DIPARTIMENTO DI MATEMATICA
"Francesco Brioschi"
POLITECNICO DI MILANO

Pricing Exotic Derivatives Exploiting Structure

Sesana, D.; Marazzina, D.; Fusai, G.

Collezione dei *Quaderni di Dipartimento*, numero **QDD 128**
Inserito negli *Archivi Digitali di Dipartimento* in data 21-7-2012

Piazza Leonardo da Vinci, 32 - 20133 Milano (Italy)

QFinLab Working Paper Series

Working Paper Series of the Quantitative Finance Laboratory *QFinLab*.
The main research areas include: mathematical finance, corporate finance, asset allocation, computational finance, financial engineering, financial time series analysis, corporate governance.

QDD 081 E. Barucci & F. Gazzola, *Prices in the utility function and demand monotonicity* (2010)

QDD 088 S. Corsaro, D. Marazzina, Z. Marino, *Wavelet techniques for option pricing on advanced architectures* (2011)

QDD 089 E. Barucci & D. Marazzina, *Optimal Investment, Stochastic Labor Income and Retirement* (2011)

QDD 090 E. Barucci & A. Cosso, *Does an equity holding tax help to stabilize a VaR regulated financial market?* (2011)

QDD 091 E. Barucci & A. Cosso, *Portfolio choices and VaR constraint with a defaultable asset* (2011)

QDD 106 G. Fusai, G. Germano, D. Marazzina, *Pricing Credit Derivatives in a Wiener-Hopf Framework* (2011)

QFinLab website: www.mate.polimi.it/qfinlab

Pricing Exotic Derivatives Exploiting Structure

Debora Sesana

Dipartimento S.E.I., Università del Piemonte Orientale A. Avogadro, via Perrone 18, I-28100 Novara, Italy

Daniele Marazzina*

Dipartimento di Matematica, Politecnico di Milano, Piazza Leonardo da Vinci 32, I-20133 Milano, Italy

Gianluca Fusai

Dipartimento S.E.I., Università del Piemonte Orientale A. Avogadro, via Perrone 18, I-28100 Novara, Italy **&** *Faculty of Finance, Cass Business School, 106 Bunhill Row, London EC1Y 8TZ, UK*

Abstract

In this paper we introduce a new fast and accurate numerical method for pricing exotic derivatives when discrete monitoring is applied. The algorithm is general and is examined in detail with reference to the CEV (Constant Elasticity of Variance) process, for which up to date no efficient procedures are available. The approach exploits the structure of the matrix arising from the numerical quadrature of the pricing backward formulas to devise a convenient factorization that helps greatly in the speed-up of the recursion. The algorithm is applied to different exotic derivatives, such as Asian, barrier, Bermudan, lookback and step options. Extensive numerical experiments confirm the theoretical results.

Keywords: CEV Process, Exotic Derivatives, Matrix Factorization, Numerical Quadrature, Option Pricing, Recursive Pricing Formula

*Corresponding author

Email addresses: debora.sesana@eco.unipmn.it (Debora Sesana), daniele.marazzina@polimi.it (Daniele Marazzina), gianluca.fusai@eco.unipmn.it (Gianluca Fusai)

1. Introduction

Over the years exotic options, such as Asian, barrier, and lookback contracts, have become more and more popular in equity markets and raised the attention in the academic research. Most of the articles in this literature price these contracts assuming a continuous monitoring, i.e., the payoff is triggered by events occurring continuously before expiry. For example, for a barrier option the barrier crossing event is monitored continuously. Standard contractual features usually specify discrete monitoring. As shown at first in [4], with reference to the geometric Brownian motion (GBM) model, price differences between continuous and discrete monitoring can be very large. However, under this contractual feature no analytical formula is at the moment available, except when the underlying evolves according to a GBM process, see [13].

The discretely monitoring pricing procedure is based on the standard backward recursion. The numerical implementation involves a recursive numerical quadrature (integration) as shown for example in [1, 12]. A considerable speed-up can be obtained when the recursion is of convolution type, since in this case we can exploit the Fast Fourier Transform (FFT) algorithm. Indeed, if the underlying asset evolves according to a exponential Lévy model then the log-price transition density has a convolution structure and the pricing recursion can be implemented by applying at each step a Fourier transform-convolution-Fourier inversion. This can be done efficiently through the use of the FFT, see for example [10, 12, 17]. The convolution structure of the transition density depends on the fact that the log-price increments follow an independent and identically distributed (i.i.d.) process. Unfortunately, if the i.i.d. assumption does not hold, the numerical integration becomes computational intensive having a cost proportional to $O(m^2)$ respect to the $O(m \log(m))$ cost of the FFT algorithm (here m refers to the number of discretization points of the integral).

The main idea of the present paper is to investigate the structure in the probability transition density of the underlying asset and of its sampling matrix. More precisely, we introduce the concept of cluster of eigenvalues of a sequence of matrices arising from the numerical quadrature of the backward recursion as we increase the number of nodes m. We formally prove (Theorem 3) that the number of significant eigenvalues (i.e., larger than a fixed tolerance ϵ) is approaching a constant r_ϵ as we take larger values of m. This result can be exploited to factorize the iteration matrix, giving a computational cost of $O(k_\epsilon m)$ operations, $k_\epsilon \approx r_\epsilon$, for the matrix-vector multiplication, instead of the standard $O(m^2)$. Given that the cost of the factorization is nearly independent on the number of monitoring dates, the

advantage of our approach will be the greatest, greater the number of monitoring dates. The algorithm is general and it is examined in detail with reference to the Constant Elasticity of Variance (CEV) model, introduced by [7, 8]. This dynamic is interesting allowing for very different transition densities and implied volatility shapes. Very few option pricing models yield fully analytical results, and most require numerical evaluations. The CEV model is not an exception.

Numerical methods for pricing derivatives under the CEV process are presented, for example, in [3, 6, 19]. These articles price derivatives contracts, like barrier [3], lookback [3, 6] and geometric Asian [19] options, via a lattice approach, such as binomial [6, 19] and trinomial [3] trees. The main limit of this approach is the slow and erratic convergence to the true price, a phenomena that is well-known to affect the tree approximation to the GBM dynamics. We can have large errors even with thousands of time steps and millions of node calculations. American options are considered in [18], where the author proposes an alternative characterization of the early exercise premium that is valid for any Markovian and diffusive underlying price process. Finally, an analytical Laplace transform approach based on the scale function of a diffusion process is pursued in [9]. These authors obtain Laplace transform of barrier and lookback option prices involving Whittaker and Bessel functions of complex argument. Option prices are then obtained via a numerical inversion of the Laplace transform. Unfortunately, the procedure is quite computational intensive mainly for lookback options: this problem requires the numerical computation of an integral involving the inverse Laplace transform.

We are not aware of investigations on the structure of the probability transition density when the FFT algorithm cannot be used, and in particular when the CEV dynamic is considered. Therefore this investigation represents the main contribution of the present paper. In addition, we show how the proposed approach can be applied to a large class of exotic derivatives such as Asian, barriers, Bermudan, lookback and step options. Extensive numerical experiments are conducted to compare the accuracy and the computational cost of our algorithm with respect to a standard backward recursive quadrature and to Monte Carlo simulation. In particular, numerical experiments confirm that the greatest benefits are achieved for a large number of monitoring dates.

The structure of the paper is as follows. First of all, in Section 2 we describe the general setup to price exotic derivatives with the discrete monitoring feature. In Section 3 we deal with the quadrature approach to solve the recursive pricing formulas, while in Section 4 we study the structure of the pricing matrix and we introduce the factorization idea. Finally, in Sec-

tion 5 we validate the pricing procedure with numerical results assuming a CEV dynamic for the underlying. Accuracy and computational cost of our pricing algorithm are compared with the above mentioned benchmarks, i.e., the classical quadrature approach and Monte Carlo simulation.

2. The Option Pricing Problem

Let us consider a derivative contract with a payoff $\phi(\cdot)$ at maturity $T = N\Delta$, where N is the number of Δ-equally spaced monitoring dates, and let S be the underlying asset price. The standard backward procedure computes the derivative price $V(S, n)$ at time $n\Delta$ through the following recursion (eventually with a modification to deal with the early exercise feature):

$$V(S,n) = e^{-r\Delta} \int_{\Omega} p(S,\xi;\Delta)\, V(\xi, n+1)\, d\xi, \quad n = N-1, \cdots, 0, \tag{1}$$

where r is the risk-free rate, and $p(S, \xi; \Delta)$ is the transition density from S at time t to ξ at time $t + \Delta$. Ω refers to the integration domain and can vary depending on the trigger event. The above recursion starts with the payoff condition at maturity $V(S, N) = \phi(S)$. We are interested in computing $V(S_0, 0)$, S_0 being the current spot price.

In the following subsections, we show how the above framework fits different exotic contracts.

2.1. Barrier Options

If we deal with barrier options, the pricing recursion (1) starts from the payoff function $\phi(S) := (\varphi(S - E))^+$, where E is the strike price and φ is a binary variables taking value 1 for calls, and -1 for puts. If we denote with L (U) the lower (upper) barrier, the domain Ω is $(L, +\infty)$ for down-and-out, (L, U) for knock-and-out, and $(0, U)$ for up-and-out barrier options.
For numerical purposes, the integration interval in (1) is truncated to $(L, \mathcal{U})$ - for down-and-out options - or $(\mathcal{L}, U)$ - for up-and-out options - with $\mathcal{L} < S_0$ ($\mathcal{U} > S_0$). The truncation is chosen such that the probability of moving from S_0 to $\mathcal{L}$ ($\mathcal{U}$) is less than a preassigned tolerance.

2.2. Bermudan Options

A Bermudan option gives the holder the right to early exercise at each monitoring date. This option is worth more than the corresponding European version, but less than the American counterparty, for which the exercise

occurs continuously. To take into account the early exercise possibility we modify (1) into:

$$V(S,n) = \max\left\{ e^{-r\Delta} \int_{\Omega} p\left(S,\xi;\Delta\right) V\left(\xi,n+1\right) d\xi,\ \phi(S) \right\},\ n = N-1,\cdots,0, \tag{2}$$

with $\Omega = (0,+\infty)$ for standard Bermudan options. If we have Bermudan contracts with a barrier trigger, then $\Omega = (L,+\infty)$, $\Omega = (0,U)$ or $\Omega = (L,U)$. Payoff function and domain truncation are as in Section 2.1.

2.3. Lookback Options

The maturity settlement of lookback options is based on the minimum or the maximum value of the underlying asset as registered during the lifetime of the option. At maturity, the holder can "look-back" and select the most favorable figure of the underlying as occurring at the monitoring dates. If we let $S(n\Delta)$ to be the asset price at the n-th monitoring date, we can define the discretely observed minimum price as

$$J_n := \min\{S(0),\cdots,S(n\Delta)\}.$$

The payoff function of a fixed-strike lookback on the minimum is given by $(E - J_N)^+$. The lookback option price at time $n\Delta$ depends on the underlying asset price S, and on the up-to-date minimum $J_n = J$ and we denote it by $V(S,J,n)$. Clearly it must be $J \leq S$. Similar considerations hold for payoffs written on the maximum.
Respect to the GBM dynamic, where a change of numeraire argument reduces the number of state variables, under a more general process specification we must keep track of both state variables, underlying price and running minimum. Given that

$$J_{n+1} = \min\{J_n, S((n+1)\Delta)\},$$

the backward recursion becomes

$$\begin{aligned} V(S,J,N) &= \left(\varphi\left(J-E\right)\right)^+, \\ V(S,J,n) &= e^{-r\Delta} \int_0^{+\infty} p(S,\xi;\Delta) V(\xi,\min\{J,\xi\},n+1) d\xi, \end{aligned} \tag{3}$$

for $n = N-1,\cdots,0$ and $S \geq J$. Here $\min\{J,\xi\}$ is the minimum value of the underlying asset at the $(n+1)$-th monitoring date given that $J_n = J$ and $S((n+1)\Delta) = \xi$. The initial option price is then given by $V(S_0,S_0,0)$.

2.4. Asian Options

Asian options are a very popular type of exotic derivative. Such as for lookback options, their pricing requires the introduction of a new state variable, i.e., the (arithmetic) average up to time $n\Delta$

$$A_n = \frac{1}{n+1}\sum_{i=0}^{n} S(n\Delta).$$

The arithmetic average follows the updating rule

$$A_{n+1} = \frac{n+1}{n+2}A_n + \frac{1}{n+2}S((n+1)\Delta),$$

so that the price of the arithmetic fixed-strike Asian option satisfies the following backward recursion:

$$V(S,A,n) = e^{-r\Delta}\int_0^{+\infty} p(S,\xi;\Delta)V\left(\xi, \frac{n+1}{n+2}A + \frac{1}{n+2}\xi, n+1\right)d\xi, \quad (4)$$

for $n = N-1,\cdots,0$, with $V(S,A,N) = (\varphi(A-E))^+$.

2.5. Step Options

Step options are similar to barrier options, but the knock-and-out feature operates only gradually. To this aim we define the occupation time I_n of the subset $\mathcal{I}$, $\mathcal{I}\subset\mathbb{R}^+$,

$$I_n = \sum_{i=1}^{n} \mathbf{1}_{\{S(i\Delta)\in\mathcal{I}\}},$$

where $\mathbf{1}_{\{S(i\Delta)\in\mathcal{I}\}}$ is the indicator function, i.e., it is equal to 1 if $S(i\Delta)\in\mathcal{I}$, 0 otherwise. Notice that I_n measures the time spent by the underlying asset in the set $\mathcal{I}$ up to time $n\Delta$. I_n takes values in $\{0,1,2,\cdots,n\}$ and satisfies the updating rule

$$I_{n+1} = I_n + \mathbf{1}_{\{S((n+1)\Delta)\in\mathcal{I}\}}.$$

Given $S(N\Delta) = S$ and $I_N = I$, the payoff of a step option with principal amortization below the barrier is

$$V(S,I,N) = \left(1-\frac{\rho}{N}I\right)^+ (\varphi(S-E))^+,$$

where ρ is the knock-out killing rate. The introduction of the knock-out range has the advantage of regularize the barrier option by making the price and the delta continuous at the barrier, see for example [16]. The price recursion for step options reads as: for $n = N-1,\cdots,0$

$$V(S,I,n) = e^{-r\Delta}\int_0^{+\infty} p(S,\xi;\Delta)V\left(\xi, I+\mathbf{1}_{\{\xi\in\mathcal{I}\}}, n+1\right)d\xi.$$

3. The Quadrature Approach

As shown in the previous section, the general pricing framework requires the numerical computation of the following recursive integral equation

$$W(x,n) = \int_a^b H(x,y;\Delta)W(y,n+1)dy, \quad \forall\, x \in (a,b), \tag{5}$$

for $n = N-1, \cdots, 0$, with $W(x,N)$ assigned. This recursion holds for European and barrier options. Bermudan options also require an early exercise clause.

If we have more than one state variable (such as for lookback, Asian and step derivatives), the function W depends on the time index n and on two state variables, so that we write $W(x,\cdot,n)$. The additional state variable is the minimum value J if lookback options are considered, the average value A for Asian contracts and the occupation time I for step options.

If we apply a quadrature formula to (5), with nodes s_i and weights w_i, $i = 0, \cdots, m-1$, we obtain

$$W(s_i,n) = \sum_{j=0}^{m-1} w_j H(s_i,s_j;\Delta)W(s_j,n+1). \tag{6}$$

If we define the matrix $\mathbf{H}_m$ as $\mathbf{H}_m = [H(s_i,s_j,\Delta)]_{i,j=0}^{m-1}$, then (6) can be written as

$$\mathbf{W}_n = \mathbf{H}_m\mathbf{D}_m\mathbf{W}_{n+1}, \quad n = N-1, \cdots, 0, \tag{7}$$

where $\mathbf{W}_n = [W(s_i,n)]_{i=0}^{m-1}$, and $\mathbf{D}_m = \mathrm{diag}(w_0,\ldots,w_{m-1})$. The matrix $\mathbf{H}_m$ is called the sampling matrix of the function H, while $\mathbf{H}_m\mathbf{D}_m$ is the iteration matrix of the backward procedure.

In the recursion (7) the size of the iteration matrix equals the number of discretization points (nodes). It is well-known that increasing the number of nodes improves the accuracy of the solution. More precisely, the speed of convergence of the quadrature error to zero can be determined by using results on the speed of convergence of the integration rule when it is applied to the integral $\int_\Omega H(\cdot,\xi;\Delta)d\xi$, as discussed in [2, Chapter 4]. In this setting the backward recursion (7) has a cost proportional to $N\,m^2$ operations. Our aim is to reduce this cost substantially by exploiting the spectral properties of the iteration matrix $\mathbf{H}_m\mathbf{D}_m$ as the matrix size m grows. This is discussed in Section 4.1.

In the following we detail how the quadrature applies to the different contractual settings.

3.1. Barrier and Bermudan Options

For barrier options, i.e., recursions (1) and (2), we have $W(x, N) = \phi(x)$, $H(x, y; \Delta) = e^{-r\Delta} p(x, y; \Delta)$, and a, b depend on the domain Ω. For example, for down-and-out barrier options, we set $a = L$ and $b = \mathcal{U}$. From the discretization of (1), we obtain a recursion of the form (7) with $\mathbf{W}_N = [\phi(s_i)]_{i=0}^{m-1}$.

If Bermudan options are considered, we have $W(x, N)$, $H(x, y; \Delta)$, and a, b as above. From the discretization of (2) we obtain

$$\begin{cases} \mathbf{W}_n^{CV} = \mathbf{H}_m \mathbf{D}_m \mathbf{W}_{n+1} \\ \mathbf{W}_n = \max\{\mathbf{W}_n^{CV}, \mathbf{\Phi}\}, \end{cases}$$

where $\mathbf{W}^{CV}$ is the continuation value, and $\mathbf{\Phi} = [\phi(s_i)]_{i=0}^{m-1} = \mathbf{W}_N$.

3.2. Other Derivatives

In this section we discuss the discretization for lookback, Asian and Step options, that are characterized by having the same sampling matrix but differ due to an additional updating at each step.

3.2.1. Lookback Options

If fixed-strike lookback put options are considered, we set $H(x, y; \Delta) = e^{-r\Delta} p(x, y; \Delta)$, $a = 0$ and $b = +\infty$ (truncated to $\mathcal{L}$ and $\mathcal{U}$ for numerical valuation). Thus the semi-discrete formulation of (3) is

$$W(s_i, J, n) = \sum_{l=0}^{m-1} w_l H(s_i, s_l; \Delta) W(s_l, \min\{J, s_l\}, n+1),$$

$i = 0, \ldots, m-1$, with $W(s_i, J, N) = (E - \min\{s_i, J\})^+$. Since J is the minimum value of the underlying asset, we discretize J on the same grid $\{s_i\}_{i=0}^{m-1}$ used for the underlying asset. More precisely, we can implement recursion (3) as follows: considering m quadrature nodes s_j and weights w_j, $j = 0, \cdots, m-1$, we define for $n = N-1, \cdots, 0$ the vectors

$$\begin{aligned} \mathbf{W}_n^j &:= [W(s_i, s_j, n)]_{i=0}^{m-1}, \\ \hat{\mathbf{W}}_n^j &:= [W(s_i, \min\{s_i, s_j\}, n)]_{i=0}^{m-1}, \end{aligned}$$

with $\hat{\mathbf{W}}_N^j = [(E - \min\{s_i, s_j\})^+]_{i=0}^{m-1}$.

The fully-discretized lookback recursion (3) is: for $n = N-1, \cdots, 0$

$$\mathbf{W}_n^j = \mathbf{H}_m \mathbf{D}_m \hat{\mathbf{W}}_{n+1}^j,$$

with $\mathbf{D}_m$ and $\mathbf{H}_m$ defined as above. Notice that $(\hat{\mathbf{W}}_n^j)_i$ corresponds to $W(s_i, s_j, n)$ (and thus to $(\mathbf{W}_n^j)_i$) only if $s_i \geq s_j$, and thus $i \geq j$. Thus, moving from $\mathbf{W}_n^j$ to $\hat{\mathbf{W}}_n^j$, an update of the minimum value is necessary for the indices i such that $s_j > s_i$. This implies that the pricing recursion can be written as: for $n = N-1, \cdots, 0$ and $j = 0, \cdots, m-1$,

$$\begin{cases} \mathbf{W}_n^j = \mathbf{H}_m \mathbf{D}_m \hat{\mathbf{W}}_{n+1}^j & \text{Recursion Step} \\ (\hat{\mathbf{W}}_n^j)_i = \begin{cases} (\mathbf{W}_n^j)_i & \text{if } i \geq j \\ (\mathbf{W}_n^i)_i & \text{if } i < j \end{cases} & \text{Updating Step} \end{cases} \tag{8}$$

Remark 1. The iteration (8) can be accelerated using a subset $\widetilde{s}$ of m/ω quadrature nodes, i.e., for $n = N-1, \cdots, 0$, for $j_\omega = 0, \cdots, m/\omega - 1$, being $(\mathbf{W}_n^{j_\omega})_i = W(s_i, \widetilde{s}_{j_\omega}, n)$, it holds

$$\begin{cases} \mathbf{W}_n^{j_\omega} = \mathbf{H}_m \mathbf{D}_m \hat{\mathbf{W}}_{n+1}^{j_\omega}, \\ (\hat{\mathbf{W}}_n^{j_\omega})_i = \begin{cases} (\mathbf{W}_n^{j_\omega})_i & \text{if } i \geq j_\omega, \\ (\mathbf{W}_n^i)_i & \text{if } i < j_\omega, \end{cases} \end{cases}$$

where the element $(\mathbf{W}_n^i)_i$ is computed by cubic interpolation if the node s_i do not belong to the subgrid $\widetilde{s}$.

3.2.2. Asian Options

For Asian options, at each step the possible new values of the state variable A do not fall on the A-grid at the previous step. Therefore an interpolation is also required at each iteration. More precisely, the semi-discrete formulation of (4) is: for $i = 0, \ldots, m-1$

$$W(s_i, A, n) = \sum_{l=0}^{m-1} w_l H(s_i, s_l; \Delta) W\left(s_l, \frac{n+1}{n+2} A + \frac{1}{n+2} s_l, n+1\right).$$

To obtain the fully-discrete formulation, we define

$$\mathbf{W}_n^j = \left(W\left(s_i, s_j, n\right)\right)_{i=0}^{m-1}, \quad j = 0, \cdots, m-1,$$

and

$$\hat{\mathbf{W}}_n^j = \left(W\left(s_i, \frac{n}{n+1} s_j + \frac{1}{n+1} s_i, n\right)\right)_{i=0}^{m-1}, \quad j = 0, \cdots, m-1.$$

Thus the discretization of (4) can be written as: for $n = N-1, \cdots, 0$, given $\mathbf{W}_{n+1}^j$, compute $\hat{\mathbf{W}}_n^j$ exploiting cubic interpolation, and then set

$$\mathbf{W}_n^j = \mathbf{H}_m \mathbf{D}_m \hat{\mathbf{W}}_{n+1}^j.$$

Again, as stated in Remark 1, we can assume that the average falls on a subset $\widetilde{s}$ of m/ω quadrature nodes, i.e.,

$$\mathbf{W}_n^{j_\omega} = \left(W\left(s_i, s_{j_\omega}, n\right)\right)_{i=0}^{m-1}, \quad j_\omega = 0, \cdots, \frac{m}{\omega} - 1.$$

3.2.3. Step Options

Finally, for step derivatives, I_n can only assume the $n+1$ values $0, 1, \cdots, n$. The payoff is discretized according to

$$\hat{\mathbf{W}}_N^j = \left(\left(1 - \frac{\rho}{N}\left(j + \mathbf{1}_{\{s_i \in \mathcal{I}\}}\right)\right)\left(\varphi(s_i - E)\right)^+\right)_{i=0}^m, \quad j = 0, \cdots, N-1.$$

Then, for $n = N-1, \cdots, 0$, we compute

$$\mathbf{W}_n^j = \mathbf{H}_m \mathbf{D}_m \hat{\mathbf{W}}_{n+1}^j, \qquad j = 0, \cdots, n,$$

and then, if $n > 0$, we construct $\hat{\mathbf{W}}_n^j$ as follows: for $i = 0, \cdots, m-1$, for $j = 0, \cdots, n-1$, if $s_i \notin \mathcal{I}$, then $\left(\hat{\mathbf{W}}_n^j\right)_i = (\mathbf{W}_n^j)_i$, otherwise $\left(\hat{\mathbf{W}}_n^j\right)_i = (\mathbf{W}_n^{j+1})_i$.

4. Matrix Factorization

Section 3 has shown how the numerical quadrature of Equation (5) gives us a recursion with iteration matrix **HD** (we omit the subscripts to simplify the notation). It turns out that this matrix is the same for European, Bermudan, Asian, lookback and step options. The knock-out trigger event in barrier options generates a different structure of the domain Ω and thus of the iteration matrix. In this section, we analyze the spectral properties of the matrix **HD**. These properties allow us to factorize **HD** by using the bidiagonalization algorithm of Golub and Reinsch [14]. The factorization consists in replacing the iteration matrix by the product $\mathbf{PJQ}^H$, where **P** and **Q** are unitary matrices, $\mathbf{Q}^H$ is the hermitian of matrix **Q**, and **J** is a bidiagonal matrix.[1]

For this purpose, at first we define the cluster point of eigenvalues of a sequence of matrices of increasing dimension (that can be obtained, for example, by increasing the number of nodes in the quadrature formula (7)). This means to analyze the asymptotic behavior of eigenvalues of a sequence of matrices, and to show that they cluster around zero. In particular, the number r_ϵ of eigenvalues greater than a fixed tolerance ϵ remains constant as we increase the number of quadrature nodes. Therefore the factorization of the matrix **HD** into the product $\mathbf{PJQ}^H$ returns a bidiagonal matrix **J** having as non-zero elements only the first k_ϵ elements, $k_\epsilon \approx r_\epsilon$, on the main and on the upper diagonals. This means that it is not necessary to perform the full factorization $\mathbf{PJQ}^H$, but we can stop at the k_ϵ-th step. If k_ϵ is much smaller than m, where m is the dimension of the matrix, this implies a

[1]For the sake of completeness, we recall that the hermitian matrix of a matrix **Q** is its transpose conjugate, and **Q** is unitary if and only if $\mathbf{QQ}^H$ is the identity matrix.

significant reduction in the computational cost. Furthermore, the particular "composition" of matrices $\mathbf{P}$ and $\mathbf{Q}$ allows us to compute the matrix-vector multiplication in $O(k_\epsilon m)$ operations instead of the $O(m^2)$ characterizing the standard recursive approach.

4.1. Spectral properties of a sequence of matrices

In this section we give the main definitions and theorems related to the spectral properties of sequences of matrices that satisfy certain conditions. In the following, we denote with $\{\mathbf{A}_m\} = \{\mathbf{A}_m\}_{m=1}^{\infty}$ a sequence of matrices in $\mathbb{C}^{m\times m}$ joined by a structural content that remains unchanged when the size varies. For brevity, we will denote the sequence simply by $\{\mathbf{A}_m\}$ and we use the symbols $\lambda_j^{(m)}$ and $\sigma_j^{(m)}$, $j = 1, \ldots, m$, to denote, respectively, the eigenvalues and the singular values of $\mathbf{A}_m$.

A property of the spectrum of a sequence of matrices is its cluster point.

Definition 1. [24] A matrix sequence $\{\mathbf{A}_m\}$ is *strongly clustered at* $s \in \mathbb{C}$ (in the eigenvalue sense), if for any $\varepsilon > 0$ the number of the eigenvalues of $\mathbf{A}_m$ off the disc $D(s,\varepsilon) := \{z : |z - s| < \varepsilon\}$ can be bounded by a pure constant q_ε possibly depending on ε, but not on m. In other words

$$q_\varepsilon(m, s) := \#\{j : \lambda_j^{(m)} \notin D(s,\varepsilon)\} = O(1), \quad m \to \infty.$$

If every $\mathbf{A}_m$ has only real eigenvalues (at least for large m) then we may assume that s is real and that the disc $D(s,\varepsilon)$ is the interval $(s-\varepsilon, s+\varepsilon)$. We replace the term "strongly" by "weakly", if $q_\varepsilon(m, s) = o(m)$, when $m \to \infty$. Similar definitions hold if we replace the term eigenvalues with singular values.

A sufficient condition under which a sequence of matrices is strongly clustered is given in the following theorem.

Theorem 1. [21, Theorem 1.2] *Let $\{\mathbf{A}_m\}$ be a sequence of matrices of strictly increasing dimension ($\mathbf{A}_m \in \mathbb{C}^{m\times m}$) with eigenvalues $|\lambda_1^{(m)}| \geq |\lambda_2^{(m)}| \geq \cdots \geq |\lambda_m^{(m)}|$ and singular values $\sigma_1^{(m)} \geq \sigma_2^{(m)} \geq \cdots \geq \sigma_m^{(m)}$. If*

- *there exist a number $N > 0$, independent of m, such that $\sigma_1^{(m)} = \|\mathbf{A}_m\|_2 \leq N$, that is $\{\mathbf{A}_m\}$ is a sequence uniformly bounded;*
- *the sequence $\{\mathbf{A}_m\}$ is strongly clustered at 0 in the singular value sense, that is, following Definition 1, $\forall \epsilon > 0\, \exists\, C = C_\epsilon$ independent of m such that $\#\{j : \sigma_j^{(m)} > \epsilon\} \leq C_\epsilon$, uniformly $\forall m$,*

then $\{\mathbf{A}_m\}$ *is strongly clustered at* 0 *in the eigenvalue sense, that is* $\forall\,\epsilon > 0\ \exists \widehat{C} = \widehat{C}_\epsilon$ *independent of* m *such that* $\#\{j : |\lambda_j^{(m)}| > \epsilon\} \leq \widehat{C}_\epsilon$, *uniformly* $\forall\, m$.

The following lemma gives us a sufficient condition under which a sequence of matrices is strongly clustered at zero.

Lemma 2. *Let* $\{\mathbf{A}_m\}$ *be a sequence of matrices (*$\mathbf{A}_m \in \mathbb{C}^{m\times m}$*), if* $\exists N > 0$, *independent of* m, *such that* $\|\mathbf{A}_m\|_F \leq N$, *where* $\|\cdot\|_F$ *is the Frobenius norm, then the sequence* $\{\mathbf{A}_m\}$ *is strongly clustered at zero in the singular value and eigenvalue sense.*

Proof. For the singular value cluster see [22, Section 4, Corollary 4.1, point 2, with $B_n = 0$]. For the cluster of the eigenvalues, if $\sigma_1^{(m)} \geq \sigma_2^{(m)} \geq \cdots \geq \sigma_m^{(m)}$ are the singular values of $\mathbf{A}_m$ and $\sigma^m = [\sigma_1^{(m)}, \sigma_2^{(m)}, \ldots, \sigma_m^{(m)}]$, we observe that

$$N \geq \|\mathbf{A}_m\|_F = \|\sigma^{(m)}\|_2 \geq \|\sigma^{(m)}\|_\infty = \sigma_1^{(m)} = \|\mathbf{A}_m\|_2,$$

then we are under the hypotheses of Theorem 1 and we can conclude that the sequence $\{\mathbf{A}_m\}$ is strongly clustered at zero in the sense of the eigenvalues.

We conclude this section with our main result on the clustering of sequences of matrices that arise from discretization of integrals of functions in two variables.

Theorem 3. *Let us define* $\mathbf{A}_m = \mathbf{K}_m\mathbf{D}_m$, *where* $\mathbf{K}_m$ *is the sampling matrix of a continuous function* k, $k(\cdot,\cdot) : \Omega \times \Omega \to \mathbb{R}$, $\Omega \subset \mathbb{R}^d$, $d \geq 1$, Ω *closed and bounded, and* $\mathbf{D}_m = \mathrm{diag}(w_0, \ldots, w_{m-1})$ *is the diagonal matrix with the weights of the quadrature formula* w_i. *Then the sequence* $\{\mathbf{A}_m\}$ *is strongly clustered at zero in the singular value and eigenvalue sense.*

Proof. We consider the Frobenius norm of the matrix $\mathbf{A}_m$:

$$\begin{aligned}\|\mathbf{A}_m\|_F^2 &= \sum_{i,j=0}^{m} (\mathbf{A}_m)_{i,j}^2 = \sum_{i,j=0}^{m} k^2(ih, jh) w_j^2 \\ &= \int_{\Omega^2} k^2(x,y)dxdy + \epsilon_m \leq C,\end{aligned}$$

where C is a constant which depends on Ω and the smoothness of the function k, and ϵ_m is the error of the quadrature formula which approaches to 0 as m increases. The application of Lemma 2 concludes the proof.

In conclusion, we observe that the above theorem can be applied to the sampling matrix $\mathbf{H}_m$ in (7). Thus we have proved that our iteration matrix is strongly clustered at zero.

4.2. Bidiagonalization

Formula (7) implies that the option price $\mathbf{W}_0$ can be obtained by performing N times the matrix-vector multiplication $\mathbf{HDv}$, starting from the payoff vector $\mathbf{W}_N$. In order to speed-up the matrix-vector product, we use the spectral properties of the matrix $\mathbf{HD}$ as discussed above. These properties allow us to factorize the iteration matrix into the product of "simpler" matrices. To this end, we consider the following theorem.

Theorem 4. [14, Theorem 1] *Let* $\mathbf{A}$ *be any* $m \times m$ *matrix with complex elements. Then* $\mathbf{A}$ *can be decomposed as* $\mathbf{A} = \mathbf{PJQ}^H$ *where* $\mathbf{P}$ *and* $\mathbf{Q}$ *are unitary matrices and* $\mathbf{J}$ *is an* $m \times m$ *bidiagonal matrix of the form*

$$\mathbf{J} = \begin{pmatrix} \alpha_1 & \beta_1 & 0 & \cdots & 0 \\ 0 & \alpha_2 & \beta_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_{m-1} \\ 0 & \cdots & \cdots & 0 & \alpha_m \end{pmatrix}.$$

Matrices $\mathbf{P}$ and $\mathbf{Q}$ are obtained as products of Householder's elementary (rank 1) matrices, i.e., matrices of the form $\mathbf{I} - 2\mathbf{x}\mathbf{x}^H$, $\mathbf{x} \in \mathbb{C}^m$, $\mathbf{x}^H\mathbf{x} = 1$. Moreover, the matrix-vector product with Householder matrix requires only $2m$ multiplicative and $2m-1$ additive operations. This means that it is not necessary to calculate explicitly the matrices $\mathbf{P}$ and $\mathbf{Q}$, but we can simply store the vectors of the Householder's matrices that generate them. If, for example, $\mathbf{P} = (\mathbf{I}-2\mathbf{x}_1\mathbf{x}_1^H)(\mathbf{I}-2\mathbf{x}_2\mathbf{x}_2^H)\cdots(\mathbf{I}-2\mathbf{x}_m\mathbf{x}_m^H)$, $\mathbf{x}_i \in \mathbb{C}^m$ it is sufficient to store the vectors $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_m$ to get all the informations necessary to calculate the matrix-vector product $\mathbf{Pv}$.

Using Theorem 4 we can factorize the matrix $\mathbf{HD}$ as $\mathbf{PJQ}^H$. In general, this algorithm is very expensive requiring $O(m^3)$ operations (see [15]). However, using the cluster property of the matrix $\mathbf{HD}$, it is possible to see experimentally that the elements $(\alpha_1, \cdots, \alpha_m,)$ on the main diagonal and those $(\beta_1, \cdots, \beta_{m-1},)$ on the upper diagonal of the matrix $\mathbf{J}$, exhibit, in modulus, the behavior shown in Figure 1. Therefore, if we consider to be non-zero only the k_ϵ elements above a certain threshold ϵ, the computation of the matrices $\mathbf{P}$, $\mathbf{J}$ and $\mathbf{Q}$ can stop at the k_ϵ-th iteration with $O(k_\epsilon m^2)$ operations. The value of k_ϵ is closely related to the fixed tolerance ϵ and to the cluster of the matrix. As a rule of thumb, if the number of eigenvalues greater than a tolerance ϵ is r_ϵ, then the number of steps k_ϵ of the algorithm is only slightly larger than r_ϵ.

Given the matrix $\mathbf{HD}$ and using the algorithm of Golub and Reinsch [14], we compute the vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{k_\epsilon}$, $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{k_\epsilon}$ and the elements

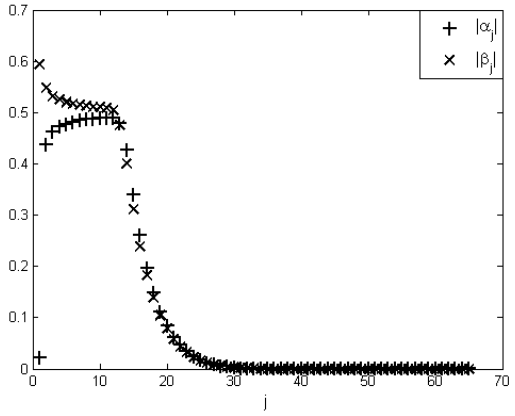


Figure 1: Trend of the absolute values of α_j on the main diagonal and β_j on the upper diagonal of the matrix $\mathbf{J}$ obtained from the discretization of the iteration matrix $\mathbf{HD}$ for a double barrier option in the lognormal model with $N = 252$, $m = 1000$. Parameters setting as in Section 5. In this case $\epsilon = 10^{-8}$, $r_\epsilon = 64$ and $j \leq k_\epsilon = 65$.

$\alpha_1, \ldots, \alpha_{k_\epsilon}$, and $\beta_1, \ldots, \beta_{k_\epsilon - 1}$, so that

$$\begin{aligned} \mathbf{P}_{k_\epsilon} &= (\mathbf{I} - 2\mathbf{x}_1\mathbf{x}_1^H)(\mathbf{I} - 2\mathbf{x}_2\mathbf{x}_2^H)\cdots(\mathbf{I} - 2\mathbf{x}_{k_\epsilon}\mathbf{x}_{k_\epsilon}^H), \\ \mathbf{Q}_{k_\epsilon} &= (\mathbf{I} - 2\mathbf{y}_1\mathbf{y}_1^H)(\mathbf{I} - 2\mathbf{y}_2\mathbf{y}_2^H)\cdots(\mathbf{I} - 2\mathbf{y}_{k_\epsilon}\mathbf{y}_{k_\epsilon}^H), \end{aligned}$$

and

$$\mathbf{P}_{k_\epsilon} \begin{pmatrix} \alpha_1 & \beta_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \beta_{k_\epsilon - 1} & \vdots \\ \vdots & & & \alpha_{k_\epsilon} & 0 \\ 0 & \cdots & \cdots & \cdots & 0 \end{pmatrix} \mathbf{Q}_{k_\epsilon}^H = \mathbf{P}_{k_\epsilon}\mathbf{J}_{k_\epsilon}\mathbf{Q}_{k_\epsilon}^H \cong \mathbf{HD}.$$

Using this factorization, we compute $\mathbf{P}_{k_\epsilon}\mathbf{J}_{k_\epsilon}\mathbf{Q}_{k_\epsilon}^H\mathbf{v}$, instead of $\mathbf{HDv}$, $\mathbf{v} \in \mathbb{C}^m$. In addition, exploiting the Householder structure of matrices $\mathbf{P}_{k_\epsilon}$ and $\mathbf{Q}_{k_\epsilon}$ we can reduce the number of operations in the matrix-vector product from $O(m^2)$ to $O(k_\epsilon m)$.

Summarizing, since in the recursive quadrature equation (7) we have to compute N matrix-vector products, the classical approach requires $O(Nm^2)$ operations against $O(k_\epsilon m^2 + N k_\epsilon m)$ operations of the factorization. Therefore we have a cost reduction if $k_\epsilon < mN/(N + m)$. In the following section we show how it is possible even to achieve a larger time reduction exploiting the structure of the matrix $\mathbf{HD}$.

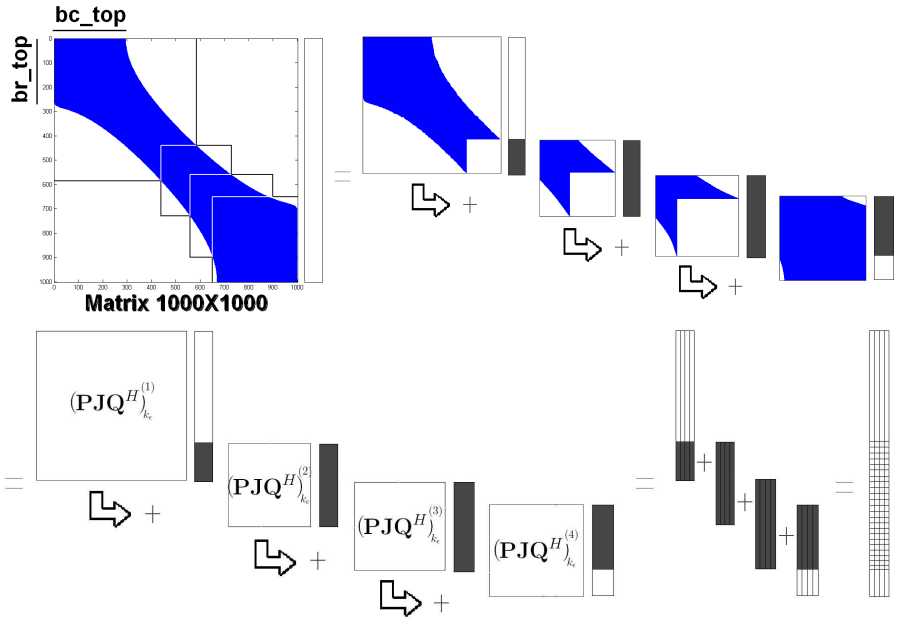


Figure 2: "Breaking into pieces" a banded matrix. The black parts of the vectors overlap in the final sum. $b_{\text{top}} = \max\{bc_{\text{top}}, br_{\text{top}}\}$

4.2.1. "Breaking into pieces" banded matrices

As discussed in Section 4.2, if the number r_ϵ of eigenvalues greater than a fixed tolerance ϵ is not small enough with respect to the size m of the given matrix (usually, in applications, we use matrices of size at most 4000×4000) or to the number of monitoring dates N, then the factorization can be very expensive and we lose the advantage of the matrix-vector multiplication. However, the matrix **HD** is 'nearly' banded, i.e., due to the transition density structure, the significant entries having values greater than a fixed tolerance are confined to a diagonal band. This behavior is illustrated in Figure 2. Thus our the idea is to "break into pieces" the matrix **HD** as shown in the same figure and to factorize separately each piece via the bidiagonalization algorithm.

More precisely, each sub piece of the sampling matrix **HD** inherits its cluster property, so that it is convenient to perform the factorization on matrices of a smaller size. This makes the procedure faster (especially when the size of the original matrix is very large). We emphasize the importance of the bandwidth of the matrix. It is clear that a too large band leads to consider "pieces of matrix" having a large size, so that we do not achieve any benefit from the suggested breakdown procedure. In this case, the standard quadrature will

remain the preferred approach. However, if a band structure is detected, once each piece is factorized we can calculate the matrix-vector product as shown in Figure 2: the original matrix (top left corner of Figure 2) is "broken" into smaller pieces (top right corner of Figure 2). Then each piece is factorized using the Golub-Reinsch algorithm (bottom left corner of Figure 2) and the matrix-vector product is computed exploiting the Householder or bidiagonal structure of the matrices involved. Finally the resulting vectors are "summed up" taking into account the overlapping parts.

5. Numerical Results

In this section we validate with numerical experiments the theoretical results of previous sections. The setup is quite general but for aim of clarity we consider the CEV process. This dynamic is indeed quite interesting allowing for very different transition densities and implied volatility shapes. For this reason, we describe it in some detail in Section 5.1. Then in Section 5.2 we show that the matrix **HD** obtained by the discretization of the CEV transition density exhibits eigenvalues strongly clustered at zero: if we increase the size (number of quadrature points) of the matrix, the number of eigenvalues greater than a given tolerance remains constant. Thus the CEV process admits the cluster property that allows us to improve the performance of the recursive approach. Finally, in Section 5.3 we apply our algorithm (hereafter denominated BP algorithm) to "break into pieces" the matrix **HD** and we apply to price exotic derivatives. This algorithm is detailed with a pseudo-code in Appendix A.

For the numerical discretization of (5) we opt for a Gauss-Legendre quadrature [20]. Numerical experiments not reported here have shown that the cluster of eigenvalues of the iteration matrix is independent of the adopted quadrature formulas, but the bandwidth of the matrix is larger for Gaussian quadrature respect to Newton-Cotes ones. However the results are similar, in terms of computational efficiency, for both classes of quadrature formulas.

All calculations were performed using Matlab R2008a on a PC Intel Core2 Quad 2.40 GHz with 3.24 GB RAM and Windows XP operating system.

5.1. The CEV Process

Even if our algorithm is quite general, from now on, we assume that the underlying asset evolves according to a CEV process [7], i.e.,

$$dS(t) = rS(t)dt + \sigma S^{\beta+1}(t)dW(t), \quad S(0) = S_0, \tag{9}$$

and thus the transition probability density is given by

$$p\left(S,\xi;\Delta\right) := e^{-r\Delta}\; p_0\left(S,\; e^{-r\Delta}\xi;\; \frac{1}{2r\beta}\left(e^{2r\beta\Delta}-1\right)\right),$$

with

$$p_0\left(S,\ \xi;\ \Delta\right) = \frac{\xi^{-2\beta-\frac{3}{2}} S^{\frac{1}{2}}}{\sigma^2|\beta|\Delta}\ e^{-\frac{S^{-2\beta}+\xi^{-2\beta}}{2\sigma^2\beta^2\Delta}}\ I_{\frac{1}{2|\beta|}}\left(\frac{S^{-\beta}\xi^{-\beta}}{\sigma^2\beta^2\Delta}\right),$$

where I_ν is the modified Bessel function of the first kind of order ν. In particular, when $\beta = 0$ we have the classical geometric Brownian process (GBM), when $\beta = -1$ we have an arithmetic Brownian motion (ABM), while when $\beta = -0.5$ the Cox-Ingersoll-Ross square-root process (SR) is obtained. For details see also [7, 9, 12]. In Figure 3 we plot the density function for different values of the leverage parameter β (left panel) and the corresponding implied volatility curve (right panel). In particular, large negative values of β generate a skewed to the left density function and a very steep implied volatility curve, as often observed in the market. The CEV process, consistently with empirical studies, allows for the volatility to depend on the price level and in addition the two are negatively correlated (leverage effect); moreover, the model is able to generate the smirk effect often observed in the market implied volatility curve. See for example [3, 6]. Unfortunately, the transition density of the CEV process is not of convolution type, thus a fast computation of the recursion via the FFT is not feasible. For these reasons, the CEV dynamic turns out to be an interesting case to test our pricing procedure.
Unless otherwise specified, we consider the same parameter setting as in [9]: the initial asset price is $S_0 = 100$, the risk-free interest rate is 10% per annum ($r = 0.1$), the volatility is $\sigma = 0.25/S_0^\beta$. Moreover, we assume that the asset pays no dividends ($q = 0$), and all options have six months to expiration ($T = 0.5$). If necessary, we truncate the integration interval as stated in Section 2.1 with a 10^{-8} tolerance.

5.2. Cluster of the HD matrix

Table 1 provides the number of eigenvalues greater, in absolute value, than $\epsilon = 10^{-11}$ and the bandwidth size b_{top} of the matrix **HD**, see Figure 2. The bandwidth size is fixed setting to zero the elements smaller, in absolute value, than 10^{-9}. The leverage parameter β in the CEV model is set equal to -0.5 (results for different values of β are reported in Appendix B, Tables B.12-B.15). We stress that this table refers to pricing problems characterized by a different iteration matrix. We notice that:

- a strong cluster at zero always occurs. For example, let us consider down-and-out options in Table 1, given a number of monitoring dates $N = 52$. The number of significative eigenvalues (greater than ϵ) is 107, independently of the matrix dimension m;

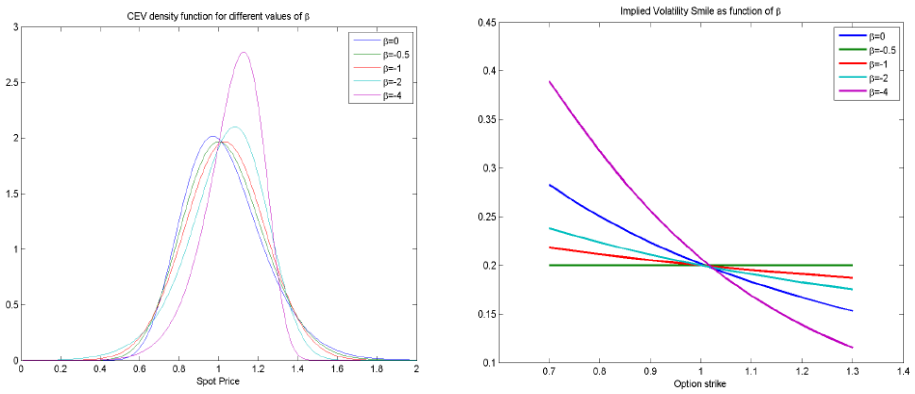


Figure 3: Density function (left) and implied volatility (right) of the CEV model for different values of β.

- the cluster increases less than linearly with respect to the number of monitoring dates N. Thus our algorithm will achieve a large time reduction when N is large. In fact, we see in Table 1 that, given m, as the number of monitoring dates increases, the same happens to the number of eigenvalues greater than ϵ, but the ratio r_ϵ/N decreases;

- the presence of barriers strongly improves the cluster. This is evident if we compare the double barrier case in Table 1 to other contracts. In particular this suggests a relative better performance of the algorithm in pricing this kind of exotics;

- changing the value of the leverage parameter β in the CEV density does not affect the cluster. This is shown in Tables B.12-B.15 in Appendix B.

We can also make some additional considerations on the bandwidth of the matrix:

- we always have banded matrices; for example in Table 1 for European, lookback, Asian or step options with $N = 52$ and $m = 4000$, the bandwidth size is $b_{\text{top}} = 493$. However, the presence of barriers increases b_{top}. Indeed barriers cut the tails of the density so that we have to sample a function that does not approach zero on the frontier of the domain. In general, the ratio b_{top}/m remains constant as m varies.

- the bandwidth size decreases as we increase the number N of monitoring dates; this is, for example, confirmed for all contracts in Table 1 when $m = 4000$ and we let N to vary from 52 to 1008;

		r_ϵ				bandwidth=b_{top}			
Contract	N	m				m			
		1000	2000	3000	4000	1000	2000	3000	4000
European	52	190	190	190	190	126	249	372	493
Lookback	104	266	267	267	267	105	208	310	411
Asian	252	411	412	412	412	83	165	247	328
Step	504	576	580	581	581	70	139	207	275
	1008	774	819	819	832	59	116	174	231
Down-and-out	52	107	107	107	107	209	414	617	818
	104	149	149	149	149	174	345	514	683
	252	230	230	230	229	139	275	410	544
	504	322	322	323	322	116	231	344	456
	1008	455	455	455	454	98	194	289	384
Up-and-out	52	115	115	115	115	183	362	540	716
	104	161	161	161	161	152	301	449	596
	252	247	247	247	247	121	239	357	474
	504	348	348	348	348	101	201	299	397
	1008	489	490	490	490	85	168	251	333
Double barrier	52	32	32	32	32	471	929	1382	1831
	104	44	44	43	43	383	757	1126	1493
	252	65	65	65	65	300	592	882	1170
	504	90	90	90	90	249	493	734	974
	1008	125	125	125	125	208	412	614	814

Table 1: Number r_ϵ of eigenvalues of **HD** greater than $\epsilon = 10^{-11}$. Legend: m is the matrix dimension, N is the number of monitoring dates, $\beta = -0.5$ is the leverage parameter in (9) and b_{top} is the bandwidth size of the matrix **HD**. Contracts are grouped according to the iteration matrix **HD**.

Since the performance of the algorithm is optimized when the cluster size r_ϵ and the bandwidth size b_{top} are both small, the above remarks suggest that this happens in all cases and the greatest benefit occurs as we increase N.

	N=252	N=504	N=1008
European	1.1604	1.3270	1.4151
Barrier Down-and-out	1.1045	1.4104	1.7289
Barrier Up-and-out	1.1784	1.4872	1.7964
Double barrier	0.7263	0.9809	1.9535
Bermudan	1.1634	1.3476	1.4189
Bermudan Down-and-out	1.1102	1.4176	1.7126
Bermudan Up-and-out	1.1891	1.5065	1.7883
Bermudan Double barrier	0.7315	0.9867	1.9412
Lookback	2.3617	2.1688	1.9565
Lookback ($\omega = 4$)	3.0116	2.8686	2.5773
Asian	1.8654	1.7169	1.5609
Asian ($\omega = 4$)	1.9707	1.9769	1.9624
Step	4.3515	3.6113	3.1951

Table 2: Speed-up values for different contracts with $\beta = -0.5$ and $m = 4000$. The initial asset price is $S_0 = 100$, the volatility is $\sigma = 0.25/S_0^\beta$, the risk-free interest rate is 10% per annum ($r = 0.1$), the asset pays no dividends ($q = 0$), and all options have six months to expiration ($T = 0.5$). The strike price E is equal to 105. Additional payoff's parameters for step options are $\rho = 0.5$ and $A = [90, 110]$. ω is the parameter in Remark 1.

In Table 2 we show the speed-up for different contracts. Here the speed-up is defined as the ratio between the CPU time for the classical recursive (Rec.) algorithm and the one for our pricing procedure (Rec.+BP). As expected, the speed-up is always greater than 1. Double barrier options are the exception when the number of monitoring dates is small, due to a too large bandwidth.

5.3. Pricing options

In this section we validate our pricing procedure comparing it to standard numerical quadrature and to Monte Carlo simulation. This has been implemented with an Euler discretization scheme with 300 steps between two consecutive monitoring dates, and 1.000.000 runs. In general Monte Carlo simulations applied to the CEV process achieve a two digits accuracy, but with a CPU time that turns to be higher than the recursive quadrature of a factor that varies from 4 to 10.[2] Extended numerical results are reported in Appendix B.

5.3.1. Barrier and Bermudan Options

In Tables 4 and 5 we price European and barrier call options. Analytical formulas for European options are available in terms of the non-central chi-square distribution, see [23]. Prices of continuously monitored barrier options, i.e., $N = \infty$, are given in [9] and reported here in Table 3.

	$\beta = 0$	$\beta = -0.5$	$\beta = -1$
European	7.0995	7.0170	6.9403
Down-and-out	6.3722	6.2554	6.1438
Up-and-out	0.6711	0.7734	0.8904
Double barrier	0.4418	0.5126	0.5945

Table 3: Prices in [9, Table 1].

β	m	Prices		CPU Times (sec.)	
		Rec.	Rec.+BP	Rec.	Rec.+BP
0	2000	7.099596	7.099596	1.01	4.60
	4000	7.099571	7.099571	3.90	18.20
-0.5	2000	7.017063	7.017063	6.09	10.16
	4000	7.016999	7.016999	23.63	39.66
-1	2000	6.940388	6.940388	8.09	12.96
	4000	6.940318	6.940318	31.40	50.87

Table 4: European call: m is the matrix dimension and β is the leverage parameter in (9). Parameters as in Table 2.

[2] We also considered an exact Monte Carlo simulation by sampling from the known transition cumulative density function, but the procedure turns out to be too time consuming and of no practical relevance.

Since European options are path-independent contracts, their pricing requires a single recursion. For this reason, in Table 4 we set $N = 1$ and the BP algorithm is not at all convenient because the factorization is too costly with respect to a single matrix-vector multiplication. However the algorithm has the same accuracy as the analytical formula: our price estimates agree with those of the first row of Table 3.

Numerical results for barrier options with $\beta = -0.5$ are given in Table 5. Prices for different values of β and for up-and-out options are reported in Appendix B, Table B.16. We notice that:

- prices computed with the BP algorithm agree with the ones from the pure recursion up to five decimal digits;
- as expected, the BP algorithm performs better as we increase the number of monitoring dates N, since the factorization has to be performed only once, and, at the same time, the bandwidth decreases (see Table 1 - we recall that our algorithm performs well if the bandwidth size is not too large). In fact, from Table 5 we notice the benefits of the BP factorization for $N = 252$ or greater;
- the algorithm works better as we increase the number of quadrature nodes m, since the cluster size r_ϵ does not increase varying m, while the computational cost of the matrix-vector multiplication increases;
- for double barrier options we observe that, increasing N, we have a trade-off between the cluster size (it improves) and the bandwidth size (it becomes larger). On this point we can make two remarks:
  1. In general, our algorithm improves the standard recursion for N larger than 252. Additional numerical tests have shown that the algorithm applied to the double barrier case can achieve a speed-up up to 3.6 when $N = 10000$.
  2. Numerical results in Table 2 show that the BP algorithm performs better for single barrier respect to double barrier options if N is lower than 1008.
- concerning the convergence of the discrete monitoring price to the continuous monitoring case, we notice a slow convergence from above of prices in Table 5 to the ones in Table 3. This justifies the pricing of discretely monitored options. For example, when $\beta = -0.5$ and $N = 10000$, a single barrier option with discrete monitoring is worth 6.2756, whilst the continuous version is 6.2554.

Experiments not reported here show similar performances for Bermudan options. The results are not affected by the presence or absence of dividends.

		Down-and-out call				Double barrier call			
N	m	Prices		CPU (sec.)		Prices		CPU (sec.)	
		Rec.	Rec. +BP	Rec.	Rec. +BP	Rec.	Rec. +BP	Rec.	Rec. +BP
52	2000	6.497277	6.497277	4.47	6.03	0.771025	0.771025	4.16	8.70
52	4000	6.497278	6.497278	17.13	23.74	0.771024	0.771024	16.12	34.37
104	2000	6.434699	6.434699	4.87	5.51	0.694140	0.694140	4.48	8.41
104	4000	6.434700	6.434700	18.73	22.22	0.694140	0.694140	17.21	32.49
252	2000	6.375374	6.375374	6.03	5.70	0.628248	0.628248	5.39	7.61
252	4000	6.375375	6.375375	23.24	21.04	0.628248	0.628248	20.68	28.47
504	2000	6.342071	6.342072	7.91	7.06	0.593922	0.593922	6.94	7.46
504	4000	6.342072	6.342073	30.65	21.73	0.593922	0.593922	26.77	27.29
1008	2000	6.317620	6.317621	11.63	11.24	0.569846	0.569846	13.99	8.47
1008	4000	6.317621	6.317623	45.04	26.05	0.569846	0.569846	54.64	27.97
10000	2000	6.275649	6.275652	72.95	122.85	0.530602	0.530603	85.20	49.35
10000	4000	6.275651	6.275652	282.11	154.93	0.530602	0.530604	331.26	91.33

Table 5: Down-and-out and double barrier call: $\beta = -0.5$ is the parameter in (9), m is the matrix dimension and N is the number of monitoring dates. Parameters as in Table 2.

5.3.2. Lookback Options

Pricing lookback options is in general more expensive than pricing barrier options, because we have to keep trace of an additional state variable, the running minimum J. Thus, we expect an increase in the CPU time with respect to barrier and Bermudan option. However, since the matrix factorization is independent on the J-grid nodes, we still expect an improvement with respect to the standard recursion. Results reported in Table 6 confirm this. In addition the two recursive algorithms show comparable accuracy. Table 7 provides, as benchmark, confidence intervals computed by the Monte Carlo algorithm. An additional speed-up can be obtained considering less nodes on the J-grid, up to ten times if we reduce by a factor of four the nodes on the J-grid ($\omega = 4$ in the mentioned table), maintaining a two decimal digits accuracy.

5.3.3. Asian Options

Asian options (see Section 2.4) share with lookback options the presence of an additional state variable. Given that the iteration matrix is the same for the two contracts, we expect a similar performance of the algorithm in the Asian as in the lookback case. Results are given in Table 8, and comments given in Section 5.3.2 still apply. For example, if we set $\omega = 4$, we reduce the CPU time by a factor of seven, still maintaining a two decimal digits accuracy. Reported option prices always fall into the Monte Carlo confidence intervals.

Finally, in Table 10 we consider the log-normal process ($\beta = 0$). In this case, indeed, it is more efficient to use a FFT approach, as in [5], or a randomization technique, as in [11]. We use the results in [5, Table 7] as benchmark. In Table 10 we also analyze the effect on the price accuracy of

N	m	Prices				CPU Times (sec.)			
		Rec. $\omega = 1$	Rec. +BP $\omega = 1$	Rec. +BP $\omega = 2$	Rec. +BP $\omega = 4$	Rec. $\omega = 1$	Rec. +BP $\omega = 1$	Rec. +BP $\omega = 2$	Rec. +BP $\omega = 4$
52	2000	14.545701	14.545701	14.544358	14.553307	592	323	140	64
52	4000	14.542978	14.542978	14.542853	14.546845	4566	1422	703	332
104	2000	14.887939	14.887940	14.886958	14.893453	1069	629	270	124
104	4000	14.886366	14.886366	14.886282	14.889097	8084	2897	1381	618
252	2000	15.191305	15.191306	15.190640	15.194984	2193	1361	598	254
252	4000	15.190981	15.190982	15.190932	15.192716	16567	7015	3212	1376
504	2000	15.353628	15.353629	15.353129	15.356335	3953	2684	1194	534
504	4000	15.354248	15.354250	15.354219	15.355452	29570	13634	5969	2573
1008	2000	15.469194	15.469194	15.468811	15.471220	6886	5117	2043	986
1008	4000	15.470853	15.470857	15.470839	15.471678	51971	26563	11362	5025

Table 6: Fixed-strike lookback put: $\beta = -0.5$ is the parameter in (9), m is the matrix dimension and N is the number of monitoring dates. Parameters as in Table 2. ω is the parameter in Remark 1.

N	Confidence Interval	CPU Times (sec.)
52	14.5242 - 14.5576	2523
104	14.8738 - 14.9073	5014
252	15.1781 - 15.2116	12104
504	15.3483 - 15.3818	24180
1008	15.4511 - 15.4846	48331

Table 7: Monte Carlo values for fixed-strike lookback put options with 1.000.000 iterations, parameters as in Table 6.

N	m	Prices				CPU Times (sec.)			
		Rec. $\omega = 1$	Rec. +BP $\omega = 1$	Rec. +BP $\omega = 2$	Rec. +BP $\omega = 4$	Rec. $\omega = 1$	Rec. +BP $\omega = 1$	Rec. +BP $\omega = 2$	Rec. +BP $\omega = 4$
52	2000	2.919996	2.919996	2.920009	2.919707	843	573	284	157
52	4000	2.920010	2.920010	2.920007	2.920017	5615	2377	1186	673
104	2000	2.928446	2.928446	2.928341	2.926760	1642	1128	543	307
104	4000	2.928465	2.928465	2.928459	2.928347	10072	4814	2379	1304
252	2000	2.933353	2.933353	2.932724	2.929454	3592	2591	1370	725
252	4000	2.933498	2.933499	2.933402	2.932702	21221	11376	5755	3140
504	2000	2.934878	2.934878	2.933790	2.929121	6448	5137	2796	1400
504	4000	2.935238	2.935238	2.934972	2.933777	38909	22662	11140	5495
1008	2000	2.935444	2.935444	2.933788	2.927539	12413	9834	5488	2992
1008	4000	2.936030	2.936031	2.935583	2.933823	71120	45563	22044	10207

Table 8: Fixed-strike Asian call: $\beta = -0.5$ is the parameter in (9), m is the matrix dimension and N is the number of monitoring dates. Parameters as in Table 2. ω is the parameter in Remark 1.

the truncation of the integration interval (see Section 2.1). The Rec.+BP algorithm always achieves a three to five decimal digits accuracy depending on the tolerance level when we truncate the domain. Significant reduction in the CPU time, without loss of accuracy, can be achieved with a sparser grid ($\omega = 2$ and $\omega = 4$) on the running average.

N	Confidence Interval	CPU Times (sec.)
52	2.9151 - 2.9356	2523
104	2.9196 - 2.9401	5014
252	2.9272 - 2.9477	12104
504	2.9283 - 2.9487	24180
1008	2.9264 - 2.9468	48331

Table 9: Monte Carlo values for fixed-strike Asian call options with 1.000.000 iterations, parameters as in Table 8.

		σ					
ω	m	Tol 10^{-8}			Tol 10^{-10}		
		0.1	0.3	0.5	0.1	0.3	0.5
1	1000	11.58113	13.66975	17.19193	11.58113	13.66991	17.19255
1	2000	11.58113	13.66977	17.19196	11.58113	13.66980	17.19236
1	4000	11.58113	13.66977	17.19192	11.58113	13.66982	17.19240
2	1000	11.58118	13.67068	17.19441	11.58120	13.67232	17.19750
2	2000	11.58113	13.66974	17.19191	11.58113	13.66982	17.19272
2	4000	11.58113	13.66977	17.19191	11.58113	13.66982	17.19239
4	1000	11.58175	13.67434	17.20552	11.58184	13.68018	17.20388
4	2000	11.58118	13.67081	17.19409	11.58121	13.67196	17.19835
4	4000	11.58113	13.66973	17.19192	11.58113	13.66981	17.19270

Table 10: Fixed-strike Asian call: A comparison between the Rec.+BP algorithm and [5, Table 7]: Gaussian case ($\beta = 0$) and strike price $E = 90$. Benchmark price: 11.58113 ($\sigma = 0.1$), 13.66981 ($\sigma = 0.3$) and 17.19239 ($\sigma = 0.5$).

5.3.4. Step Options

Numerical results given in Table 11 show that our algorithm applied to step options achieves the same accuracy as the direct recursive procedure: they agree up to the sixth digit, but with a strong reduction in the computational time. Thus, also for this kind of contracts the Rec.+BP algorithm is more efficient than the plain recursion and Monte Carlo simulation.

N	m	Prices		CPU Times (sec.)		Monte Carlo values	
		Rec.	Rec.+BP	Rec.	Rec.+BP	Confidence Interval	CPU Times (sec.)
52	2000	5.580878	5.580878	13.08	7.46	5.5772-5.6133	2446.38
52	4000	5.581670	5.581670	50.64	23.71		
104	2000	5.564554	5.564554	35.21	13.98	5.5473-5.5832	4881.28
104	4000	5.565345	5.565345	134.63	35.45		
252	2000	5.554981	5.554981	154.47	60.41	5.5414-5.5772	11811.86
252	4000	5.555772	5.555773	589.53	135.57		
504	2000	5.551620	5.551620	533.47	241.04	5.5367-5.5725	23616.72
504	4000	5.552411	5.552412	2003.21	554.70		
1008	2000	5.549940	5.549940	1823.83	1044.87	5.5276-5.5634	47231.25
1008	4000	5.550731	5.550732	6931.57	2169.42		

Table 11: Step call: m is the matrix dimension, N is the number of monitoring dates. Parameters as in Table 2.

References

[1] Andricopoulos, A. D., Widdicks, M., Duck, P. W., & Newton, D. P. (2003). Universal option valuation using quadrature methods. *Journal of Financial Economics*, 67, 447-471.

[2] Atkinson, K. E. (2009). *The Numerical Solution of Integral Equations of the Second Kind*. Cambridge University Press.

[3] Boyle, P. P., & Tian, Y. S. (1999). Pricing lookback and barrier options under the CEV Process. *The Journal of Financial and Quantitative Analysis*, 34-2, 241-264.

[4] Broadie, M., Glasserman, P., & Kou., S. G. (1997). A continuity correction for the discrete barrier options. *Mathematical Finance*, 7, 325-349.

[5] Černý, A., & Kyriakou, I. (2011). An improved convolution algorithm for discretely sampled Asian options. *Quantitative Finance*, 11-3, 381-389.

[6] Costabile, M. (2006). On pricing lookback options under the CEV process. *Decisions in Economics and Finance*, 29, 139-153.

[7] Cox, J. (1996). Notes on option pricing I: Constant elasticity of variance diffusions. *Journal of Portfolio Management*, 22, 15-17.

[8] Cox, J., & Ross, S. (1976). The valuation of options for alternative stochastic processes. *Journal of Financial Economics*, 3, 145-166.

[9] Davydov, D., & Linetsky, V. (2001). Pricing and hedging path-dependent options under the CEV process. *Management Science*, 47-7, 949-965.

[10] Fusai, G., Marazzina, D., Marena, M., & Ng, M. (2012). Z-transform and preconditioning techniques for option pricing. *Quantitative Finance*, to appear.

[11] Fusai, G., Marazzina, D., & Marena, M. (2011). Pricing discretely monitored Asian options by maturity randomization. *SIAM Journal on Financial Mathematics*, 2, 383-403.

[12] Fusai, G., & Recchioni, M. C. (2007). Analysis of quadrature methods for pricing discrete barrier options. *Journal of Economic Dynamics and Control*, 31, 826-860.

[13] Fusai, G., Abrahams, D., & Sgarra, C. (2006). An exact analytical solution of discrete barrier options. *Finance and Stochastics*, 10, 1-26.

[14] Golub, G. H., & Kahan, W. (1965). Calculating the singular values and pseudo-inverse of a matrix. *SIAM Journal on Numerical Analysis* 2, 205-224.

[15] Golub, G. H., & Van Loan, C. F. (1996). *Matrix Computations*. (3rd ed.). The Johns Hopkins University Press, Baltimore, Maryland.

[16] Linetsky, V. (1999). Step Options. *Mathematical Finance*, 9, 55-96.

[17] Lord, R., Fang, F., Bervoets, F., & Oosterlee, C. W. (2008). A fast and accurate FFT-based method for pricing early-exercise options under Levy processes. *SIAM Journal on Scientific Computing*, 30, 1678-1705.

[18] Nunes, J. P. V. (2009). Pricing American options under the Constant Elasticity of Variance model and subject to bankruptcy. *Journal of Financial and Quantitative Analysis*, 44, 1231-1263.

[19] Peng, B. (2006). Pricing geometric Asian option under the CEV process. *International Economic Journal*, 20-4, 515-522.

[20] Quarteroni, A., Sacco, R., & Saleri, F. (2000). *Numerical Mathematics*. Springer-Verlag New York.

[21] Serra Capizzano, S., Bertaccini, D., & Golub, G. H. (2005). How to deduce a proper eigenvalue cluster from a proper singular value cluster in the nonnormal case. *SIAM Journal on Matrix Analysis and Applications*, 27-1, 82-86.

[22] Serra Capizzano, S. (2001). Spectral behavior of matrix sequences and discretized boundary value problems. *Linear Algebra and its Applications*, 337, 37-78.

[23] Schroder, M. (1989). Computing the Costant Elasticity of Variance option pricing formula. *The Journal of Finance*, 44-1, 211-219.

[24] Tyrtyshnikov, E. E. (1996). A unifying approach to some old and new theorems on distribution and clustering. *Linear Algebra and its Applications*, 232, 1-43.

Appendix A. Algorithm to "break into pieces" a matrix

In order to "break into pieces" the matrix **HD** as explained in Section 4.2.1, we proceed, recursively, as follows: we denote by

`fact(A)` the factorization (bidiagonalization) procedure of a matrix `A`;

`bandRtop(A)` the number of elements larger than a tolerance ϵ in the first row of `A`;

`bandCtop(A)` the number of elements larger than a tolerance ϵ in the first column of `A`;

`bandRend(A)` the number of elements larger than a tolerance ϵ in the last row of `A`;

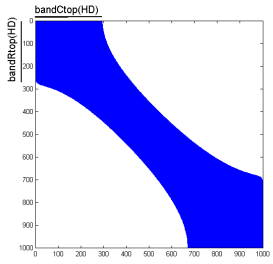
`bandCend(A)` the number of elements larger than a tolerance ϵ in the last column of `A`;

`dim(A)` the number of rows (columns) of a square matrix `A`.

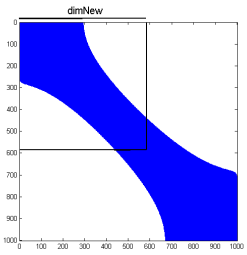
The algorithm, described in Figure A.4, is the following.

```
function = Cut(HD,epsilon)
while(1)
  compute bandRtop(HD);
  compute bandCtop(HD);    %(see Figure A.4A)
  put dimNew=2*max(bandCtop(HD),bandRtop(HD)); %(see Figure A.4B)
  If dimNew >= dim(HD)
    fact(HD);
    BREAK;    %(see Figure A.4C)
  else
    put HDnew=HD(1:dimNew,1:dimNew);    %(see Figure A.4D-E)
    compute bandRend(HDnew);
    compute bandCend(HDnew);    %(see Figure A.4F)
    put maxRCend=max(bandaRend(HDnew),bandaCend(HDnew));
    put HDnew(dimNew-maxRCend:end,dimNew-maxRCend:end)=0; %(see Figure A.4G)
    fact(HDnew);
    put HD=HD(dimNew-maxRCend:end,dimNew-maxRCend:end); %(see Figure A.4H-I)
  end
end
```

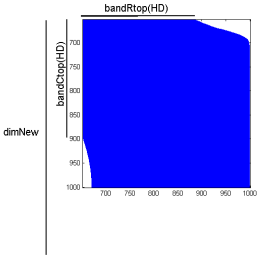
The value 2 in the fifth line of code has been experimentally verified to be the best one.



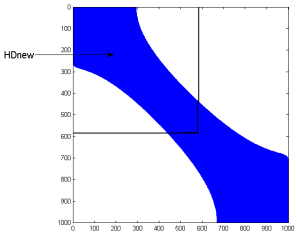
A: initial matrix **HD**



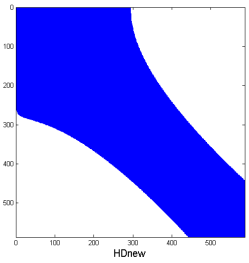
B



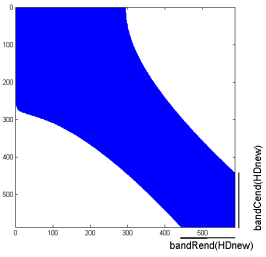
C: last submatrix, see Figure 2



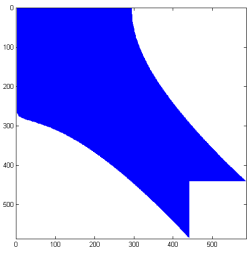
D: first submatrix considered



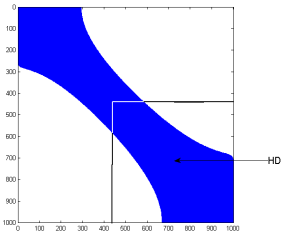
E: enlargement of Figure D



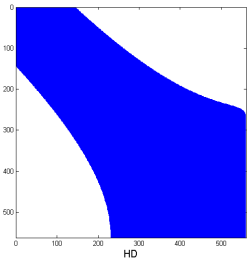
F



G



H: new matrix **HD**



I: enlargement of Figure H

Figure A.4: "Breaking into pieces" a banded matrix.

Appendix B. Supplementary Material

		r_ϵ				bandwidth=b_{top}			
β	N	m				m			
		1000	2000	3000	4000	1000	2000	3000	4000
0	52	192	192	192	192	99	195	291	387
	104	269	270	270	270	82	163	243	322
	252	407	417	417	417	65	130	193	256
	504	538	586	587	587	55	109	162	215
	1008	681	811	828	828	46	91	136	181
-0.5	52	190	190	190	190	126	249	372	493
	104	266	267	267	267	105	208	310	411
	252	411	412	412	412	83	165	247	328
	504	576	580	581	581	70	139	207	275
	1008	774	819	819	832	59	116	174	231
-1	52	190	190	190	190	172	341	509	675
	104	267	267	267	267	145	287	427	567
	252	412	412	412	412	116	230	343	455
	504	580	581	581	581	98	193	288	383
	1008	818	819	819	819	82	163	243	323
-2	52	152	152	152	152	135	268	399	529
	104	214	214	214	214	114	225	336	446
	252	332	332	332	332	91	181	270	358
	504	467	468	469	469	77	152	227	301
	1008	621	661	662	662	65	128	191	254

Table B.12: European, Asian, Lookback, Standard Bermudan and Step options have the same iteration matrix **HD**. In the Table we give the number r_ϵ of eigenvalues greater than $\epsilon = 10^{-11}$ and the bandwidth size. Legend: m is the matrix dimension, N is the number of monitoring dates, β is the parameter in (9) and b_{top} is defined as in Figure 2.

		r_ϵ				bandwidth=b_{top}			
β	N	m				m			
		1000	2000	3000	4000	1000	2000	3000	4000
0	52	105	105	105	105	182	359	535	710
	104	146	146	146	146	151	298	445	590
	252	225	225	225	225	120	237	353	469
	504	316	316	316	316	100	198	296	393
	1008	444	445	445	445	84	167	248	330
-0.5	52	107	107	107	107	209	414	617	818
	104	149	149	149	149	174	345	514	683
	252	230	230	230	229	139	275	410	544
	504	322	322	323	322	116	231	344	456
	1008	455	455	455	454	98	194	289	384
-1	52	111	111	111	111	229	454	676	897
	104	156	155	155	155	192	380	566	751
	252	239	239	239	239	153	303	452	600
	504	336	336	336	336	129	255	380	505
	1008	473	473	473	473	108	215	320	425
-2	52	117	117	117	117	192	379	565	749
	104	164	164	164	164	161	318	475	630
	252	252	252	252	252	129	255	380	505
	504	355	355	355	355	108	215	320	425
	1008	499	500	500	500	91	181	270	358

Table B.13: Down-and-out European and Bermudan call. The notation is as in Table B.12.

		r_ϵ				bandwidth=b_{top}			
β	N	m				m			
		1000	2000	3000	4000	1000	2000	3000	4000
0	52	119	119	119	119	162	321	478	635
	104	166	166	166	166	135	267	398	528
	252	256	256	256	256	107	212	316	420
	504	359	360	360	360	90	178	265	352
	1008	505	506	507	507	75	149	222	295
-0.5	52	115	115	115	115	183	362	540	716
	104	161	161	161	161	152	301	449	596
	252	247	247	247	247	121	239	357	474
	504	348	348	348	348	101	201	299	397
	1008	489	490	490	490	85	168	251	333
-1	52	111	111	111	111	229	452	674	894
	104	156	155	155	155	191	379	565	749
	252	239	239	239	239	153	303	452	600
	504	336	336	336	336	129	255	380	504
	1008	473	473	473	473	108	214	320	424
-2	52	69	69	69	69	208	411	612	812
	104	97	97	97	97	173	343	511	678
	252	149	149	149	149	138	273	407	540
	504	210	210	210	210	116	229	342	453
	1008	295	295	295	295	97	193	287	381

Table B.14: Up-and-out call European and Bermudan. The notation is as in Table B.12.

		r_ϵ				bandwidth=b_{top}			
β	N	m				m			
		1000	2000	3000	4000	1000	2000	3000	4000
0	52	32	32	32	32	465	917	1364	1807
	104	43	43	43	43	377	744	1107	1468
	252	64	64	64	64	294	581	865	1147
	504	88	88	88	88	244	482	718	953
	1008	122	122	122	122	203	402	600	796
-0.5	52	32	32	32	32	471	929	1382	1831
	104	44	44	43	43	383	757	1126	1493
	252	65	65	65	65	300	592	882	1170
	504	90	90	90	90	249	493	734	974
	1008	125	125	125	125	208	412	614	814
-1	52	33	33	33	33	477	940	1398	1853
	104	44	44	44	44	389	769	1145	1518
	252	66	66	66	66	306	604	900	1194
	504	92	91	91	91	255	504	750	996
	1008	127	127	127	127	213	421	628	833
-2	52	34	34	34	34	436	859	1278	1694
	104	46	46	46	46	356	703	1047	1388
	252	69	69	69	69	280	553	823	1092
	504	96	96	96	96	233	461	686	911
	1008	133	133	133	133	195	385	574	762

Table B.15: Double barrier European and Bermudan call. The notation is as in Table B.12.

			Down-and-out call				Up-and-out call				Double barrier call			
β	N	m	Prices		CPU Times (sec.)		Prices		CPU Times (sec.)		Prices		CPU Times (sec.)	
			Rec.	Rec.+BP	Rec.	Rec.+BP	Rec.	Rec.+BP	Rec.	Rec.+BP	Rec.	Rec.+BP	Rec.	Rec.+BP
0	52	2000	6.604402	6.604402	1.54	3.04	0.884296	0.884296	1.60	2.80	0.681155	0.681155	1.28	5.61
	52	4000	6.604392	6.604392	6.12	12.57	0.884296	0.884296	6.37	11.44	0.681155	0.681155	5.09	22.08
	104	2000	6.544377	6.544377	1.95	2.60	0.822557	0.822557	2.05	2.35	0.609453	0.609453	1.59	5.42
	104	4000	6.544368	6.544368	7.68	10.69	0.822557	0.822557	8.11	9.83	0.609452	0.609452	6.32	20.95
	252	2000	6.487436	6.487437	3.10	2.74	0.768701	0.768701	3.31	2.80	0.548320	0.548320	2.48	4.57
	252	4000	6.487428	6.487428	12.21	9.96	0.768701	0.768701	13.02	9.20	0.548320	0.548320	9.81	17.43
	504	2000	6.455458	6.455458	5.05	4.18	0.740206	0.740206	5.36	4.41	0.516598	0.516598	4.01	4.53
	504	4000	6.455450	6.455450	19.69	10.88	0.740206	0.740206	20.96	10.51	0.516598	0.516598	15.82	16.24
	1008	2000	6.431973	6.431974	8.75	8.38	0.720007	0.720008	9.33	9.12	0.494401	0.494401	11.11	5.42
	1008	4000	6.431965	6.431967	34.12	15.24	0.720007	0.720008	36.27	16.05	0.494401	0.494401	43.78	17.02
	10000	2000	6.391650	6.391652	70.44	117.66	0.686660	0.686660	73.34	128.37	0.458320	0.458320	82.40	45.61
	10000	4000	6.391642	6.391643	272.71	143.39	0.686660	0.686661	283.23	158.21	0.458320	0.458321	325.58	81.60
-0.5	52	2000	6.497277	6.497277	4.47	6.03	0.998684	0.998684	4.53	5.71	0.771025	0.771025	4.16	8.70
	52	4000	6.497278	6.497278	17.13	23.74	0.998683	0.998683	17.31	22.66	0.771024	0.771024	16.12	34.37
	104	2000	6.434699	6.434699	4.87	5.51	0.933797	0.933797	4.97	5.42	0.694140	0.694140	4.48	8.41
	104	4000	6.434700	6.434700	18.73	22.22	0.933796	0.933796	19.20	21.31	0.694140	0.694140	17.21	32.49
	252	2000	6.375374	6.375374	6.03	5.70	0.876956	0.876956	6.24	5.68	0.628248	0.628248	5.39	7.61
	252	4000	6.375375	6.375375	23.24	21.04	0.876956	0.876956	24.17	20.51	0.628248	0.628248	20.68	28.47
	504	2000	6.342071	6.342072	7.91	7.06	0.846791	0.846791	8.32	7.22	0.593922	0.593922	6.94	7.46
	504	4000	6.342072	6.342073	30.65	21.73	0.846790	0.846790	32.20	21.65	0.593922	0.593922	26.77	27.29
	1008	2000	6.317620	6.317621	11.63	11.24	0.825369	0.825370	12.32	11.74	0.569846	0.569846	13.99	8.47
	1008	4000	6.317621	6.317623	45.04	26.05	0.825369	0.825369	47.57	26.48	0.569846	0.569846	54.64	27.97
	10000	2000	6.275649	6.275652	72.95	122.85	0.789931	0.789932	76.40	130.61	0.530602	0.530603	85.20	49.35
	10000	4000	6.275651	6.275652	282.11	154.93	0.789930	0.789931	294.67	163.17	0.530602	0.530604	331.26	91.33
-1	52	2000	6.395544	6.395544	3.79	5.57	1.127531	1.127531	4.64	6.52	0.872938	0.872938	3.51	8.14
	52	4000	6.395542	6.395542	14.50	22.20	1.127534	1.127534	17.91	26.29	0.872938	0.872938	13.28	31.17
	104	2000	6.330371	6.330371	4.17	5.05	1.059619	1.059619	4.79	5.67	0.790679	0.790679	3.78	7.57
	104	4000	6.330370	6.330369	15.92	19.86	1.059621	1.059621	18.60	22.50	0.790679	0.790679	14.54	30.41
	252	2000	6.268625	6.268626	5.32	5.17	0.999873	0.999873	5.64	5.47	0.719811	0.719811	4.68	6.93
	252	4000	6.268624	6.268624	20.43	18.33	0.999875	0.999875	21.71	19.67	0.719811	0.719811	17.99	26.24
	504	2000	6.233978	6.233979	7.19	6.51	0.968068	0.968068	7.50	6.70	0.682749	0.682749	7.94	6.96
	504	4000	6.233977	6.233978	27.80	19.15	0.968071	0.968071	29.01	19.77	0.682748	0.682748	31.13	25.10
	1008	2000	6.208546	6.208548	10.93	10.77	0.945441	0.945442	11.45	10.92	0.656689	0.656689	13.26	7.88
	1008	4000	6.208545	6.208547	42.11	24.06	0.945444	0.945444	44.11	24.19	0.656689	0.656689	51.73	25.25
	10000	2000	6.164904	6.164904	71.94	125.83	0.907932	0.907932	76.30	126.07	0.614098	0.614098	83.90	49.39
	10000	4000	6.164903	6.164904	278.78	156.93	0.907934	0.907935	295.46	157.10	0.614097	0.614099	327.11	88.16

Table B.16: Down-and-out, Up-and-out and Double barrier call: m is the matrix dimension, N is the number of monitoring dates and β is the parameter in (9). The initial asset price is $S_0 = 100$, the risk-free interest rate is 10% per annum ($r = 0.1$), the volatility is $\sigma = 0.25/S_0^{\beta}$, the asset pays no dividends ($q = 0$), and all options have six months to expiration ($T = 0.5$). The strike price E is equal to 105.

β	N	m	Prices				CPU Times (sec.)				Monte Carlo values	
			Rec. $\omega=1$	Rec.+BP $\omega=1$	Rec.+BP $\omega=2$	Rec.+BP $\omega=4$	Rec. $\omega=1$	Rec.+BP $\omega=1$	Rec.+BP $\omega=2$	Rec.+BP $\omega=4$	Confidence Interval	CPU Times (sec.)
0	52	2000	14.362044	14.362044	14.368058	14.369366	597	309	132	59	14.3423 - 14.3737	2504
	52	4000	14.362427	14.362427	14.365849	14.367109	4539	1452	709	330		
	104	2000	14.688093	14.688093	14.692382	14.693144	1070	602	258	110	14.6684 - 14.6997	4975
	104	4000	14.688808	14.688808	14.691229	14.692165	8130	2948	1401	626		
	252	2000	14.976531	14.976532	14.979313	14.979588	2255	1381	600	263	14.9614 - 14.9926	11996
	252	4000	14.977817	14.977818	14.979365	14.980017	17062	6831	3096	1335		
	504	2000	15.130593	15.130594	15.132574	15.132582	4043	2710	1119	548	15.1186 - 15.1498	23949
	504	4000	15.132509	15.132511	15.133593	15.134096	30530	13481	5914	2585		
	1008	2000	15.240091	15.240095	15.241506	15.241316	7303	5145	2061	1119	15.2282 - 15.2593	47910
	1008	4000	15.242882	15.242885	15.243636	15.244035	54829	26279	11411	5038		
-0.5	52	2000	14.545701	14.545701	14.544358	14.553307	592	323	140	64	14.5242 - 14.5576	2523
	52	4000	14.542978	14.542978	14.542853	14.546845	4566	1422	703	332		
	104	2000	14.887939	14.887940	14.886958	14.893453	1069	629	270	124	14.8738 - 14.9073	5014
	104	4000	14.886366	14.886366	14.886282	14.889097	8084	2897	1381	618		
	252	2000	15.191305	15.191306	15.190640	15.194984	2193	1361	598	254	15.1781 - 15.2116	12104
	252	4000	15.190981	15.190982	15.190932	15.192716	16567	7015	3212	1376		
	504	2000	15.353628	15.353629	15.353129	15.356335	3953	2684	1194	534	15.3483 - 15.3818	24180
	504	4000	15.354248	15.354250	15.354219	15.355452	29570	13634	5969	2573		
	1008	2000	15.469194	15.469194	15.468811	15.471220	6886	5117	2043	986	15.4511 - 15.4846	48331
	1008	4000	15.470853	15.470857	15.470839	15.471678	51971	26563	11362	5025		
-1	52	2000	14.756810	14.756811	14.760338	14.758654	447	332	145	67	14.7353 - 14.7712	2504
	52	4000	14.758292	14.758292	14.757768	14.757772	3403	1441	710	340		
	104	2000	15.120440	15.120440	15.122999	15.121791	847	630	272	125	15.1081 - 15.1442	4974
	104	4000	15.121902	15.121902	15.121527	15.121535	6507	2911	1384	624		
	252	2000	15.443344	15.443345	15.445053	15.444263	1881	1434	632	251	15.4216 - 15.4577	12001
	252	4000	15.445043	15.445044	15.444800	15.444812	14273	6938	3168	1368		
	504	2000	15.616345	15.616346	15.617602	15.617035	3446	2679	1113	549	15.60411 - 15.6403	23960
	504	4000	15.618467	15.618469	15.618295	15.618309	26085	13904	6095	2656		
	1008	2000	15.739613	15.739613	15.740549	15.740141	6332	5212	2089	963	15.7278 - 15.7640	47922
	1008	4000	15.742423	15.742427	15.742302	15.742317	47523	26561	11375	5076		

Table B.17: Fixed-strike lookback put: m is the matrix dimension, N is the number of monitoring dates and β is the parameter in (9). The initial asset price is $S_0 = 100$, $\sigma = 0.25/S_0^{\beta}$, the risk-free interest rate is 10% per annum ($r = 0.1$), the volatility is $\sigma = 0.25/S_0^{\beta}$, the asset pays no dividends ($q = 0$), and all options have six months to expiration ($T = 0.5$). The strike price E is equal to 105. ω is the parameter in Remark 1.

β	N	m	Prices				CPU Times (sec.)				Monte Carlo values	
			Rec. $\omega=1$	Rec.+BP $\omega=1$	Rec.+BP $\omega=2$	Rec.+BP $\omega=4$	Rec. $\omega=1$	Rec.+BP $\omega=1$	Rec.+BP $\omega=2$	Rec.+BP $\omega=4$	Confidence Interval	CPU Times (sec.)
0	52	2000	2.971650	2.971650	2.971659	2.971273	832	554	272	151	2.9684 - 2.9899	2504
	52	4000	2.971649	2.971649	2.971651	2.971663	5255	2387	1287	644		
	104	2000	2.979972	2.979972	2.979856	2.978041	1552	1122	547	297	2.9730 - 2.9945	4975
	104	4000	2.979976	2.979976	2.979980	2.979828	9667	4879	2325	1207		
	252	2000	2.984833	2.984834	2.984143	2.980487	3407	2485	1368	724	2.9792 - 3.0008	11996
	252	4000	2.984937	2.984937	2.984850	2.984018	20237	11290	5451	2693		
	504	2000	2.986369	2.986370	2.985192	2.980024	6494	4907	2736	1471	2.9802 - 3.0018	23949
	504	4000	2.986659	2.986660	2.986394	2.985004	37182	22672	11095	5397		
	1008	2000	2.986954	2.986955	2.985223	2.978418	11913	9699	5417	2951	2.9728 - 2.9944	47910
	1008	4000	2.987461	2.987462	2.987004	2.984990	70096	44719	21764	10988		
-0.5	52	2000	2.919996	2.919996	2.920009	2.919707	843	573	284	157	2.9151 - 2.9356	2523
	52	4000	2.920010	2.920010	2.920007	2.920017	5615	2377	1186	673		
	104	2000	2.928446	2.928446	2.928341	2.926760	1642	1128	543	307	2.9196 - 2.9401	5014
	104	4000	2.928465	2.928465	2.928459	2.928347	10072	4814	2379	1304		
	252	2000	2.933353	2.933353	2.932724	2.929454	3592	2591	1370	725	2.9272 - 2.9477	12104
	252	4000	2.933498	2.933499	2.933402	2.932702	21221	11376	5755	3140		
	504	2000	2.934878	2.934878	2.933790	2.929121	6448	5137	2796	1400	2.9283 - 2.9487	24180
	504	4000	2.935238	2.935238	2.934972	2.933777	38909	22662	11140	5495		
	1008	2000	2.935444	2.935444	2.933788	2.927539	12413	9834	5488	2992	2.9264 - 2.9468	48331
	1008	4000	2.936030	2.936031	2.935583	2.933823	71120	45563	22044	10207		
-1	52	2000	2.870184	2.870184	2.870212	2.869682	709	557	285	158	2.8617 - 2.8812	2504
	52	4000	2.870184	2.870184	2.870184	2.870193	4322	2307	1181	629		
	104	2000	2.878753	2.878753	2.878602	2.876448	1368	1120	540	306	2.8670 - 2.8864	4974
	104	4000	2.878759	2.878759	2.878760	2.878558	8724	4627	2323	1221		
	252	2000	2.883723	2.883724	2.882903	2.878681	3191	2686	1360	736	2.8696 - 2.8891	12001
	252	4000	2.883859	2.883859	2.883741	2.882776	18885	11500	5717	2719		
	504	2000	2.885274	2.885274	2.883894	2.877906	6221	4976	2675	1464	2.8734 - 2.8929	23960
	504	4000	2.885618	2.885619	2.885298	2.883710	35444	22953	11326	5387		
	1008	2000	2.885842	2.885842	2.883806	2.875826	11770	9963	5642	3000	2.8787 - 2.8982	47922
	1008	4000	2.886430	2.886431	2.885892	2.883583	69791	45781	22172	10841		

Table B.18: Fixed-strike Asian call: m is the matrix dimension, N is the number of monitoring dates and β is the parameter in (9). The initial asset price is $S_0 = 100$, $\sigma = 0.25/S_0^\beta$, the risk-free interest rate is 10% per annum ($r = 0.1$), the volatility is $\sigma = 0.25/S_0^\beta$, the asset pays no dividends ($q = 0$), and all options have six months to expiration ($T = 0.5$). The strike price E is equal to 105. ω is the parameter in Remark 1.

β	N	m	Prices		CPU Times (sec.)		Monte Carlo values	
			Rec.	Rec.+BP	Rec.	Rec.+BP	Confidence Interval	CPU Times (sec.)
0	52	2000	5.652152	5.652152	9.86	4.09	5.6518-5.7025	2434.59
	52	4000	5.653228	5.653228	37.81	13.50		
	104	2000	5.635620	5.635620	30.66	10.02	5.6331-5.6716	4857.49
	104	4000	5.636695	5.636695	117.49	26.51		
	252	200	5.625925	5.625925	145.48	56.39	5.6248-5.6632	11756.54
	252	4000	5.626999	5.627000	553.91	126.05		
	504	2000	5.622521	5.622521	507.55	239.23	5.6224-5.6608	23505.95
	504	4000	5.623595	5.623596	1919.93	528.45		
	1008	2000	5.620819	5.620820	1804.38	1054.60	5.6068-5.6452	46992.97
	1008	4000	5.621893	5.621894	6764.08	2196.21		
-0.5	52	2000	5.580878	5.580878	13.08	7.46	5.5772-5.6133	2446.38
	52	4000	5.581670	5.581670	50.64	23.71		
	104	2000	5.564554	5.564554	35.21	13.98	5.5473-5.5832	4881.28
	104	4000	5.565345	5.565345	134.63	35.45		
	252	2000	5.554981	5.554981	154.47	60.41	5.5414-5.5772	11811.86
	252	4000	5.555772	5.555773	589.53	135.57		
	504	2000	5.551620	5.551620	533.47	241.04	5.5367-5.5725	23616.72
	504	4000	5.552411	5.552412	2003.21	554.70		
	1008	2000	5.549940	5.549940	1823.83	1044.87	5.5276-5.5634	47231.25
	1008	4000	5.550731	5.550732	6931.57	2169.42		
-1	52	2000	5.503713	5.503713	10.75	7.32	5.4874-5.5213	2436.01
	52	4000	5.503624	5.503624	41.19	26.01		
	104	2000	5.487607	5.487607	29.01	14.10	5.4634-5.4970	4858.14
	104	4000	5.487518	5.487518	110.89	35.87		
	252	2000	5.478162	5.478162	132.63	61.44	5.4638-5.4975	11756.46
	252	4000	5.478073	5.478074	509.16	134.53		
	504	2000	5.474846	5.474846	478.82	229.71	5.4407-5.4762	23504.15
	504	4000	5.474757	5.474758	1824.61	559.48		
	1008	2000	5.473189	5.473189	1744.86	1047.12	5.4639-5.4976	46996.17
	1008	4000	5.473100	5.473101	6619.11	2196.39		

Table B.19: Step call: m is the matrix dimension, N is the number of monitoring dates. The initial asset price is $S_0 = 100$, $\sigma = 0.25/S_0^{\beta}$, the risk-free interest rate is 10% per annum ($r = 0.1$), the volatility is $\sigma = 0.25/S_0^{\beta}$, the asset pays no dividends ($q = 0$), and all options have six months to expiration ($T = 0.5$). The payoff parameters are $\rho = 0.5$, $E = 105$, and $A = [90, 110]$.