Les Cahiers du GERAD ISSN: 0711–2440

Dynamic programming and parallel computing for valuing two-dimensional American-style options

H. Ben-Ameur, M. Ben Abdellatif, B. Rémillard

G–2016–48

June 2016

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Dynamic programming and parallel computing for valuing two-dimensional American-style options

Hatem Ben-Ameur [a]
Malek Ben Abdellatif [b]
Bruno Rémillard [c]

[a] *GERAD & Department of Management Sciences, HEC Montréal, Montréal (Québec), Canada H3T 2A7*

[b] *Department of Management Sciences, HEC Montréal, Montréal (Québec), Canada H3T 2A7*

[c] *GERAD & CRM & Department of Management Sciences, HEC Montréal, Montréal (Québec), Canada H3T 2A7*

atem.ben-ameur@hec.ca
malek.ben-abdellatif@hec.ca
bruno.remillard@hec.ca

June 2016

Les Cahiers du GERAD
G–2016–48

Abstract: We propose a dynamic program coupled with finite elements for valuing two-dimensional American-style options. To speed-up our procedure, we use parallel computing at every step of the recursion. Our model is flexible because it accommodates a large family of option contracts signed on two underlying assets that move according to a lognormal vector process. The same procedure can be adapted to accommodate a larger family of derivative contracts and state-process dynamics. Our numerical experiments show convergence and efficiency, positioning our method as a viable alternative to traditional methodologies based on trees, finite differences, and Monte Carlo simulation.

Keywords: American options, two-dimensional state spaces, dynamic programming, finite elements, parallel computing

1 Introduction

We propose a dynamic program for valuing two-dimensional American-style options. Parallel computing is used to reduce its running time and enhance its efficiency. Two-dimensional options are traded over the counter (OTC). Examples include rainbow, quanto, and basket options. According to the Bank of International Settlements (BIS), the gross market value of total OTC options was 438 billion US dollars in 2015. These options are used for hedging market risk related to financial assets, natural resources, and commodities.

American-style options cannot be evaluated in closed form and must be approximated in some way. The literature discusses several methodologies based on Monte Carlo simulation, trees, and finite differences. This holds for one-dimensional as well as multidimensional state spaces.

The multidimensional lattice approach assumes a discrete model that usually converges to a continuous counterpart. It runs in two steps. The first step, which is the most challenging, is forward and used for the lattice construction (the overall state space). The second step is backward and is used for evaluation purposes (Boyle, 1988, Boyle et al., 1989, Kamrad and Ritchken, 1991).

Finite differences (FD) is a backward methodology that assumes state and time discretizations. At each step of the recursion, one solves a partial differential equation with boundary conditions, which characterizes the option's value. While Dockendorf and Paxson (2015) use FD for valuing multidimensional European-style options, Hartley (2000) and Berridge and Schumacher (2008) use FD for valuing their American counterparts.

Simulation-based methodologies run in two steps. The first step is forward, and involves simulating a random sample of the underlying assets' trajectories, which can be seen as a random lattice. The second step, which is the most challenging, is backward and evaluates the option contract and identifies its optimal exercise strategy. At any given decision date, the simulated trajectories do not intersect (almost surely), which weakens the evaluation step via the fundamental theorem of asset pricing in no-arbitrage markets. The solution proposed in the literature is threefold: bundling methods (Tilley, 1993, Barraquand and Martineau, 1995, Boyle et al., 1997, Raymar and Zwecher, 1997, Jin et al., 2007, 2013, Broadie and Glasserman, 1997, Bally and Pages, 2003a,b, Bally and Printems, 2005), regression and/or global approximations (Carriere, 1996, Tsitsiklis and Van Roy, 1999, Longstaff and Schwartz, 2001, Broadie and Glasserman, 1997), and duality methods (Haugh and Kogan, 2004, Rogers, 2002, Andersen and Broadie, 2004, Del Moral et al., 2012).

Dynamic programming (DP) coupled with finite elements has been used with success for valuing one-dimensional American-style options in pure-diffusion models (Ben-Ameur et al., 2002) and jump-diffusion models (Ben-Ameur et al., 2016). Design-wise, one-dimensional DP can be naturally extended to higher dimensions and combined with parallel computing in order to reduce computation time.

DP starts running at the option's maturity, where the option's value function is known. The option's value is computed on a given grid of points. Then, DP alternates between interpolation at step $n+1$ and evaluation at step n, and moves backward from maturity down to the origin. At each step of the recursion, DP acts as follows: one first uses a piecewise polynomial and interpolates the option's value function from the grid points to the overall state space; then one uses no-arbitrage pricing and approximates the option's value function at the previous step on (possibly) the same grid points. These last computational tasks are not sequential and, thus, can be parallelized.

This paper is organized as follows. While Section 2 presents the model, Section 3 describes our dynamic program and discusses parallel computing. Section 4 is a numerical investigation, and Section 5 concludes.

2 The model

We consider a frictionless market in which two stocks, S^1 and S^2, are traded continuously and move according to a bivariate log-normal process. The risk-free rate, r, is assumed to be constant. This market is known to be arbitrage free and complete. Thus, there exists a unique risk-neutral probability measure $\mathbb{Q}$ under which

the state process (S^1, S^2) moves according to the following stochastic differential equation:

$$\frac{dS_t^i}{S_t^i} = (r - d_i)dt + \sigma_i dW_t^i, \qquad \text{for } i = 1, 2, \tag{1}$$

where d_i is the continuous dividend rate of stock i, σ_i is its log-return volatility, and (W^1, W^2) is a bivariate correlated Brownian motion with

$$\text{Cor}(W_t^1, W_t^2) = \rho, \quad \text{for all } t > 0.$$

The solution of (1) can be written as

$$S_u^i = S_t^i \exp\left[\left(r - d_i - \frac{\sigma_i^2}{2}\right)(u - t) + \sigma_i\left(W_u^i - W_t^i\right)\right], \tag{2}$$

for $0 \le t \le u$.

An American option on (S^1, S^2) with maturity T is defined by its cash-flow process, $\kappa(t, x, y) \ge 0$, for $0 \le t \le T$, $x > 0$, $y > 0$, where $x = S_t^1$ and $y = S_t^2$. This is the option's value under exercise. Its European counterpart is characterized by

$$\kappa(t, x, y) = 0, \quad \text{for } 0 \le t < T.$$

Examples include the exchange option:

$$\kappa(t, x, y) = \max(x - y, 0),$$

the call-on-max option:

$$\kappa(t, x, y) = \max(\max(x, y) - K, 0),$$

and the put-on-min option:

$$\kappa(t, x, y) = \max(K - \min(x, y), 0),$$

where K is the the option's strike price. The exchange option gives the option holder the right to exchange S^2 for S^1; the call-on-max option gives the right to purchase the higher-priced asset at the strike price K; and the put-on-min gives the right to sell the lower-priced asset at the strike K.

This setting is described in detail in Stulz (1982) and Johnson (1987) where closed-form solutions for the above-mentioned European options are given. We report them in Appendix A.

3 Dynamic programming and parallel computing

First, we describe the dynamic programming approach where the assets are modeled by a general Markov process. Then, we present the use of parallel computing to improve efficiency.

3.1 Dynamic programming

We consider a two-dimensional Bermudan option, characterized by its exercise value $v_t^e(x, y) = \kappa(t, x, y)$ and $N + 1$ exercise opportunities, $t_0 = 0, t_1, \ldots, t_N = T$.

Let $\mathcal{G}$ be a set of grid points $\{(a_1, b_1), (a_1, b_2), \ldots, (a_p, b_q)\}$ such that $\max(\Delta a_k, \Delta b_l) \to 0$ and $\mathbb{Q}[(S_t^1, S_t^2) \in [a_p, \infty) \times \mathbb{R}_+^* \cup \mathbb{R}_+^* \times [b_q, \infty)] \to 0$, when p and $q \to 0$. Let $a_0 = b_0 = 0$ and $a_{p+1} = b_{q+1} = \infty$. The rectangle $[a_i, a_{i+1}) \times [b_j, b_{j+1})$ is designated by R_{ij}.

For simplicity, assume that the state process (S^1, S^2) is Markov and homogeneous, $t_{n+1} - t_n = \Delta t$ a positive constant, the grid points $\mathcal{G}$ fixed along the recursion, and $\kappa(t, x, y) = \kappa(x, y)$. To reach the pure American option, let $\Delta t \to 0$.

Define the transition tables T^{00}, T^{10}, T^{01}, and T^{11} as follows:

$$\begin{aligned} T^{\nu\mu}_{klij} &= \mathbb{E}^*\left[(S^1_{t_{n+1}})^\nu (S^2_{t_{n+1}})^\mu \mathbb{I}\left((S^1_{t_{n+1}}, S^2_{t_{n+1}}) \in R_{ij}\right) \mid \right. \\ &\qquad \left. (S^1_{t_n}, S^2_{t_n}) = (a_k, b_l)\right], \quad \text{for } \nu \text{ and } \mu \in \{0,1\}. \end{aligned} \tag{3}$$

For example, T^{00}_{klij} represents the transition probability that the Markov process (S^1, S^2) moves from (a_k, b_l) at t_n and visits the rectangle R_{ij} at t_{n+1}. These transition parameters, which are at the heart of the dynamic-programming approach, can be considered a fixed cost as long as the Markov state process is homogeneous, $t_{n+1} - t_n$ is a positive constant, and the grid points $\mathcal{G}$ do not depend on time. We derive and provide closed-form solutions for them in Appendix B.

Assume that an approximation of the option's value function is available at a future decision date t_{n+1} on $\mathcal{G}$, indicated by $\tilde{v}_{n+1}(a_k, b_l)$, for $k = 1, \ldots, p$ and $l = 1, \ldots, q$. This is not really a strong assumption since the option's value function is known at maturity in closed form, that is, $\tilde{v}_N = v_N = v^e_N$. DP acts as follows:

1. Use a bilinear piecewise polynomial and interpolate the option's value function $\tilde{v}_{n+1}$ at t_{n+1} from $\mathcal{G}$ to the overall state space $[0, \infty)^2$ by setting

$$\begin{aligned} \hat{v}_{n+1}(x, y) &= \sum_{i=0}^{p} \sum_{j=0}^{q} \left(\alpha^{n+1}_{ij} + \beta^{n+1}_{ij} x + \gamma^{n+1}_{ij} y + \delta^{n+1}_{ij} xy\right) \times \\ &\qquad \mathbb{I}\left((x, y) \in R_{ij}\right), \end{aligned} \tag{4}$$

   where the local coefficients α^{n+1}_{ij}, β^{n+1}_{ij}, γ^{n+1}_{ij}, and δ^{n+1}_{ij}, for $i = 1, \ldots, p-1$ and $j = 1, \ldots, q-1$, are obtained in closed form from the system of linear equations:

$$\begin{cases} \hat{v}_{n+1}(a_i, b_j) = \tilde{v}_{n+1}(a_i, b_j) \\ \hat{v}_{n+1}(a_{i+1}, b_j) = \tilde{v}_{n+1}(a_{i+1}, b_j) \\ \hat{v}_{n+1}(a_i, b_{j+1}) = \tilde{v}_{n+1}(a_i, b_{j+1}) \\ \hat{v}_{n+1}(a_{i+1}, b_{j+1}) = \tilde{v}_{n+1}(a_{i+1}, b_{j+1}) \end{cases} \tag{5}$$

   and the rest of them are set to their adjacent counterparts;

2. Use non-arbitrage pricing and approximate the option's holding value function at t_n on $\mathcal{G}$:

$$\begin{aligned} &\tilde{v}^h_n(a_k, b_l) \\ &= \mathbb{E}^*\left[e^{-r\Delta t} \hat{v}_{n+1}(S^1_{t_{n+1}}, S^2_{t_{n+1}}) \mid (S^1_{t_n}, S^2_{t_n}) = (a_k, b_l)\right] \\ &= e^{-r\Delta t} \sum_{i,j} \left(\alpha^{n+1}_{ij} T^{00}_{klij} + \beta^{n+1}_{ij} T^{10}_{klij} + \gamma^{n+1}_{ij} T^{01}_{klij} + \delta^{n+1}_{ij} T^{11}_{klij}\right); \end{aligned} \tag{6}$$

3. Approximate the option's value function at t_n on $\mathcal{G}$:

$$\tilde{v}_n(a_k, b_l) = \max(v^e_n(a_k, b_l), \tilde{v}^h_n(a_k, b_l)); \tag{7}$$

4. Go to step 1 and repeat until $n = 0$.

Equation (6) splits the option's holding value into two parts: the local coefficients are related to the option contract and the transition parameters to the dynamics of the state process. All in all, the option's holding value is a sum of local future-value pieces, times their associated transition parameters, discounted back at the risk free-rate. The same equation shows that DP assumes a space discretization, but not a time discretization, and does respect the true dynamics of the state process as long as the transition parameters in Equation (3) are known in closed form. Finally, Equation (4) shows that DP ends up with an interpolation of $v_0(x, y)$, defined on the overall state space. Thus, the first and second derivatives of $v_0(x, y)$ with respect to x and y then become available, among other sensitivity coefficients.

Higher-order two-dimensional piecewise approximations are more accurate but require a higher computing time, and vice versa. We find that the bilinear piecewise interpolation of Equation (4) is an acceptable compromise.

3.2 Parallel computing

Parallel computing uses multiple central processing units (CPUs) simultaneously to speed-up complex computations. For C programming, used herein to achieve our numerical experiments, there are two libraries used for parallel computing: MPI and OpenMP.

The Message Passing Interface (MPI) library allows the computing process to exchange information between the running CPU environments in order to achieve a given job. Each CPU has access to a certain memory space. MPI requires case-sensitive programming changes from the serial code to its parallel version.

Parallel computing can also run when all CPUs share the same memory space. Open Multi Processing (OpenMP) is a library that allows one to implement parallel computing with a minimal change to the serial code. However, shared-memory supercomputers are extremely expensive, and thus somewhat inaccessible.

MPI and OpenMP are compatible with Fortran and C languages. Parallel computing is also feasible under other software packages, e.g., Graphics Processing Unit (GPU) for Matlab and R.

The easiest way to parallelize DP is to submit the computation tasks associated to a given grid point (a_k, b_l), for $k = 1, \ldots, p$ and $l = 1, \ldots, q$, to a single CPU. Our parallel code acts as follows.

1. This single CPU computes once and locally stores the overall grid points (a_i, b_j) and the exercise values $\kappa(a_i, b_j)$, for $i = 1, \ldots, p$ and $j = 1, \ldots, q$.
2. Following Equation (3), it also computes once and locally stores the $4 \times (p + 1)(q + 1)$ transition parameters T^{00}_{klij}, T^{10}_{klij}, T^{01}_{klij}, and T^{11}_{klij}, for $i = 0, \ldots, p$ and $j = 0, \ldots, q$.
3. Following Equation (4)–(5), it computes and stores at step $n+1$ the local coefficients α^{n+1}_{ij}, β^{n+1}_{ij}, γ^{n+1}_{ij}, and δ^{n+1}_{ij}, for $i = 0, \ldots, p$ and $j = 0, \ldots, q$.
4. Following Equation (6)–(7), it computes and stores at step n the option's holding value $\tilde{v}^h_n(a_k, b_l)$ then the overall value $\tilde{v}_n(a_k, b_l)$.
5. The same CPU exports $\tilde{v}_n(a_k, b_l)$ to a selected CPU, the so-called master CPU.
6. The master CPU collects $\tilde{v}_n(a_k, b_l)$, for $k = 1, \ldots, p$ and $l = 1, \ldots, q$, and sends them back to all running CPUs.
7. Go to step 3 and repeat until $n = 0$.

Since the number of CPUs available to the analyst is usually less than the grid size pq, we submit the same number of grid points to each CPU. Fixing this number for each grid size pq is a question of efficiency.

Assume the same program is run twice with n and kn CPUs, where n and $k \in \mathbb{N}^*$. Let τ_1 and τ_2 be the computing times of the first and second run, respectively. In the best case scenario, the expected computing time declines by the same factor k, that is,

$$E[\tau_2] = \frac{E[\tau_1]}{k},$$

which results in a relative efficiency ratio

$$\frac{E[\tau_1]/E[\tau_2]}{k} = 1.$$

In fact, this ratio is usually lower than one, since the running CPUs exchange some information during the computing process, as in steps 5-6, and the parallel code behaves partially as the serial code, as in step 1. A relative efficiency ratio higher than 75% is highly desirable.

We use the supercomputer Briarée managed by Calcul Québec and Compute Canada;[1] it is equipped with 8064 CPUs (cores). These 8064 cores are divided in 672 computing nodes, each equipped with two six-core processors running at a speed of 2.667 GHz. Thus, each computing node includes 12 cores. The number of

[1]The operation of this supercomputer is funded by the Canada Foundation for Innovation (CFI), ministère de l'Économie, de la Science et de l'Innovation du Québec (MESI) and the Fonds de recherche du Québec - Nature et technologies (FRQ-NT).

computing nodes, $\bar{n}$, called for parallel computing must be specified by the programmer ($\bar{n} \leq 672$), which results in $12 \times \bar{n}$ cores. Briarée has a total memory space of 26.72 TB, split between the computing nodes. Given the architecture of Briarée's hardware (Figure 1), the number of grid points submitted to each core is

$$\frac{pq}{12 \times \bar{n}} \in \mathbb{N}^*.$$

The code lines are written in C and compiled with GCC. We use the MPI library to access parallel computing.

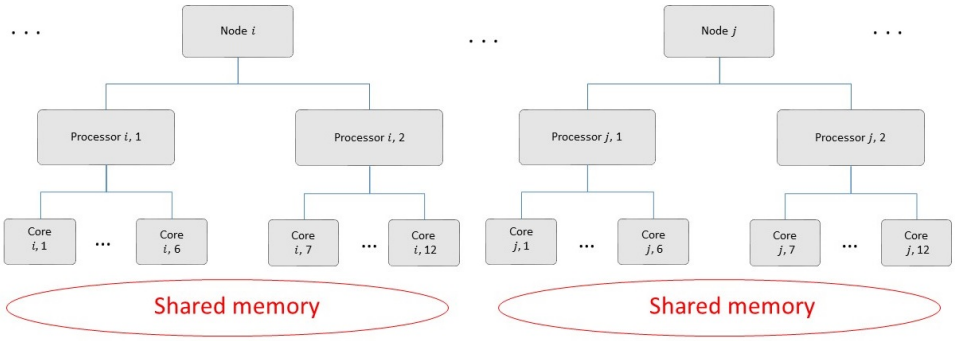

Figure 1: Briarée's architecture

4 Numerical investigation

4.1 European options

Table 1 compares DP to Boyle (1988), who uses a two-dimensional trinomial tree for valuing European put-on-min options. The closed-form solution for this contract is given in Stulz (1982) (see also Appendix A). Set $S_0^1 = S_0^2 = 40$, $d_1 = d_2 = 0$, $\sigma_1 = 0.2$, $\sigma_2 = 0.3$, $\rho = 0.5$, $r = 5\%$ (effective) $\equiv 0.04879$ (continuously compounded), $T = 7$ months $\equiv 0.58333$ years. Following the constraints of Calcul Québec and Compute Canada, we select $\bar{n}$ as the highest integer lower than 200 that ensures an efficiency ratio higher than 75%, where $pq/(12\bar{n}) \in \mathbb{N}^*$. As explained in Section 3, DP does not need a time discretization. For comparison

Table 1: European put-on-min options
DP vs. Boyle (1988)

	DP with a grid size pq				
	72^2	144^2	300^2	Boyle	Closed form
$K = 35$	1.411	1.392	1.388	1.425	1.387
40	3.837	3.805	3.800	3.778	3.798
45	7.543	7.508	7.501	7.475	7.500
$12\times\bar{n}$	576	1728	1800		
Total CPU	0.93	5.40	34.48		
Linear CPU	0.65	4.11	11.99	10 time steps	
$K = 35$	1.504	1.410	1.391	1.392	1.387
40	3.970	3.832	3.805	3.795	3.798
45	7.694	7.537	7.507	7.499	7.500
$12\times\bar{n}$	576	1728	1800		
Total CPU	2.14	10.29	67.01		
Linear CPU	1.83	8.85	46.91	50 time steps	

purposes, however, we run DP with the same number of time steps as assumed in Boyle (1988). As expected, when the number of time steps is low, DP behaves almost perfectly, whereas the binomial tree is less accurate. For high number of time steps, the binomial tree converges and becomes as accurate as DP.

Boyle (1988) does not report his computing times. Each DP's CPU time (in seconds) can be split into a fixed cost, associated to the transition parameters, and a linear cost, associated to the backward recursion. Our numerical experiments show that the fixed cost accounts for an important portion of the total CPU time. The relevant DP's computing time is the linear CPU time, since the transition parameters can be computed only once or twice a day, following the model-estimation step.

4.2 American options

We compare DP to alternative methodologies for valuing American-style options, that is, the lattice approach, finite differences, and Monte Carlo simulation, among other ad hoc procedures.

Detemple et al. (2003) consider an American call-on-min option. They propose several approximations of the option's exercise frontier at each decision date, which results in a lower and an upper bound for the option's value. These bounds are then estimated by Monte Carlo simulation of size $50,000$. Table 2 reports DP values versus Detemple et al. (2003). The parameters are $K = 100$, $d_1 = d_2 = 0.05$, $\rho = 0$, $\sigma_1 = \sigma_2 = 0.2$, $T = 1$, and $r = 0.06$. DP values are always between their lower and upper counterparts.

Table 3 compares DP to Boyle's (1988) trinomial tree. The parameters are given in Section 4.1. All in all, DP values are close to Boyle's (1988) values. Consistent with the analysis of Section 4.1, the gap between the competing approximations is larger when the number of exercise opportunities is low; DP is expected to be more accurate.

While Monte Carlo simulation is combined with a dual approach by Rogers (2002), it is combined with a bundling approach by Jin et al. (2007). Their random samples are of size $10,000$ and $60,000$, respectively. We report their respective 95% confidence intervals. Hartley (2000) uses finite differences. The parameters are $K = 100$, $d_1 = d_2 = 0$, $\rho = 0$, $\sigma_1 = \sigma_2 = 0.6$, $r = 0.06$, and $T = 0.5$.

DP values, obtained with $p = q = 300$, almost always belong to their associated 95% confidence intervals and compare extremely well with Hartley's (2000) values, which are described by Rogers (2002) as extremely accurate.

Table 2: American call-on-min options
DP vs. Detemple et al. (2003)

		DP with a grid size pq				
S_0^2	S_0^1	72^2	144^2	300^2	Lower bound	Upper bound
90	90	0.893	0.799	0.782	0.63	0.97
	100	1.691	1.560	1.536	1.27	1.88
	110	2.526	2.364	2.335	1.92	2.77
	120	3.177	2.984	2.950	2.49	3.37
100	100	3.392	3.233	3.205	2.62	3.98
	110	5.341	5.174	5.145	4.26	6.36
	120	6.680	6.608	6.576	5.51	7.58
	130	7.639	7.402	7.361	6.36	7.95
110	110	9.455	9.362	9.348	10.00	13.66
	120	12.225	12.118	12.098	10.37	14.50
	130	13.511	13.335	13.304	11.58	14.53
	140	14.147	13.912	13.869	12.51	14.44
$12\times\bar{n}$		576	1728	1800		
Total CPU		2.93	15.25	81.88		
Linear CPU		2.89	15.05	78.76	100 time steps	

Table 3: American put-on-min options
DP vs. Boyle (1988)

	DP with a grid size pq			
	72^2	144^2	300^2	Boyle
$K = 35$	1.436	1.416	1.413	1.450
40	3.918	3.887	3.881	3.870
45	7.713	7.678	7.671	7.645
$12\times\bar{n}$	576	1728	1800	
Total CPU	1.01	5.79	36.63	
Linear CPU	0.67	4.51	13.75	10 time steps
$K = 35$	1.535	1.440	1.422	1.423
40	4.064	3.926	3.899	3.892
45	7.880	7.727	7.697	7.689
$12\times\bar{n}$	576	1728	1800	
Total CPU	1.96	9.94	66.45	
Linear CPU	1.71	8.46	46.43	50 time steps

Table 4: American put-on-min options – DP vs. Rogers (2002), Jin et al. (2007), and Hartley (2000)

	DP with a grid size pq					
(S_0^1, S_0^2)	72^2	144^2	300^2	Rogers	Jin et al.	Hartley
(80, 80)	37.938	37.416	37.312	[37.35, 37.65]	[37.1000, 37.4022]	37.30
(80, 100)	32.775	32.205	32.091	[32.12, 32.26]	[31.8421, 32.1451]	32.08
(80, 120)	29.826	29.265	29.152	[29.18, 29.32]	[28.8860, 29.2434]	29.14
(100, 100)	25.889	25.205	25.066	[24.93, 25.23]	[24.8296, 25.1576]	25.06
(100, 120)	21.779	21.067	20.920	[20.89, 21.09]	[20.6850, 20.9932]	20.91
(120,120)	16.864	16.102	15.942	[15.99, 16.19]	[15.6737, 16.0017]	15.92
$12\times\bar{n}$	576	1728	1800			
Total CPU	1.66	8.70	46.72	180	24	
Linear CPU	1.60	8.41	43.58		51 time steps	

In addition, DP shows competitive CPU times. These could have been further drastically reduced with access to the overall hardware capabilities. These results reinforce DP as a viable alternative for valuing two-dimensional American options.

5 Conclusion

We propose a dynamic program coupled with piecewise bilinear approximations for valuing two-dimensional American options. We use parallel computing to speed up efficiency. This methodology presents two major advantages with respect to its competitors, giving that it assumes a space but not a time discretization, and a numerical but not a statistical error. Our investigation shows that DP competes well against its alternative methodologies in terms of accuracy. Although DP's CPU times are competitive, they can be further drastically reduced through access to the overall hardware capabilities.

This paper paves the way for a few useful extensions. The same DP procedure can accommodate a larger family of derivative contracts, such as Asian options and barrier options, and more complex state processes, such as two-dimensional jump diffusions and GARCH processes, as long as the transition parameters can be computed efficiently. The extension to higher dimensions is challenging but feasible. Monte Carlo simulation is certainly required for valuing option contracts in high state-space dimensions, but DP can firstly be combined with quasi–Monte Carlo simulation in moderate state-space dimensions, where the latter deterministic approach is known to be more efficient than the former random approach.

Appendix A Closed-form solutions for European options

The closed-form solutions given below are taken from Stulz (1982).

A.1 Exchange option

The price of an European exchange option giving the right to exchange S^2 against S^1 at maturity date T, evaluated at date t, is given by

$$E(S_t^1, S_t^2, T-t) = S_t^1 e^{-d_1(T-t)}\Phi(d_+) - S_t^2 e^{-d_2(T-t)}\Phi(d_-),$$

where

$$\begin{aligned} d_\pm &= \frac{ln(S_t^1/S_t^2) - (d_1 - d_2 \pm \sigma^2/2)(T-t)}{\sigma\sqrt{(T-t)}}, \\ \sigma &= \sqrt{\sigma_1^2 + \sigma_2^2 - 2\sigma_1\sigma_2\rho}, \end{aligned}$$

and $\Phi(\cdot)$ is the cumulative density function of the univariate standard normal distribution.

A.2 Call-on-max option

The price of an European call-on-max option with maturity date T and strike price K, evaluated at date t, is given by

$$\begin{aligned} C_{\max}(S_t^1, S_t^2, K, T-t) &= S_t^1 e^{-d_1(T-t)}\Phi(d_1^1, d_{12}, \rho_{12}) + S_t^2 e^{-d_2(T-t)}\Phi(d_1^2, d_{21}, \rho_{21}) - \\ &\quad Ke^{-r(T-t)}(1 - \Phi(-d_2^1, -d_2^2, \rho)), \end{aligned}$$

where

$$\begin{aligned} \sigma &= \sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2, \\ d_{ij} &= \frac{log(S_t^i/S_t^j) + (d_j - d_i + \sigma^2/2)(T-t)}{\sigma\sqrt{(T-t)}}, \\ \rho_{ij} &= \frac{\sigma_i - \rho\sigma_j}{\sigma}, \\ d_2^i &= \frac{log(S_t^i/K) + (r - d_i - \sigma_i^2/2)(T-t)}{\sigma_i\sqrt{T-t}}, \\ d_1^i &= d_2^i + \sigma_i\sqrt{T-t}, \end{aligned}$$

and $\Phi(\cdot,\cdot,\rho)$ is the cumulative density function of the bivariate standard normal distribution with correlation coefficient ρ.

A.3 Put-on-min option

The price of a European put-on-min option with maturity T and strike price K, evaluated at date t, is given by

$$P_{\min}(S_t^1, S_t^2, K, T-t) = e^{-r(t-t)}K - C_{\min}(S_t^1, S_t^2, 0, T-t) + C_{\min}(S_t^1, S_t^2, K, T-t),$$

where $C_{\min}(S_t^1, S_t^2, K, T-t)$ is the price of the European call-on-min option with strike price K and maturity T, evaluated at date t as follows.

$$\begin{aligned} C_{\min}(S_t^1, S_t^2, K, T-t) &= S_t^1 e^{-d_1(T-t)}\Phi(d_1^1, d_{12}', -\rho_{12}) + S_t^2 e^{-d_2(T-t)}\Phi(d_1^2, d_{21}', -\rho_{21}) - \\ &\quad Ke^{-r(T-t)}\Phi(d_2^1, d_2^2, \rho)), \end{aligned}$$

where

$$d_{ij}' = \frac{log(S_t^i/S_t^j) + (d_j - d_i - \sigma^2/2)(T-t)}{\sigma\sqrt{(T-t)}}.$$

Appendix B Transition parameters

The transition parameters $T_{klij}^{\nu\mu}$ for ν and $\mu \in \{0,1\}$, $k \in \{1,\ldots,p\}$, $l \in \{1,\ldots,q\}$, $i \in \{0,\ldots,p\}$, and $j \in \{0,\ldots,q\}$ are calculated as follows.

$$\begin{aligned}
T_{klij}^{00} &= \mathbb{E}^*\left[\mathbb{I}\left((S^1_{t_{n+1}}, S^2_{t_{n+1}}) \in R_{ij}\right) \mid (S^1_{t_n}, S^2_{t_n}) = (a_k, b_l)\right] \\
&= \mathbb{Q}\left[(S^1_{t_{n+1}}, S^2_{t_{n+1}}) \in R_{ij} \mid (S^1_{t_n}, S^2_{t_n}) = (a_k, b_l)\right] \\
&= \int_{x_{k,i}}^{x_{k,i+1}} \int_{y_{l,j}}^{y_{l,j+1}} \phi(z_1, z_2, \rho) dz_1 dz_2 \\
&= \Phi(x_{k,i+1}, y_{l,j+1}, \rho) - \Phi(x_{k,i}, y_{l,j+1}, \rho) - \Phi(x_{k,i+1}, y_{l,j}, \rho) + \Phi(x_{k,i}, y_{l,j}, \rho),
\end{aligned}$$

where

$$\begin{aligned}
x_{k,i} &= \left(\log\left(a_i/a_k\right) - \left(r - \delta_1 - \sigma_1^2/2\right)\Delta t\right) / \left(\sigma_1 \Delta t\right) \\
y_{l,j} &= \left(\log\left(b_j/b_l\right) - \left(r - \delta_2 - \sigma_2^2/2\right)\Delta t\right) / \left(\sigma_2 \Delta t\right).
\end{aligned}$$

The functions $\phi(\cdot,\cdot,\rho)$ and $\Phi(\cdot,\cdot,\rho)$ are respectively the density and the cumulative density functions of the bivariate standard normal distribution with correlation coefficient ρ. The function $\Phi(\cdot,\cdot,\rho)$ is computed according to Genz (2004).

$$\begin{aligned}
T_{klij}^{10} &= \mathbb{E}^*\left[S^1_{t_{n+1}} \mathbb{I}\left((S^1_{t_{n+1}}, S^2_{t_{n+1}}) \in R_{ij}\right) \mid (S^1_{t_n}, S^2_{t_n}) = (a_k, b_l)\right] \\
&= \int_{x_{k,i}}^{x_{k,i+1}} \int_{y_{l,j}}^{y_{l,j+1}} a_k \exp\left((r - d_1 - \sigma_1^2/2)\Delta t + \sigma_1\sqrt{\Delta t} z_1\right) \times \phi(z_1, z_2, \rho) dz_1 dz_2 \\
&= w_k^1 \int_{x_{k,i} - \sigma_1\Delta t}^{x_{k,i+1} - \sigma_1\Delta t} \int_{y_{l,j} - \rho\sigma_1\Delta t}^{y_{l,j+1} - \rho\sigma_1\Delta t} \phi(u_1, u_2, \rho) du_1 du_2 \\
&= w_k^1 \big[\Phi(x_{k,i+1} - \sigma_1\Delta t, y_{l,j+1} - \rho\sigma_1\Delta t, \rho) - \\
&\quad \Phi(x_{k,i} - \sigma_1\Delta t, y_{l,j+1} - \rho\sigma_1\Delta t, \rho) - \\
&\quad \Phi(x_{k,i+1} - \sigma_1\Delta t, y_{l,j} - \rho\sigma_1\Delta t, \rho) + \\
&\quad \Phi(x_{k,i} - \sigma_1\Delta t, y_{l,j} - \rho\sigma_1\Delta t, \rho)\big],
\end{aligned}$$

where $w_k^1 = a_k \exp\left((r - d_1 - \sigma_1^2/2)\Delta t + \sigma_1^2 \Delta t^2/2\right)$.

$$\begin{aligned}
T_{klij}^{01} &= \mathbb{E}^*\left[S^2_{t_{n+1}} \mathbb{I}\left((S^1_{t_{n+1}}, S^2_{t_{n+1}}) \in R_{ij}\right) \mid (S^1_{t_n}, S^2_{t_n}) = (a_k, b_l)\right] \\
&= \int_{x_{k,i}}^{x_{k,i+1}} \int_{y_{l,j}}^{y_{l,j+1}} b_l \exp\left((r - d_2 - \sigma_2^2/2)\Delta t + \sigma_2\sqrt{\Delta t} z_2\right) \times \phi(z_1, z_2, \rho) dz_1 dz_2 \\
&= w_l^2 \int_{x_{k,i} - \rho\sigma_2\Delta t}^{x_{k,i+1} - \rho\sigma_2\Delta t} \int_{y_{l,j} - \sigma_2\Delta t}^{y_{l,j+1} - \sigma_2\Delta t} \phi(u_1, u_2, \rho) du_1 du_2 \\
&= w_l^2 \big[\Phi(x_{k,i+1} - \rho\sigma_2\Delta t, y_{l,j+1} - \sigma_2\Delta t, \rho) - \\
&\quad \Phi(x_{k,i} - \rho\sigma_2\Delta t, y_{l,j+1} - \sigma_2\Delta t, \rho) - \\
&\quad \Phi(x_{k,i+1} - \rho\sigma_2\Delta t, y_{l,j} - \sigma_2\Delta t, \rho) + \\
&\quad \Phi(x_{k,i} - \rho\sigma_2\Delta t, y_{l,j} - \sigma_2\Delta t, \rho)\big],
\end{aligned}$$

where $w_l^2 = b_l \exp\left((r - d_2 - \sigma_2^2/2)\Delta t + \sigma_2^2 \Delta t^2/2\right)$.

$$
\begin{aligned}
T^{11}_{klij} &= \mathbb{E}^*\left[S^1_{t_{n+1}} S^2_{t_{n+1}} \mathbb{I}\left((S^1_{t_{n+1}}, S^2_{t_{n+1}}) \in R_{ij}\right) \mid (S^1_{t_n}, S^2_{t_n}) = (a_k, b_l)\right] \\
&= \int_{x_{k,i}}^{x_{k,i+1}} \int_{y_{l,j}}^{y_{l,j+1}} a_k \exp\left((r - d_1 - \sigma_1^2/2)\Delta t + \sigma_1\sqrt{\Delta t} z_1\right) \times \\
&\quad b_l \exp\left((r - d_2 - \sigma_2^2/2)\Delta t + \sigma_2\sqrt{\Delta t} z_2\right) \phi(z_1, z_2, \rho) dz_1 dz_2 \\
&= w^1_k w^2_l \exp\left(\rho\sigma_1\sigma_2\Delta t^2\right) \times \\
&\quad \int_{x_{k,i} - (\sigma_1 + \rho\sigma_2)\Delta t}^{x_{k,i+1} - (\sigma_1 + \rho\sigma_2)\Delta t} \int_{y_{l,j} - (\rho\sigma_1 + \sigma_2)\Delta t}^{y_{l,j+1} - (\rho\sigma_1 + \sigma_2)\Delta t} \phi(u_1, u_2, \rho) du_1 du_2 \\
&= w^1_k w^2_l \exp\left(\rho\sigma_1\sigma_2\Delta t^2\right) \times \\
&\quad \big[\Phi(x_{k,i+1} - (\sigma_1 + \rho\sigma_2)\Delta t, y_{l,j+1} - (\rho\sigma_1 + \sigma_2)\Delta t, \rho) - \\
&\quad \Phi(x_{k,i} - (\sigma_1 + \rho\sigma_2)\Delta t, y_{l,j+1} - (\rho\sigma_1 + \sigma_2)\Delta t, \rho) - \\
&\quad \Phi(x_{k,i+1} - (\sigma_1 + \rho\sigma_2)\Delta, y_{l,j} - (\rho\sigma_1 + \sigma_2)\Delta t, \rho) + \\
&\quad \Phi(x_{k,i} - (\sigma_1 + \rho\sigma_2)\Delta, y_{l,j} - (\rho\sigma_1 + \sigma_2)\Delta t, \rho)\big].
\end{aligned}
$$

References

Andersen, L. and Broadie, M. (2004). Primal-dual simulation algorithm for pricing multidimensional American options. Management Science, 50(9):1222–1234.

Bally, V. and Pages, G. (2003a). Error analysis of the optimal quantization algorithm for obstacle problems. Stochastic Processes and their Applications, 106(1):1–40.

Bally, V. and Pages, G. (2003b). A quantization algorithm for solving multidimensional discrete-time optimal stopping problems. Bernoulli, 9(6):1003–1049.

Bally, V. and Printems, J. (2005). A quantization tree method for pricing and hedging multidimensional American options. Mathematical Finance, 15(1):119–168.

Barraquand, J. and Martineau, D. (1995). Numerical valuation of high dimensional multivariate American securities. Journal of Financial and Quantitative Analysis, 30(3):383–405.

Ben-Ameur, H., Breton, M., and L'Ecuyer, P. (2002). A dynamic programming procedure for pricing American-style asian options. Management Science, 48(5):625–643.

Ben-Ameur, H., Chérif, R., and Rémillard, B. (2016). American-style options in jump-diffusion models: estimation and evaluation. Quantitative Finance. In Press.

Berridge, S. and Schumacher, J. (2008). An irregular grid approach for pricing high-dimensional american options. Journal of Computational and Applied Mathematics, 222(1):94–111.

Boyle, P., Broadie, M., and Glasserman, P. (1997). Monte Carlo methods for security pricing. Journal of Economic Dynamics and Control, 21(8):1267–1321.

Boyle, P. P. (1988). A lattice framework for option pricing with two state variables. Journal of Financial and Quantitative Analysis, 23(1):1–12.

Boyle, P. P., Evnine, J., and Gibbs, S. (1989). Numerical evaluation of multivariate contingent claims. Review of Financial Studies, 2(2):241–250.

Broadie, M. and Glasserman, P. (1997). Pricing American-style securities using simulation. Journal of Economic Dynamics and Control, 21(8):1323–1352.

Carriere, J. F. (1996). Valuation of the early-exercise price for options using simulations and nonparametric regression. Insurance: Mathematics and Economics, 19(1):19–30.

Del Moral, P., Rémillard, B., and Rubenthaler, S. (2012). Monte Carlo approximations of American options that preserve monotonicity and convexity. In Numerical Methods in Finance, pages 115–143. Springer.

Detemple, J., Feng, S., and Tian, W. (2003). The valuation of American call options on the minimum of two dividend-paying assets. Annals of Applied Probability, 13(3):953–983.

Dockendorf, J. and Paxson, D. A. (2015). Sequential real rainbow options. The European Journal of Finance, 21(10-11):867–892.

Genz, A. (2004). Numerical computation of rectangular bivariate and trivariate normal and t probabilities. Statistics and Computing, 14(3):251–260.

Hartley, P. (2000). Pricing a multi-asset American option. Working paper. University of Bath.

Haugh, M. B. and Kogan, L.(2004). Pricing American options: a duality approach. Operations Research, 52(2):258–270.

Jin, X., Li, X., Tan, H. H., and Wu, Z. (2013). A computationally efficient state-space partitioning approach to pricing high-dimensional American options via dimension reduction. European Journal of Operational Research, 231(2):362–370.

Jin, X., Tan, H. H., and Sun, J. (2007). A state-space partitioning method for pricing high-dimensional American-style options. Mathematical Finance, 17(3):399–426.

Johnson, H. (1987). Options on the maximum or the minimum of several assets. Journal of Financial and Quantitative Analysis, 22(3):277–283.

Kamrad, B. and Ritchken, P. (1991). Multinomial approximating models for options with k state variables. Management Science, 37(12):1640–1652.

Longstaff, F. A. and Schwartz, E. S. (2001). Valuing American options by simulation: a simple least-squares approach. The Review of Financial Studies, 14(1):113–147.

Raymar, S. B. and Zwecher, M. J. (1997). A Monte Carlo valuation of American call options on the maximum of several stocks. Journal of Derivatives, 5(1):7–23.

Rogers, L. C. G. (2002). Monte Carlo valuation of American options. Mathematical Finance, 12(3):271–286.

Stulz, R. (1982). Options on the minimum or the maximum of two risky assets: analysis and applications. Journal of Financial Economics, 10(2):161–185.

Tilley, J. A. (1993). Valuing American options in a path simulation model. Transactions of the Society of Actuaries, 45:83–104.

Tsitsiklis, J. N. and Van Roy, B. (1999). Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives. IEEE Transactions on Automatic Control, 44(10):1840–1851.