Numerical methods for impulse control problem

Dawid Dopierała

February 2022

1 Introduction to the problem

This report is based on [1], and we will only consider the case given in the chapter 7 of the paper. First of all, let us consider a probability space (Ω, F_t, P) with price process $(X_t)_{t\geq 0}$ and Brownian Motion $(W_t)_{t\geq 0}$ on (Ω, F_t, P) both adapted to filtration $(F_t)_{t\geq 0}$. Then, a classical Black-Scholes model of evolution of asset price X is:

$$dX_t = rX_t dt + \sigma X_t dW_t, \tag{1}$$

where $r, \sigma > 0$ and $X_0 = x$. Of course, we can imagine that we got two different markets with two different stocks and we can cash our asset and go to another market, but every switch has a fixed cost c. In our case we can represent the process of switching between markets with a process $(\gamma_t)_{t\geq 0}$ defined by $\gamma_t := \sum_{k\geq 0} i_k \mathbb{1}_{[\tau_k, \tau_{k+1})}(t)$, where $(\tau_k)_{k\geq 0}$ are stopping times (when we switch the market) and i_k are random variable adapted to the filtration F_{τ_k} with values in $I = \{1, 2\}$ (into which market do we switch). Therefore, we got the following controlled equation:

$$dX_t^\gamma = (r + \nu(\gamma_t)(\mu - r))X_t^\gamma dt + \sigma\nu(\gamma_t)X_t^\gamma dW_t, \tag{2}$$

where $\mu > 0$ and $\nu(i) = i - 1$, so we have assumed that we one of the assets is non risky.

Now keeping in mind that when taking into consideration our expected present value we have to discount both the money and the costs of switching between markets the objective function is:

$$J(x, \gamma) := E\left[\int_0^\infty e^{-r\cdot t} l(X_t^\gamma)dt - \sum_{k\geq 0} e^{-r\cdot\tau_{k+1}} c\right], \tag{3}$$

where $l(x) := 0.5 - |x - 1|$ is the reward function. So now, if we denote by A^i the set of possible control strategies starting with state i ($i_0 = i$ and $\tau_0 = 0$) the value functions are:

$$u_i = \sup_{\gamma\in A^i} J(x, \gamma). \tag{4}$$

It means, that by the Hamilton Bellman Jacobi equations $\forall_{i\neq j}$ and for $x \in (0, \infty)$ we got:

$$\begin{aligned} &min\left[-r \cdot x \cdot u_1'(x) + ru_1(x) - l(x), u_1(x) - u_2(x) + c\right] = 0, \\ &min\left[-\frac{1}{2}\sigma^2 u_2''(x) - \mu \cdot x \cdot u_2'(x) + ru_2(x) - l(x), u_2(x) - u_1(x) + c\right] = 0. \end{aligned} \tag{5}$$

We will use $c = 1/8$, $\mu = 0.06$, $r = 0.02$, $\sigma = 0.2$ and consider only the part of the space $x \in (0, 2)$ setting $u_i(0) = u_i(2) = 0$ (the first one follows from the above equation and the second one is an assumption). Considering a uniform grid $x_k = k \cdot h$, $k = 1, 2, ..., N$ on $(0, 2)$ and approximations $u_{i,k}$ of $u_i(x_k)$ we have to use discretization of the derivatives such that $\forall_{l\neq k}$ equation for $u_{i,k}$ has to be non-increasing with respect to $u_{i,l}$. For this reason I am using the following approximations:

$$\begin{aligned} u_i'(x_k) &\approx \frac{u_{i,k+1} - u_{i,k}}{h}, \\ u_i''(x_k) &\approx \frac{u_{i,k+1} - 2u_{i,k} + u_{i,k-1}}{h^2}. \end{aligned} \tag{6}$$

Now we define vectors $\vec{\alpha}$, $\vec{\beta}$, $\vec{r}$ as:

$$\begin{aligned} \alpha_k &:= \frac{\sigma^2 x_k^2}{2h^2}, \\ \beta_k &:= \frac{x_k}{h}, \\ r_k &:= r, \end{aligned} \tag{7}$$

we get that the matrices necessary for the discrete approximations are as follows: A_0 is $N \times N$ upper diagonal matrix with $\vec{r} + r\vec{\beta}$ on the diagonal and $-r\vec{\beta}$ (without the last element of the vector) on the upper diagonal, while A_1 is $N \times N$ tridiagonal matrix with $2\vec{\alpha} + \vec{r} + \mu\vec{\beta}$ on the diagonal, $-\vec{\alpha}$ (without the first element of the vector) on the lower diagonal and $-\vec{\alpha} - \mu\vec{\beta}$ (without the last element of the vector) on the upper diagonal. If we now also denote Id as the $N \times N$ identity matrix and $\mathbf{0}$ as $N \times N$ zero matrix we can define:

$$\mathbf{A} := \begin{bmatrix} A_0 & \vec{0} \\ \vec{0} & A_1 \end{bmatrix}, \mathbf{M} := \begin{bmatrix} Id & -Id \\ -Id & Id \end{bmatrix}, \tag{8}$$

and $\vec{u} := (\vec{u_0}, \vec{u_1})^T$ then the discretized equations (5) become (keep in mind that since u_i are zeros on the boundaries we do not have to use a vector accounting for the boundary conditions):

$$min\left(\mathbf{A}\vec{u} - \vec{l}, \mathbf{M}\vec{u} - \vec{b}\right) = 0, \tag{9}$$

where $\vec{b}$ is the vector with all elements equal $-c$.

2 Numerical computations

2.1 Algorithms

To solve (9) we can use policy iteration scheme. We initialize $\omega^{(0)} \in \{0,1\}^{2N}$, then for $k \geq 0$ and $k \in \{0,1,...,N-1\}$ compute:

$$\begin{aligned} A_i^{(k)} &= (1-\omega_i^{(k)})A_i + \omega_i^{(k)} M_i, \\ b_i^{(k)} &= (1-\omega_i^{(k)})\vec{l}_i + \omega_i^{(k)}\vec{b}_i, \end{aligned} \tag{10}$$

then we set[1]:

$$\begin{aligned} & u^{(k)} = A^{(k),-1} b^{(k)}, \\ & \omega^{k+1} \in arg \min_{\omega \in \{0,1\}^{2N}} \left[(1-\omega)(\mathbf{A}\vec{u}^{(k)} - \vec{l}) + \omega(\mathbf{M}\vec{u}^{(k)} - \vec{b}) \right]. \end{aligned} \tag{11}$$

We finish the iterations either when the number of iterations is big enough (bigger than some k_{max}) or when the following condition is satisfied:

$$\frac{\left\|\vec{u}^{(k)} - \vec{u}^{(k-1)}\right\|_\infty}{max(scale, \left\|\vec{u}^{(k)}\right\|_\infty)} < tol, \tag{12}$$

where *scale* and *tol* are some real positive numbers.
To improve the above algorithm we can use penalized equation. The idea is very simple, for $\rho > 0$ we consider problem:

$$\mathbf{A}\vec{u} - \vec{l} + \rho \cdot \min \left[\mathbf{M}\vec{u} - \vec{b}, 0 \right] = 0, \tag{13}$$

which can also be written as:

$$\min \left[\mathbf{A}\vec{u} - \vec{l}, (\rho\mathbf{M} + \mathbf{A})\,\vec{u} - \left(\rho\vec{b} + \vec{l}\right) \right] = 0. \tag{14}$$

Now, when we take $\rho \to \infty$ then clearly the solution of above equation is the same as the solution of (9). To solve the above equation we can simply use the policy iteration scheme (just like for the non penalized equation).

We can also try out two variants of the PSOR algorithm: one when we just iterate over successive rows and one when iterate over both u_i at the same time, but again both will fail so I dont include those in the report (the code is in the notebook).

2.2 Outcome of numerical computations

First of all, for all computations we set $tol = 0.0001$, $scale = 1$ and $k_{max} = 200$ (maximum number of algorithm iterations). One of the graphs that we got from the iterated policy scheme is:

[1]Of course we do not compute the inverse of A, we just solve the linear system.

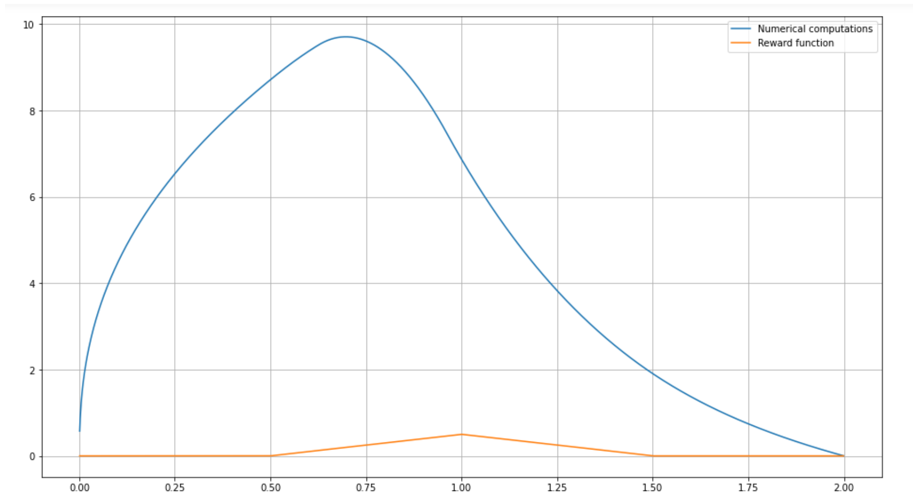
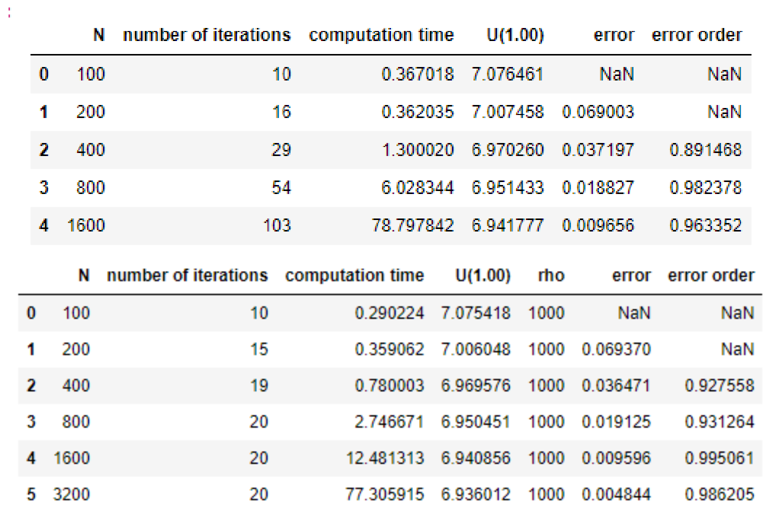


Figure 1: Graph for the iterated policy scheme for N=1600.

The above graph is the same as in the [1]. The table for the iterated policy scheme and penalized policy with $\rho = 1000$ is as follows (keep in mind that the number of iterations changes for every row, so we should take the computed numerical order with a grain of salt):



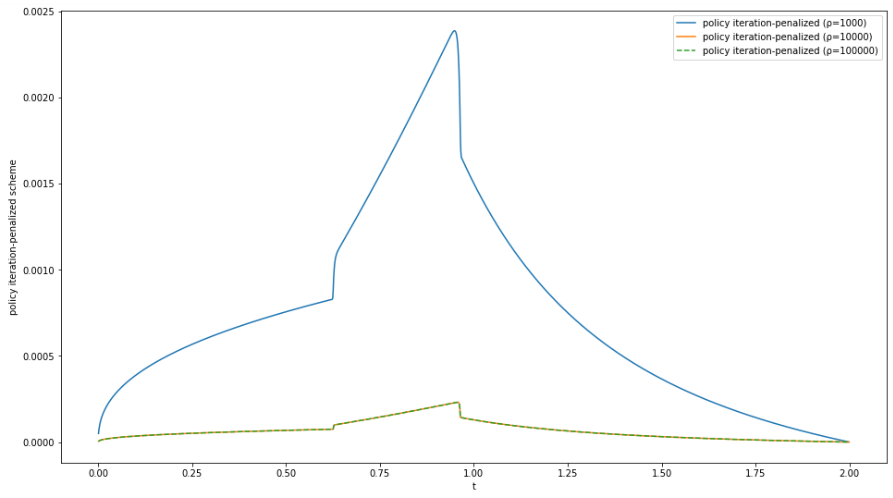
	N	number of iterations	computation time	U(1.00)	error	error order
0	100	10	0.367018	7.076461	NaN	NaN
1	200	16	0.362035	7.007458	0.069003	NaN
2	400	29	1.300020	6.970260	0.037197	0.891468
3	800	54	6.028344	6.951433	0.018827	0.982378
4	1600	103	78.797842	6.941777	0.009656	0.963352

	N	number of iterations	computation time	U(1.00)	rho	error	error order
0	100	10	0.290224	7.075418	1000	NaN	NaN
1	200	15	0.359062	7.006048	1000	0.069370	NaN
2	400	19	0.780003	6.969576	1000	0.036471	0.927558
3	800	20	2.746671	6.950451	1000	0.019125	0.931264
4	1600	20	12.481313	6.940856	1000	0.009596	0.995061
5	3200	20	77.305915	6.936012	1000	0.004844	0.986205

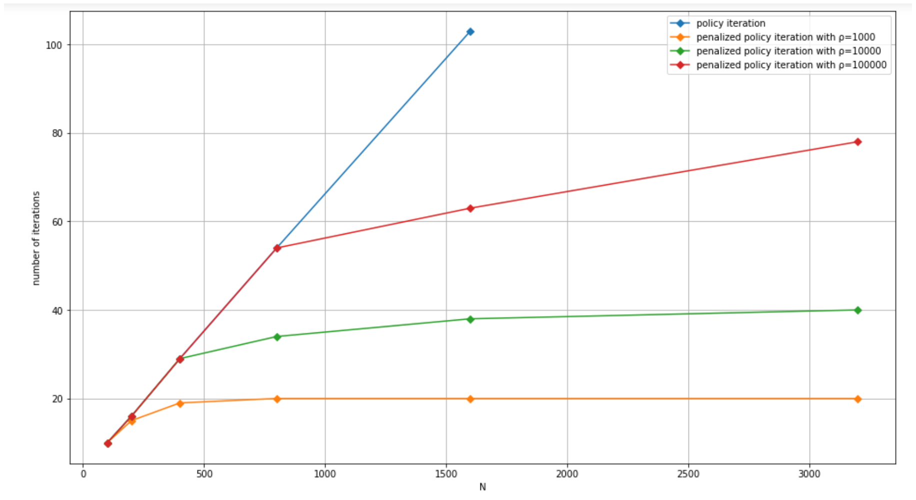
Figure 2: Table for policy iteration scheme (above) and penalized policy iteration with $\rho = 1000$ (below).

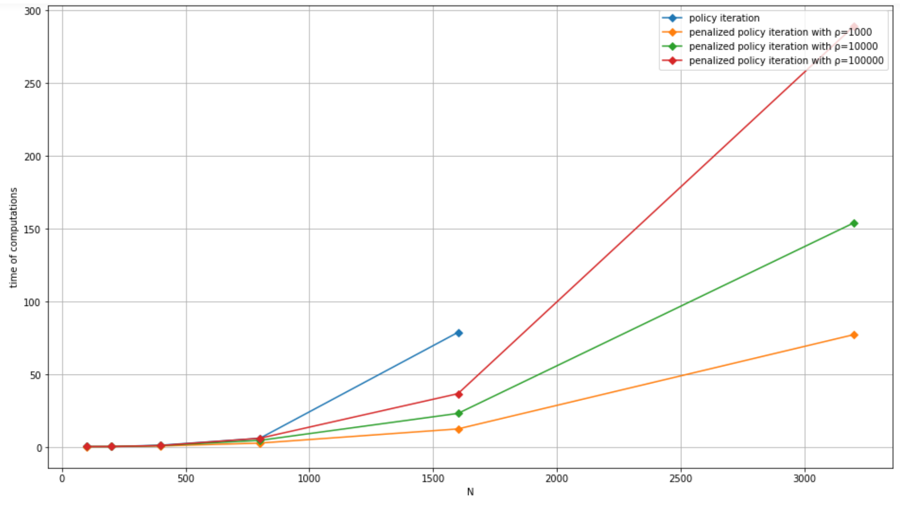
Clearly the penalized scheme computes way faster, and order of both schemes seems to be 1[2]. We can look at the graph of the difference between the outcome of the policy iteration for the vanilla problem and the penalized one (in this case for N=800):

[2]We should remember that we compute the error with respect to the previous computation instead of the true value (which we sadly do not know), which as we had seen in the first homework may gives us wrong order.



The above graph is the same as in the [1]. Interestingly enough, the shape of the difference seems to be the same for every ρ (it's just scaled on the y axis). Now we can make some graphs to look on how does the time of computations and number of iterations depends on N for various algorithms:





As we can see for the penalized problem the number of iterations seems to increase very slowly with N (something like square root), while the number of iterations for the vanilla problem increases linearly. In contrary, for the time of the computation all algorithms tends to increase in the same manner (something like square). The second part is in the contrary with the outcome in the paper, probably because the code used there was well optimized.

References

[1] C. Reisinger, Y. Zhang, "Error estimates of penalty schemes for quasi-variational inequalities arising from impulse control problems", arXiv:1901.07841v2