<http://www.diva-portal.org>

Postprint

This is the accepted version of a paper published in *BIT Numerical Mathematics*. This paper has been peer-reviewed but does not include the final publisher proof-corrections or journal pagination.

Citation for the original published paper (version of record):

Shcherbakov, V. (2016)

Radial basis function partition of unity operator splitting method for pricing multi-asset American options.

BIT Numerical Mathematics, 56: 1401-1423

<http://dx.doi.org/10.1007/s10543-016-0616-y>

Access to the published version may require subscription.

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-284299>

Radial basis function partition of unity operator splitting method for pricing multi-asset American options

Victor Shcherbakov

Department of Information Technology, Uppsala University, Sweden

Abstract

The operator splitting method in combination with finite differences has been shown to be an efficient approach for pricing American options numerically. Here, the operator splitting formulation is extended to the radial basis function partition of unity method. An approach that has previously often been used together with radial basis function methods to deal with the free boundary arising in American option pricing is to solve a penalised version of the Black–Scholes equation. It is shown that the operator splitting technique outperforms the penalty approach when used with the radial basis function partition of unity method. Numerical experiments are performed for one, two and three underlying assets. The advantage of the operator splitting technique grows with the number of dimensions.

1 Introduction

The Benchmark Project in Option Pricing (BENCHOP) [20] was recently launched. In this project established methods for option pricing were tested and compared on different benchmark problems. The metric of the comparison was the time to achieve a relative error tolerance of 10^{-4} .

This work is inspired by the BENCHOP project, where among other methods, the radial basis function partition of unity method (RBF–PUM) was tested. For many of the benchmark problems RBF–PUM was competitive with finite difference (FD) methods, but for the American option problems it performed significantly worse. In the project RBF–PUM employed the penalty method [13] for handling the free exercise boundary, while the FD methods used the operator splitting (OS) formulation [9].

The penalty approach allows one to remove the early exercise boundary and solve the problem on a fixed domain by adding a (usually non-linear) penalty function, whose size is adjusted by a penalty parameter ϵ . A property of this approach is that the error introduced by the penalty is proportional to the penalty parameter size. In order to avoid solving a non-linear problem in the BENCHOP we used an implicit-explicit numerical scheme, where the penalty function was explicitly treated. This type of scheme was proposed in [13] stating that the condition imposed by the explicit treatment of the penalty would be rather mild. Nevertheless, the time step is strongly dependent on and proportional to the penalty parameter size. Thus, to reach the given tolerance (10^{-4}) we had to use a very small penalty parameter, which imposed a severe condition on the time step size to maintain stability. This resulted in long execution times.

The operator splitting approach works on the linear complementarity problem (LCP), where a fully implicit scheme can be easily applied. Consequently, it does not enforce any constraint on the time step size. We believe that the use of the OS technique made all FD methods noticeably faster than RBF-PUM for the American option pricing problem.

Thus, the primary goal of this work is to extend the operator splitting approach to RBF-PUM, which to the author's knowledge has not yet been done. Then, we compare the results of the new method to those obtained by the penalty approach in the BENCHOP project. Furthermore, for the penalty approach we also implement a fully implicit scheme as well as a fully explicit scheme to compare against the previously implemented implicit-explicit scheme. In the fully implicit case, we employ Newton's method to handle the non-linearity. It turns out that Newton's method on average requires only a few iterations to converge, regardless of the problem's dimension. This provides an overall efficient scheme.

The layout of this paper is structured as follows. In Section 2, we introduce the Black-Scholes equation for the American multi-asset put option. In Section 3, we discuss the operator splitting method and the penalty method for American options. In Section 4, we give an overview of RBF-PUM, set up the discrete formulation and briefly discuss implementation details. In Section 5, we test the three penalty schemes and the operator splitting scheme for the one-dimensional case, and then we proceed to higher dimensional cases with the best candidates. In Section 6, we give some conclusions.

2 The Black-Scholes equation

Under the Black-Scholes market assumptions [1] we consider the d -dimensional Black-Scholes equation, whose solution defines the price of an American put option written on d assets

$$\frac{\partial V}{\partial t} - \mathcal{L}V = 0, \quad \vec{x} \in \Omega, t \in (0, T], \quad (2.1)$$

$$\mathcal{L} = \frac{1}{2} \sum_{i,j=1}^d \sigma_i \sigma_j \rho_{ij} x_i x_j \frac{\partial^2}{\partial x_i \partial x_j} + \sum_{i=1}^d r x_i \frac{\partial}{\partial x_i} - r, \quad (2.2)$$

where V is the value of the option, $\vec{x} = (x_1, \dots, x_d)$ defines the spot prices of the d underlying assets, d is the number of assets in the portfolio, σ_i is the volatility of asset i , ρ_{ij} is the correlation between assets i and j , r is the risk-free interest rate, t is the backward time, i.e., time to maturity, and T is the maturity time of the option. Dividends could also be taken into account, but since we consider only American put options, the dividends are not necessary to allow for the possibility of early exercise, as for example in the case of call options (for more details see [8]). The domain Ω is a subdomain of $\mathbb{R}_+^d$, which falls inside the early exercise boundary $\Gamma(\vec{x}, t)$.

The payoff function for the put option is given by:

$$\Phi(\vec{x}) = \max \left(K - \sum_{i=1}^d \theta_i x_i, 0 \right), \quad (2.3)$$

where K is the strike price and θ_i is the weight of the i -th asset in the portfolio. The payoff used defines the option value at the time of maturity. Since the problem is stated in backward time, the payoff provides the initial condition for equation (2.1)

$$V(\vec{x}, 0) = \Phi(\vec{x}), \quad \vec{x} \in \Omega. \quad (2.4)$$

In addition, equation (2.1) is subject to the following boundary condition at the near-field boundary

$$V(0, t) = K, \quad t \in [0, T], \quad (2.5)$$

and at the free boundary

$$V(\vec{x}, t) = \Phi(\vec{x}), \quad \vec{x} \in \Gamma(\vec{x}, t), t \in [0, T], \quad (2.6)$$

$$\frac{\partial V}{\partial x_i}(\vec{x}, t) = -\theta_i, \quad \vec{x} \in \Gamma(\vec{x}, t), t \in [0, T]. \quad (2.7)$$

Outside the free boundary the solution is given by $V(\vec{x}, t) = \Phi(\vec{x})$.

Remark 1. The near-field boundary can be represented by the point $\vec{x} = 0$. At the boundaries $v_i = \{\vec{x} | \vec{x} \in \Omega, \vec{x} \neq 0, x_i = 0\}$, the spatial operator (2.2) is degenerate and reduces to a $(d - 1)$ -dimensional operator. Fichera [4] derived general conditions for when to impose boundary conditions for parabolic PDEs with degenerate diffusion operators. In the case of the Black–Scholes operator, boundary conditions should not be imposed at v_i unless required for numerical purposes.

The problem (2.1)–(2.7) can also be reformulated as a linear complementarity problem

$$(V - \Phi) \geq 0, \quad (2.8)$$

$$\left(\frac{\partial V}{\partial t} - \mathcal{L}V \right) \geq 0, \quad (2.9)$$

$$\left(\frac{\partial V}{\partial t} - \mathcal{L}V \right) (V - \Phi) = 0. \quad (2.10)$$

The formulations (2.1)–(2.7) and (2.8)–(2.10) are equivalent and have the same solution.

3 Methodology

In this section we introduce two approaches for solving the Black–Scholes equation for multi-asset American put options.

3.1 The penalty method

We use the type of penalty function for American put options that was introduced by Nielsen et al. [13] and has been used later by several authors together with finite differences [14] as well as radial basis functions [3, 17, 18].

The penalty function is given by

$$P(V) = \frac{e r K}{V + e - q}, \quad (3.1)$$

where e is a penalty parameter, which has to be chosen sufficiently small, and $q(\vec{x})$ is a barrier function equal to the non-zero part of the payoff,

$$q(\vec{x}) = K - \sum_{i=1}^d \theta_i x_i. \quad (3.2)$$

The form of the penalty term has also been generalised for application to American call options in [18].

Adding the penalty term to the Black–Scholes equation allows us to convert the free boundary problem to a fixed domain problem. The error introduced by the penalty is expected to be $\mathcal{O}(e)$. This was discussed in [13] for finite difference methods, and also experimentally shown for RBF methods in [18], although the analytically derived bound in the same work was $\mathcal{O}(e^{1/2})$. The modified equation takes the form

$$\frac{\partial V}{\partial t} - \mathcal{L}V - P(V) = 0, \quad \vec{x} \in \tilde{\Omega}, t \in [0, T]. \quad (3.3)$$

Equation (3.3) is defined over the entire $\mathbb{R}_+^d$, but to enable numerical simulations we use a truncated domain $\tilde{\Omega}$, which is truncated at $x_i = x_\infty$ in each direction. We impose the following boundary conditions

$$V(0, t) = K, \quad t \in [0, T], \quad (3.4)$$

$$V(\vec{x}, t) = 0, \quad \vec{x} \in \gamma(\vec{x}), t \in [0, T], \quad (3.5)$$

where $\gamma(\vec{x}) = \{\vec{x} \mid \vec{x} \in \tilde{\Omega}, x_i = x_\infty, i = 1, \dots, d\}$ is called the far-field boundary. The initial condition becomes

$$V(\vec{x}, 0) = \Phi(\vec{x}), \quad \vec{x} \in \tilde{\Omega}. \quad (3.6)$$

A typical choice for x_∞ falls between $2dK$ and $4dK$, depending on the problem parameters, where d is the problem dimension and K is the strike price.

3.2 The operator splitting method

In contrast to the penalty approach, the operator splitting method [9] deals with the Black–Scholes equation in the linear complementarity form (2.8)–(2.10). The main idea of this method is to split the time integration step into two fractional steps, where the partial differential equation is integrated together with an auxiliary variable λ in the first step, and then the solution and λ are updated to fulfill the American option constraint in the second step. Such a splitting scheme introduces an error, which turns out to be sufficiently small to not influence the order of the time integration scheme [10]. It is also possible in each time step to ignore the free boundary and then apply the American constraint explicitly [7], but this approach reduces the order of the time integration, while the splitting method uses the auxiliary variable λ to improve the accuracy. More details about the operator splitting method for pricing American options can be found in [10] and references therein.

In the splitting based approach the original linear complementarity problem (2.8)–(2.10) is reformulated to account for the auxiliary variable λ as

$$\lambda = \frac{\partial V}{\partial t} - \mathcal{L}V, \quad \vec{x} > 0, t \in [0, T], \quad (3.7)$$

$$(V - q)\lambda = 0, \quad \vec{x} > 0, t \in [0, T], \quad (3.8)$$

$$(V - q) \geq 0, \quad \lambda \geq 0, \quad \vec{x} > 0, t \in [0, T], \quad (3.9)$$

$$V = \Phi, \quad \vec{x} > 0, t = 0, \quad (3.10)$$

$$V = K, \quad \vec{x} = 0, t \in [0, T], \quad (3.11)$$

$$V = 0, \quad \vec{x} \in \gamma(\vec{x}), t \in [0, T]. \quad (3.12)$$

4 Time discretisation and space approximation

The space approximation is performed by the radial basis function partition of unity method. This is a localised modification of the global RBF method, where local approximants are constructed in overlapping subdomains $\{\Omega_i\}_{i=1}^M$, which form an open cover of the entire computational domain $\tilde{\Omega}$. The local approximants are combined by the partition of unity functions $\{w^i\}_{i=1}^M$ subordinated to the open cover $\{\Omega_i\}_{i=1}^M$ into a global approximation function. The locality of the partition of unity technique allows for a significant reduction of the computational effort compared with the global method.

We divide our computational domain into M overlapping spherical patches $\{\Omega_i\}_{i=1}^M$ with radii R_i and N_i computational nodes in each. Thus, in every patch we can construct a local time-dependent RBF approximation over the nodes $\vec{x}_j$, $j = 1, \dots, N_i$,

$$v^i(t, \vec{x}) = \sum_{j=1}^{N_i} \alpha_j^i(t) \phi(\varepsilon \|\vec{x} - \vec{x}_j\|) = \sum_{j=1}^{N_i} \alpha_j^i(t) \phi_j(\vec{x}), \quad (4.1)$$

where $\phi_j(\vec{x})$ is a radial basis function centered at node $\vec{x}_j$, $\alpha_j^i(t)$ are coefficients to be determined and ε is the shape parameter, which defines the width of the basis functions. Common choices of basis functions can be found in Table 1.

Table 1: Commonly used radial basis functions.

RBF	$\phi(r)$
Multiquadric	$(1 + (\varepsilon r)^2)^{1/2}$
Inverse multiquadric	$(1 + (\varepsilon r)^2)^{-1/2}$
Inverse quadratic	$(1 + (\varepsilon r)^2)^{-1}$
Gaussian	$e^{-(\varepsilon r)^2}$
Thin plate spline	$r^2 \ln(r)$

The local approximants are combined into the global approximant by

$$v(t, \vec{x}) = \sum_{i=1}^M w^i(\vec{x}) v^i(t, \vec{x}) = \sum_{i=1}^M w^i(\vec{x}) \sum_{j=1}^{N_i} \alpha_j^i(t) \phi_j(\vec{x}), \quad (4.2)$$

where w^i is a partition of unity function compactly supported on Ω_i and satisfying the property

$$\sum_{i=1}^M w^i(\vec{x}) = 1. \quad (4.3)$$

The partition of unity can be constructed by Shepard's method [19]

$$w^i(\vec{x}) = \frac{\psi^i(\vec{x})}{\sum_{k=1}^M \psi^k(\vec{x})}, \quad (4.4)$$

where $\psi^i(\vec{x})$ is a compactly supported function on Ω_i . We use C^2 compactly supported Wendland functions [21], since the Black–Scholes problem (2.1)–(2.7) requires at least C^2 continuity of the partition of unity weights

$$\psi(r) = \begin{cases} (1-r)^4(4r+1), & \text{if } 0 \leq r \leq 1 \\ 0, & \text{if } r > 1. \end{cases} \quad (4.5)$$

For a more detailed discussion on radial basis function partition of unity methods see [18].

4.1 The penalty method

The total number of node points in the entire $\tilde{\Omega}$ is $N = N_I + N_B$, where N_I is the number of interior nodes and N_B is the number of boundary nodes. In order to determine the values of $\alpha_j^i(t)$ we use collocation at the N node points. For the interior nodes $\vec{x}_j$, $j = 1, \dots, N_I$ we use equation (3.3), and for the near-field and far-field boundary points $\vec{x}_j$, $j = N_I + 1, \dots, N$ we use equations (3.4) and (3.5), respectively. Thus, $\vec{v}_I(t) = [v(t, \vec{x}_1), \dots, v(t, \vec{x}_{N_I})]^T$ and $\vec{v}_B(t) = [v(t, \vec{x}_{N_I+1}), \dots, v(t, \vec{x}_N)]^T$. By evaluating (4.2) at the node points, we get

$$\begin{bmatrix} \vec{v}_I(t) \\ \vec{v}_B(t) \end{bmatrix} = \begin{bmatrix} A_{II} & A_{BI} \\ A_{IB} & A_{BB} \end{bmatrix} \begin{bmatrix} \vec{\alpha}_I(t) \\ \vec{\alpha}_B(t) \end{bmatrix}, \quad (4.6)$$

where the global coefficient matrix A consists of elements $a_{jk} = w^i(\vec{x}_k) \phi(\varepsilon \|\vec{x}_j - \vec{x}_k\|)$ and has a nearly block-diagonal structure, where the i -th block corresponds to the i -th patch. The blocks overlap due to the overlapping structure of the open cover $\{\Omega_i\}_{i=1}^M$. Consequently,

$$\mathcal{L} \vec{v}_I(t) = \begin{bmatrix} L_{II} & L_{IB} \end{bmatrix} \begin{bmatrix} \vec{\alpha}_I(t) \\ \vec{\alpha}_B(t) \end{bmatrix} = \begin{bmatrix} L_{II} & L_{IB} \end{bmatrix} A^{-1} \begin{bmatrix} \vec{v}_I(t) \\ \vec{v}_B(t) \end{bmatrix} \quad (4.7)$$

$$= \begin{bmatrix} C_{II} & C_{IB} \end{bmatrix} \begin{bmatrix} \vec{v}_I(t) \\ \vec{v}_B(t) \end{bmatrix}, \quad (4.8)$$

where the matrix L consists of elements $l_{jk} = \mathcal{L}(w^i(\vec{x}_k) \phi(\varepsilon \|\vec{x}_j - \vec{x}_k\|))$, $j = 1, \dots, N_I$ and $k = 1, \dots, N$. Note that A^{-1} exists, because for standard choices of basis functions A is non-singular.

For time marching we select the unconditionally stable second order backward differentiation formula (BDF-2) [6, p. 401]. The BDF-2 scheme involves three time levels. To initiate the method, often the BDF-1 scheme (Euler backward) is used for the first time step. Therefore two different matrices need to be factorised. In order to avoid this we use the BDF-2 scheme as described in [11].

We split the time interval $[0, T]$ into N_t non-uniform steps of length $k^n = t^{n+1} - t^n$, $n = 1, \dots, N_t$, then

$$(E - \beta_0^n \mathcal{L}) \vec{v}_I^1 = \vec{v}_I^0, \quad (4.9)$$

$$(E - \beta_0^n \mathcal{L}) \vec{v}_I^n = \beta_1^n \vec{v}_I^{n-1} - \beta_2^n \vec{v}_I^{n-2} + \beta_0^n P(\vec{v}_I^{n-1}), \quad n = 2, \dots, N_t, \quad (4.10)$$

where E is an identity matrix of the proper size and

$$\beta_0^n = k^n \frac{1 + \omega_n}{1 + 2\omega_n}, \quad \beta_1^n = \frac{(1 + \omega_n)^2}{1 + 2\omega_n}, \quad \beta_2^n = \frac{\omega_n^2}{1 + 2\omega_n}, \quad (4.11)$$

and $\omega_n = k^n / k^{n-1}$, $n = 2, \dots, N_t$. In [11] it is shown how the time steps can be chosen in such a way that $\beta_0^n \equiv \beta_0$. Then the coefficient matrix is the same in all time steps and only one matrix factorisation is needed.

For the boundary node points we enforce the following condition

$$\vec{v}_B^n = \vec{f}_B^n, \quad n = 1, \dots, N_t, \quad (4.12)$$

where

$$\vec{f}_B = [f(t, \vec{x}_{N_I+1}), \dots, f(t, \vec{x}_N)]^T, \quad (4.13)$$

and

$$f(t, \vec{x}) = \begin{cases} K, & \text{if } \vec{x} = 0 \\ 0, & \text{if } \vec{x} \in \gamma(\vec{x}). \end{cases} \quad (4.14)$$

This leads to a linear system, which needs to be solved at each time step

$$\begin{bmatrix} E_I - \beta_0 C_{II} & -\beta_0 C_{IB} \\ 0 & E_B \end{bmatrix} \begin{bmatrix} \vec{v}_I^n \\ \vec{v}_B^n \end{bmatrix} = \begin{bmatrix} \vec{f}_I^n \\ \vec{f}_B^n \end{bmatrix}, \quad (4.15)$$

where

$$\vec{f}_I^n = \beta_1^n \vec{v}_I^{n-1} - \beta_2^n \vec{v}_I^{n-2} + \beta_0^n P(\vec{v}_I^{n-1}). \quad (4.16)$$

In short notations we write it as

$$B \vec{v}^n = \vec{f}^n. \quad (4.17)$$

Note that to avoid solving a non-linear system the penalty term is treated explicitly here. It imposes some condition on the time step size. Nielsen shows [13] that in case of finite differences for put options the condition is

$$\Delta t \leq \frac{e}{rK}, \quad (4.18)$$

where e is the penalty parameter, r is the risk-free interest rate, K is the strike price and in our case $\Delta t = \max\{k^n\}_{n=1}^{N_t}$. Experiments for RBFs seem to give nearly the same bound [18].

We also implement an entirely implicit BDF-2 scheme, where the penalty is defined at the new time level. In this case instead of the linear system (4.15) we obtain the following non-linear system

$$\begin{bmatrix} E_I - \beta_0 C_{II} & -\beta_0 C_{IB} \\ 0 & E_B \end{bmatrix} \begin{bmatrix} \vec{v}_I^n \\ \vec{v}_B^n \end{bmatrix} - \beta_0^n \begin{bmatrix} P(\vec{v}_I^n) \\ 0 \end{bmatrix} = \begin{bmatrix} \vec{f}_I^n \\ \vec{f}_B^n \end{bmatrix}, \quad (4.19)$$

where

$$\vec{f}_I^n = \beta_1^n \vec{v}_I^{n-1} - \beta_2^n \vec{v}_I^{n-2}, \quad (4.20)$$

and $\vec{f}_B^n$ remains as in (4.13). In order to solve the arising non-linear system we employ the Newton method.

To complete the picture we also consider a second-order explicit scheme. For this purpose we select Heun's scheme. This results in the following linear system

$$\vec{w}_I^n = \vec{v}_I^{n-1} + \Delta t [C \vec{v}^{n-1} + P(\vec{v}_I^{n-1})], \quad (4.21)$$

$$\vec{w}_B^n = \vec{v}_B^{n-1}, \quad (4.22)$$

$$\vec{v}_I^n = \vec{v}_I^{n-1} + 0.5 \Delta t [C \vec{w}^n + P(\vec{w}_I^n) + C \vec{v}^{n-1} + P(\vec{v}_I^{n-1})], \quad (4.23)$$

$$\vec{v}_B^n = \vec{f}_B^n, \quad (4.24)$$

where $\vec{w}^n$ is an auxiliary variable, $\vec{f}_B^n$ as in (4.13), and

$$C = \begin{bmatrix} C_{II} & C_{IB} \end{bmatrix}. \quad (4.25)$$

Apart from the constraint on the time step size enforced by the penalty, there will be an additional constraint that comes from the explicit time discretisation.

A test of the above three schemes will tell us which method is better suited for the penalised Black–Scholes equation. The point of view, which has been common in the literature, is that the semi-implicit scheme is an optimal choice, because it exempts us from solving a non-linear system, while the enforced condition on the time step is not as severe as it could be if one used an entirely explicit discretisation scheme.

We will see that it turns out that the implicit penalty method is the most efficient out of the three modifications. Therefore we decide to present an algorithm only for the implicit version of the penalty method (see Algorithm 1).

Algorithm 1 Evaluate an American option by the implicit penalty method

```
1: Initialise  $\mathbf{v} = \Phi$  and  $\mathbf{y} = \Phi$ 
2: Set tolerance  $\tau$ 
3: for each time step from 2 to  $N_t$  do
4:   while  $\text{err} \geq \tau$  do Newton iterations on the nonlinear system (4.19)
5:     Evaluate the penalty function at  $\mathbf{y}$   $P = erK/(\mathbf{y} + e - q)$ 
6:     Define the Jacobian  $J = B - \beta_0 P/(\mathbf{y} + e - q)$ 
7:     Solve  $\mathbf{v} = \mathbf{y} - J^{-1}(B\mathbf{y} + \beta_0 P + f)$ 
8:     Update  $\mathbf{y} = \mathbf{v}$ 
9:   end while
10: end for
```

4.2 The operator splitting method

As in the previous section we use collocation to determine the coefficients $\alpha_j^i(t)$ and the BDF-2 scheme for time evolution. The main steps of the discretisation for the operator splitting method can be repeated in the same manner as for the penalty method.

The operator splitting method has to be performed in two fractional steps at each time step. At the first fractional step the linear system is solved and then, at the second fractional step, the solution is corrected to satisfy the American option constraint $V \geq \Phi$ and the auxiliary variable $\tilde{\lambda}^n$ is updated. As it was shown in [10] the order of the splitting is the same as the order of the time integration scheme, i.e., second order. The splitting does not reduce the order of accuracy since the American option constraint is treated in a separate fractional time step. Thus, the splitting scheme allows for performing the time integration without any essential loss of accuracy. Thereby, problem (3.7)–(3.12) in the discrete case takes form

$$B\tilde{\mathbf{v}}^n = \vec{g}^n + \beta_0^n \tilde{\lambda}^{n-1} \quad (4.26)$$

$$\vec{\mathbf{v}}^n - \tilde{\mathbf{v}}^n = \beta_0^n (\tilde{\lambda}^n - \tilde{\lambda}^{n-1}) \quad (4.27)$$

$$(\vec{\mathbf{v}}^n - \vec{q})\tilde{\lambda}^n = 0, \quad (4.28)$$

$$(\vec{\mathbf{v}}^n - \vec{q}) \geq 0, \quad \tilde{\lambda}^n \geq 0, \quad (4.29)$$

where $n = 1, \dots, N_t$, B is the same matrix as in equation (4.17), $\vec{g}^n = [\vec{g}_I^n, \vec{g}_B^n]^T$ with

$$\vec{g}_I^n = \beta_1^n \vec{v}_I^{n-1} - \beta_2^n \vec{v}_I^{n-2} \quad (4.30)$$

and $\vec{g}_B^n = \vec{f}_B^n$. The initial value for the auxiliary variable is $\tilde{\lambda}^0 = 0$.

Below we present Algorithm 2 for the OS method for the American option problem, which highlights the main implementation steps.

5 Numerical experiments

In this section we compute American put option values V , and their first and second spatial derivatives, in financial terms known as the Greeks Δ and Γ , with the above mentioned methods. We evaluate option values and Greeks at three predefined asset values, which correspond to “in-the-money”, “at-the-money” and “out-of-the-money” cases. All solutions for option prices are benchmarked against accurate reference solutions V_r , which are obtained by Fourier methods. The relative deviation of the approximated values V from the reference values V_r is restricted to

Algorithm 2 Evaluate an American option by the operator splitting method

- 1: Initialise $\tilde{\mathbf{v}} = \Phi$ and $\lambda = 0$
 - 2: Factorise $B = LU$
 - 3: **for** each time step from 2 to N_t **do**
 - 4: Solve for $\tilde{\mathbf{v}}$ from (4.27) using L and U factors $\tilde{\mathbf{v}} = U^{-1}L^{-1}(g + \beta_0\lambda)$
 - 5: Update $\lambda_1 = \lambda$, $\lambda = 0$
 - 6: Solve for $\mathbf{v}$ from (4.28) $\mathbf{v} = \tilde{\mathbf{v}} + \beta_0(\lambda - \lambda_1)$
 - 7: Where the solution falls below the payoff, project it to the payoff $\mathbf{v} = \Phi$
 - 8: Update $\lambda = \lambda_1 + (\mathbf{v} - \tilde{\mathbf{v}})/\beta_0$
 - 9: **end for**
-

be less than 10^{-4} . By practitioners, this precision is considered sufficient for financial purposes. Therefore, if the methods succeed in meeting this restriction, then the main criteria, which determines the advantage of one method over another, would be the *shorter computational time*.

In other words, all numerical experiments are performed with the aim *to get the relative error δ below the tolerance 10^{-4} in the shortest time*,

$$\delta = \left\| \frac{V - V_r}{V_r} \right\|_{\infty}. \quad (5.1)$$

All corresponding errors and run times are recorded and presented in the tables. To eliminate randomness, *each code was run 10 times*, then the two longest and the two shortest times were removed and the mean over the remaining six times was measured. Error profiles for the option values evaluated on the cube $[80, 120]^d$, where d denotes the dimension, are also presented in the figures below. We see the domain $[80, 120]^d$ as the most relevant from both the financial viewpoint and the mathematical viewpoint, because for the chosen parameter sets it includes the strike and the free boundary.

All reference solutions are obtained by using highly accurate versions of Fourier methods. The FGL method [12] in the single-asset case, and the COS method [16] in the double-asset case. For the RBF-PUM experiments we select the multiquadric basis function $\phi = \sqrt{1 + \varepsilon^2 r^2}$.

The experiments are carried out on a machine with an Intel Core i7 processor, 2.3 GHz and 16 GB RAM. All codes are implemented serially in MATLAB R2014b.

To facilitate an easy reproduction of our results, we publish parameter values, that were used in the experiments, in Table 2. Parameters for different methods within the same experiment may differ, because they are fine-tuned to give an accurate solution within “optimal” time.

Remark 2. We do not guarantee that the chosen parameters and domain sizes are the true optimal ones and nothing more optimal can be found. We use quotation marks to denote “minimal”, “optimal”, “shortest” in terms of what we could find within a reasonable amount of fine-tuning.

5.1 The non-uniform grid

In all our experiments we exploit the flexibility of RBF-PUM to easily handle non-uniform grids. The non-uniform grid that we used has grid points clustered around the strike price K . The node coordinates in each direction i are defined by

$$x_{i,j} = K + l \sinh(\xi_j), \quad j = 1, \dots, \sqrt[d]{N}, \quad (5.2)$$

Table 2: Parameters that were used in the experiments. PEX denotes the explicit penalty method, PIMEX denotes the implicit-explicit (IMEX) penalty method, PIM denotes the implicit penalty method, and OS denotes the operator splitting method.

Method		1D	2D	3D
PEX	Domain	$\vec{x} \in [0, 3K]$		
	Shape param	$\varepsilon = 0.17$		
	Pen param	$e = 10^{-4}$		
	Num parts	$M = 12$		
PIMEX	Domain	$\vec{x} \in [0, 3K]$		
	Shape param	$\varepsilon = 0.2$		
	Pen param	$e = 10^{-4}$		
	Num parts	$M = 35$		
PIM	Domain	$\vec{x} \in [0.5K, 2K]$	$\vec{x} \in [0, 4K]^2$	
	Shape param	$\varepsilon = 0.2$	$\varepsilon = 0.1$	
	Pen param	$e = 10^{-5}$	$e = 10^{-6}$	
	Num parts	$M = 5$	$M = 144$	
	Newton tol	$\tau = 10^{-4}$	$\tau = 10^{-4}$	
OS	Domain	$\vec{x} \in [0.5K, 2K]$	$\vec{x} \in [0, 4K]^2$	$\vec{x} \in [0, 6K]^3$
	Shape param	$\varepsilon = 0.6$	$\varepsilon = 0.1$	$\varepsilon = 0.1$
	Num parts	$M = 3$	$M = 144$	$M = 512$

where $\xi_j \in [\xi_*, \xi^*]$ are equidistant points, and l is the parameter that defines the amount of clustering. Requiring that the nodes $x_{i,j}$ should fall into the interval $[0, x_\infty]$ we can find ξ_* and ξ^*

$$\xi_* = \sinh^{-1}(-K/l), \quad (5.3)$$

$$\xi^* = \sinh^{-1}((x_\infty - K)/l). \quad (5.4)$$

The amount of clustering $l = 0.4$ for all the experiments. We choose equal size partitions to enable the construction of a higher order approximant in the strike region, which is critical from the financial point of view.

In Figure 1 we can see the non-uniform grid and partitioning that were used in the two-dimensional experiments for the OS method. The use of the partition of unity technique allows for a significant sparsification of the linear system. For example, with 144 partitions only 5.6% of the elements remain. For the numerical experiments we permute the matrix elements according to the sparse reverse Cuthill-McKee ordering [2].

Remark 3. If a global RBF method were to be used then the system would be dense with all 100% of the elements non-zero.

5.2 The single-asset case

We evaluate the solution at three points $x = [90; 100; 110]$, which correspond to “in-the-money”, “at-the-money” and “out-of-the-money” cases. The strike price $K = 100$, the volatility $\sigma = 0.15$, the risk-free interest rate $r = 0.03$, and the time of maturity $T = 1$ year.

The first column of Table 3 contains names of the methods that were used. The FGL (Fourier–Gauss–Laguerre) method [12] is the reference solution. The second column displays

Figure 1: Left: The non-uniform grid with 5776 computational nodes (76 per dimension) that was used in the two-dimensional OS experiments. The partition of unity is performed using 144 circular patches of equal size. Right: The RBF-PUM coefficient matrix that was obtained on this grid. It contains only 5.6% non-zero elements.

Table 3: The American put option reference values and the relative error $(V - V_r)/V_r$ in the approximations for one underlying asset. The grid sizes and the execution times for the operator splitting method and for the three versions of the penalty method are also presented. The * denotes the reference solution.

Method	$N \times N_t$	Time (s)	Values		
			$x = 90$	$x = 100$	$x = 110$
FGL*	-	-	10.726487	4.820608	1.828208
PEX	232×14000	1.5421	0.000048	0.000017	-0.000055
PIMEX	604×6500	0.4977	0.000068	0.000084	0.000097
PIM	74×75	0.0378	0.000056	0.000031	0.000069
OS	87×480	0.0249	0.000021	-0.000098	-0.000061

the “minimal” grid size that allows to get the relative error down to 10^{-4} . The third column displays the execution times of the four methods in seconds, and the last three columns show the reference option values for the indicated spot prices, and the relative differences $(V - V_r)/V_r$ of the numerical solutions from the reference.

We can see that among the penalty methods the implicit version is preferable. Its run time is at least 13 times shorter than any of the other versions. A key factor that makes PIM faster than PEX and PIMEX is that it requires fewer grid points and time steps to achieve the given tolerance. This is a consequence of the implicit interpretation of the penalty function, which allows one to remove any dependence of the time step on the penalty parameter. Furthermore, for PIM only a few Newton iterations per time step are needed and the overall computational time is not severely affected.

Remark 4. It is worthwhile to notice that the number of grid points is different for the three

penalty implementations. This is mostly due to the size of the penalty parameter combined with the focus on the time efficiency of each implementation. Let us consider PIM. This method has no constraint on the time step implied by the penalty size. Therefore we can select $e = 10^{-5}$, that is, the error introduced by the penalty approach will be an order of magnitude lower than the required tolerance. It simply means that we need 74 space nodes and 75 time step to have an accurate approximation. On the other hand, the efficiency of the PIMEX implementation suffers from the penalty parameter size. The smaller the parameter the smaller time steps we have to take. In the experiment for PIMEX we used $e = 10^{-4}$ and it led to 6500 time steps and 604 space points to suppress the error introduced by the penalty, which roughly of the same size as the tolerance. If in this case we chose $e = 10^{-5}$ we would need around 74 space points (as in the PIM case), but then the constraint on the time step size would become even more severe and we would require around 150000 steps. This would negatively affect the overall computational time. A similar argument applies to the PEX case.

However, the operator splitting method performs 1.5 times faster than the fastest of the penalty modifications. If RBF-PUM had employed the operator splitting method, instead of the IMEX penalty method, in the BENCHOP project, it would have been significantly more competitive to finite difference methods.

The RBF-PUM IMEX penalty implementation is the same one as was used for the American put option in the BENCHOP project [20], where it was compared with the FD operator splitting method within the same scope on the same machine (let us call it “the cluster”). The run time on “the cluster” for the RBF-PUM IMEX penalty method was 3.6 seconds and for the FD OS method it was on average (out of three different modifications) 0.059 seconds. On the machine which is used for the current experiments (let us call it “the laptop”) the run time for the RBF-PUM IMEX penalty method is 0.4977 seconds. That is, “the laptop” is about *seven* times faster than “the cluster”. The number seven was also verified by running other codes, which are available on the BENCHOP project website, on “the laptop” and comparing the obtained run times with the values presented in the BENCHOP paper. Thus, using the transitive relation we can conclude that the RBF-PUM OS implementation, which takes 0.0249 seconds on “the laptop”, would be seven times slower on “the cluster” and, therefore, take 0.1743 seconds. If we compare this number with the FD OS method, it will be just *three* times slower, instead of 60 times slower as with the RBF-PUM IMEX method. But even this three times difference will be diminished in higher dimensions, because already for two-dimensional problems RBF-PUM performed better than any of the FD methods [20].

Figure 2 displays the error profiles for the given four methods. We can see that towards the boundaries the errors are increasing. This is caused by the use of the adapted grid in combination with equal size partitions. The outer partitions contain fewer computational nodes, consequently the local approximants are less accurate.

Remark 5. Note that the OS error vanishes to the left of the free boundary, since the solution in that part of the domain is set to be equal to the payoff. The same behaviour can be observed in the other error plots for the OS method. In the case of the penalty methods, the penalised Black–Scholes equation has to be solved in the entire domain leading to a nonzero error to the left of the free boundary.

5.3 The double-asset case

Similarly, we evaluate the solution at three points $x = [90, 100; 100, 100; 100, 110]$, which correspond to “in-the-money”, “at-the-money” and “out-of-the-money” cases. The notation $x = [90, 100]$ denotes that the value of the first asset $x_1 = 90$ and the value of the second asset

Figure 2: The single-asset case. Relative error profiles of the operator splitting method and the three versions of the penalty method. The magenta dotted line denotes the strike. The dark blue dotted line denotes the free boundary location at the final time $\Gamma(\vec{x}, T)$.

$x_2 = 100$. The strike price $K = 100$, the volatility $\sigma_1 = \sigma_2 = 0.15$, the correlation between the assets $\rho = 0.5$, the risk-free interest rate $r = 0.03$, and the time of maturity $T = 1$ year.

Table 4: The American put option reference values and the relative error $(V - V_r)/V_r$ in the approximations for two underlying asset. The grid sizes and the execution times for the operator splitting method and for the implicit penalty method are also presented. The * denotes the reference solution.

Method	$\sqrt{N} \times N_t$	Time (s)	Values		
			$x = [90, 100]$	$x = [100, 100]$	$x = [100, 110]$
COS*	-	-	6.649395	4.051099	2.325955
PIM	73×22	62.9980	0.000026	0.000028	0.000045
OS	76×160	14.6647	0.000055	0.000010	-0.000023

For the two-dimensional experiments we keep only one version of the penalty method—the implicit version—because it performed better for the one-dimensional problem. In Table 4, we can see the reference option values obtained by the COS method [16] (based on a cosine expansion) and the relative difference $(V - V_r)/V_r$ between the approximated values and the reference values, together with the “minimal” grid sizes, which allow for getting the error down to 10^{-4} at the three specified points, and the execution times in seconds. In this test, the advantage of the operator splitting method over the penalty method becomes even more evident. The OS method is *four* times more efficient than PIM.

In Figure 3 we plot contours of the error in the option values in logarithmic scale. We see that the error is larger in the lower left and in the upper right corners. This depends on the lower accuracy of the local approximants in the partitions covering those areas, because they contain fewer computational node points. The OS error vanishes to the left of the free boundary.

Figure 3: The double-asset case. Left: The relative error contour in logarithmic scale for the implicit penalty method. Right: The relative error contour in logarithmic scale for the operator splitting method. The black dotted line denotes the strike. The blue dotted line schematically denotes the free boundary location at the final time $\Gamma(\vec{x}, T)$.

5.4 The multi-asset case

We do not possess a reference solution for the three-dimensional problem, therefore we just display the price surface of an American option written on three assets (see Figure 4), and option values at three points $x = [90, 100, 90; 100, 100, 100; 110, 100, 110]$, that correspond to ‘in-the-money’, ‘at-the-money’ and ‘out-of-the-money’ cases (see Table 5). The strike price $K = 100$, the volatility $\sigma_1 = \sigma_2 = \sigma_3 = 0.15$, the correlation between the assets $\rho_{12} = \rho_{13} = \rho_{23} = 0.5$, the risk-free interest rate $r = 0.03$, and the time of maturity $T = 1$ year.

The calculations are only performed for the operator splitting method, because the execution time of the implicit penalty method becomes unreasonably long even with only a few grid nodes per dimension. Hence, it makes the use of the implicit penalty method computationally unfeasible, while the operator splitting method allows computations on moderately large numbers of grid nodes per dimension (≈ 40) within reasonable time. Based on the experience from the one- and two-dimensional problems, this number of nodes would allow for calculating the price up to the third decimal digit, i.e., sufficiently accurately.

Table 5: Option values of an American put option written on three assets obtained by the operator splitting method.

Method	Values		
	$x = [90, 100, 90]$	$x = [100, 100, 100]$	$x = [110, 100, 110]$
OS	7.536933	3.727650	1.673915

Figure 4: Left: Visualisation of the three-dimensional option price surface. Right: Option values evaluated on the cube $[80, 120]^3$. The bars denote option values, and the axes denote values of the three underlying assets.

5.5 Convergence

To validate that the methods are numerically well-behaved we study the error convergence of the OS and PIM methods in the one- and two-dimensional cases with respect to the number of grid points. In one case the error is taken as the maximum of the relative errors at the three points indicated above (see Figure 5 Centre), in the other case the error is taken as the root mean square error over the cube $[80, 120]^d$, which in our tests includes both the strike and the free boundary (see Figure 5 Left). We also consider the execution time with respect to the number of computational nodes per dimension (see Figure 5 Right). The shape parameter and the number of partitions are kept fixed (see Table 2 for the values) while the convergence is being tested.

All the methods locally exhibit high algebraic convergence rates, when the error is measured at the three selected points, which is expected for RBF based methods [17]. However, a measurement at three points does not generate a norm, therefore we also look at the root mean square error over the region $[80, 120]^d$. Since the solution of the American option pricing problem has a discontinuity of the second derivative at the free boundary position, optimal convergence rates of high order methods cannot be expected. If the problem was smooth, we could expect exponential convergence globally, but in our case V is only two times weakly differentiable, i.e., $V \in W_p^2(\tilde{\Omega})$. Embedding theorems proven in [15] state

$$\|V\|_{L_q(\tilde{\Omega})} \leq Ch^{2-d\left(\frac{1}{p}-\frac{1}{q}\right)} |V|_{W_p^2(\tilde{\Omega})}, \quad (5.5)$$

where h is a measure of the node distance, C is some constant, $1 \leq p < \infty$, $1 \leq q \leq \infty$, and d is the dimension. Globally we measure the error in the L_2 -norm, therefore, the convergence can be at most of the order 2.5 for one-dimensional problems and at most of the order 3 for two-dimensional problems. However, for the one-dimensional case we observe a superconvergence for both solver strategies. In fact, if we recall Figure 2, we can notice that the errors at the

Figure 5: Left: The root mean square error convergence with respect to the number of points per dimension in logarithmic scale. The RMS error is measured over the region $[80, 120]^d$, which includes both the strike and the free boundary. Centre: The relative error convergence with respect to the number of points per dimension in logarithmic scale. The relative error is measured at the three selected points. Right: The execution time with respect to the number of points per dimension in logarithmic scale. $N_i = N$ in the one-dimensional case and $N_i = \sqrt{N}$ in the two-dimensional case.

“tail” $[110, 120]$ are larger than in the rest of the domain due to the non-uniform grid. It turns out that the error at the “tail” dominates the error in the rest of the domain and, when refined, converges faster. Since the “tail” belongs to the part of the domain where the Black–Scholes equation is solved and the solution does not experience discontinuities of the derivatives, a high convergence rate is possible. Figure 6 displays the convergence rates for the PIM and OS methods at the free boundary (although it is just one point in one-dimensional case) and over the “tail”. We can see that at the free boundary the convergence, as expected, is about the order 2.5, while over the “tail” the convergence rate is significantly higher, and the “tail error” dominates the “free boundary error” for all grid sizes. For the two-dimensional case the high local convergence rates drop down to expected values, when measured over the selected region. However, we observe some saturation at the error level about $7 \cdot 10^{-3}$, which is caused by inaccurate local approximations in the farthest from the strike partitions. This error does not get properly resolved when refined.

The implicit penalty method turns out to converge a little faster than the operator splitting method, although it is slower in terms of run time. The reason is that PIM performs Newton iterations to deal with the non-linearity in each time step. The execution time for both methods is almost independent of the number of points in the single-asset case, because the time spent on solving the system is negligible compared with the time for construction of local RBF matrices. In the double-asset case, solving the system of equations becomes the most time consuming operation, and we observe a nearly quadratic growth of the run time with the total number of grid points (4th order growth corresponding to $\sqrt{N}$) for both methods, which goes in line with the theory for sparse banded matrices. The total theoretical number of operations to solve the system in both cases would be $\mathcal{O}(Np^2)$, where p is the band width of the matrix [5]. In our case

the band width is roughly proportional to the square root of the total size, i.e., $p^2 \sim N$. That is, the total cost would be proportional to $\mathcal{O}(N^2)$.

Figure 6: Error convergence profiles for the single-asset problem. Left: The root mean square error convergence with respect to the number of points per dimension in logarithmic scale for the OS method. In blue the error measured at the free boundary point. In black the error measured over the “tail”, region $[110, 120]$. Right: The root mean square error convergence with respect to the number of points per dimension in logarithmic scale for PIM. In red the error measured at the free boundary point. In black the error measured over the “tail”, region $[110, 120]$.

5.6 Greeks

The Greeks represent the sensitivities of the solution to the option pricing problem with respect to different variables and parameters. They are used in particular for hedging portfolios. In this section we examine the sensitivities Δ and Γ , which are defined as follows

$$\Delta = \frac{\partial V}{\partial \vec{x}} \quad \text{and} \quad \Gamma = \frac{\partial^2 V}{\partial \vec{x}^2}. \quad (5.6)$$

In the case of RBF based methods, the Greeks can be evaluated directly from the solution to the Black–Scholes equation at a very small computational cost. However, we know that the American option price has a discontinuity in the second derivative at the free boundary. Hence, obtaining accurate results for the Greeks can be a challenge, especially calculations of the Γ .

In Figure 7 we plot the absolute errors in the Δ and Γ sensitivities for the single-asset American options obtained by the OS and PIM implementations. The reference solution is obtained by the FGL method. Both methods give quite accurate solutions, although, as expected, there is some increase in the error towards the free boundary location due to the discontinuity of the second derivative of the option price. Values of the Greeks at the three selected points can be found in Table 6.

In Figure 8 we plot the absolute errors in the Δ and Γ sensitivities for the double-asset problem. The Greeks get reasonably resolved over the whole domain of interest, especially near

Figure 7: Top: Absolute error in the Δ Greek obtained by the PIM and OS methods. Bottom: Absolute error in the Γ Greek obtained by the PIM and OS methods. The magenta dotted line denotes the strike. The dark blue dotted line denotes the free boundary location at the final time $\Gamma(\vec{x}, T)$.

Table 6: The Greeks for the American put option on one underlying asset. The Δ_r and Γ_r are the values obtained by the FGL method. For the approximation methods PIM and OS the differences in the values $\Delta - \Delta_r$ and $\Gamma - \Gamma_r$ are presented.

Method	Greek	Values		
		$x = 90$	$x = 100$	$x = 110$
FGL*	Δ_r	-0.766760	-0.427163	-0.192335
	Γ_r	0.036996	0.029551	0.017327
PIM	$\Delta - \Delta_r$	-0.001749	0.000234	0.000400
	$\Gamma - \Gamma_r$	-0.002753	0.000112	-0.000012
OS	$\Delta - \Delta_r$	0.001423	-0.000527	0.000208
	$\Gamma - \Gamma_r$	0.000023	0.003411	-0.000011

the strike region for both evaluation methods. However, the precision near the optimal exercise boundary is poor due to the discontinuity in the second derivative. This part of the domain can be better resolved if required, but it will lead to an extra computational effort. Values of the Greeks at the three selected points can be found in Table 7.

In summary, both solver strategies turn out to be equally good alternatives for computation of the primary sensitivities Δ and Γ . Since the Greeks are obtained at a low additional computational cost, the run times for obtaining the solution plus the Greeks will be similar to the values presented in Tables 3 and 4. That is, the OS method will be significantly faster than PIM and should be preferred.

Table 7: The Greeks for the American put option on two underlying asset. The Δ_r and Γ_r are the values obtained by the COS method. For the approximation methods PIM and OS the differences in the values $\Delta - \Delta_r$ and $\Gamma - \Gamma_r$ are presented.

Method	Greek	Values		
		$x = [90, 100]$	$x = [100, 100]$	$x = [100, 110]$
COS*	Δ_r	-0.306132	-0.212623	-0.137449
	Γ_r	0.009540	0.009006	0.005861
PIM	$\Delta - \Delta_r$	-0.003495	-0.000263	0.000932
	$\Gamma - \Gamma_r$	0.001029	-0.000139	0.000778
OS	$\Delta - \Delta_r$	-0.002634	-0.000286	0.000924
	$\Gamma - \Gamma_r$	0.001069	-0.000171	0.000786

Figure 8: Top Left: The log of the absolute error in the Δ Greek obtained by the OS method. Top Right: The log of the absolute error in the Δ Greek obtained by the PIM. Bottom Left: The log of the absolute error in the Γ Greek obtained by the OS method. Bottom Right: The log of the absolute error in the Γ Greek obtained by the PIM. The black dotted line denotes the strike. The blue dotted line denotes the free boundary location at the final time $\Gamma(\vec{x}, T)$.

6 Conclusion

The operator splitting approach for pricing American options has been extended to the radial basis function partition of unity method. It has been shown suitable for multi-asset options. The penalty approach has been implemented in three versions: implicit, implicit-explicit and explicit. Previously the implicit-explicit formulation has been commonly used with radial basis functions [3, 17, 18], because it allowed to avoid solving a non-linear problem, while enforcing a “rather mild” constraint on the time step size, which still led to unnecessary extra calculations in the time integration. This work shows that, if the partition of unity technique is employed, the implicit version should be preferred. The Newton method requires only a few iterations to converge regardless of the problem dimension. Thus, it allows to avoid restriction of the time step by the penalty size and preclude unnecessary calculations.

The operator splitting method has been compared with its penalty counterparts, where we could observe a significant advantage of the OS method in terms of computational time. In three dimensions and higher, only the OS method is practically useful, because the execution times for the penalty methods become unreasonably long.

We have also studied the convergence of the OS and the PIM methods in one and two dimensions. The methods are consistent with the change of grid sizes and locally exhibit high algebraic convergence rates, that is characteristic for localised RBF methods. However, if the errors are measured over a region, which includes the optimal exercise boundary, then the convergence rates decrease due to an insufficient smoothness of the solution.

We can say that if RBF-PUM had been used in combination with the operator splitting approach in the BENCHOP project, the results of RBF-PUM would have been comparable to the FD methods. We also assume that it will be more beneficial to use the operator splitting method with RBF-PUM rather than with FD methods for pricing options written on several assets, because radial basis function methods are better suited for higher-dimensional problems, due to higher order accuracy and ease of implementation. It also has to be taken into account that the high order accuracy of RBF-PUM reduces the memory requirements in multiple dimensions, which is problematic for FD methods due to the curse of dimensionality.

Finally, we should mention that the Δ and Γ Greeks can be obtained by both PIM and OS implementations at an insignificant additional computational effort. Both solver strategies are equally successful in computing the Greeks. However, the OS method is faster and should be preferred over PIM.

Acknowledgements

The author would like to thank Marjon Ruijter (Rabobank, Utrecht) for the provided reference solution for the two-dimensional problem, Elisabeth Larsson (Uppsala University) for fruitful discussions, and Slobodan Milovanović (Uppsala University) for help with visualising results. The author also thanks the anonymous peer reviewers for valuable comments, which definitely improved this manuscript.

References

- [1] Black, F., Scholes, M.: The pricing of options and corporate liabilities. *J. Polit. Econ.* **81**(3), 637–654 (1973). DOI 10.1086/260062
- [2] Cuthill, E., McKee, J.: Reducing the bandwidth of sparse symmetric matrices. In: *Proceedings of the 1969 24th National Conference, ACM '69*, pp. 157–172. ACM, New York, NY, USA (1969). DOI 10.1145/800195.805928
- [3] Fasshauer, G.E., Khaliq, A.Q.M., Voss, D.A.: Using mesh free approximation for multi-asset American option problems. *J. Chinese Inst. Engrs.* **27**(4), 563–571 (2004)
- [4] Fichera, G.: Sulle equazioni differenziali lineari ellittico-paraboliche del secondo ordine. *Atti Accad. Naz. Lincei. Mem. Cl. Sci. Fis. Mat. Nat. Sez. I. VIII, Ser. 5* pp. 3–30 (1956)
- [5] Golub, G.H., Van Loan, C.F.: *Matrix Computations*, fourth edn. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD (2013)
- [6] Hairer, E., Nørsett, S., Wanner, G.: *Solving Ordinary Differential Equations I. Nonstiff problems*, second edn. Springer-Verlag, Berlin (2000). DOI 10.1007/978-3-540-78862-1
- [7] Hon, Y.C.: A quasi-radial basis functions method for American options pricing. *Comput. Math. Appl.* **43**(3-5), 513–524 (2002). DOI 10.1016/S0898-1221(01)00302-9
- [8] Hull, J.: *Options, Futures and Other Derivatives*. Pearson Prentice Hall, Upper Saddle River, New Jersey (2009)
- [9] Ikonen, S., Toivanen, J.: Operator splitting methods for American option pricing. *Appl. Math. Lett.* **17**(7), 809–814 (2004). DOI 10.1016/j.aml.2004.06.010
- [10] Ikonen, S., Toivanen, J.: Operator splitting methods for pricing American options under stochastic volatility. *Numer. Math.* **113**(2), 299–324 (2009). DOI 10.1007/s00211-009-0227-5
- [11] Larsson, E., Åhlander, K., Hall, A.: Multi-dimensional option pricing using radial basis functions and the generalized Fourier transform. *J. Comput. Appl. Math.* **222**(1), 175–192 (2008). DOI 10.1016/j.cam.2007.10.039
- [12] Lindström, E., Ströjby, J., Brodén, M., Wiktorsson, M., Holst, J.: Sequential calibration of options. *Comput. Statist. Data Anal.* **52**(6), 2877–2891 (2008). DOI 10.1016/j.csda.2007.08.009
- [13] Nielsen, B.F., Skavhaug, O., Tveito, A.: Penalty and front-fixing methods for the numerical solution of American option problems. *J. Comput. Finance* **5**(4), 69–97 (2002)
- [14] Nielsen, B.F., Skavhaug, O., Tveito, A.: Penalty methods for the numerical solution of American multi-asset option problems. *J. Comput. Appl. Math.* **222**(1), 3–16 (2008). DOI 10.1016/j.cam.2007.10.041
- [15] Rieger, C., Zwicknagl, B.: Sampling inequalities for infinitely smooth functions, with applications to interpolation and machine learning. *Adv. Comput. Math.* **32**(1), 103–129 (2010). DOI 10.1007/s10444-008-9089-0
- [16] Ruijter, M.J., Oosterlee, C.W.: Two-dimensional Fourier cosine series expansion method for pricing financial options. *SIAM J. Sci. Comput.* **34**(5), B642–B671 (2012). DOI 10.1137/120862053
- [17] Safdari-Vaighani, A., Heryudono, A., Larsson, E.: A radial basis function partition of unity collocation method for convection-diffusion equations. *J. Sci. Comput.* **64**(2), 341–367 (2015). DOI 10.1007/s10915-014-9935-9
- [18] Shcherbakov, V., Larsson, E.: Radial basis function partition of unity methods for pricing vanilla basket options. *Comput. Math. Appl.* **71**(1), 185–200 (2016). DOI 10.1016/j.camwa.2015.11.007
- [19] Shepard, D.: A two-dimensional interpolation function for irregularly-spaced data. In: *Proceedings of the 1968 23rd ACM National Conference, ACM '68*, pp. 517–524. ACM, New York, NY, USA (1968). DOI 10.1145/800186.810616
- [20] von Sydow, L., Höök, L.J., Lindström, E., Milovanović, S., Persson, J., Shcherbakov, V., Shpolyanskiy, Y., Samuel, S., Toivanen, J., Waldén, J., Wiktorsson, M., Levesley, J., Li, J., Oosterlee, C., Ruijter, M., Toropov, A., Zhao, Y.: BENCHOP — The BENCHmarking project in Option Pricing. *Int. J. Comput. Math.* **92**(12), 2361–2379 (2015). DOI 10.1080/00207160.2015.1072172
- [21] Wendland, H.: Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Adv. Comput. Math.* **4**(4), 389–396 (1995). DOI 10.1007/BF02123482