Neumann condition) and $m \in [2, 3]$.

Using the function Φ, we can deduce U (Figure 14).

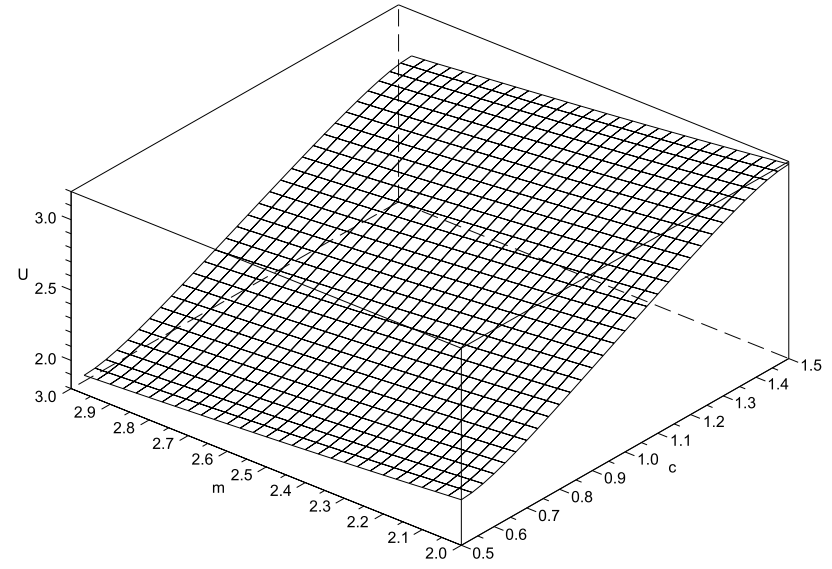


Figure 14: U with $c \in [0.5, 1.5]$ (and Neumann condition) and $m \in [2, 3]$.

From U, we can compute the trajectory for m (Figure 15).

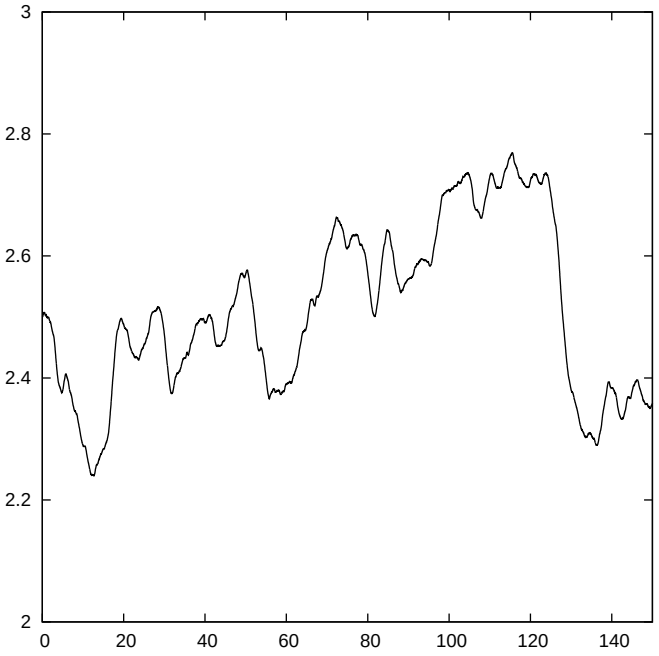


Figure 15: Trajectory of m.

Finally, oil prices can be deduced from the available production capacities (Figure 16).

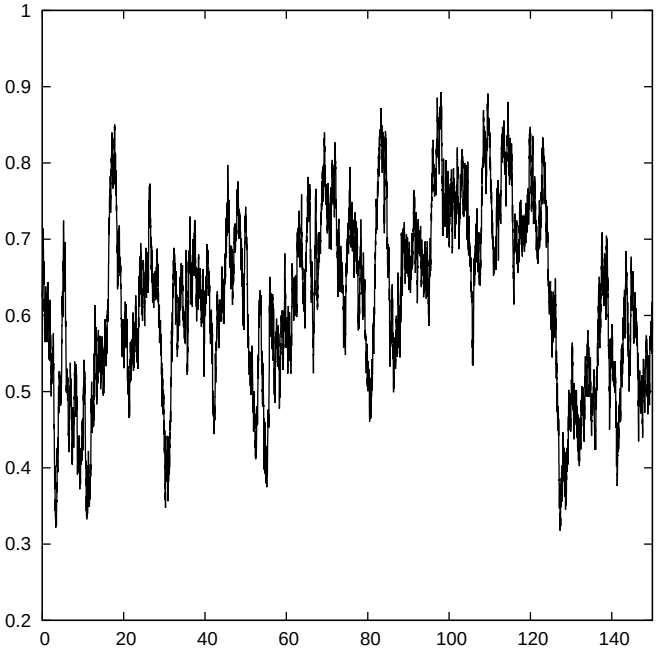


Figure 16: Oil prices.

6 Comments

In the above model, we studied the reaction of the oil producers to demand shocks. Since supply adjustments are costly and can only be done progressively by the choices of each producer entering the market, the demand shocks are instantaneously transformed into prices shocks and then slowly attenuated through quantity adjustments.

If we consider indeed a *textbook* supply-demand graph, we see (Figure 17) that if demand increases instantaneously (the change in demand being unexpected, contrary to the case of section 4 above), instead of going from the equilibrium point A to the new equilibrium point C, the system instantaneously goes to point B (inducing an increase in prices) and then progressively goes along the demand curve toward the equilibrium point C, unless other shocks happen in the meanwhile as it is the case in our numerical examples.

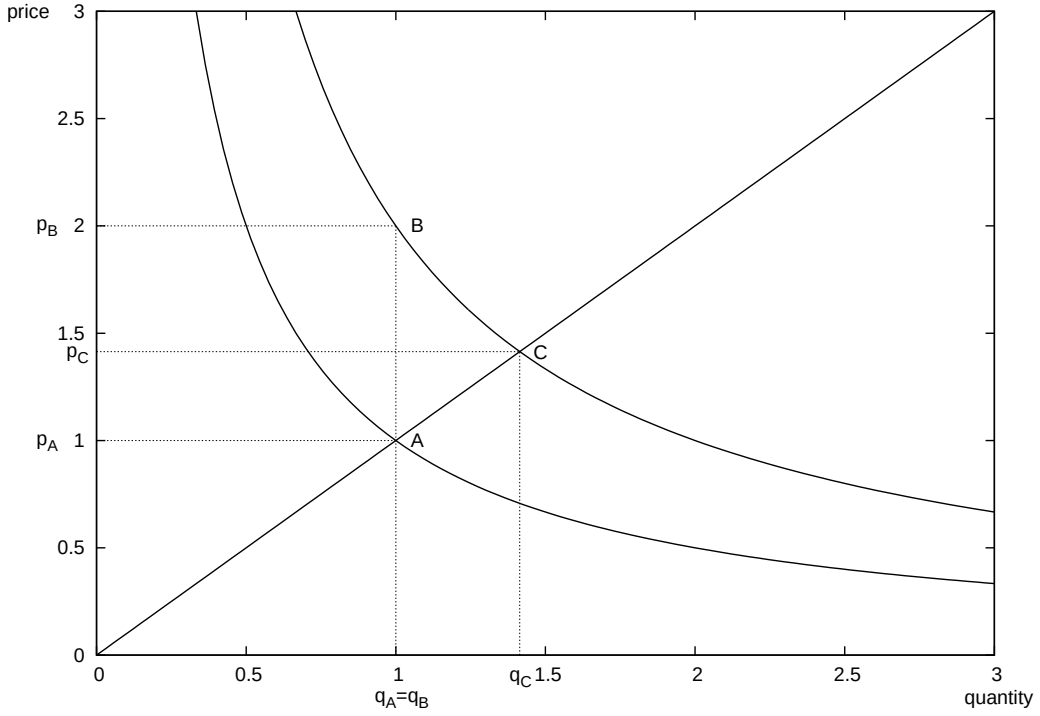


Figure 17: Supply-Demand Graph with/without capacity constraints.

If we look now more carefully at the above numerical results, focusing on the last 20 years, we see that the total supply (Figure 19) roughly follows the trend in the trajectory of c (Figure 18). However, from c to m, the variations are smoothen a lot and in terms of prices, we see (Figure 20) that, due to the form of the demand function, the variation of the demand shifter c is almost perfectly transmitted to prices.

As a consequence, our model appears as a way to describe the complex transmission of a demand shock to prices: the price reaction function.

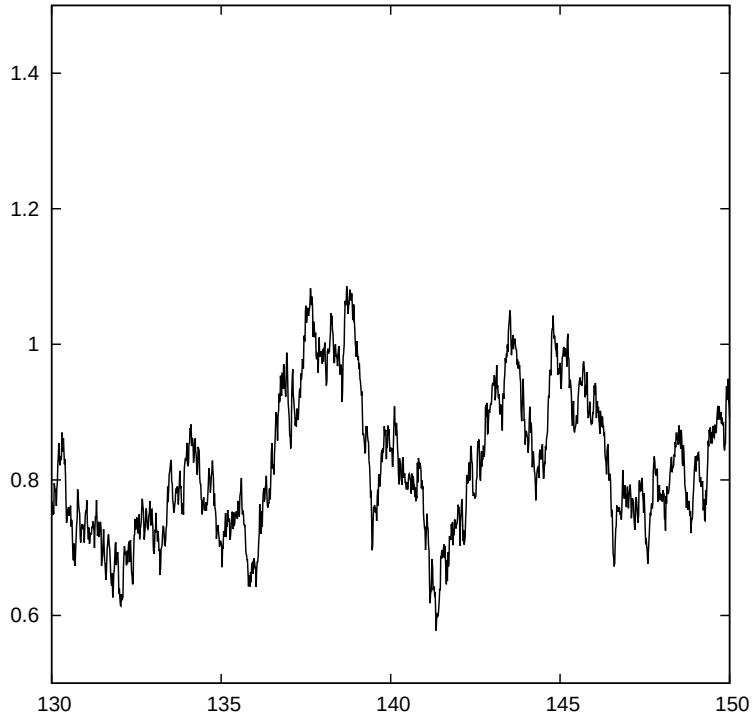


Figure 18: Focus on the trajectory of c with $\alpha = 0.2$, $\bar{c} = 1$ and $\sigma = 0.15$. Reflections have been introduced at $c = 0.5$ and $c = 1.5$.

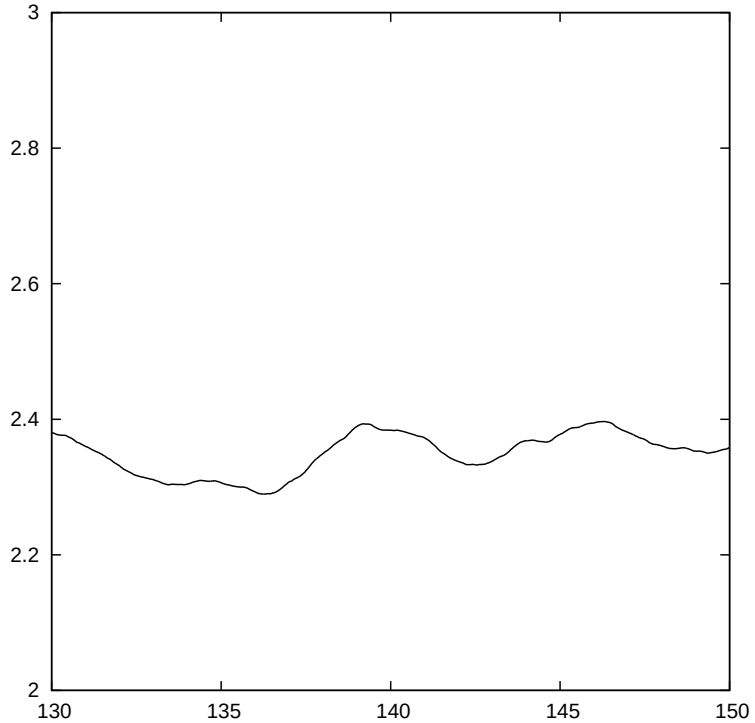


Figure 19: Focus on the trajectory of m.

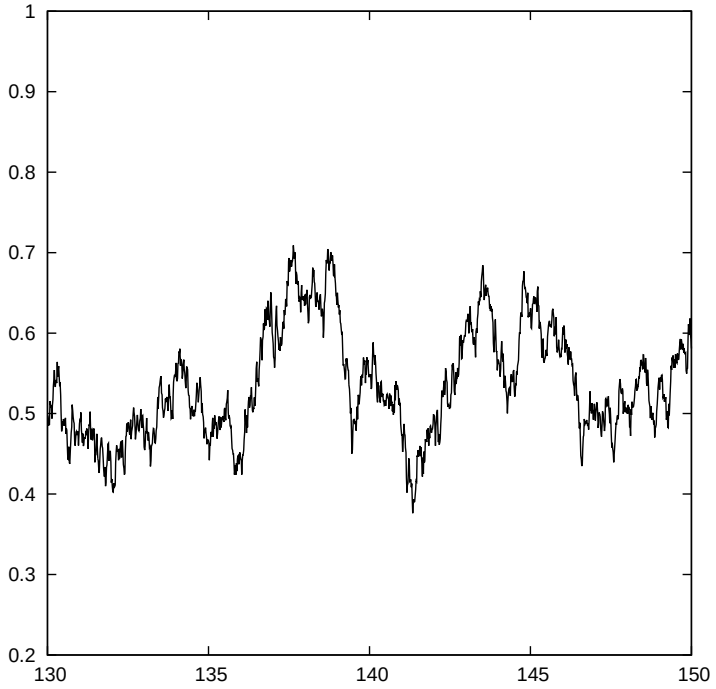


Figure 20: Focus on the oil prices.

But our model is also, and most importantly, able to explain a lot of phenomena depending on the demand function. For instance, taking an inverse demand function of the form $P(c,m) = H\left(\frac{c}{m}\right)$ with $H(z) = \begin{cases} 1, & \text{if } z \leq \frac{1}{2} \\ \frac{1}{2}, & \text{otherwise} \end{cases}$ we can observe a skewness in price distribution and hence a few excursion of prices at the high price (Figures 21 to 23).

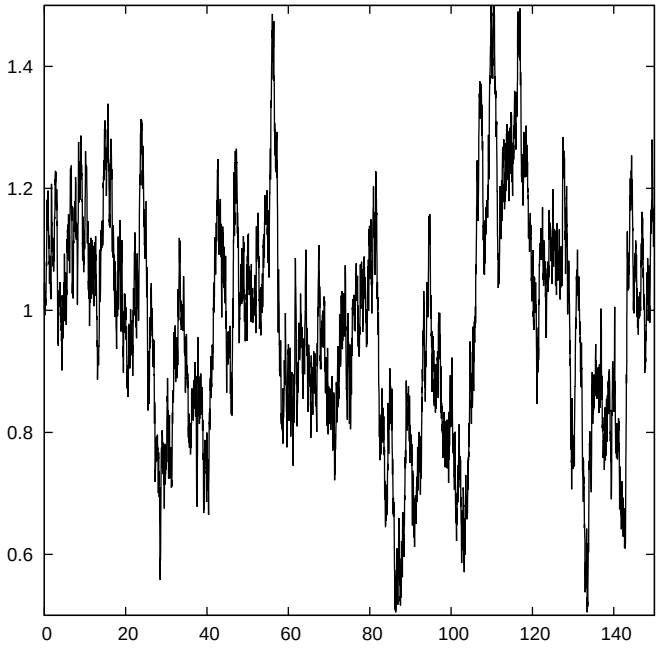


Figure 21: Trajectory of c with $\alpha = 0.2$, $\bar{c} = 1$ and $\sigma = 0.15$.

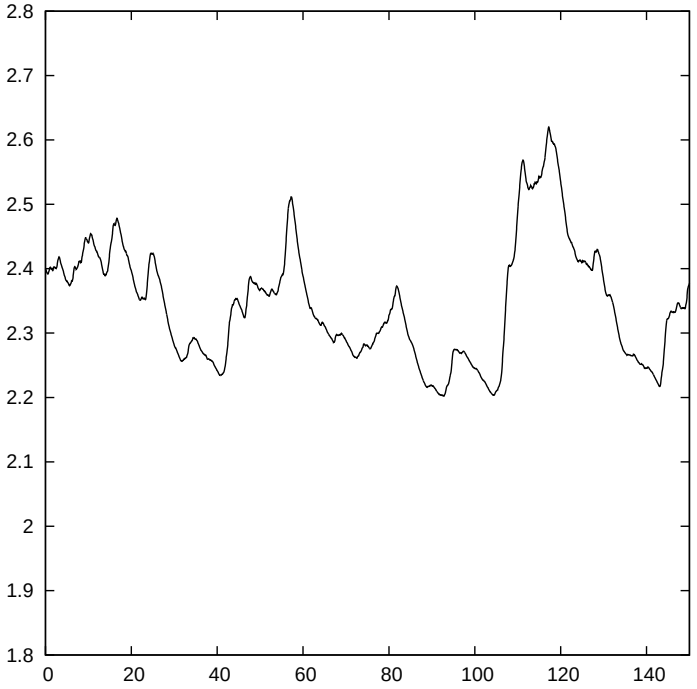


Figure 22: Trajectory of m.

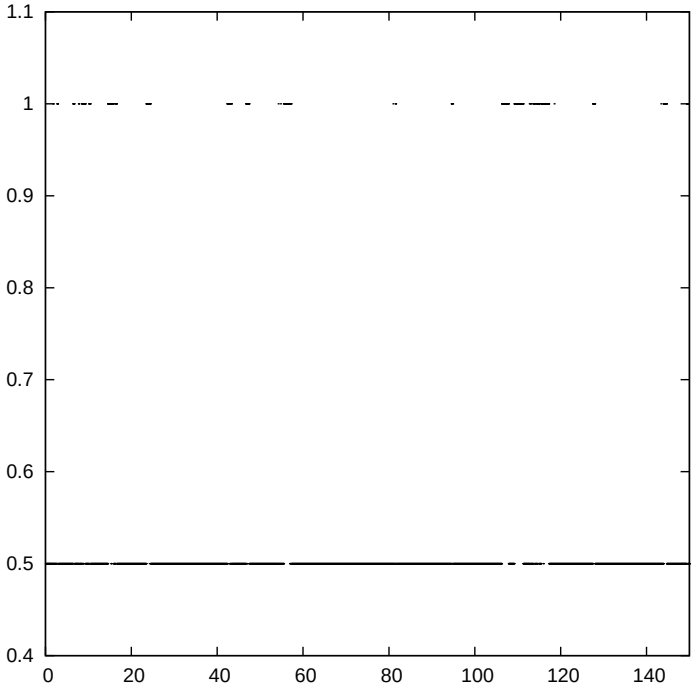


Figure 23: Oil prices.

Part II

The planning problem approach

1 Introduction

The planning problem approach presented in a previous example is in fact a new general approach developed for this work on oil production. It is effectively a new approach to deal with mean field games that relies on an equivalence between decentralized problems and planning problems for some specific games that we call potential games, due to an analogy with some preceding works. A similar approach was indeed used for mean field games in *reduced form*[9] and can be generalized to cases in which the value function depends not only on each player's state but also on the distribution of the players' state. This more general case especially allows for games with common noise.

2 The decentralized problem

We present the result for a discrete state space as it will be used in the remainder of this paper. However, although the result will be used on a simple case for which states are successive states in the production capacity building process, we consider in this part the more general case of a connected graph[10]. The result will be proved in a framework that can have slight variants and it

[9] Let us indeed consider the mean field games equations as in [7, 8, 9]:

$$(HJB) \qquad \partial_t u + \frac{\sigma^2}{2}\Delta u + H(\nabla u) = -f(x,m)$$

$$(K) \qquad \partial_t m + \nabla \cdot (m \partial_p H(\nabla u, m)) = \frac{\sigma^2}{2}\Delta m$$

with

$$u(T,x) = g(x, m_T), \qquad m(0,x) = m_0(x)$$

These equations correspond to a problem in which infinitely many agents try to maximize a similar criterion depending on their own position in the state space and on the distribution of the positions of all the agents or players in the state space. Now, if there exist two functions $m \mapsto F(m)$ and $m \mapsto G(m)$ such that the derivatives of F and G in the variational sense are respectively f and g then m can be derived from the following maximization problem

$$\sup_{(\alpha_t)_{t\geq 0}} \mathbb{E}\left[\int_0^T \left(F(m(t)) - H^*(\alpha_t)\right) dt + G(m(T))\right]$$

$$\text{s.t.} \quad \partial_t m + \nabla \cdot (m\alpha_t) = \frac{\sigma^2}{2}\Delta m, \qquad m(0) = m_0$$

The optimal control is the same and u is the adjoint state associated to this maximization problem.

[10] The result is however proved in the case of a finite horizon problem to avoid problems linked to transversality conditions

must be noted that the only important restriction is that Hamiltonian functions have to be independent of the distribution of the players in the state space (as it was the case for the analogous result concerning mean field games in reduced form). In other words, the result we will prove cannot be generalized to models involving congestion effects such as traffic jam models.

2.1 Setup

Let us consider a state space $\mathcal{S}$ consisting of N states indexed from 1 to N. These states are the nodes of a connected graph and we denote for each $i \in \mathcal{S}$, the neighbors of the node i by $\mathcal{V}_i$. This graph is supposed to be directed and consequently $j \in \mathcal{V}_i$ means that there is an edge from i to j (but not necessarily from j to i).
We consider a continuum of agents. Each agent can be in any state of the state space and the number of agent in state i is denoted m_i.
Each agent's state is a continuous Markov chain that we generically denote X_t.
Transition probabilities are chosen by each agent and we suppose that no edge can be created. In practice, if an agent is in state i (or, equivalently, if $X_t = i$), then he chooses $(\delta_{ij}(t))_{j\in\mathcal{V}_i}$ and the probability for X_{t+dt} to be $j \in \mathcal{V}_i$ is $\delta_{ij}(t)dt$. This choice is costly and we introduce for each state/node i a cost function $C_i((\delta_{ij})_{j\in\mathcal{V}_i})$ that is a convex and increasing function[11].

Each state is associated to a payoff and we associate to state i the instantaneous payoff $f_i(m_1, \dots, m_N)$ and a final payoff $g_i(m_1, \dots, m_N)$.

Now, coming to the objective function, we suppose that each agent maximizes:

$$\sup_{(\delta_{ij}(t))_{i\in\mathcal{S},j\in\mathcal{V}_i,t\geq 0}} \mathbb{E}\left[\int_0^T \left(f_{X_t}(m_1(t),\dots,m_N(t)) - C_{X_t}((\delta_{X_t j})_{j\in\mathcal{V}_{X_t}})\right) e^{-rt}\, dt \right.$$
$$\left. + g_{X_T}(m_1(T),\dots,m_N(T))\right]$$

with

$$\forall i \in \mathcal{S}, m_i'(t) = \sum_{\{k/i\in\mathcal{V}_k\}} \delta_{ki}^*(t) m_k(t) - \sum_{j\in\mathcal{V}_i} \delta_{ij}^*(t) m_i(t)$$

$(\delta_{ij}^*(t))_{i\in\mathcal{S},j\in\mathcal{V}_i}$ being the optimal instantaneous transition probability.

This problem enters the class of general mean field games and we will prove in the next section that, for some payoff functions $(f_1, \dots, f_N)$ and $(g_1, \dots, g_N)$, this problem faced by infinitely many players (and called for this very reason a decentralized problem) derived from an optimization problem faced by a planner that can be solved more easily.

[11] This function equals 0 when $(\delta_{ij})_{j\in\mathcal{V}_i} = 0$.

2.2 Hamilton-Jacobi Equations

Let us start with the Hamilton-Jacobi equations that characterize the decentralized problem. Since there are N states, we need to write down N Hamilton-Jacobi equations, one for each state.

Let us denote by $u_i(t, m_1, \ldots, m_N)$ the value function associated to being in state i at time t when the distribution of agents across the different states is given by the tuple $(m_1, \ldots, m_N)$.

From classical control theory we obtain the following system of partial differential equations consisting of N coupled Hamilton-Jacobi equations:

$$\forall i \in \mathcal{S}, \quad \frac{\partial u_i}{\partial t}(t, m_1, \ldots, m_N) - r u_i(t, m_1, \ldots, m_N) + f_i(m_1, \ldots, m_N)$$
$$+ \sup_{(\delta_{ij})_{j \in \mathcal{V}_i}} \left[\sum_{j \in \mathcal{V}_i} \delta_{ij}(u_j(t, m_1, \ldots, m_N) - u_i(t, m_1, \ldots, m_N)) - C_i((\delta_{ij})_{j \in \mathcal{V}_i}) \right]$$
$$+ \sum_{j=1}^{N} \frac{\partial u_i}{\partial m_j}(t, m_1, \ldots, m_N) \left[\sum_{k/j \in \mathcal{V}_k} \delta^*_{kj}(t) m_k - \sum_{k \in \mathcal{V}_j} \delta^*_{jk}(t) m_j \right] = 0$$

with the terminal value

$$u_i(T, m_1, \ldots, m_N) = g_i(m_1, \ldots, m_N)$$

Now let us introduce the Hamiltonian functions

$$H_i((p_{ij})_{j \in \mathcal{V}_i}) = \sup_{(\delta_{ij})_{j \in \mathcal{V}_i}} \left[\sum_{j \in \mathcal{V}_i} \delta_{ij} p_{ij} - C_i((\delta_{ij})_{j \in \mathcal{V}_i}) \right]$$

We have $\forall i \in \mathcal{S}, \forall j \in \mathcal{V}_i$:

$$\delta^*_{ij}(t) = \frac{\partial H_i}{\partial p_{ij}}((u_j(t, m_1, \ldots, m_N) - u_i(t, m_1, \ldots, m_N))_{j \in \mathcal{V}_i})$$

Hence, the system of Hamilton-Jacobi equations can be written in a simpler way:

$$\forall i \in \mathcal{S}, \quad 0 = \frac{\partial u_i}{\partial t} - r u_i + f_i + H_i((u_j - u_i)_{j \in \mathcal{V}_i})$$
$$+ \sum_{j=1}^{N} \frac{\partial u_i}{\partial m_j} \left[\sum_{\{k/j \in \mathcal{V}_k\}} \frac{\partial H_k}{\partial p_{kj}}((u_j - u_k)_{j \in \mathcal{V}_k}) m_k - \sum_{k \in \mathcal{V}_j} \frac{\partial H_j}{\partial p_{jk}}((u_k - u_j)_{k \in \mathcal{V}_j}) m_j \right]$$

with the terminal condition $u_i(T) = g_i$.

This system of N coupled partial differential equations may *reduce* to a single Hamilton-Jacobi-Bellman equation in the case of "potential games". This is what we are going to prove in the next section.

3 Potential games and reduction to a planning problem

3.1 Theoretical aspects

In this section, we suppose that there exists a function $(m_1, \dots, m_N) \mapsto F(m_1, \dots, m_N)$ such that $\forall i \in \mathcal{S}$, $\frac{\partial F}{\partial m_i} = f_i(m_1, \dots, m_N)$ and similarly, we suppose that there exists a function $(m_1, \dots, m_N) \mapsto G(m_1, \dots, m_N)$ such that $\forall i \in \mathcal{S}$, $\frac{\partial G}{\partial m_i} = g_i(m_1, \dots, m_N)$.

In that case, the game is called a potential game and we have the following result:

Proposition 3. *Let us suppose there exist two functions F and G such that $\nabla F = (f_1, \dots, f_N)'$ and $\nabla G = (g_1, \dots, g_N)'$.*
Let us consider the following optimization problem – called the planning problem.

$$\sup_{(\delta_{ij}(t))_{i\in\mathcal{S},j\in\mathcal{V}_i,t\geq 0}} \left[\int_0^T \left(F(m_1(t), \dots, m_N(t)) - \sum_{i=1}^N C((\delta_{ij}(t))_{j\in\mathcal{V}_i}) m_i \right) e^{-rt} dt + G(m_1(t), \dots, m_N(t)) \right]$$

with

$$\forall i \in \mathcal{S}, \quad m_i'(t) = \sum_{\{k/i\in\mathcal{V}_k\}} \delta_{ki}(t) m_k(t) - \sum_{j\in\mathcal{V}_i} \delta_{ij}(t) m_i(t)$$

If the value function $(t, m_1, \dots, m_N) \mapsto \Phi(t, m_1, \dots, m_N)$ associated to this problem is a smooth function, then we have:

$$\forall i \in \mathcal{S}, \quad u_i = \frac{\partial \Phi}{\partial m_i}$$

and the optimal control $(\delta_{ij}^*(t))_{i\in\mathcal{S},j\in\mathcal{V}_i}$ is the same in both the planning problem and in the decentralized problem.

Proof:

The value function Φ associated to the problem verifies the following Hamilton-Jacobi equation:

$$0 = \frac{\partial \Phi}{\partial t}(t, m_1, \dots, m_N) - r\Phi(t, m_1, \dots, m_N) + F(m_1, \dots, m_N)$$

$$+ \sup_{(\delta_{ij})_{1\leq i\leq N, j\in\mathcal{V}_i}} \sum_{1\leq i\leq N} \left[\frac{\partial\Phi}{\partial m_i}(t, m_1, \ldots, m_N) \left(\sum_{\{k/i\in\mathcal{V}_k\}} \delta_{ki} m_k - \sum_{j\in\mathcal{V}_i} \delta_{ij} m_i \right) - C_i((\delta_{ij})_{j\in\mathcal{V}_i}) m_i \right]$$

with the terminal conditions $\Phi(T, m_1, \ldots, m_N) = G(m_1, \ldots, m_N)$.

Reordering the terms, we get:

$$0 = \frac{\partial\Phi}{\partial t}(t, m_1, \ldots, m_N) - r\Phi(t, m_1, \ldots, m_N) + F(m_1, \ldots, m_N)$$
$$+ \sum_{1\leq i\leq N} m_i \sup_{(\delta_{ij})_{j\in\mathcal{V}_i}} \left[\sum_{j\in\mathcal{V}_i} \delta_{ij} \left(\frac{\partial\Phi}{\partial m_j}(t, m_1, \ldots, m_N) - \frac{\partial\Phi}{\partial m_i}(t, m_1, \ldots, m_N) \right) - C_i((\delta_{ij})_{j\in\mathcal{V}_i}) \right]$$

Using the Hamiltonian functions $(H_i)_i$ introduced previously, this equation becomes:

$$0 = \frac{\partial\Phi}{\partial t} - r\Phi + F + \sum_{1\leq i\leq N} m_i H_i \left(\left(\frac{\partial\Phi}{\partial m_j} - \frac{\partial\Phi}{\partial m_i} \right)_{j\in\mathcal{V}_i} \right)$$

If we differentiate this equation with respect to m_k we get:

$$0 = \frac{\partial^2\Phi}{\partial t \partial m_k} - r\frac{\partial\Phi}{\partial m_k} + \frac{\partial F}{\partial m_k} + H_k \left(\left(\frac{\partial\Phi}{\partial m_j} - \frac{\partial\Phi}{\partial m_k} \right)_{j\in\mathcal{V}_k} \right)$$
$$+ \sum_{1\leq i\leq N} m_i \sum_{j\in\mathcal{V}_i} \left(\frac{\partial^2\Phi}{\partial m_j \partial m_k} - \frac{\partial^2\Phi}{\partial m_i \partial m_k} \right) \frac{\partial H_i}{\partial p_{ij}} \left(\left(\frac{\partial\Phi}{\partial m_j} - \frac{\partial\Phi}{\partial m_i} \right)_{j\in\mathcal{V}_i} \right)$$

Hence, denoting $\forall i \in \mathcal{S}, v_i = \frac{\partial\Phi}{\partial m_i}$, we get:

$$0 = \frac{\partial v_k}{\partial t} - r v_k + f_k + H_k \left((v_j - v_k)_{j\in\mathcal{V}_k} \right)$$
$$+ \sum_{1\leq i\leq N} m_i \sum_{j\in\mathcal{V}_i} \left(\frac{\partial v_k}{\partial m_j} - \frac{\partial v_k}{\partial m_i} \right) \frac{\partial H_i}{\partial p_{ij}} \left((v_j - v_i)_{j\in\mathcal{V}_i} \right)$$

Reordering the terms we get:

$$0 = \frac{\partial v_k}{\partial t} - r v_k + f_k + H_k \left((v_j - v_k)_{j\in\mathcal{V}_k} \right)$$
$$+ \sum_{1\leq i\leq N} \frac{\partial v_k}{\partial m_i} \left[\sum_{\{j/i\in\mathcal{V}_j\}} \frac{\partial H_j}{\partial p_{ji}}((v_i - v_j)_{i\in\mathcal{V}_j}) m_j - \sum_{j\in\mathcal{V}_i} \frac{\partial H_i}{\partial p_{ij}}((v_j - v_i)_{j\in\mathcal{V}_i}) m_i \right]$$

Since $v_k(T) = \frac{\partial G}{\partial m_k} = g_k$, we indeed have that $v_k = \frac{\partial\Phi}{\partial m_k} = u_k$ the value function of the decentralized problem.

Moreover, the optimal control in the planning problem are characterized by:

$$\delta_{ij}^* = \frac{\partial H_i}{\partial p_{ij}} \left(\left(\frac{\partial \Phi}{\partial m_j} - \frac{\partial \Phi}{\partial m_i} \right)_{j \in \mathcal{V}_i} \right)$$

and these equations are the same as in the decentralized problem.

□

The above proposition shows that we can solve the decentralized problem thanks to the introduction of a planning problem. Hence, instead of N Hamilton-Jacobi equations, we only have one Hamilton-Jacobi equation to solve.

It is also noteworthy that this result can be slightly generalized to cases in which there are exogenous sources of new agents entering the system at some nodes. This straightforward generalization will be used in our model for oil production.

3.2 Practical use

The above result applies for potential games and this class of games may seem to be quite small since the payoff functions $(f_1, \dots, f_N)$ associated to the states must come from a single function F (with $\nabla F = (f_1, \dots, f_N)'$) and similarly for the terminal payoff[12].

In practice, a simple class of potential games corresponds to games for which payoff functions are local functions in the sense that $\forall i \in \mathcal{S}$, $f_i(m_1, \dots, m_N)$ only depends on m_i.
In that case, denoting the function f_i by $f_i : m_i \mapsto f_i(m_i)$, the function F entering the planning problem's objective function can simply be:

$$F(m_1, \dots, m_N) = \sum_{i=1}^{N} F_i(m_i)$$

where $\forall i \in \mathcal{S}$, F_i is an antiderivative of f_i.

Models of this kind are quite common and the model we will present for production capacity building within the oil industry enters this category. In our models, the different states will correspond to different stages in the building of production capacities and the payoff functions will be 0 for all states but one: the last one corresponding to actual production.

[12] In practice, we will often consider infinite horizon problem. Hence, the condition on the terminal payoff will be assumed to be 0 in numerical computations and we only need to focus on the payoff functions $(f_1, \dots, f_N)$.

Part III

A production capacity building model

In this part, we present the central application of the planning problem approach to the "time-to-build" literature. Oil industry is indeed characterized by a long industrial process going from geological studies to actual oil extraction and this process involves risk at each step. The geological studies may induce drilling at some place where drilling is more complicated and costly than expected, drilling may eventually end up to a dry well, geopolitical or social tensions may slow the extraction process, etc.
In terms of modeling it means that an increase in demand cannot be immediately satisfied by new supply and will rather induce an increase in prices. New projects to build production capacities may be launched but these projects take time to result in new production capacities, if at all.

In the first section of this part, we build the first brick of the model that only involves production. To take account of the risk involved in extraction and to model the fact that this risk may be partially controlled, we opted for a crude modeling in which producers pay a cost to reduce the probability that a given production unit breaks down (the production unit being definitively lost in that case). Then, the "time-to-build" model is presented with several steps before ending up to a production unit. Each step corresponds to an industrial stage and the risk is modeled by a probability to go from one industrial stage to the next one, each producer paying a cost to foster this transition from one step to the next.

1 A first brick

1.1 The model

Let us start with a model in which production capacities are immediately available for production. Contrary to the models of the first part in which the producer controls new production capacities, we suppose now that there is a constant exogenous flow of new producers with a unitary production capacity and that each producer controls, up to a cost, the probability of a break down or equivalently the probability of staying in the market. The advantage of such a modeling over the models of the first part is that it can be generalized to take account of "time-to-build" effects.

Each producer with unitary production capacity is going to maximize an objective function similar to the objective function of the first models. The difference is that optimization is not

anymore on the production capacity itself but rather on the intensity of the Poisson process that governs the final failure of the production unit.

Denoting λ as above, the intensity of the controlled Poisson process, the optimization problem is:

$$\sup_{(\lambda(t))_t} \mathbb{E}\left[\int_0^\tau \left(p(c(t), m(t)) - C_{prod}(\lambda(t))\right) e^{-rt} dt\right]$$

where the stopping time τ is linked to λ by $\mathbb{P}\left(\tau \leq t + dt | \tau > t\right) = \lambda(t)dt$.
C_{prod} is a positive and decreasing function such as $C_{prod}(\lambda) = \frac{1}{2k\lambda^2}$ since one needs to pay more to have a better chance to use a given production capacity for a long time.
Now, if we choose a dynamics for c identical to the one used in the preceding models ($dc(t) = -\alpha(c(t) - \bar{c})dt + \sigma dW_t$), then the number m of available production capacities satisfies the following equation:

$$m'(t) = 1 - \lambda^*(c(t), m(t))m(t)$$

where λ^* is the optimal control that only depends on c and m.

1.2 Hamilton-Jacobi-Bellman equation and the counterpart planning problem

Let us denote $(c, m) \mapsto U(c, m)$ the value function associated to this decentralized problem.

U verifies the following partial differential equation:

$$p(c, m) - rU(c, m) + \partial_m U(c, m)(1 - \lambda^*(c, m)m) + \sup_\lambda \left(-\lambda U - C_{prod}(\lambda)\right)$$

$$-\alpha(c - \bar{c})\partial_c U(c, m) + \frac{\sigma^2}{2}\partial^2_{cc} U(c, m) = 0$$

If we introduce $H_{prod}(p) = \sup_\lambda \left(\lambda p - C_{prod}(\lambda)\right)$, then the optimal control is:

$$\lambda^*(c, m) = H'_{prod}(-U(c, m))$$

Hence, the partial differential equation satisfied by U is:

$$p(c, m) - rU(c, m) + \partial_m U(c, m)(1 - H'_{prod}(-U(c, m))m) + H_{prod}(-U(c, m))$$

$$-\alpha(c - \bar{c})\partial_c U(c, m) + \frac{\sigma^2}{2}\partial^2_{cc} U(c, m) = 0$$

This equation derives for the Hamilton-Jacobi-Bellman equation of a global problem. Let us indeed consider a benevolent planner that faces the following problem:

$$\sup_{(\lambda(t))_t} \int_0^\infty (\pi(c(t), m(t)) - C_{prod}(\lambda(t))m(t))\, e^{-rt} dt$$

$$\text{s.t.} \quad m'(t) = 1 - \lambda(t)m(t)$$

with $\frac{\partial \pi}{\partial m} = p$.

Then the value function Φ of this problem satisfies:

$$\pi(c, m) - r\Phi(c, m) + \sup_\lambda (\partial_m \Phi(c, m)(1 - \lambda m) - C_{prod}(\lambda)m)$$

$$-\alpha(c - \bar{c})\partial_c \Phi(c, m) + \frac{\sigma^2}{2}\partial^2_{cc}\Phi(c, m) = 0$$

This can be equivalently written as:

$$\pi(c, m) - r\Phi(c, m) + \partial_m \Phi(c, m) + m \sup_\lambda (-\lambda \partial_m \Phi(c, m) - C_{prod}(\lambda))$$

$$-\alpha(c - \bar{c})\partial_c \Phi(c, m) + \frac{\sigma^2}{2}\partial^2_{cc}\Phi(c, m) = 0$$

i.e.:

$$\pi(c, m) - r\Phi(c, m) + \partial_m \Phi(c, m) + mH_{prod}(-\partial_m \Phi(c, m))$$

$$-\alpha(c - \bar{c})\partial_c \Phi(c, m) + \frac{\sigma^2}{2}\partial^2_{cc}\Phi(c, m) = 0$$

As in Proposition 3 of the preceding part, we see that U and $\partial_m \Phi$ satisfy the same equation.

Now, we want to add some steps before a production capacity is available for production and this is the purpose of the next section.

2 Time to build

2.1 Setup of the model

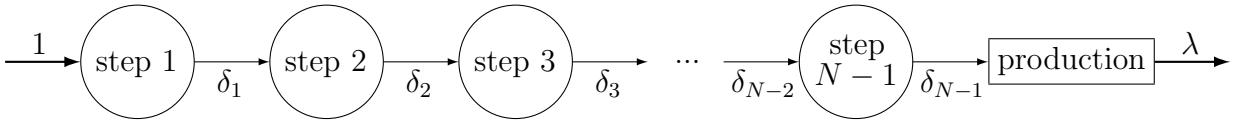
Let us suppose that the industrial process to obtain an actual production capacity is made of $N - 1$ steps. Flows of new possible producers are the same as above but they enter the system with a project to build one unit of production capacity (the project being initially in step 1).

We model the progress in production capacity building through an instantaneous probability to go from a given step to the next one. We indeed introduce a set of instantaneous probability $\delta_1, \ldots, \delta_{N-1}$ where $\delta_i dt$ represents the probability to go from step i to the next one between t and $t + dt$, step N corresponding to actual production.

Our hypothesis is now that each producer can foster the process and pays a cost $C_i(\delta_i)$ to have an instantaneous probability δ_i associated to step i.
Once the $N - 1$ steps has been gone through, the production capacity is available for actual production and the model is the same as in the preceding section.

Hence, the model, although there is a source term, enters the framework developed in the preceding part with N nodes (here $N - 1$ stages with no payoff and a particular state associated to production). In this last state (sometimes denoted N for convenience), the payoff is $p(c, m)$. Edges of the graph are between successive nodes to model the transition from one step to the following one.

The model can be summed up by the following graph representation:



2.2 Partial differential equations

In this mean field game, each agent computes an optimal control that takes the form of a N-tuple $(\delta_1^*, \ldots, \delta_{N-1}^*, \lambda^*)$. Because agents are identical, although they are not all in the same state, the optimal controls are identical across agents and we can consider $(\delta_1^*, \ldots, \delta_{N-1}^*, \lambda^*)$ to be the instantaneous probability of the Markov chain that describe the system as a whole. Hence, if we denote by $m_1, \ldots, m_N$ the number of producers in each of the $N - 1$ steps of production capacity building and if, for notational simplification, we denote by m_N the current available total production capacity, then the dynamics of the m_is are:

$$\begin{aligned} m_1'(t) &= 1 - \delta_1^*(t) m_1(t) \\ m_i'(t) &= \delta_{i-1}^*(t) m_{i-1}(t) - \delta_i^*(t) m_i(t), \qquad 2 \leq i \leq N-1 \\ m_N'(t) &= \delta_{N-1}^*(t) m_{N-1}(t) - \lambda^*(t) m_N(t) \end{aligned}$$

Now, turning to the determination of the optimal control $(\delta_1^*, \ldots, \delta_{N-1}^*, \lambda^*)$, we have to introduce a value function for each $N-1$ step and a value function for the production state. Since we can restrict ourselves to stationary solutions[13], let us denote these value functions $u_1(c, m_1, \ldots, m_N)$ for state 1, ..., $u_{N-1}(c, m_1, \ldots, m_N)$ for state $N-1$ and eventually $u_N(c, m_1, \ldots, m_N)$ for the value function associated to the production state (similarly, the optimal control will now depend on $(c, m_1, \ldots, m_N)$).

First, we can write the $N-1$ equations satisfied by $(u_1, \ldots, u_{N-1})$:

$$\forall 1 \leq i \leq N-1, \qquad 0 = -ru_i - \alpha(c-\bar{c})\partial_c u_i + \frac{\sigma^2}{2}\partial^2_{cc} u_i$$
$$+(1-\delta_1^* m_1)\partial_{m_1} u_i + \sum_{j=2}^{N-1}(\delta_{j-1}^* m_{j-1} - \delta_j^* m_j)\partial_{m_j} u_i$$
$$+(\delta_{N-1}^* m_{N-1} - \lambda^* m_N)\partial_{m_N} u_i + \sup_{\delta_i}\left(\delta_i(u_{i+1}-u_i) - C_i(\delta_i)\right)$$

and denoting $H_i(p) = \sup_\delta \left(\delta p - C_i(\delta_i)\right)$, we have:

$$\delta_i^* = H_i'(u_{i+1} - u_i)$$

Now, since there is a payoff when production occurs, the equation for u_N is different from the preceding ones and we get:

$$\forall 1 \leq i \leq N-1, \qquad 0 = -ru_N + p(c, m_N) - \alpha(c-\bar{c})\partial_c u_N + \frac{\sigma^2}{2}\partial^2_{cc} u_N$$
$$+(1-\delta_1^* m_1)\partial_{m_1} u_N + \sum_{j=2}^{N-1}(\delta_{j-1}^* m_{j-1} - \delta_j^* m_j)\partial_{m_j} u_N$$
$$+(\delta_{N-1}^* m_{N-1} - \lambda^* m_N)\partial_{m_N} u_N + \sup_{\lambda}\left(-\lambda u_N - C_{prod}(\lambda)\right)$$

and denoting $H_{prod}(p) = \sup_\lambda \left(\lambda p - C_{prod}(\lambda)\right)$, we have:

$$\lambda^* = H_{prod}'(-u_N)$$

Now, in the framework of the preceding part, the payoffs associated to states 1 to $N-1$ are 0 and the payoff associated to the production state is $p(c, m_N)$. Hence, we consider an antiderivative $\pi(c, m_N)$ of $p(c, m_N)$ with respect to m_N. We know there exists a planning problem equivalent to our problem. Thus, we can solve only one partial differential equation, then find the optimal control $(\delta_1^*, \ldots, \delta_{N-1}^*, \lambda^*)$ and eventually compute the equilibrium dynamics for supply and prices.

[13] the time dependence being absorbed by the variable c.

The problem of the social planner as in Proposition 3 of the preceding part writes here:

$$\sup_{(\delta_1,\ldots,\delta_{N-1},\lambda)} \int_0^\infty \left(\pi(c(t), m_N(t)) - \left(\sum_{i=1}^{N-1} C_i(\delta_i(t)) m_i(t) \right) - C_{prod}(\lambda(t)) m_N(t) \right) e^{-rt} dt$$

where:

$$\begin{aligned} m_1'(t) &= 1 - \delta_1(t) m_1(t) \\ m_i'(t) &= \delta_{i-1}(t) m_{i-1}(t) - \delta_i(t) m_i(t), \qquad 2 \leq i \leq N-1 \\ m_N'(t) &= \delta_{N-1}(t) m_{N-1}(t) - \lambda(t) m_N(t) \end{aligned}$$

If Φ is the value function associated to this problem then Φ verifies the following Hamilton-Jacobi equation:

$$0 = -r\Phi + \pi(c, m_N) - \alpha(c - \bar{c})\partial_c \Phi + \frac{\sigma^2}{2} \partial^2_{cc} \Phi$$

$$+ \sup_{(\delta_1,\ldots,\delta_{N-1},\lambda)} \left[(1 - \delta_1 m_1)\partial_{m_1}\Phi + \sum_{j=2}^{N-1} (\delta_{j-1} m_{j-1} - \delta_j m_j)\partial_{m_j}\Phi \right.$$

$$\left. + (\delta_{N-1} m_{N-1} - \lambda m_N)\partial_{m_N}\Phi - \sum_{j=1}^{N-1} C_j(\delta_j) m_j - C_{prod}(\lambda) m_N \right]$$

Reordering the terms we obtain:

$$0 = -r\Phi + \pi(c, m_N) - \alpha(c - \bar{c})\partial_c \Phi + \frac{\sigma^2}{2} \partial^2_{cc} \Phi$$

$$+ \partial_{m_1}\Phi + \sum_{j=1}^{N-1} m_j \sup_{\delta_j} \left(\delta_j (\partial_{m_{j+1}}\Phi - \partial_{m_j}\Phi) - C_j(\delta_j) \right) + m_N \sup_{\lambda} \left(-\partial_{m_N}\Phi \lambda - C_{prod}(\lambda) \right)$$

Hence, using the hamiltonian functions introduced before we get:

$$0 = -r\Phi + \pi(c, m_N) - \alpha(c - \bar{c})\partial_c \Phi + \frac{\sigma^2}{2} \partial^2_{cc} \Phi$$

$$+ \partial_{m_1}\Phi + \sum_{j=1}^{N-1} m_j H_j(\partial_{m_{j+1}}\Phi - \partial_{m_j}\Phi) + m_N H_{prod}\left(-\partial_{m_N}\Phi \right)$$

and the optimal controls are:

$$\forall 1 \leq j \leq N-1, \qquad \delta_j^* = H_j' \left(\partial_{m_{j+1}}\Phi - \partial_{m_j}\Phi \right)$$

and

$$\lambda^* = H'_{prod}(-\partial_{m_N}\Phi)$$

3 Numerics

We present below numerical examples for the model we presented above. First, we focus on the $N = 1$ case and then consider the timeto build model with $N = 2$.

3.1 The model with $N = 1$

We first consider the $N = 1$ case.
The cost function associated to λ is $C_{prod}(\lambda) = \frac{1}{2k\lambda^2}$ with $k = 12$.
The inverse demand function is $P(t, m) = c(t)m^{-\eta}$ with $\eta = 0.5$ and c an Ornstein-Uhlenbeck process:

$$dc(t) = -\alpha(c(t) - \bar{c})dt + \sigma dW_t$$

where $\alpha = 0.2$, $\bar{c} = 1$ and $\sigma = 0.15$.
Profits are discounted at rate $r = 25\%$. A trajectory for c is drawn for 150 years on Figure 24.

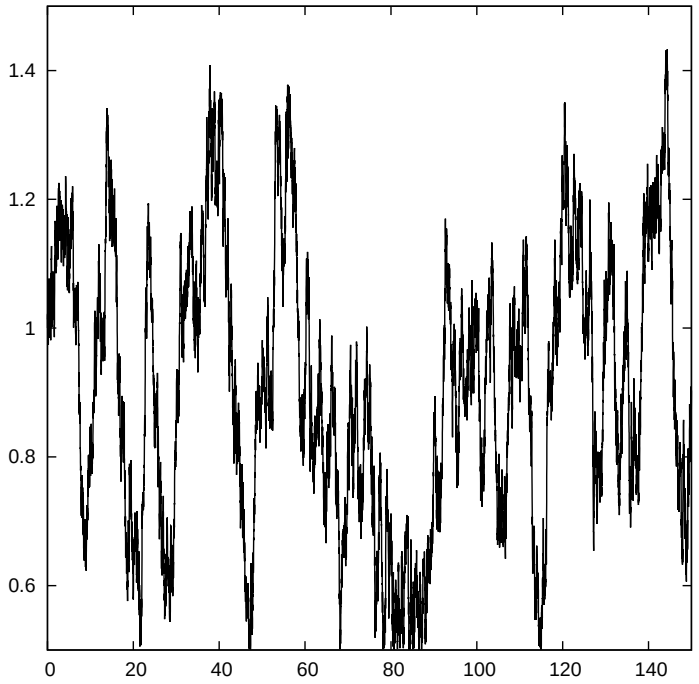


Figure 24: Trajectory of c with $\alpha = 0.2$, $\bar{c} = 1$ and $\sigma = 0.15$. Reflections have been introduced at $c = 0.5$ and $c = 1.5$.

Then Φ is solved using implicit methods for stationary solutions of Hamilton-Jacobi-Bellman

equations (Figure 25).

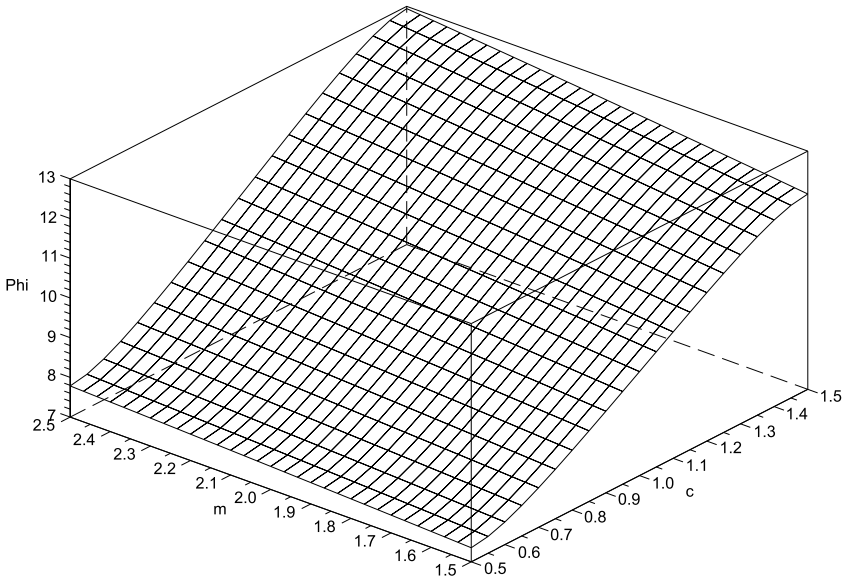


Figure 25: Resolution of the equation for Φ with $c \in [0.5, 1.5]$ (Neumann condition) and $m \in [2, 3]$.

Using the function Φ, we can deduce U (Figure 26).

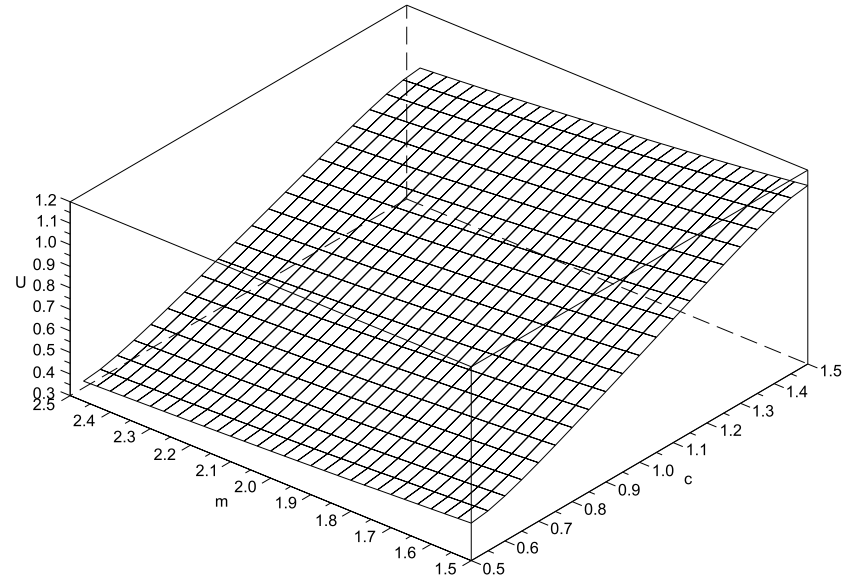


Figure 26: U with $c \in [0.5, 1.5]$ (and Neumann condition) and $m \in [2, 3]$.

We see that U is naturally increasing with c since an increase in c induces an increase in

expected profits. Similarly, U is decreasing with m.

Now, from U, we can compute the trajectory for m (Figure 27).

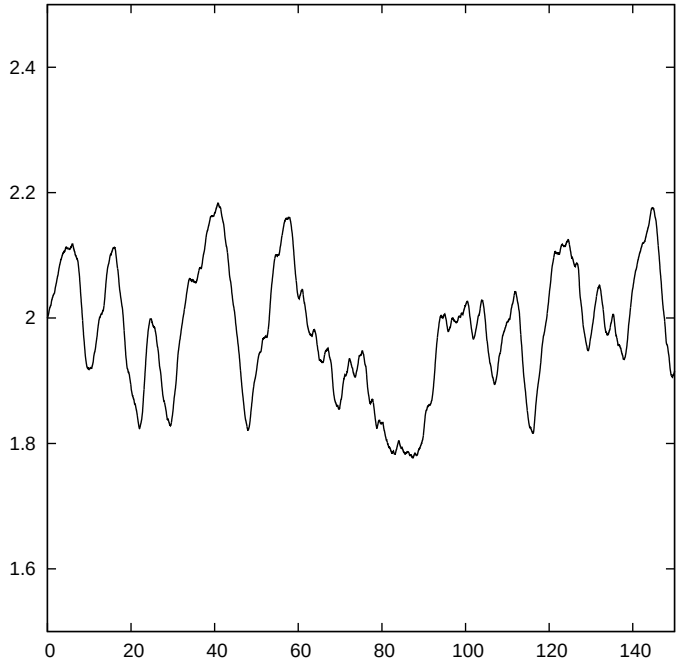


Figure 27: Trajectory of m.

Finally, oil prices can be deduced from the available production capacities (Figure 28).

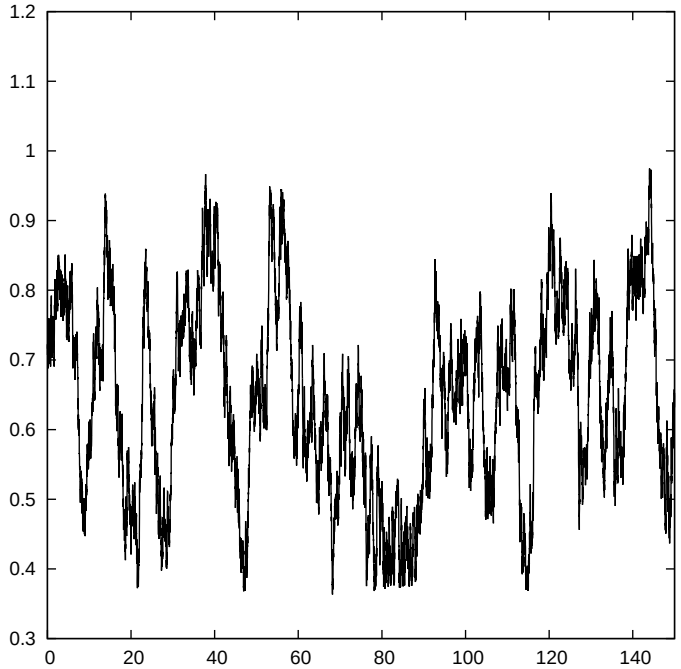


Figure 28: Oil prices.

3.2 The model with $N = 2$

Let us now turn to the $N = 2$ case.

The cost function associated to δ_1 is $C_1(\delta_1) = \frac{\delta_1^2}{2k_1}$ with $k_1 = 1.4$.
The cost function associated to λ is $C_{prod}(\lambda) = \frac{1}{2k_2\lambda^2}$ with $k_2 = 12$.

The inverse demand function is $P(t, m) = c(t)m^{-\eta}$ with $\eta = 0.5$ and c an Ornstein-Uhlenbeck process:

$$dc(t) = -\alpha(c(t) - \bar{c})dt + \sigma dW_t$$

where $\alpha = 0.2$, $\bar{c} = 1$ and $\sigma = 0.15$. Profits are discounted at rate $r = 25\%$.

A trajectory for c is drawn for 150 years on Figure 29.

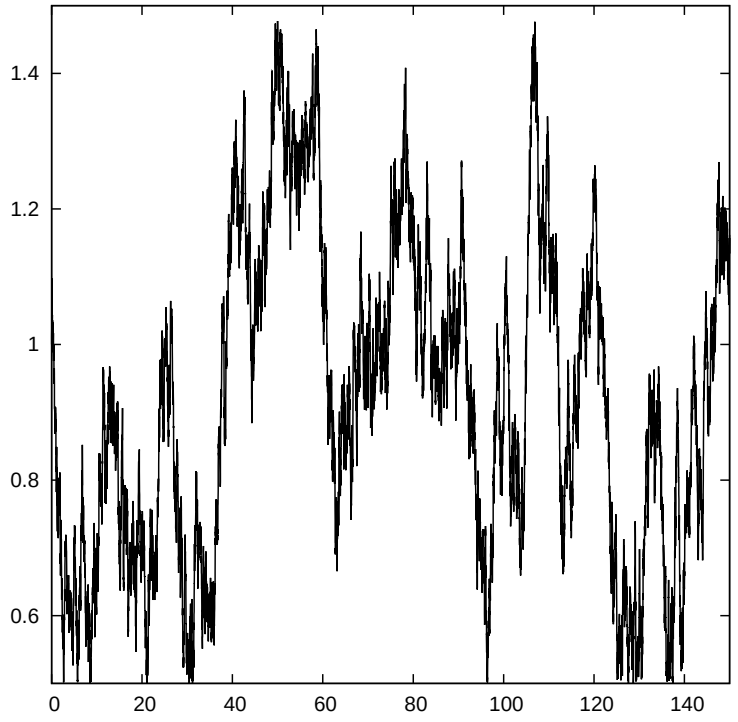


Figure 29: Trajectory of c with $\alpha = 0.2$, $\bar{c} = 1$ and $\sigma = 0.15$. Reflections have been introduced at $c = 0.5$ and $c = 1.5$.

Then Φ is solved using implicit methods for stationary solutions of Hamilton-Jacobi-Bellman equations (Figure 30).

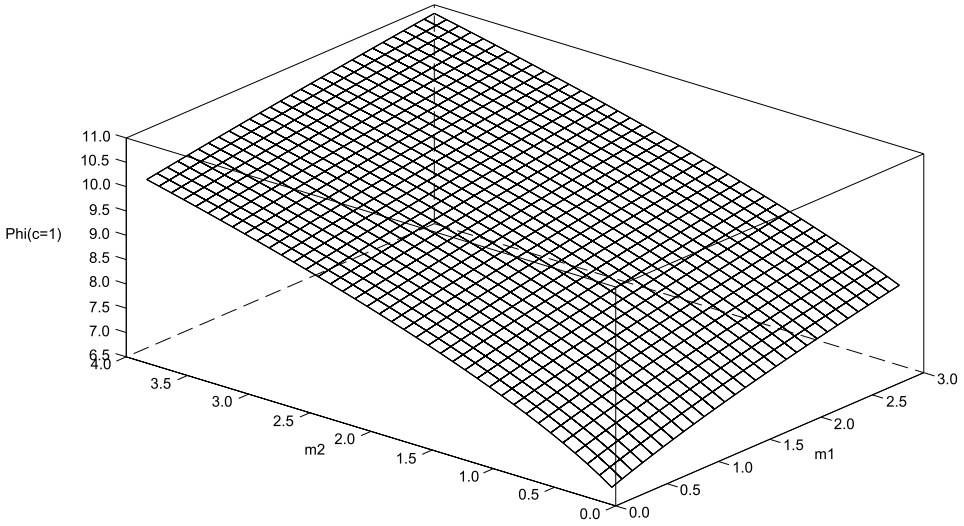


Figure 30: Resolution of the equation for Φ. The surface represented corresponds to $c = 1$.

Using the function Φ, we can deduce U_1 (Figure 31).

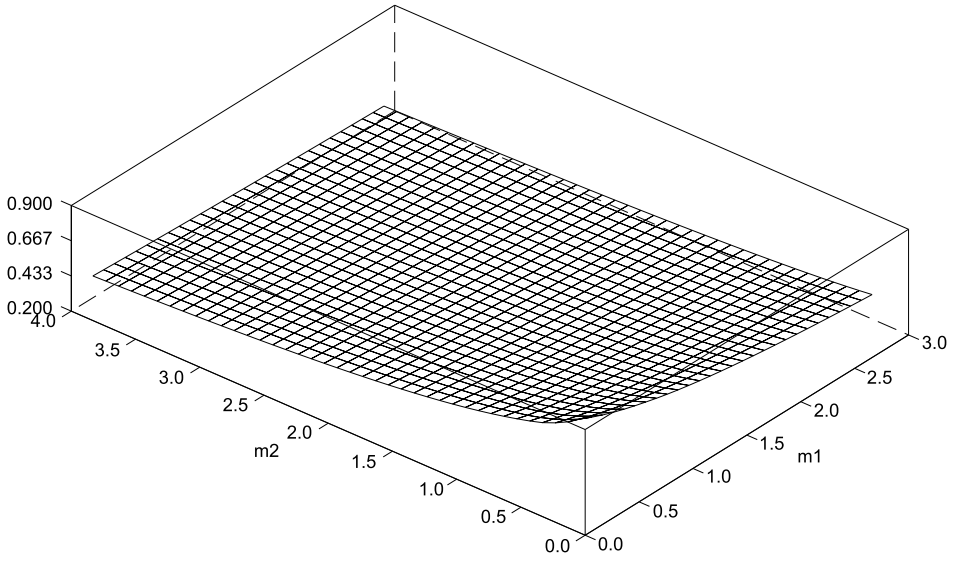


Figure 31: U_1. The surface represented corresponds to $c = 1$.

Similarly, we obtain U_2 (Figure 32).

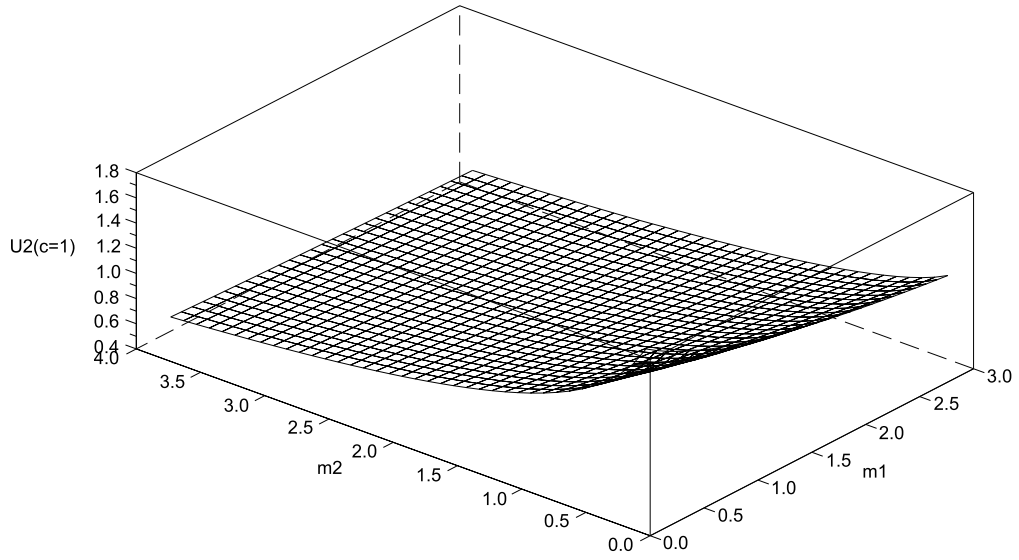


Figure 32: U_2. The surface represented corresponds to $c = 1$.

From (U_1, U_2), we can compute the trajectory for m_1 (Figure 33) and m_2 (Figure 34).

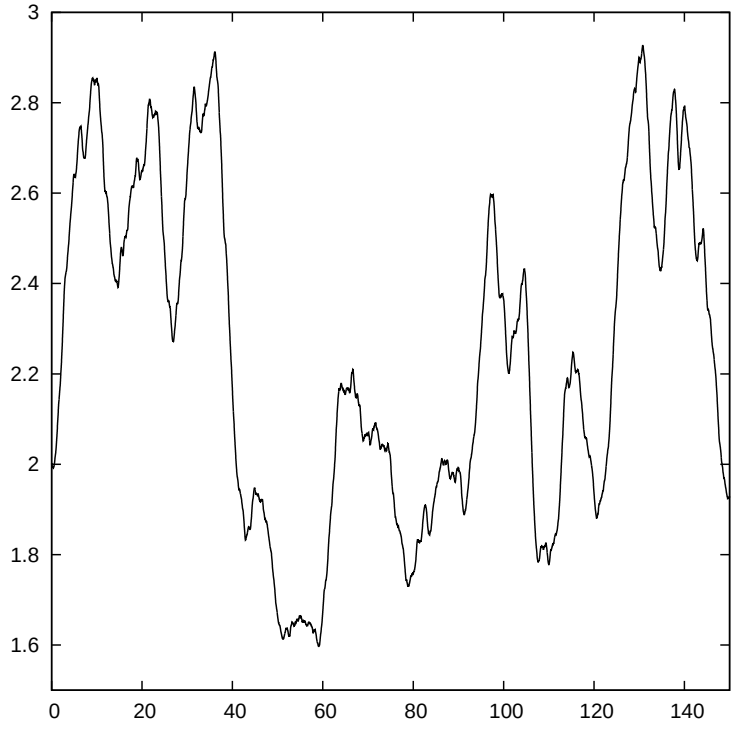


Figure 33: Trajectory of m_1.

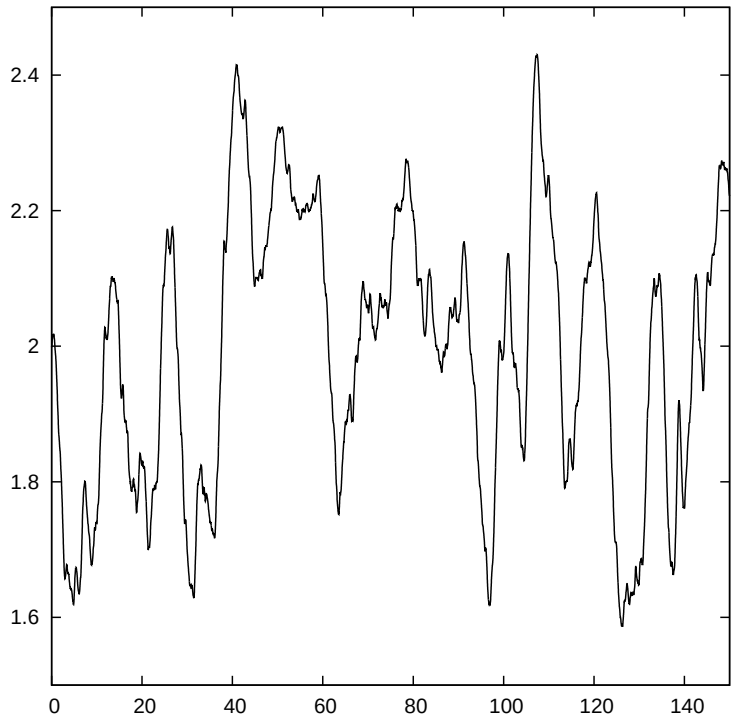


Figure 34: Trajectory of m_2.

Finally, oil prices can be deduced from the available production capacities (Figure 28).

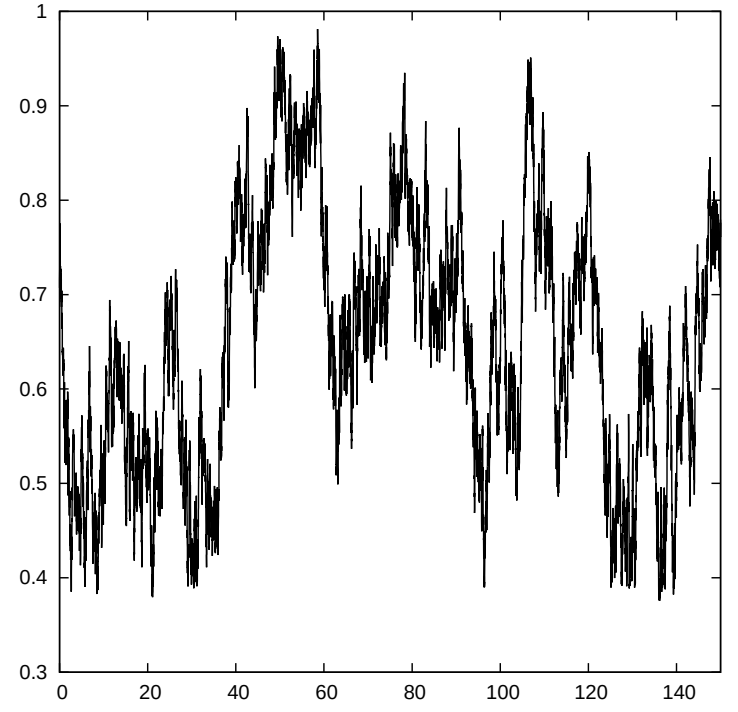


Figure 35: Oil prices.

To analyze the mechanisms involved in this model, let us remark first that there is a constant

flow[14] of new projects arriving in state 1. Then, when c is increasing, agents with production units in step 1 are willing to pay more to have a chance to proceed to the actual production step. Producers in step 2 are also willing to pay more to stay in the market, but the situation may evolve as more and more production capacities are available for production and consequently negate the increase in price due to the increase in c. On the other hand, when c is decreasing, agents in step 1 have less incentive to transform their project into actual production capacities. Similarly, producers in step 2 want to pay less to stay in the market.

Hence, to understand the important impact of the time-to-build effect, let us suppose that c was low for quite a long period. In that case, see for instance around $t = 36$ or around $t = 128$ on Figures 33 and 34, m_2 is low since there is no incentive to stay in the market. In turn, m_1 is very high since the constant flow of new projects has accumulated in step 1 (as long as there was too many producers in step 2).
Then, when c increases, the agents in step 1 want to produce before the others and there is a rush to the production state. That is why we observe a higher volatility in the available production capacity when there is a time-to-build effect.

This setting is in fact very rich and the effects we obtain depend on the choices for the cost functions. If for instance the cost to go from step 1 to 2 is very high, then the price must increase a lot before the agents pay the cost to have a better chance to produce. Hence, our framework allows us to model the important rise in price due to the delay in obtaining new production capacity.

[14]The model of the first part could be plugged to this model to make the arrival flow endogenous.

Conclusion

In this text we present a new model to describe the progressive adjustment of production capacities to changes in demand. The new modeling framework we provide includes new tools from mean field games theory and especially tools to tackle, both theoretically and numerically, the case of common noise. Using very recent works by J.-M. Lasry and P.-L. Lions for Hamilton-Jacobi-Bellman equations in infinite dimension and adapting them to the simpler case of a finite state space, we were able to propose a model where the time to build a new production capacity is taken into account in a flexible way since agents may decide whether or not to foster the arrival of these new production capacities.

This time-to-build model paves the road for further work in which exploration decisions or inventory management may be taken into account. Moreover, it opens a new field of research in the design of numerical schemes for the HJB equation arising from a planning problem where all the states enter the value function.

References

[1] M.J. Chambers and R.E. Bailey. A theory of commodity price fluctuations. *The Journal of Political Economy*, 104(5):924–957, 1996.

[2] A. Deaton and G. Laroque. On the behaviour of commodity prices. *The Review of Economic Studies*, 59(1):1, 1992.

[3] A. Deaton and G. Laroque. Competitive storage and commodity price dynamics. *The Journal of Political Economy*, 104(5):896–923, 1996.

[4] L.C. Evans. Partial Differential Equations (Graduate Studies in Mathematics, Vol. 19). 2009.

[5] J.D. Hamilton. Oil and the macroeconomy. *The New Palgrave Dictionary of Economics*, 2, 2008.

[6] F.E. Kydland and E.C. Prescott. Time to build and aggregate fluctuations. *Econometrica: Journal of the Econometric Society*, pages 1345–1370, 1982.

[7] J.-M. Lasry and P.-L. Lions. Jeux à champ moyen i. le cas stationnaire. *C. R. Acad. Sci. Paris*, 343(9), 2006.

[8] J.-M. Lasry and P.-L. Lions. Jeux à champ moyen ii. horizon fini et contrôle optimal. *C. R. Acad. Sci. Paris*, 343(10), 2006.

[9] J.-M. Lasry and P.-L. Lions. Mean field games. *Japanese Journal of Mathematics*, 2(1), Mar. 2007.

[10] S. Majd and R.S. Pindyck. Time to build, option value, and investment decisions. *Journal of financial Economics*, 18(1):7–27, 1987.

[11] J.M. Lasry O. Guéant and P.L. Lions. Long-run Oil Production. In *The Economics of Sustainable Development*, 2009.

[12] J.M. Lasry O. Guéant and P.L. Lions. Mean field games and applications. In *Paris Princeton Lectures on Mathematical Finance*, 2010.

[13] R.S. Pindyck. Energy price increases and macroeconomic policy. *The Energy Journal*, 1(4):1–20, 1980.

[14] R.S. Pindyck. Uncertainty and exhaustible resource markets. *The Journal of Political Economy*, 88(6):1203–1225, 1980.

[15] R.S. Pindyck. Models of resource markets and the explanation of resource price behaviour. *Energy Economics*, 3(3):130–139, 1981.

Appendix

Source code for the computations of I.3:

```
exec("one_box_constant_educ.sci");

// solutions

fd = mopen('one_box_constant_educ.dat','w');
lambda = 0.2;
k=0.2;
c=1;
r=0.05;
nb_t=1001;
T=20;
dt = T/(nb_t-1);
m0 = 0.2;
[mtheta,utheta] = one_box_constant_educ(m0,lambda,k,c,r,nb_t,T,500,0.01);
for i=1:nb_t
    mfprintf(fd, '%f %f %f %f %f %f %f\n', (i-1)*dt, utheta(200,i), mtheta(200,i),
             utheta(400,i), mtheta(400,i), utheta(500,i), mtheta(500,i));
end
mclose(fd);

fd = mopen('one_box_constant_educ_bis.dat','w');
lambda = 0.2;
k=0.2;
c=1;
r=0.05;
nb_t=1001;
T=20;
dt = T/(nb_t-1);
m0 = 6;
[mtheta,utheta] = one_box_constant_educ(m0,lambda,k,c,r,nb_t,T,500,0.01);
for i=1:nb_t
    mfprintf(fd, '%f %f %f %f %f %f %f\n', (i-1)*dt, utheta(200,i), mtheta(200,i),
```

```
            utheta(400,i), mtheta(400,i), utheta(500,i), mtheta(500,i));
end
mclose(fd);


function [mtheta,utheta] = one_box_constant_educ(m0,lambda,k,c,r,nb_t,
                                                  T,nb_theta,dtheta)
    dt = T/(nb_t-1);
    m_inf = (k*c/lambda/(lambda+r))^(2/3);
    u_inf = lambda * m_inf / k;
    mtheta = zeros(nb_theta,nb_t);
    utheta = zeros(nb_theta,nb_t);
    mtheta(1,:) = m0;
    utheta(1,:) = u_inf;
    for theta=2:nb_theta
        integ = 0;
        utheta(theta,nb_t) = u_inf;
        for t=nb_t-1:-1:1
            integ = integ + dt * exp(-(lambda+r)*(t-1)*dt) *c
                    * 1 /mtheta(theta-1,t)^0.5;
            utheta(theta,t) = dtheta*(exp(-(lambda+r)*(nb_t-t)*dt)*u_inf
            + exp((lambda+r)*(t-1)*dt)*integ) + (1 - dtheta)* utheta(theta-1,t);
        end
        integ=0;
        mtheta(theta,1) = m0;
        for t=2:nb_t
            integ = integ + dt * exp(lambda*(t-1)*dt) *k * utheta(theta-1,t);
            mtheta(theta,t) = dtheta*(exp(-lambda*(t-1)*dt)*m0
            + exp(-lambda*(t-1)*dt)*integ) + (1-dtheta)*mtheta(theta-1,t);
        end
    end
endfunction
```

Source code for the computations of I.4.1:

```
exec("one_box_choc_permanent.sci");
```

```
// solutions

fd = mopen('one_box_choc_permanent.dat','w');
lambda = 0.2;
k=0.2;
c_1=1;
c_2=1.5;
r=0.05;
nb_t=1001;
T=30;
T1=3;
T2=5;
dt = T/(nb_t-1);
m0 = (k*c_1/lambda/(lambda+r))^(2/3);
[m,u,c]=one_box_choc_perm(m0,lambda,k,c_1,c_2,r,nb_t,T1,T2,T,0.01,800);
for i=1:nb_t
    mfprintf(fd, '%f %f %f %f %f\n', (i-1)*dt, c(i), u(i), m(i), c(i)/m(i)^0.5);
end
mclose(fd);


function [m,u,c]=one_box_choc_perm(m0,lambda,k,c_min,c_max,r,nb_t,
                                   T1,T2,T,dtheta,nb_theta)
    dt = T/(nb_t-1);
    m_inf = (k*c_max/lambda/(lambda+r))^(2/3);
    u_inf = lambda * m_inf / k;
    c = zeros(1,nb_t);
    for t=1:nb_t
        if (t-1)*dt<T1 then
            c(t) = c_min;
        elseif (t-1)*dt<T2
            c(t) = (((t-1)*dt - T1)*c_max + (T2-(t-1)*dt)*c_min)/(T2-T1);
        else
            c(t) = c_max;
        end
```

```
    end
    m = m0*ones(1,nb_t);
    u = u_inf*ones(1,nb_t);
    new_m = zeros(1,nb_t);
    new_u = zeros(1,nb_t);
    for theta=2:nb_theta
        integ = 0;
        new_u(1,nb_t) = u_inf;
        for t=nb_t-1:-1:1
            integ = integ + dt * exp(-(lambda+r)*(t-1)*dt)*c(t)
                                * (1/sqrt(m(1,t)));
            new_u(t) = dtheta*(exp(-(lambda+r)*(nb_t-t)*dt)*u_inf
            + exp((lambda+r)*(t-1)*dt)*integ) + (1 - dtheta)* u(1,t);
        end
        integ=0;
        new_m(1,1) = m0;
        for t=2:nb_t
            integ = integ + dt * exp(lambda*(t-1)*dt) *k * u(1,t);
            new_m(1,t) = dtheta*(exp(-lambda*(t-1)*dt)*m0
            + exp(-lambda*(t-1)*dt)*integ) + (1-dtheta)*m(1,t);
        end
        u=new_u;
        m=new_m;
    end
endfunction
```

Source code for the computations of I.4.2:

```
exec("one_box_choc_transitoire.sci");

// solutions

fd = mopen('one_box_choc_transitoire.dat','w');
lambda = 0.2;
k=0.2;
c_1=1;
```

```
c_2=1.5;
r=0.05;
nb_t=3001;
T=30;
T1=3;
T2=5;
dt = T/(nb_t-1);
m0 = (k*c_1/lambda/(lambda+r))^(2/3);
[m,u,c]=one_box_choc_tran(m0,lambda,k,c_1,c_2,r,nb_t,T1,T2,T,0.01,800);
for i=1:nb_t
    mfprintf(fd, '%f %f %f %f %f\n', (i-1)*dt, c(i), u(i), m(i), c(i)/m(i)^0.5);
end
mclose(fd);


function [m,u,c]=one_box_choc_tran(m0,lambda,k,c_min,c_max,r,nb_t,
                                   T1,T2,T,dtheta,nb_theta)
    dt = T/(nb_t-1);
    m_inf = (k*c_min/lambda/(lambda+r))^(2/3);
    u_inf = lambda * m_inf / k;
    c = zeros(1,nb_t);
    for t=1:nb_t
        if (t-1)*dt<T1 then
            c(t) = c_min;
        elseif (t-1)*dt<(T1+T2)/2
            c(t) = (((t-1)*dt - T1)*c_max
                   + ((T1+T2)/2-(t-1)*dt)*c_min)/((T2-T1)/2);
        elseif (t-1)*dt < T2
            c(t) = (((t-1)*dt - (T1+T2)/2)*c_min
                   + (T2-(t-1)*dt)*c_max)/((T2-T1)/2);
        else
            c(t) = c_min;
        end
    end
    m = m0*ones(1,nb_t);
    u = u_inf*ones(1,nb_t);
```

```
    new_m = zeros(1,nb_t);
    new_u = zeros(1,nb_t);
    for theta=2:nb_theta
        integ = 0;
        new_u(1,nb_t) = u_inf;
        for t=nb_t-1:-1:1
            integ = integ + dt * exp(-(lambda+r)*(t-1)*dt)*c(t)
                                * (1/sqrt(m(1,t)));
            new_u(t) = dtheta*(exp(-(lambda+r)*(nb_t-t)*dt)*u_inf
            + exp((lambda+r)*(t-1)*dt)*integ) + (1 - dtheta)* u(1,t);
        end
        integ=0;
        new_m(1,1) = m0;
        for t=2:nb_t
            integ = integ + dt * exp(lambda*(t-1)*dt) *k * u(1,t);
            new_m(1,t) = dtheta*(exp(-lambda*(t-1)*dt)*m0
                        + exp(-lambda*(t-1)*dt)*integ) + (1-dtheta)*m(1,t);
        end
        u=new_u;
        m=new_m;
    end
endfunction
```

Source code for the computations of I.5:

```
exec("calc_Phi_2.sci");

// solutions

lambda = 0.2;
k=0.2;
r=0.05;
eta=0.5;
alpha=0.2;
c_bar=1;
sigma=0.15;
```

```
c_min=0.5;
c_max=1.5;
nb_c=31;
m_min=2;
m_max=3;
nb_m=31;
theta=150;
dtheta=1/50;


[Phi]=calc_Phi_2(lambda,k,r,eta,alpha,c_bar,sigma,c_min,c_max,nb_c,
                 m_min,m_max,nb_m,theta,dtheta);

c = linspace(c_min,c_max,nb_c);
m = linspace(m_min,m_max,nb_m);

fd = mopen('Phi_2.dat','w');
for i=1:nb_c
    for j=1:nb_m
        mfprintf(fd, '%f %f %f\n', c(i), m(j), Phi(i,j));
    end
end
mclose(fd);



exec("vect_c_random.sci");
c0=1;
T=150;
nb_t=10000;
c_dyn=vect_c_random(alpha,c_bar,sigma,c0,T,nb_t,c_min,c_max);
exec("calc_m.sci");
m0 = 2.5;
m_dyn=calc_m(k,lambda,c_min,c_max,nb_c,m_min,m_max,nb_m, Phi,c_dyn, m0,T,nb_t);
dt = T /(nb_t-1);
```

```
fd = mopen('c_m.dat','w');
for t=1:nb_t
    mfprintf(fd, '%f %f %f %f\n', (t-1)*dt, c_dyn(t),
                  m_dyn(t), c_dyn(t)*m_dyn(t)^(-eta));
end
mclose(fd);
f=gcf();
xset("colormap",graycolormap(128))
plot3d1(c,m,Phi,220,10,"c@m@Phi",[0 2 4])
xs2eps(f,"Phi.eps")
clf();

for i=1:31
    U(i,:)= 30*diff(Phi(i,:));
end

f=gcf();
xset("colormap",graycolormap(128))
plot3d1(c,m(1:30),U,220,30,"c@m@U",[0 2 4])
xs2eps(f,"U.eps")


function [Phi]=calc_Phi_2(lambda,k,r,eta,alpha,c_bar,sigma,c_min,c_max,nb_c,
                          m_min,m_max,nb_m,theta,dtheta)
    dc = (c_max-c_min) / (nb_c-1);
    dm = (m_max-m_min) / (nb_m-1);
    nb_theta = floor(theta/dtheta) + 1;
    grille = 50*ones(nb_c,nb_m);
    grille_new = zeros(nb_c,nb_m);
    c = linspace(c_min,c_max,nb_c);
    m = linspace(m_min,m_max,nb_m);

    A = zeros(nb_c,nb_c);
    A(1,1)=1;
    A(1,2)=-1;
    A(nb_c,nb_c) = 1;
```

```
A(nb_c,nb_c-1) = -1;
for l=2:nb_c-1
    A(l,l-1) = -0.5;
    A(l,l) = 1;
    A(l,l+1) = -0.5;
end
A = sigma^2*dtheta/(dc)^2 * A + eye(nb_c,nb_c);
A = inv(A);

for k_theta=1:nb_theta
    for j=1:nb_m
       for i=1:nb_c
           terme1 = -r*grille(i,j) + 1/(1-eta) * c(i) *m(j)^(1-eta);
           if j==nb_m then
              gradmplus = grille(i,j)-grille(i,j-1);
              gradmmoins = grille(i,j)-grille(i,j-1);
           elseif j==1 then
              gradmplus = grille(i,j+1)-grille(i,j);
              gradmmoins = grille(i,j+1)-grille(i,j);
           else
              gradmplus = grille(i,j+1)-grille(i,j);
              gradmmoins = grille(i,j)-grille(i,j-1);
           end
           gradmplus = gradmplus / dm;
           gradmmoins = gradmmoins / dm;
           if k*gradmplus-lambda*m(j) >= 0 then
               terme2 = -lambda*m(j)*gradmplus + k/2 * (gradmplus)^2;
           elseif k*gradmmoins-lambda*m(j) <= 0
               terme2 = -lambda*m(j)*gradmmoins + k/2 * (gradmmoins)^2;
           else
               terme2 = -(lambda*m(j))^2/2/k
           end
           if i==nb_c then
              gradcplus = grille(i,j)-grille(i-1,j);
              gradcmoins = grille(i,j)-grille(i-1,j);
           elseif i==1 then
```

```
                gradcplus = grille(i+1,j)-grille(i,j);
                gradcmoins = grille(i+1,j)-grille(i,j);
            else
                gradcplus = grille(i+1,j)-grille(i,j);
                gradcmoins = grille(i,j)-grille(i-1,j);
            end
            gradcplus = gradcplus / dc;
            gradcmoins = gradcmoins / dc;
            if c(i) > c_bar then
                terme3 = -alpha*(c(i)-c_bar)*gradcmoins;
            else
                terme3 = -alpha*(c(i)-c_bar)*gradcplus;
            end
            grille_new(i,j) = grille(i,j)
                            + dtheta * (terme1 + terme2 + terme3);
        end
    end
    for j=1:nb_m
        grille(:,j) = A*grille_new(:,j);
    end
end
Phi=grille;
endfunction


function c=vect_c_random(alpha,c_bar,sigma,c0,T,nb_t,c_min,c_max)
    dt = T /(nb_t-1);
    c = zeros(1,nb_t);
    c(1,1) = c0;
    dw = sqrt(dt)*rand(1,nb_t-1,"normal");
    for i=1:nb_t-1
        c(1,i+1) = c(1,i)-alpha*(c(1,i)-c_bar)*dt + sigma*dw(1,i);
        if c(1,i+1) < c_min then
            c(1,i+1) = 2*c_min - c(1,i+1);
        elseif c(1,i+1) > c_max then
            c(1,i+1) = 2*c_max - c(1,i+1);
```

```
        end
    end
endfunction


function m_dyn=calc_m(k,lambda,c_min,c_max,nb_c,
                      m_min,m_max,nb_m, Phi,c_dyn, m0,T,nb_t)
    dt = T /(nb_t-1);
    m_dyn = zeros(1,nb_t);
    m_dyn(1) = m0;
    dc = (c_max-c_min) / (nb_c-1);
    dm = (m_max-m_min) / (nb_m-1);
    for t=1:nb_t-1
        i = 1+floor((c_dyn(1,t)-c_min)/dc);
        nu = (c_dyn(1,t)-c_min)/dc - floor((c_dyn(1,t)-c_min)/dc);
        j = 1+floor((m_dyn(1,t)-m_min)/dm);
        u_1 = (Phi(i,j+1)-Phi(i,j))/dm;
        u_2 = (Phi(i+1,j+1)-Phi(i+1,j))/dm;
        u = nu * u_2 + (1-nu)*u_1;
        m_dyn(1,t+1) = m_dyn(1,t)*(1-lambda * dt)+k*u*dt;
    end
endfunction
```

Source code for the computation of a solution with excursions to high prices of I.6:

```
function [Phi]=calc_Phi_dem41(lambda,k,r,alpha,c_bar,sigma,c_min,c_max,nb_c,
               m_min,m_max,nb_m,theta,dtheta)
    dc = (c_max-c_min) / (nb_c-1);
    dm = (m_max-m_min) / (nb_m-1);
    nb_theta = floor(theta/dtheta) + 1;
    grille = zeros(nb_c,nb_m);
    c = linspace(c_min,c_max,nb_c);
    m = linspace(m_min,m_max,nb_m);

    A = zeros(nb_c,nb_c);
    A(1,1)=0.5;
```

```
A(1,2)=-0.5;
A(nb_c,nb_c) = 0.5;
A(nb_c,nb_c-1) = -0.5;
for l=2:nb_c-1
    A(l,l-1) = -0.5;
    A(l,l) = 1;
    A(l,l+1) = -0.5;
end
A = sigma^2*dtheta/(dc)^2 * A + eye(nb_c,nb_c);
A = inv(A);
xi=2;
for k_theta=1:nb_theta
    for j=1:nb_m
       for i=1:nb_c
           if m(j)< xi*c(i) then
               picm = 1*m(j);
           else
               picm = 1*xi*c(i) + 0.5*(m(j)-xi*c(i));
           end
           terme1 = -r*grille(i,j) + picm;
           if j==nb_m then
              gradmplus = grille(i,j)-grille(i,j-1);
              gradmmoins = grille(i,j)-grille(i,j-1);
           elseif j==1 then
              gradmplus = grille(i,j+1)-grille(i,j);
              gradmmoins = grille(i,j+1)-grille(i,j);
           else
              gradmplus = grille(i,j+1)-grille(i,j);
              gradmmoins = grille(i,j)-grille(i,j-1);
           end
           gradmplus = gradmplus / dm;
           gradmmoins = gradmmoins / dm;
           terme2 = -lambda*m(j)*gradmmoins + k/2 * (gradmplus)^2;
           if i==nb_c then
              gradcplus = grille(i,j)-grille(i-1,j);
              gradcmoins = grille(i,j)-grille(i-1,j);
```

```
                elseif i==1 then
                   gradcplus = grille(i+1,j)-grille(i,j);
                   gradcmoins = grille(i+1,j)-grille(i,j);
                else
                   gradcplus = grille(i+1,j)-grille(i,j);
                   gradcmoins = grille(i,j)-grille(i-1,j);
                end
                gradcplus = gradcplus / dc;
                gradcmoins = gradcmoins / dc;
                if c(i) > c_bar then
                    terme3 = -alpha*(c(i)-c_bar)*gradcmoins;
                else
                    terme3 = -alpha*(c(i)-c_bar)*gradcplus;
                end
                grille(i,j) = grille(i,j)
                            + dtheta * (terme1 + terme2 + terme3);
            end
            grille(:,j) = A*grille(:,j);
        end
    end
    Phi=grille;
endfunction
```

Source code for the computation of a solution to the model described in III.1:

```
exec("calc_Phi_1_box.sci");
exec("build_1_box.sci");

// solutions

k=12;
r=0.25;
eta=0.5;
alpha=0.2;
c_bar=1;
sigma=0.15;
```

```
c_min=0.5;
c_max=1.5;
nb_c=21;
m_min=1.5;
m_max=2.5;
nb_m=31;
theta=30;
dtheta=1/60;

[Phi]=calc_Phi_1_box(k,r,eta,alpha,c_bar,sigma,c_min,c_max,nb_c,
                     m_min,m_max,nb_m,theta,dtheta)

c = linspace(c_min,c_max,nb_c);
m = linspace(m_min,m_max,nb_m);

exec("vect_c_random.sci");
c0=1;
T=150;
nb_t=10000;
c_dyn=vect_c_random(alpha,c_bar,sigma,c0,T,nb_t,c_min,c_max);
exec("calc_m_1_box.sci");
m0 = 2;
m_dyn=calc_m_1_box(k,c_min,c_max,nb_c,m_min,m_max,nb_m, Phi,c_dyn, m0,T,nb_t);
dt = T /(nb_t-1);

fd = mopen('c_m_1_box.dat','w');
for t=1:nb_t
    mfprintf(fd, '%f %f %f %f\n', (t-1)*dt, c_dyn(t), m_dyn(t),
                                  c_dyn(t)*m_dyn(t)^(-eta));
end
mclose(fd);

f=gcf();
xset("colormap",graycolormap(128))
plot3d1(c,m,Phi,220,10,"c@m@Phi",[0 2 4])
xs2eps(f,"Phi_1_box.eps")
```

```
clf();

for i=1:21
    U(i,:)= 30*diff(Phi(i,:));
end

f=gcf();
xset("colormap",graycolormap(128))
plot3d1(c,m(1:30),U,220,40,"c@m@U",[0 2 4])
xs2eps(f,"U_1_box.eps")


function [Phi]=calc_Phi_1_box(k,r,eta,alpha,c_bar,sigma,c_min,c_max,nb_c,
                              m_min,m_max,nb_m,theta,dtheta)
    dc = (c_max-c_min) / (nb_c-1);
    dm = (m_max-m_min) / (nb_m-1);
    nb_theta = floor(theta/dtheta) + 1;
    [u1,m1,lamda,phi_ref]=build_1_box(k,r,eta);
    grille = zeros(nb_c,nb_m);
    grille_new = zeros(nb_c,nb_m);
    mid_m = (nb_m+1)/2;
    for i =1:nb_m
        for c=1:nb_c
            grille(c,i) = phi_ref + (i-mid_m)*u1*dm;
        end
    end
    c = linspace(c_min,c_max,nb_c);
    m = linspace(m_min,m_max,nb_m);
    A = zeros(nb_c,nb_c);
    A(1,1)=1;
    A(1,2)=-1;
    A(nb_c,nb_c) = 1;
    A(nb_c,nb_c-1) = -1;
    for l=2:nb_c-1
        A(l,l-1) = -0.5;
        A(l,l) = 1;
```

```
    A(l,l+1) = -0.5;
end
A = sigma^2*dtheta/(dc)^2 * A + eye(nb_c,nb_c);
A = inv(A);
for k_theta=1:nb_theta
    printf("%d\n",k_theta);
    for j=1:nb_m
       for i=1:nb_c
           terme1 = -r*grille(i,j) + 1/(1-eta) * c(i) *m(j)^(1-eta);
           if j==nb_m then
              gradmplus = grille(i,j)-grille(i,j-1);
              gradmmoins = grille(i,j)-grille(i,j-1);
           elseif j==1 then
              gradmplus = grille(i,j+1)-grille(i,j);
              gradmmoins = grille(i,j+1)-grille(i,j);
           else
              gradmplus = grille(i,j+1)-grille(i,j);
              gradmmoins = grille(i,j)-grille(i,j-1);
           end
           gradmplus = max(gradmplus/dm,0.0001);
           gradmmoins = max(gradmmoins/dm,0.001);
           lambdaplus = (k*gradmplus)^(-1/3);
           lambdamoins = (k*gradmmoins)^(-1/3);
           if (1-lambdaplus*m(j)) > 0 then
               terme2 = gradmplus - m(j) * 3/2 * k^(-1/3)*gradmplus^(2/3);
           elseif (1-lambdamoins*m(j)) < 0
               terme2 = gradmmoins - m(j) * 3/2 * k^(-1/3)*gradmmoins^(2/3);
           else
               gradm = m(j)^3/k;
               terme2 = gradm - m(j) * 3/2 * k^(-1/3)*gradm^(2/3);
           end
           if i==nb_c then
              gradcplus = grille(i,j)-grille(i-1,j);
              gradcmoins = grille(i,j)-grille(i-1,j);
           elseif i==1 then
              gradcplus = grille(i+1,j)-grille(i,j);
```

```
                gradcmoins = grille(i+1,j)-grille(i,j);
            else
                gradcplus = grille(i+1,j)-grille(i,j);
                gradcmoins = grille(i,j)-grille(i-1,j);
            end
            gradcplus = gradcplus / dc;
            gradcmoins = gradcmoins / dc;
            if c(i) > c_bar then
                terme3 = -alpha*(c(i)-c_bar)*gradcmoins;
            else
                terme3 = -alpha*(c(i)-c_bar)*gradcplus;
            end
            grille_new(i,j) = grille(i,j)
                              + dtheta * (terme1 + terme2 + terme3);
        end
    end
    for j=1:nb_m
        grille(:,j) = A*grille_new(:,j);
    end
  end
  Phi=grille;
endfunction


function m1_dyn=calc_m_2_box(k1,k2,c_min,c_max,nb_c,
                             m_min,m_max,nb_m, Phi,c_dyn, m1_0,m2_0,T,nb_t)
  dt = T /(nb_t-1);
  m_dyn = zeros(1,nb_t);
  m_dyn(1) = m0;
  dc = (c_max-c_min) / (nb_c-1);
  dm = (m_max-m_min) / (nb_m-1);
  for t=1:nb_t-1
    i = 1+floor((c_dyn(1,t)-c_min)/dc);
    nu = (c_dyn(1,t)-c_min)/dc - floor((c_dyn(1,t)-c_min)/dc);
    j = 1+floor((m_dyn(1,t)-m_min)/dm);
    u_1 = (Phi(i,j+1)-Phi(i,j))/dm;
```

```
        u_2 = (Phi(i+1,j+1)-Phi(i+1,j))/dm;
        u = nu * u_2 + (1-nu)*u_1;
        lambda = (k*u)^(-1/3)
        m_dyn(1,t+1) = m_dyn(1,t)*(1-lambda * dt)+dt;
    end
endfunction


function [u1,m1,lamda,Phi]=build_1_box(k,r,eta)
    u1=5;
    err=10;
    for j=1:20
        fu1 = r*u1+(3/2)*k^(-1/3)*u1^(2/3)-k^(-eta/3)*u1^(-eta/3);
        fpu1 = r+ k^(-1/3)*u1^(-1/3)+ eta/3 * k^(-eta/3)*u1^(-eta/3-1);
        u1 = max(0.0001,u1 - fu1/fpu1);
    end
    m1 = (k*u1)^(1/3);
    lamda = (k*u1)^(-1/3);
    Phi = (m1^(1-eta)/(1-eta) - 1/(2*k*lamda^2))/r;
endfunction
```

Source code for the computation of a solution to the model described in III.2:

```
exec("calc_Phi_2_box.sci");
exec("build_2_box.sci")

// solutions

k1=1.4;
k2=12;
r=0.25;
eta=0.5;
alpha=0.2;
c_bar=1;
sigma=0.15;
c_min=0.5;
c_max=1.5;
```

```
nb_c=11;
m1_min=0.2;
m1_max=3;
nb_m1=29;
m2_min=0.2;
m2_max=4;
nb_m2=39;
theta=30;
dtheta=1/100;

[Phi,gc1,gc2]=calc_Phi_2_box(k1,k2,r,eta,alpha,c_bar,sigma,c_min,c_max,nb_c,
                    m1_min,m1_max,nb_m1,m2_min,m2_max,nb_m2,theta,dtheta);

c = linspace(c_min,c_max,nb_c);
m1 = linspace(m1_min,m1_max,nb_m1);
m2 = linspace(m2_min,m2_max,nb_m2);

exec("vect_c_random.sci");
c0=1;
T=150;
nb_t=10000;
c_dyn=vect_c_random(alpha,c_bar,sigma,c0,T,nb_t,c_min,c_max);
exec("calc_m_2_box.sci");
m1_0 = 2;
m2_0 = 2;
[m1_dyn,m2_dyn]=calc_m_2_box(k1,k2,c_min,c_max,nb_c,m1_min,m1_max,nb_m1,
                            m2_min,m2_max,nb_m2,Phi,c_dyn, m1_0,m2_0,T,nb_t);
dt = T /(nb_t-1);

fd = mopen('c_m_2_box.dat','w');
for t=1:nb_t
    mfprintf(fd, '%f %f %f %f %f\n', (t-1)*dt, c_dyn(t), m1_dyn(t),
                m2_dyn(t), c_dyn(t)*m2_dyn(t)^(-eta));
end
mclose(fd);
```

```
f=gcf();
xset("colormap",graycolormap(128))
plot3d1(m1,m2,squeeze(Phi(6,:,:)),220,40,"m1@m2@Phi(c=1)",[0 2 4])
xs2eps(f,"Phi_2_box_c1.eps")
clf();

for j=1:29
    for k =1:38
        U2(j,k)= 10*(Phi(6,j,k+1)-Phi(6,j,k));
    end
end
for k=1:39
    for j=1:28
        U1(j,k)= 10*(Phi(6,j+1,k) - Phi(6,j,k));
    end
end

f=gcf();
xset("colormap",graycolormap(128))
plot3d1(m1(1:28),m2,U1,220,60,"m1@m2@U1(c=1)",[0 2 4])
xs2eps(f,"U1_2_box.eps")
clf();

f=gcf();
xset("colormap",graycolormap(128))
plot3d1(m1,m2(1:38),U2,220,60,"m1@m2@U2(c=1)",[0 2 4])
xs2eps(f,"U2_2_box.eps")
clf();


function [Phi,grilles_codes,grilles_codes_2]=calc_Phi_2_box(k1,k2,r,eta,alpha,
         c_bar,sigma,c_min,c_max,nb_c,m1_min,m1_max,nb_m1,
         m2_min,m2_max,nb_m2,theta,dtheta)
    dc = (c_max-c_min) / (nb_c-1);
    dm1 = (m1_max-m1_min) / (nb_m1-1);
    dm2 = (m2_max-m2_min) / (nb_m2-1);
```

```
nb_theta = floor(theta/dtheta)+1;
[u1,u2,m1,m2,delta1,lamda,phi_ref]=build_2_box(k1,k2,r,eta);
grille = zeros(nb_c,nb_m1,nb_m2);
grille_new = zeros(nb_c,nb_m1,nb_m2);
mid_m1 = (nb_m1+1)/2;
mid_m2 = (nb_m2+1)/2;
for i =1:nb_m1
    for j=1:nb_m2
        for c=1:nb_c
            grille(c,i,j) = phi_ref + (i-mid_m1)*u1*dm1 + (j-mid_m2)*u2*dm2;
        end
    end
end

grilles_codes = zeros(nb_m1,nb_m2);
grilles_codes_2 = zeros(nb_m1,nb_m2);

c = linspace(c_min,c_max,nb_c);
m1 = linspace(m1_min,m1_max,nb_m1);
m2 = linspace(m2_min,m2_max,nb_m2);

A = zeros(nb_c,nb_c);
A(1,1)=1;
A(1,2)=-1;
A(nb_c,nb_c) = 1;
A(nb_c,nb_c-1) = -1;
for l=2:nb_c-1
    A(l,l-1) = -0.5;
    A(l,l) = 1;
    A(l,l+1) = -0.5;
end
A = sigma^2*dtheta/(dc)^2 * A + eye(nb_c,nb_c);
A = inv(A);

for k_theta=1:nb_theta
    printf("%d\n",k_theta);
```

```
printf("%f %f %f %f\n", grilles_codes(1,1),grilles_codes(1,nb_m2),
       grilles_codes(nb_m1,1),grilles_codes(nb_m1,nb_m2));
for k=1:nb_m2
    for j=1:nb_m1
       for i=1:nb_c
           terme1 = -r*grille(i,j,k) + 1/(1-eta) * c(i) *m2(k)^(1-eta);
           if j==nb_m1 then
              gradm1plus = 2*(grille(i,j,k)-grille(i,j-1,k))
                         -(grille(i,j-1,k)-grille(i,j-2,k));
              gradm1moins = grille(i,j,k)-grille(i,j-1,k);
           elseif j==1 then
              gradm1plus = grille(i,j+1,k)-grille(i,j,k);
              gradm1moins = 2*(grille(i,j+1,k)-grille(i,j,k))
                          -(grille(i,j+2,k)-grille(i,j+1,k));
           else
              gradm1plus = grille(i,j+1,k)-grille(i,j,k);
              gradm1moins = grille(i,j,k)-grille(i,j-1,k);
           end
           gradm1plus = gradm1plus / dm1;
           gradm1moins = gradm1moins / dm1;
           if k==nb_m2 then
              gradm2plus = 2*(grille(i,j,k)-grille(i,j,k-1))
                          -(grille(i,j,k-1)-grille(i,j,k-2));
              gradm2moins = grille(i,j,k)-grille(i,j,k-1);
           elseif k==1 then
              gradm2plus = grille(i,j,k+1)-grille(i,j,k);
              gradm2moins = 2*(grille(i,j,k+1)-grille(i,j,k))
                          -(grille(i,j,k+2)-grille(i,j,k+1));;
           else
              gradm2plus = grille(i,j,k+1)-grille(i,j,k);
              gradm2moins = grille(i,j,k)-grille(i,j,k-1);
           end
           gradm2plus = gradm2plus / dm2;
           gradm2moins = gradm2moins / dm2;

           deltaplusplus = k1*(gradm2plus-gradm1plus);
```

```
deltaplusmoins = k1*(gradm2moins-gradm1plus);
deltamoinsmoins = k1*(gradm2moins-gradm1moins);
deltamoinsplus = k1*(gradm2plus-gradm1moins);
lambdaplus = (max(k2*gradm2plus,0.0001))^(-1/3);
lambdamoins = (max(k2*gradm2moins,0.0001))^(-1/3);

m1dotplusplus = 1 - deltaplusplus * m1(j);
m2dotplusplus = deltaplusplus * m1(j) - lambdaplus * m2(k);

if i==11 then
        grilles_codes_2(j,k) = deltaplusplus ;
end
m1dotplusmoins = 1 - deltaplusmoins * m1(j);
m2dotplusmoins = deltaplusmoins * m1(j)
               - lambdamoins * m2(k);
m1dotmoinsplus = 1 - deltamoinsplus * m1(j);
m2dotmoinsplus = deltamoinsplus * m1(j)
               - lambdaplus * m2(k);
m1dotmoinsmoins = 1 - deltamoinsmoins * m1(j);
m2dotmoinsmoins = deltamoinsmoins * m1(j)
                - lambdamoins * m2(k);

if (m1dotplusplus > 0) & (m2dotplusplus > 0) then
    if i==11 then
        grilles_codes(j,k) = 0;
    end
    terme2 = gradm1plus + k1*m1(j)/2
           * (gradm2plus-gradm1plus)^2
           - 3/2*m2(k)* k2^(-1/3)*max(gradm2plus,0)^(2/3);
elseif (m1dotplusmoins > 0) & (m2dotplusmoins < 0) then
    if i==11 then
        grilles_codes(j,k) = 1;
    end
    terme2 = gradm1plus + k1*m1(j)/2
    * (gradm2moins-gradm1plus)^2
    - 3/2*m2(k)* k2^(-1/3)*max(gradm2moins,0)^(2/3);
```

```
elseif (m1dotmoinsplus < 0) & (m2dotmoinsplus > 0) then
    if i==11 then
        grilles_codes(j,k) = 2;
    end
    terme2 = gradm1moins + k1*m1(j)/2
    * (gradm2plus-gradm1moins)^2
    - 3/2*m2(k)* k2^(-1/3)*max(gradm2plus,0)^(2/3);
elseif (m1dotmoinsmoins < 0) & (m2dotmoinsmoins < 0) then
    if i==11 then
        grilles_codes(j,k) = 3;
    end
    terme2 = gradm1moins + k1*m1(j)/2
    * (gradm2moins-gradm1moins)^2
    - 3/2*m2(k)* k2^(-1/3)*max(gradm2moins,0)^(2/3);
else // on est près d'un cas d'égalité
    m1dotpos = (m1dotplusplus > 0) & (m1dotplusmoins > 0)
             & (m1dotmoinsplus > 0) & (m1dotmoinsmoins > 0);
    m1dotneg = (m1dotplusplus < 0) & (m1dotplusmoins < 0)
             & (m1dotmoinsplus < 0) & (m1dotmoinsmoins < 0);
    m2dotpos = (m2dotplusplus > 0) & (m2dotplusmoins > 0)
             & (m2dotmoinsplus > 0) & (m2dotmoinsmoins > 0);
    m2dotneg = (m2dotplusplus < 0) & (m2dotplusmoins < 0)
             & (m2dotmoinsplus < 0) & (m2dotmoinsmoins < 0);
    if m1dotpos then
        if m2dotpos | m2dotneg then
            printf("Erreur %d %d %d\n",i,j,k)
        else // on décentre pour le 2...
            if i==11 then
                grilles_codes(j,k) = 0;
            end
            terme2 = gradm1plus + k1*m1(j)/2
            * (gradm2plus-gradm1plus)^2
            - 3/2*m2(k)* k2^(-1/3)*max(gradm2plus,0)^(2/3);
        end
    elseif m1dotneg then
        if m2dotpos | m2dotneg then
```

```
                printf("Erreur %d %d %d\n",i,j,k)
            else // on décentre pour le 2...
                if i==11 then
                    grilles_codes(j,k) = 2;
                end
                terme2 = gradm1moins + k1*m1(j)/2
                * (gradm2plus-gradm1moins)^2
                - 3/2*m2(k)* k2^(-1/3)*max(gradm2plus,0)^(2/3);
            end
        else
            if m2dotpos then
                if i==11 then
                    grilles_codes(j,k) = 0;
                end
                terme2 = gradm1plus + k1*m1(j)/2
                * (gradm2plus-gradm1plus)^2
                - 3/2*m2(k)* k2^(-1/3)*max(gradm2plus,0)^(2/3);
            elseif m2dotneg then
                if i==11 then
                    grilles_codes(j,k) = 1;
                end
                terme2 = gradm1plus + k1*m1(j)/2
                * (gradm2moins-gradm1plus)^2
                - 3/2*m2(k)* k2^(-1/3)*max(gradm2moins,0)^(2/3);
            else
                if i==11 then
                    grilles_codes(j,k) = -1;
                end
                gradm2 = m2(k)^3/k2;
                gradm1 = gradm2 - 1/(k1*m1(j));
                terme2 = gradm1 + k1*m1(j)/2
                * (gradm2-gradm1)^2
                - 3/2*m2(k)* k2^(-1/3)*max(gradm2,0)^(2/3);
            end
        end
    end
```

```
                    if i==nb_c then
                       gradcplus = grille(i,j,k)-grille(i-1,j,k);
                       gradcmoins = grille(i,j,k)-grille(i-1,j,k);
                    elseif i==1 then
                       gradcplus = grille(i+1,j,k)-grille(i,j,k);
                       gradcmoins = grille(i+1,j,k)-grille(i,j,k);
                    else
                       gradcplus = grille(i+1,j,k)-grille(i,j,k);
                       gradcmoins = grille(i,j,k)-grille(i-1,j,k);
                    end
                    gradcplus = gradcplus / dc;
                    gradcmoins = gradcmoins / dc;
                    if c(i) > c_bar then
                        terme3 = -alpha*(c(i)-c_bar)*gradcmoins;
                    else
                        terme3 = -alpha*(c(i)-c_bar)*gradcplus;
                    end
                    grille_new(i,j,k) = grille(i,j,k)
                                      + dtheta * (terme1 + terme2 + terme3);
                end
            end
        end
        for k=1:nb_m2
            for j=1:nb_m1
                grille(:,j,k) = A*grille_new(:,j,k);
            end
        end
    end
    Phi=grille;
endfunction


function [m1_dyn,m2_dyn]=calc_m_2_box(k1,k2,c_min,c_max,nb_c,
         m1_min,m1_max,nb_m1,m2_min,m2_max,nb_m2,Phi,c_dyn, m1_0,m2_0,T,nb_t)
    dt = T /(nb_t-1);
    m1_dyn = zeros(1,nb_t);
```

```
    m1_dyn(1) = m1_0;
    m2_dyn = zeros(1,nb_t);
    m2_dyn(1) = m2_0;
    dc = (c_max-c_min) / (nb_c-1);
    dm1 = (m1_max-m1_min) / (nb_m1-1);
    dm2 = (m2_max-m2_min) / (nb_m2-1);
    for t=1:nb_t-1
        i = 1+floor((c_dyn(1,t)-c_min)/dc);
        nu = (c_dyn(1,t)-c_min)/dc - floor((c_dyn(1,t)-c_min)/dc);
        j = 1+floor((m1_dyn(1,t)-m1_min)/dm1);
        u1_1 = (Phi(i,j+1,k)-Phi(i,j,k))/dm1;
        u1_2 = (Phi(i+1,j+1,k)-Phi(i+1,j,k))/dm1;
        u1 = nu * u1_2 + (1-nu)*u1_1;
        k = 1+floor((m2_dyn(1,t)-m2_min)/dm2);
        u2_1 = (Phi(i,j,k+1)-Phi(i,j,k))/dm2;
        u2_2 = (Phi(i+1,j,k+1)-Phi(i+1,j,k))/dm2;
        u2 = nu * u2_2 + (1-nu)*u2_1;
        lambda = (k2*u2)^(-1/3);
        delta = k1*(u2-u1);
        m1_dyn(1,t+1) = m1_dyn(1,t)*(1-delta * dt)+dt;
        m2_dyn(1,t+1) = m2_dyn(1,t)*(1-lambda * dt)+delta*m1_dyn(1,t)*dt;
    end
endfunction


function [u1,u2,m1,m2,delta1,lamda,Phi]=build_2_box(k1,k2,r,eta)
    u2=5;
    err=10;
    for i=1:20
        fu2 = r*u2+(3/2)*k2^(-1/3)*u2^(2/3)-k2^(-eta/3)*u2^(-eta/3);
        fpu2 = r+ k2^(-1/3)*u2^(-1/3)+ eta/3 * k2^(-eta/3)*u2^(-eta/3-1);
        u2 = max(0.0001,u2 - fu2/fpu2);
    end
    m2 = (k2*u2)^(1/3);
    u1 = u2 - (-r+sqrt(r^2+2*r*k1*u2))/k1;
    m1 = 1/k1/(u2-u1);
```

```
    delta1 = k1*(u2-u1);
    lamda = (k2*u2)^(-1/3);
    Phi = (m2^(1-eta)/(1-eta) - delta1^2/2/k1 - 1/(2*k2*lamda^2))/r;
endfunction
```