Pricing Financial Derivatives using Radial Basis Function generated Finite Differences with Polyharmonic Splines on Smoothly Varying Node Layouts

Slobodan Milovanović

Department of Information Technology
Uppsala University
Sweden

Abstract

In this paper, we study the benefits of using polyharmonic splines and node layouts with smoothly varying density for developing robust and efficient radial basis function generated finite difference (RBF-FD) methods for pricing of financial derivatives. We present a significantly improved RBF-FD scheme and successfully apply it to two types of multidimensional partial differential equations in finance: a two-asset European call basket option under the Black–Scholes–Merton model, and a European call option under the Heston model. We also show that the performance of the improved method is equally high when it comes to pricing American options. By studying convergence, computational performance, and conditioning of the discrete systems, we show the superiority of the introduced approaches over previously used versions of the RBF-FD method in financial applications.

Keywords: Pricing of Financial Derivatives; Radial Basis Function generated Finite Differences; Polyharmonic Splines; Node Placing.

1 Introduction

Pricing of financial derivatives is one of the core processes in financial markets. When it comes to popularity, options play one of the main roles on that stage of contingent assets, as they offer more freedom to their holders compared to other similar instruments. Unlike futures, options are contracts that grant the right,

but not the obligation to buy or sell an underlying asset at a set price on or before a certain date. Option pricing is often a necessary process for making investment decisions, managing risk and calibrating financial models. The prices of these financial instruments are computed by individuals and institutions across the world many times a day using a plethora of methods. Some of those derivatives may be priced analytically, e.g., European options under the assumptions of the famous Black–Scholes–Merton model [3, 30]. However, if we consider other types of options, such as American options or basket options, or perhaps we want to use different pricing models (e.g., a local stochastic volatility model) — in general, we are not able to derive analytical solutions. In those cases, we need to use different numerical methods to approximate the prices of such options. A diverse overview of the numerical schemes used for option pricing can be seen in the BENCHOP project [39, 40]. The results of that work illustrate how challenging pricing problems for different methods can be and how important it is to carefully choose and appropriately develop numerical methods in order to build efficient pricing tools.

In this paper, we focus on improvements and adaptations of the Radial Basis Function generated Finite Difference (RBF-FD) methods for pricing multi-asset options and options under multi-factor models. Both of those pricing problems can be formulated as time-dependent multidimensional partial differential equations (PDEs). As a high-order, mesh-free and sparse numerical method from the RBF family, together with the Radial Basis Function Partition of Unity (RBF-PU) method [35, 36, 33], RBF-FD shows strong potential when it comes to solving multidimensional PDEs. We develop on top of the previous results of using RBF-FD for financial engineering [31, 14, 24, 27, 23, 22], and use important recent advancement of the RBF-FD approximation in other disciplines [1, 8], to build stable, accurate, and fast solvers for multidimensional PDEs in finance. The main features of the developed solvers address the previous problems of choosing the RBF shape parameter by using polyharmonic splines (PHSs), and instabilities induced by high condition numbers of the differentiation matrices by using node layouts with smoothly varying density.

The remainder of this article is organized as follows. In Section 2, we formulate the RBF-FD method for option pricing problems and motivate using PHSs and node layouts with smoothly varying density. Then, in Section 3, we demonstrate the benefits of the introduced method on a two-asset European call basket option and American put basket option under the Black–Scholes–Merton model, and a European call option under the Heston model. Finally, in Section 4, we draw conclusions and recommend some future research directions.

2   Radial Basis Function generated Finite Difference Methods

The RBF-FD methods belong to the family of RBF methods. Using the RBF methods for approximating solutions of PDEs dates back to the beginning of the nineties in the previous century [26, 25]. Ever since, these methods have been used in different fields, including computational finance [6, 18, 32]. Although the classic RBF methods (also referred to as global RBF methods) possess some desirable properties such as high order convergence and mesh-free domain discretization, they are featured with dense system matrices which in many cases have very large condition numbers. To overcome these weaknesses, several localized RBF approaches with advanced features were introduced, among which RBF-FD [37, 42] and RBF-PU [41] are the most popular and still actively developed.

In order to apply the method, we observe option pricing problems on the truncated computational domain $\Omega \subset \mathbb{R}^d$ in the following PDE form

$$\frac{\partial}{\partial t} u(t, \underline{x}) + \mathcal{L}u(t, \underline{x}) = 0, \qquad \underline{x} \in \Omega, \tag{1}$$

$$\mathcal{B}u(t, \underline{x}) = f(t, \underline{x}), \quad \underline{x} \in \partial\Omega, \tag{2}$$

$$u(T, \underline{x}) = g(\underline{x}), \qquad \underline{x} \in \Omega, \tag{3}$$

where $u(t, \underline{x})$ is the option price; $\mathcal{L}$ is the differential operator of the model; $\mathcal{B}$ is the boundary differential operator that with $f(t, \underline{x})$ defines the boundary conditions for the pricing problem; $g(\underline{x})$ is the payoff function; $\underline{x}$ is the spatial variable representing underlying assets or stochastic factors and t is the time variable.

To construct an RBF-FD approximation, we scatter N nodes across the computational domain Ω. For each node $\underline{x}_j$, we define an array of nodes $\mathbf{x}_j$ consisting of $n_j - 1$ neighboring nodes and $\underline{x}_j$ itself, and consider it as a stencil of size n_j centered at $\underline{x}_j$. The differential operator $\mathcal{L}$ defined in (1) is approximated in every node $\underline{x}_j$ as

$$\mathcal{L}u(\underline{x}_j) \approx \sum_{i=1}^{n_j} w_j^i u_j^i \equiv \mathbf{w}_j u(\mathbf{x}_j), \quad j = 1, \dots, N, \tag{4}$$

where $u_j^i \equiv u(\underline{x}_j^i)$ and $\underline{x}_j^i$ is a locally indexed node in $\mathbf{x}_j$, while $\mathbf{w}_j$ is the array of differentiation weights for the stencil centered at $\underline{x}_j$. In the standard RBF-FD methods, the weights w_j^i are calculated by enforcing (4) to be exact for RBFs

centered at each of the nodes in $\mathbf{x}_j$ yielding

$$
\begin{bmatrix}
\phi(\|\underline{x}_j^1 - \underline{x}_j^1\|) & \cdots & \phi(\|\underline{x}_j^1 - \underline{x}_j^{n_j}\|) \\
\vdots & \ddots & \vdots \\
\phi(\|\underline{x}_j^{n_j} - \underline{x}_j^1\|) & \cdots & \phi(\|\underline{x}_j^{n_j} - \underline{x}_j^{n_j}\|)
\end{bmatrix}
\begin{bmatrix}
w_j^1 \\
\vdots \\
w_j^{n_j}
\end{bmatrix}
=
\begin{bmatrix}
\mathcal{L}\phi(\|\underline{x}_j - \underline{x}_j^1\|) \\
\vdots \\
\mathcal{L}\phi(\|\underline{x}_j - \underline{x}_j^{n_j}\|)
\end{bmatrix}.
\tag{5}
$$

In theory on RBF interpolation, it is known that (5) forms a nonsingular system of equations. Therefore, a unique set of weights can be computed for each node. We arrange those weights in a differentiation matrix L in order to build a discrete spatial operator that approximates $\mathcal{L}$. Since $n_j \ll N$, the resulting differentiation matrix is sparse.

2.1 Smoothly Varying Node Layouts

Although the RBF-FD methods are of a mesh-free nature, we observe in practice that the conditioning of the numerical scheme is highly sensitive to the choice of the node layout. Namely, if a discretized computational domain contains non-smooth changes in density of the node layout, it is very likely that the stencils constructed across those areas will have very large condition numbers, and therefore make the entire approximation unstable. Using Cartesian grids with RBF-FD is one way to be safe, but that is far from optimal when it comes to approximation accuracy and it severely limits the adaptive potentials of the method. Some effort has been made to build custom node layouts for pricing basket options in [31], but that approach is hardly generalizable to other problems.

For a successful implementation of RBF-FD methods, we need to be able to quickly generate node layouts with smoothly varying density. According to [11], current methods for node scattering can be seen as either *iterative* methods or *advancing front* methods. The most known example of the iterative type are minimal energy distributions, where a repelling force between the nodes is formulated such that it concentrates the nodes in the areas where increased accuracy is needed [9]. While these methods produce excellent node layouts, in many cases it is computationally inefficient to build node layouts through iterations. Advancing front methods, on the other hand, build node layouts starting from the boundaries until they fill up the domain, which is usually more efficient.

In this work, we focus on the node placing algorithm, introduced in [11]. The algorithm is of an advancing front type, with an addition of node repelling iterations which further improve node layout quality close to the boundaries. The first ingredients that are necessary for using the algorithm are the boundaries of the computational domain Ω and the radius function $R(\underline{x})$, which controls

density of the node layout on that domain. Next, we define a rectangle that is slightly larger than our computational domain, and we fill it up using an advancing front type basic node placing scheme that can be found in [11]. Now, the nodes in the rectangle are scattered according to the radius function $R(\underline{x})$. Then, we superpose the node set in the rectangle with the boundary nodes of our computational domain. At this step, we can use the opportunity to also place the nodes of special interest together with the boundary nodes, as these will stay in their positions in the final node layout. This is extremely useful when we are interested to know the solution at a particular point in the computational domain. Following that, we discard all nodes outside the domain boundary and also the ones that are inside the domain, but within a distance of $\frac{1}{2}R(\underline{x})$ from the boundary nodes. Finally, we run $a \in \mathbb{N}$ node repel steps on the nodes that are close to the boundary, i.e., $b \in \mathbb{N}$ nearest neighbors to each boundary node, in order to smooth out the irregularities in those areas. This is implemented by using a repel force proportional to r^{-3}, where $r \in \mathbb{R}$ presents the distances between the nodes. The force is set to act between the b neighboring nodes, and it is used in iterations to push the nodes towards the optima of the local potential wells.

One of the great advantages of such smoothly varying node layouts is that we can place the nodes exactly at the points where we want to know the option price without disturbing the smoothness of the layout. This is great for accurate option pricing for the given spot prices of the underlying assets. With traditional node layouts, we need to use interpolation in order to estimate option prices at the desired points, which introduces an additional error and computational load, or we have to disturb the node layout smoothness by placing nodes at the coordinates of interest.

2.2 Polyharmonic Splines

Many RBFs (e.g., Gaussian, multiquadric, inverse quadratic) were considered for approximating differential operators in the literature. Although such approximations are featured with great properties, the linear systems of equations that need to be solved in order to obtain the weights w_j^i are often ill-conditioned. Several past works [4, 12, 10, 29, 13, 7] addressed this problem by adding low-order polynomials together with RBFs into the presented interpolation. Moreover, the shape parameter, which is present in most of the RBFs, needs to be chosen carefully in order to have a stable approximation. The problem of choosing the shape parameter for Gaussian-based RBF-FD schemes is thoroughly examined for option pricing problems in [31], but still remains unsolved for general applications.

Nevertheless, recent developments [1, 8], show that the RBF-FD approxi-

mation can be greatly improved by using high order polynomials together with PHSs as RBFs in the interpolation. With that approach, it seems as if the polynomial degree takes the role of controlling the rate of convergence. This allows us to use piecewise smooth PHSs as RBFs without a shape parameter, since the approximation accuracy is no longer controlled by the smoothness of the RBFs. Still, the RBFs do contribute to reduction of approximation errors, and they are necessary in order to have both stable and accurate approximation. We define the PHS function in (6) and show some examples in Figure 1,

$$\phi(r) = \begin{cases} r^q, & q \in \{2k-1\}, \\ r^q \ln(r), & q \in \{2k\}, \end{cases} \tag{6}$$

where $k \in \mathbb{N}$. The results in [8] show that there is no significant difference between using odd and even degrees of PHSs in practical applications of RBF-FD. Consequently, we use odd degrees due to their slightly simpler form.
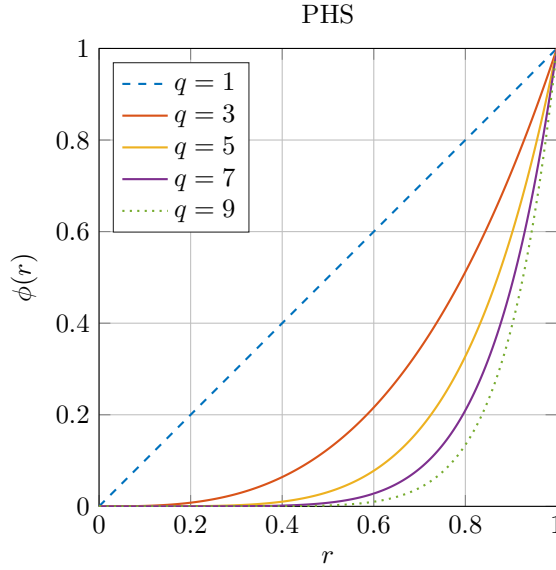

Figure 1: Polyharmonic splines of different odd degrees.

Taking everything into account, the linear system that we need to solve to obtain

the differentiation weights for each node in our problems is

$$\begin{bmatrix} A & P^T \\ P & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w}_j \\ \gamma_j \end{bmatrix} = \begin{bmatrix} \mathcal{L}\phi(\|\underline{x}_j - \underline{x}_j^1\|) \\ \vdots \\ \mathcal{L}\phi(\|\underline{x}_j - \underline{x}_j^{n_j}\|) \\ \mathcal{L}p_1(\underline{x}_j) \\ \vdots \\ \mathcal{L}p_{m_j}(\underline{x}_j) \end{bmatrix}, \tag{7}$$

where A is the RBF matrix and $\mathbf{w}_j$ is the array of differentiation weights, both shown on the left-hand side of (5); P is the matrix of size $m_j \times n_j$ that contains all monomials up to degree p (corresponding to m_j monomial terms) that are evaluated in each node $\underline{x}_j^i$ of the stencil $\mathbf{x}_j$ and $\mathbf{0}$ is a zero square matrix of size $m_j \times m_j$; γ_j is the array of dummy weights that should be discarded and $\{p_1, p_2, \ldots, p_{m_j}\}$ is the array of monomial functions indexed by their position relative to the total number of monomial terms m_j, such that it contains all the combinations of monomial terms up to degree p.

Compared to standard FD discretizations, where differential operators are approximated only on one-dimensional Cartesian grids, meaning that high-dimensional operators need to be discretized separately in each direction, in the RBF-FD approximations dimensionality does not make the problem more difficult. When it comes to the boundary nodes and the nodes that are close to the boundary, the nearest neighbor based stencils automatically form according to the shape of the boundary and require no special treatment for computing the differentiation weights. The only data that is required for approximation of differential operators are Euclidian distances between the nodes. This means that (5) represents a way to approximate a differential operator in any number of dimensions. Although the FD weights can be directly derived and the RBF-FD weights need to be obtained by solving a small linear system for each node, this task is perfectly parallelizable and that extra cost can be well justified by the desirable features of the method.

After the weights are computed and stored in the differentiation matrix, an approximation of (1) can be presented in the form of the following semi-discrete equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{u}(t) = L\mathbf{u}(t), \tag{8}$$

$$\mathbf{u}(T) = g(\mathbf{x}), \tag{9}$$

where $\mathbf{u}(t) \equiv u(t, \mathbf{x})$ is the semi-discrete numerical solution of the pricing equation, while $\mathbf{x}$ is the array of all nodes in the computational domain. To compute the option price $\mathbf{u}(t)$, we need to integrate (8) backwards in time.

2.3 Integration in Time

For the time discretization we use the second order backward differentiation formula (BDF2) [16, p. 401]. The BDF2 scheme involves three time levels. To initiate the method, the BDF1 (Euler backward) scheme is often used for the first time step. Thus, two different matrices would need to be factorized. In order to avoid this, we use BDF2 with BDF1 as described in [28], so that we get a single differentiation matrix.

We split the time interval $[0, T]$ into M non-uniform steps of length $\tau^l = t^{M-l} - t^{M-l+1}$, $l = 1, \dots, M$ and define the BDF2 weights as

$$\beta_0^l = \tau^l \frac{1 + \omega_l}{1 + 2\omega_l}, \quad \beta_1^l = \frac{(1 + \omega_l)^2}{1 + 2\omega_l}, \quad \beta_2^l = \frac{\omega_l^2}{1 + 2\omega_l}, \tag{10}$$

where $\omega_l = \tau^l / \tau^{l-1}$, $l = 2, \dots, M$. In [28] it is shown how the time steps can be chosen in such a way that $\beta_0^l \equiv \beta_0$. Therefore, the coefficient matrix is the same in all time steps and only one matrix factorization is needed.

Applying the BDF2 scheme to (8) we obtain a fully discretized system of equation

$$\underbrace{(E - \beta_0 L)}_{C} \mathbf{u}^l = \beta_1^l \mathbf{u}^{l-1} - \beta_2^l \mathbf{u}^{l-2} \tag{11}$$

where E is the identity matrix of the appropriate size.

To solve this system, we employ the iterative GMRES method with an incomplete LU factorization as the preconditioner.

3   Numerical Experiments

Here, we demonstrate the advantages of the improved RBF-FD method with PHSs on smoothly varying node layouts against the classic RBF-FD setups previously used for option pricing, e.g., in [31, 39]. We consider three pricing problems. We start with a two-dimensional European call option under the Black–Scholes–Merton model as a simple example. Then, we demonstrate that the method works just as well when used on a more challenging American put basket option under the same model. Finally, as an advanced case, we show the results for a European call option under the local stochastic volatility Heston model.

For all of the considered problems, we scale the operator $\mathcal{L}$ such that we can perform the RBF-FD approximation on a unit domain and then we rescale the result in order to obtain the actual option price. We consider three different node layouts.

- The first one is the equidistant Cartesian grid which we refer to as `cartesian`.

- The second layout is the adapted non-uniform node layout used in [31] and based on [21, 15], which we call `adapted`. Conceptually, on a one-dimensional unit domain, we construct this node layout from N equidistant nodes

$$z_i = \operatorname{arcsinh}\left(-\frac{\hat{K}}{H}\right) + (i-1)\Delta z,$$

$i = 1, \ldots, N$, where $\hat{K}$ is the scaled strike price position, H is the density parameter, and

$$\Delta z = \frac{1}{N}\left[\operatorname{arcsinh}\left(\frac{1-\hat{K}}{H}\right) - \operatorname{arcsinh}\left(-\frac{\hat{K}}{H}\right)\right].$$

Then, the final layout would consist of the nodes

$$x_i = \hat{K} + H \cdot \sinh(z_i).$$

In our experiments, we use this one-dimensional concept to construct appropriate node layouts with $H = 0.1$. We choose the parameter H empirically, based on several computational experiments.

- The third is the newly introduced smoothly varying node layout, constructed by a node placing algorithm from [11] which we present in Section 2.1 and denote here as `smooth`. We use

$$R(\mathbf{x}) = \frac{1}{\sqrt{N}}\left(\left(\frac{(\mathbf{x}^{(1)} - X_1)\cos(G) + (\mathbf{x}^{(2)} - X_2)\sin(G)}{P}\right)^2 \right. $$
$$\left. + \left(\frac{(\mathbf{x}^{(1)} - X_1)\sin(G) - (\mathbf{x}^{(2)} - X_2)\cos(G)}{Q}\right)^2 + 1\right),$$

as the radius function, where $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ are the components of the vector $\mathbf{x}$, while X_1, X_2, P, Q, and G are real parameters intended to control node scattering. As for the local node adjustment, we consider $b = 32$ boundary neighbors with $a = 4$ repelling iterations. We choose these parameters empirically.

By carefully choosing the density controlling parameters, we adapt the former two node layouts to have a higher node density around the discontinuity in the terminal condition, while maintaining desired conditioning and accuracy.

As RBFs, we use PHSs of degree $q = 5$ and augment them with monomials of up to degree $p = 4$. This should correspond to an RBF-FD method of fourth order. Nevertheless, since the terminal conditions of the equations that we are solving are not smooth (i.e., the first derivative of the payoff function is discontinuous at the strike price), we can only expect second order convergence. The

reason we are not using monomials of degree $p = 2$ is that the approximation accuracy and efficiency is still higher with $p = 4$, even though both approximations converge with order two. Since the problems are of dimension $D = 2$, this gives a polynomial space of size $m = \binom{p+D}{p} = 15$, which we use to set the size of the RBF-FD stencils to $n = 5m = 75$, according to the empirical guidelines from [1, 8].

To identify the nearest neighbors for the stencil construction in an efficient way, we employ the k-D tree algorithm [2].

For the plots demonstrating the computational performance, we use MAT-LAB implementations of the presented methods on a laptop equipped with a 2.3 GHz Intel Core i7 CPU and 16 GB of RAM. Moreover, the RBF-FD weights computation is performed in parallel using the parallel toolbox command `parfor` with 4 workers. We implement the time integration with the `nofill` setting for the incomplete LU factorization to produce the preconditioner for the GMRES solver. We set the `tol` parameter in GMRES to 10^{-8} for all the experiments. To speed up the convergence, we use the values from the previous time step as the initial value for the next time step. For all experiments, time is discretized into $M = 100$ steps. This is just enough to keep the time discretization error smaller than the spatial discretization error in all considered cases.

3.1 Multi-Asset Options

A multi-asset option that depends on D underlying risky assets $S_d(t)$, $d = 1, \ldots, D$ under the Black–Scholes–Merton model with an assumed risk free bond $B(t)$, follows the dynamics

$$
\begin{aligned}
\mathrm{d}B(t) &= rB(t)\,\mathrm{d}t, \\
\mathrm{d}S_1(t) &= \mu_1 S_1(t)\,\mathrm{d}t + \sigma_1 S_1(t)\,\mathrm{d}W_1(t), \\
\mathrm{d}S_2(t) &= \mu_2 S_2(t)\,\mathrm{d}t + \sigma_2 S_2(t)\,\mathrm{d}W_2(t), \\
&\vdots \\
\mathrm{d}S_D(t) &= \mu_D S_D(t)\,\mathrm{d}t + \sigma_D S_D(t)\,\mathrm{d}W_D(t),
\end{aligned}
\tag{12}
$$

where t is time, r is the risk free interest rate, μ_d are the drifts and σ_d are the volatilities of S_d, and W_d are the Wiener processes. The Wiener processes are correlated such that $\mathrm{d}W_i(t)\,\mathrm{d}W_j(t) = \rho_{i,j}\,\mathrm{d}t$. In this multidimensional setting, an option with payoff function $g(S_1(T), \ldots, S_D(T))$, where T is the time of maturity of the option, can be priced under the risk-neutral measure $\mathbb{Q}$ as

$$
u(S_1(t), \ldots, S_D(t), t) = e^{-r(T-t)} \mathbb{E}_t^{\mathbb{Q}}[g(S_1(T), \ldots, S_D(T))].
\tag{13}
$$

Traditionally, Monte Carlo methods are used to estimate the expected value in (13) for multi-asset options. In order to apply RBF-FD, we look at the

corresponding multidimensional Black–Scholes–Merton equation

$$\frac{\partial u}{\partial t} + \mathcal{L}u = 0, \tag{14}$$

$$u(s_1, s_2, \ldots, s_D, T) = g(s_1, s_2, \ldots, s_D), \tag{15}$$

where

$$\mathcal{L}u \equiv r \sum_i^D s_i \frac{\partial u}{\partial s_i} + \frac{1}{2} \sum_{i,j}^D \rho_{i,j} \sigma_i \sigma_j s_i s_j \frac{\partial^2 u}{\partial s_i \partial s_j} - ru, \tag{16}$$

and for an arithmetic call option

$$g(s_1, s_2, \ldots, s_D) = \max\left(\frac{1}{D}\sum_{d=1}^D s_d - K, \ 0\right),$$

while for an arithmetic put option

$$g(s_1, s_2, \ldots, s_D) = \max\left(K - \frac{1}{D}\sum_{d=1}^D s_d, \ 0\right),$$

with K as the strike price.

3.1.1 European Call Basket Option under The Black–Scholes–Merton Model

We perform an experiment with a two-asset European call basket option where $r = 0.03$, $\sigma_1 = \sigma_2 = 0.15$, and $\rho = 0.5$, with $T = 1$ and $K = 100$. The error is measured at three points

$$\mathbf{x}_{\text{BS}} = \begin{pmatrix} 90 & 90 \\ 100 & 100 \\ 110 & 110 \end{pmatrix},$$

which are close to the strike price K. The relative location of these points on a scaled domain can be seen in the right most plot of Figure 2 denoted with yellow pentagons. For the `cartesian` and `adapted` node layouts, we employ cubic interpolation to approximate the values at the desired points. We then use the maximum error from those three points $\Delta u_{\max}$ to measure convergence.

We set up the computational domain such that the far field boundary at each axis is $s_d^{\max} = 8K$ and use the advantages of a mesh-free framework to eliminate the unnecessary computations in the upper right half of the domain by setting the far field boundary diagonally across what would usually be a standard tensor product domain. The node where all underlying assets are equal to $s_d^{\min} = 0$, we consider as a close field boundary. There, we set a Dirichlet boundary condition $u(\mathbf{x}_{\text{CF}}, t) = 0$, and at the far field boundary we set

11

$u(\mathbf{x}_{\mathrm{FF}}, t) = \frac{1}{D} \sum_{i=1}^{D} s_i - K \exp(-rt)$. An example of the node layouts for this problem where $N \approx 1000$ is shown in Figure 2.
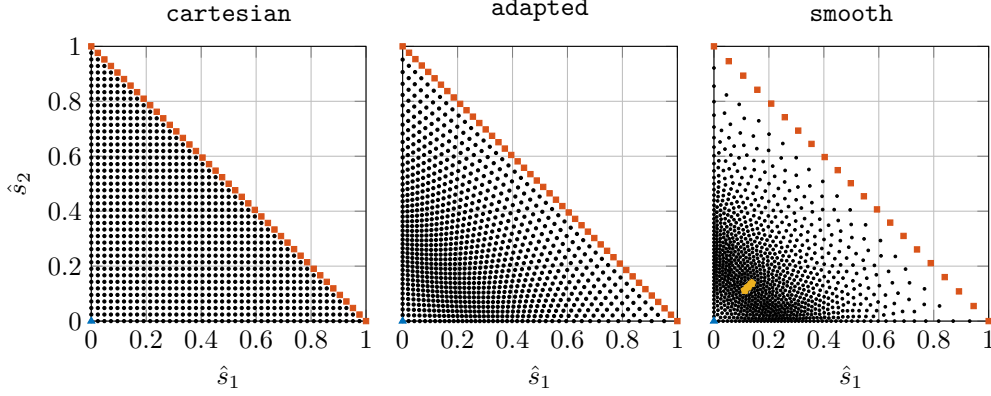

Figure 2: Node layouts on a scaled computational domain used in the numerical experiments. Blue triangles denote the nodes where the close field boundary conditions are applied, red squares denote the nodes where the far field boundary conditions are applied, and yellow pentagons denote the nodes in which the error is measured.

In the plots, the points $\mathbf{x}_{\mathrm{CF}}$ are marked by blue triangles, while the $\mathbf{x}_{\mathrm{FF}}$ points are marked by red squares. The parameters used to obtain the `smooth` node layout are $P = 0.25$, $Q = 0.75$, $G = \frac{\pi}{4}$, while $X_1 = X_2 = \hat{K} = \frac{1}{8}$.

By conducting numerical experiments we have observed second order convergence of the RBF-FD methods as the node layout is refined, which can be seen in Figure 3. From the figure, it is clear that the RBF-FD method on the `smooth` node layout is significantly more accurate than the other two, for the same amount of computational time. The `adapted` layout is also performing better than the standard `cartesian` layout, which is in line with the previous findings presented in [31]. In the left-hand side plot of Figure 3, we see that all of the RBF-FD methods are more accurate (relative to the node count) than the standard FD method. This is expected as the `smooth` layout has higher node density in the area where the error is measured. The right-hand side plot of Figure 3 shows that the RBF-FD method on the `cartesian` node layout can barely compete with the standard FD method, which is reasonable as the computation of the differentiation weights slows it down. Nevertheless, the RBF-FD method evidently outperforms the FD method on the remaining two node layouts.
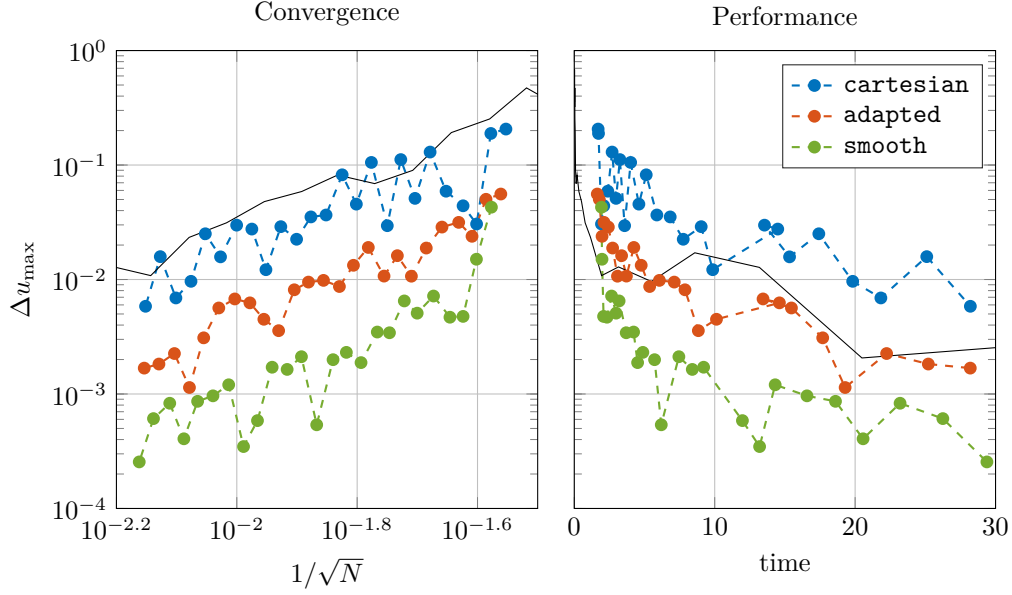
Figure 3: Performance of the RBF-FD method with PHSs for a two-asset European call basket option under the Black–Scholes–Merton model on different node layouts. The left-hand side plot shows the error against the average node layout density and the right-hand side plot shows the error against the computational time measured in seconds. The solid black line represents the performance of the standard FD method of second order.

From the past experience of working with the RBF-FD methods, we also observe the condition number of the differentiation matrix in order to be able to anticipate potential numerical instabilities. The condition numbers for the three setups are presented in Figure 4. This figure also shows the benefits of using the `smooth` layouts versus the others, although the `adapted` layout is also showing a significantly lower condition number growth compared to `cartesian` as the node density increases.
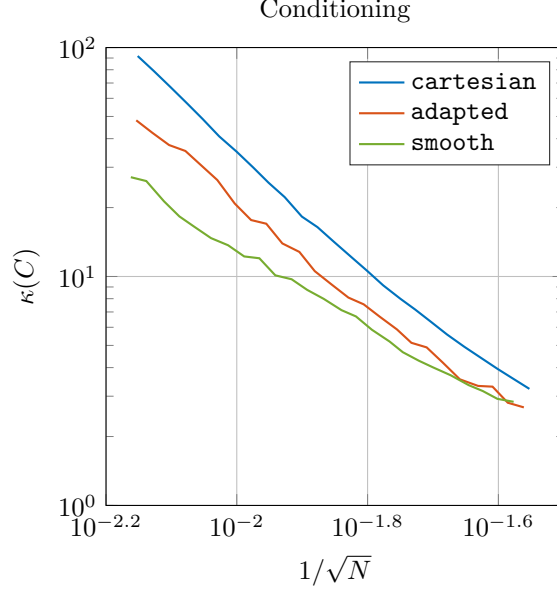
13

Figure 4: The condition number of the differentiation matrix C as a function of the average node density for different node layouts when pricing a two-asset European Call basket option.

3.1.2 American Put Basket Option under The Black–Scholes–Merton Model

When it comes to American options, these financial derivatives can be exercised at any $t \leq T$, as opposed to the European options that can only be exercised at $t = T$. Instead of using a PDE as a model, for American options we need to formulate the pricing task as a linear complementarity problem (LCP)

$$\frac{\partial u}{\partial t} + \mathcal{L}u \geq 0,$$

$$u(s_1, s_2, \ldots, s_D, t) \geq g(s_1, s_2, \ldots, s_D), \tag{17}$$

$$\left(\frac{\partial u}{\partial t} + \mathcal{L}u\right)\left(u(s_1, s_2, \ldots, s_D, t) - g(s_1, s_2, \ldots, s_D)\right) = 0,$$

with the initial data $g(s_1, s_2, \ldots, s_D)$. In order to solve (17), we use the operator splitting method [19, 20, 34], combined with the RBF-FD and BDF2 methods, in the same way as it was used in [31].

We conduct this experiment using the same parameters as for the European call basket option. The only difference are the boundary conditions, because now we deal with a put option. The boundary condition at the far-field boundary nodes is set to be $u(\mathbf{x}_{\mathbf{FF}}, t) = 0$. The option price at the close-field boundary is kept at the value of the payoff function at that point, since that node is behind the free boundary characteristic for American options.
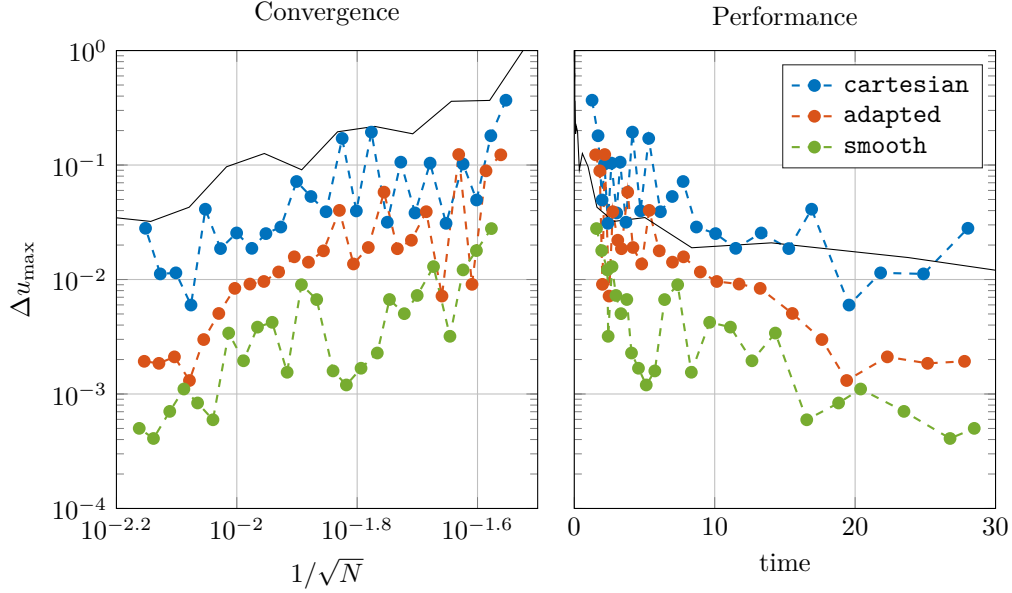
14

Figure 5: Performance of the RBF-FD method with PHSs for a two-asset American put basket option under the Black–Scholes–Merton model on different node layouts. The left-hand side plot shows the error against the average node layout density and the right-hand side plot shows the error against the computational time measured in seconds. The solid black line represents the performance of the standard FD method of second order.

The performance results in Figure 5 look very similar to the European case. Numerical experiments show that the same can be stated for the conditioning.

3.2 Multi-Factor Model

Models with multiple stochastic factors are introduced to capture market features better than the standard Black–Scholes–Merton model. It is well known that the Black–Scholes–Merton framework fails to model heavy tails of return distributions and volatility skews. For instance, the interest for local volatility models started with the work of Dupire [5] and ever since they have been becoming increasingly popular. In this section, we use one of the most popular stochastic local volatility models known as the Heston model [17].

The dynamics of the Heston models is

$$\mathrm{d}S(t) = rS(t)\,\mathrm{d}t + \sqrt{V(t)}S(t)\,\mathrm{d}W_s(t), \quad S(0) = s, \tag{18}$$

$$\mathrm{d}V(t) = \kappa(\eta - V(t))\,\mathrm{d}t + \sigma\sqrt{V(t)}\,\mathrm{d}W_v(t), \quad V(0) = v, \tag{19}$$

where $S(t)$ is the underlying asset price, $V(t)$ is its stochastic volatility, σ is the constant volatility of volatility, κ is the speed of mean reversion of the

volatility process, η is the mean reversion level, r is the risk-free interest rate, $W_s(t)$ and $W_v(t)$ are correlated Wiener processes with constant correlation ρ, i.e., $\mathrm{d}W_s(t)\,\mathrm{d}W_v(t) = \rho\,\mathrm{d}t$.

By applying the Itô lemma and the Feynman–Kac theorem, a PDE for the Heston model reads as

$$\frac{\partial u}{\partial t} + \mathcal{L}u = 0, \tag{20}$$

$$u(s, v, T) = \max(s - K,\ 0), \tag{21}$$

where

$$\mathcal{L}u \equiv \frac{1}{2}vs^2\frac{\partial^2 u}{\partial s^2} + \rho\sigma vs\frac{\partial^2 u}{\partial s\partial v} + \frac{1}{2}\sigma^2 v\frac{\partial^2 u}{\partial v^2} + rs\frac{\partial u}{\partial s} + \kappa(\eta - v)\frac{\partial u}{\partial v} - ru, \tag{22}$$

K is the strike price and s and v are deterministic representations of the stochastic asset price and volatility processes, respectively.

3.2.1 European Call Option under The Heston Model

We perform an experiment with a European call option where $r = 0.03$, $\kappa = 2$, $\eta = 0.0225$, $\sigma = 0.25$, and $\rho = -0.5$, while $K = 100$. We choose three evaluation points close to the strike price K at which we compute the option value results

$$\mathbf{x}_{\texttt{HST}} = \begin{pmatrix} 90 & 0.0225 \\ 100 & 0.0225 \\ 110 & 0.0225 \end{pmatrix},$$

The relative location of these points on a scaled domain can be seen in the right most plot of Figure 6 denoted with yellow pentagons. For the `cartesian` and `adapted` node layouts, we employ cubic interpolation to approximate the values at the desired points. Then, we use the maximum error from those three points $\Delta u_{\max}$ to measure convergence.

We setup the computational domain such that the far field boundaries are at $s^{\max} = 4K$ and $v^{\max} = 0.5$. At the points where $s = 0$, we set a simple Dirichlet boundary condition $u(\mathbf{x}_{\texttt{CF}}, t) = 0$, and at the far field $u(\mathbf{x}_{\texttt{FF}}, t) = s - K\exp(-rt)$. The points $\mathbf{x}_{\texttt{CF}}$ are marked by blue triangles, while the $\mathbf{x}_{\texttt{FF}}$ points are marked by red squares in the plots of Figure 6. We leave the volatility boundaries without enforcing any conditions and compute the option values at those points through the RBF-FD approximation, the same way as in the inner domain.

Similarly to the previous examples, we consider three different node layouts for this problem and we name them correspondingly. In this case the `adapted` node layout is not constructed diagonally, as the discontinuity in the terminal condition of this problem is orthogonal to s axis. This node layout has been adapted to cluster the points close to the strike. In case of the `smooth` node

layout, the nodes are pushed closer to the strike and towards the lower volatility boundary. This is done by choosing the node layout parameters as $P = 0.75$, $Q = 0.25$, and $G = 0$, while $X_1 = \hat{K} = \frac{1}{4}$ and $X_2 = \frac{1}{2}$ 0.0225 . An example of the node layouts where $N \approx 1000$ is shown in Figure 6.
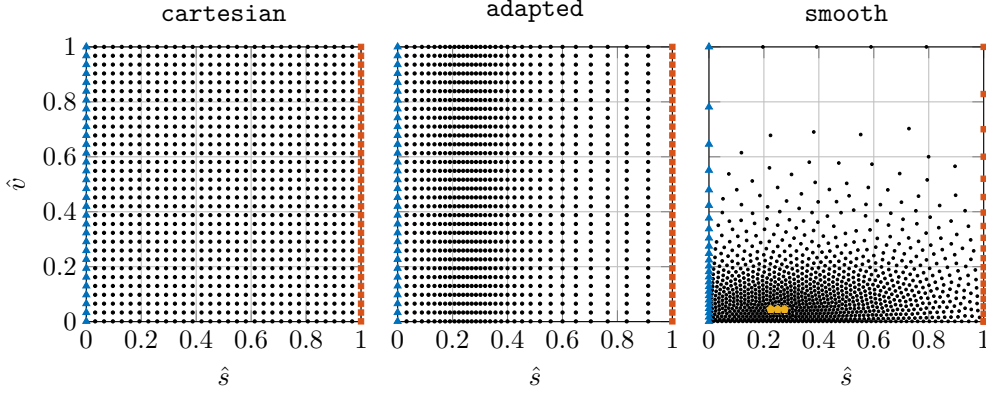

Figure 6: Node layouts on a scaled computational domain used in the numerical experiments. Blue triangles denote the nodes where the close field boundary conditions are applied, red squares denote the nodes where the far field boundary conditions are applied, and yellow pentagons denote the nodes in which the error is measured.

When it comes to performance of the RBF-FD methods under the Heston model, Figure 7 shows how important it is to have a smoothly varying density. The figure clearly demonstrates the dominance of the RBF-FD method with the `smooth` layout, both in accuracy and computational time. In this figure, we can also see how the `adapted` layout, which was initially designed for basket option problems, starts as a bit better than `cartesian`, but eventually encounters numerical instabilities as the layout density fails to be smooth enough to support the stencils under the Heston operator. Figure 8 shows the `smooth` node layout as a tremendously better conditioned scheme compared to the other two.
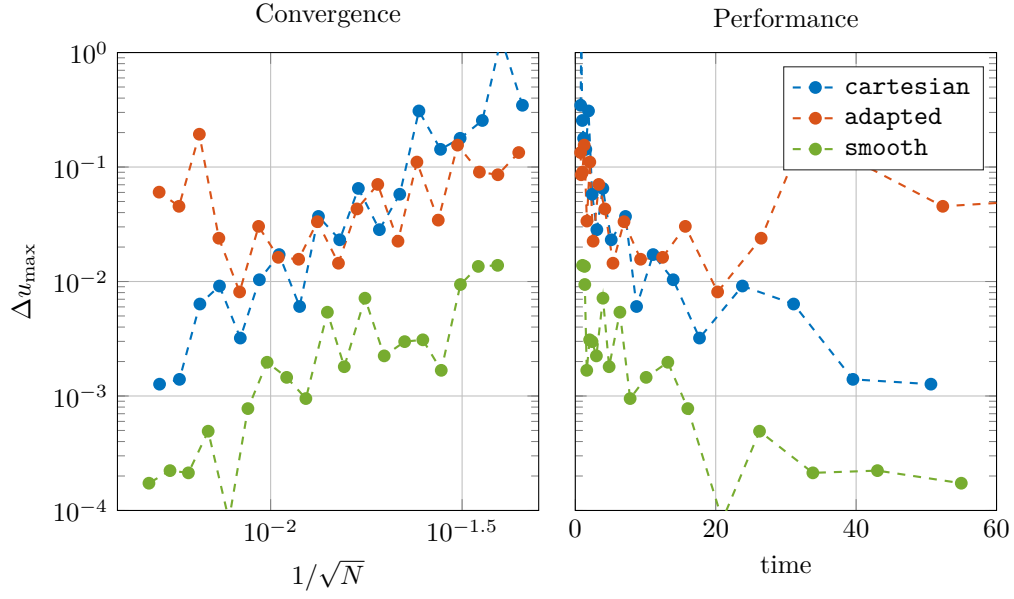
17

Figure 7: Performance of the RBF-FD method with PHSs for an European call option under the Heston model on different node layouts. The left-hand side plot shows the error against the average node layout density and the right-hand side plot shows the error against the computational time measured in seconds.
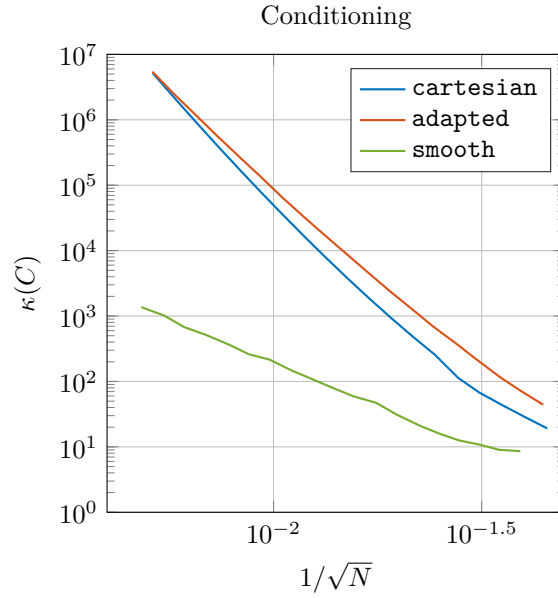

Figure 8: The condition number of the differentiation matrix as a function of the average node density for different node layouts.

4 Conclusions

In this paper we study the benefits of using PHSs and node layouts with smoothly varying density for developing robust and efficient RBF-FD methods for option pricing. We present the improved RBF-FD scheme and successfully apply it to two types of multidimensional PDEs in finance: two-dimensional European call and American put basket options under the Black–Scholes–Merton model, and a European call option under the Heston model. We show numerically that the performance of the method is equally high when it comes to pricing American options compared to the European ones. By studying convergence, computational performance, and conditioning of the discrete systems, we demonstrate the desirable properties of the introduced approaches.

The implemented RBF-FD methods significantly outperformed the standard FD method in the numerical experiments, despite the computational overhead from the differentiation weights. As the computation of the differentiation weights is parallelizable, the performance dominance should be even higher when machines with higher number of cores are used.

Using PHSs as RBFs, augmented with polynomials, in the RBF-FD approximations, show to be hassle free as a result of absence of the shape parameter. The PHSs take control of stabilizing the stencils as the degree of the augmented polynomials in the approximation dictates the formal order of the method.

Although the used smoothly varying density node placing algorithm works only in two-dimensional domains, some recent work has been done to come up with more robust and efficient ways to construct adaptable smooth node layouts in higher dimensions [38]. Research on efficient generation of high-dimensional node layouts is expected to give a significant improvement in performance of the higher-dimensional RBF-FD methods and improve the competitiveness of these methods in different financial applications.

Acknowledgements

References

[1] BAYONA, V., FLYER, N., FORNBERG, B., AND BARNETT, G. A. On the role of polynomials in RBF–FD approximations: II. Numerical solution of elliptic PDEs. *Journal of Computational Physics 332* (2017), 257–273.

[2] BENTLEY, J. L. Multidimensional binary search trees used for associative searching. *Communications of the ACM 18*, 9 (1975), 509–517.

[3] BLACK, F., AND SCHOLES, M. The pricing of options and corporate liabilities. *J. Polit. Econ. 81* (1973), 637–654.

[4] DAVYDOV, O., AND OANH, D. T. Adaptive meshless centres and RBF stencils for Poisson equation. *Journal of Computational Physics 230*, 2 (2011), 287–304.

[5] DUPIRE, B., ET AL. Pricing with a smile. *Risk 7*, 1 (1994), 18–20.

[6] FASSHAUER, G. E., KHALIQ, A. Q. M., AND VOSS, D. A. Using meshfree approximation for multi-asset american options. *Journal of the Chinese Institute of Engineers 27*, 4 (2004), 563–571.

[7] FLYER, N., BARNETT, G. A., AND WICKER, L. J. Enhancing finite differences with radial basis functions: experiments on the Navier–Stokes equations. *Journal of Computational Physics 316* (2016), 39–62.

[8] FLYER, N., FORNBERG, B., BAYONA, V., AND BARNETT, G. A. On the role of polynomials in RBF–FD approximations: I. Interpolation and accuracy. *Journal of Computational Physics 321* (2016), 21–38.

[9] FLYER, N., AND LEHTO, E. Rotational transport on a sphere: Local node refinement with radial basis functions. *Journal of Computational Physics 229*, 6 (2010), 1954–1969.

[10] FLYER, N., LEHTO, E., BLAISE, S., WRIGHT, G. B., AND ST-CYR, A. A guide to RBF-generated finite differences for nonlinear transport: Shallow water simulations on a sphere. *Journal of Computational Physics 231*, 11 (2012), 4078–4095.

[11] FORNBERG, B., AND FLYER, N. Fast generation of 2-D node distributions for mesh-free PDE discretizations. *Computers & Mathematics with Applications 69*, 7 (2015), 531–544.

[12] FORNBERG, B., AND LEHTO, E. Stabilization of RBF-generated finite difference methods for convective PDEs. *Journal of Computational Physics 230*, 6 (2011), 2270–2285.

[13] FORNBERG, B., LEHTO, E., AND POWELL, C. Stable calculation of Gaussian-based RBF–FD stencils. *Comput. Math. Appl. 65*, 4 (Feb. 2013), 627–637.

[14] GOLBABAI, A., AND MOHEBIANFAR, E. A new stable local radial basis function approach for option pricing. *Computational Economics* (2016), 1–18.

[15] HAENTJENS, T., AND IN'T HOUT, K. J. ADI schemes for pricing American options under the Heston model. *Applied Mathematical Finance 22*, 3 (2015), 207–237.

[16] HAIRER, E., NØRSETT, S., AND WANNER, G. *Solving Ordinary Differential Equations I. Nonstiff problems*, second ed. Springer-Verlag, Berlin, 2000.

[17] HESTON, S. L. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The review of financial studies 6*, 2 (1993), 327–343.

[18] HON, Y.-C., AND MAO, X.-Z. A radial basis function method for solving options pricing model. *Financial Engineering 8*, 1 (1999), 31–49.

[19] IKONEN, S., AND TOIVANEN, J. Operator splitting methods for American option pricing. *Applied mathematics letters 17*, 7 (2004), 809–814.

[20] IKONEN, S., AND TOIVANEN, J. Operator splitting methods for pricing American options under stochastic volatility. *Numerische Mathematik 113*, 2 (2009), 299–324.

[21] IN'T HOUT, K., AND FOULON, S. ADI finite difference schemes for option pricing in the Heston model with correlation. *Int. J. Numer. Anal. Model 7*, 2 (2010), 303–320.

[22] KADALBAJOO, M. K., KUMAR, A., AND TRIPATHI, L. P. Application of radial basis function with L-stable Padé time marching scheme for pricing exotic option. *Computers & Mathematics with Applications 66*, 4 (2013), 500–511.

[23] KADALBAJOO, M. K., KUMAR, A., AND TRIPATHI, L. P. Application of the local radial basis function-based finite difference method for pricing American options. *International Journal of Computer Mathematics 92*, 8 (2015), 1608–1624.

[24] KADALBAJOO, M. K., KUMAR, A., AND TRIPATHI, L. P. An efficient numerical method for pricing option under jump diffusion model. *International Journal of Advances in Engineering Sciences and Applied Mathematics 7*, 3 (2015), 114–123.

[25] KANSA, E. J. Multiquadrics — A scattered data approximation scheme with applications to computational fluid-dynamics — I surface approximations and partial derivative estimates. *Computers & Mathematics with applications 19*, 8-9 (1990), 127–145.

[26] Kansa, E. J. Multiquadrics — A scattered data approximation scheme with applications to computational fluid-dynamics — II solutions to parabolic, hyperbolic and elliptic partial differential equations. *Computers & mathematics with applications 19*, 8-9 (1990), 147–161.

[27] Kumar, A., Tripathi, L. P., and Kadalbajoo, M. K. A numerical study of Asian option with radial basis functions based finite differences method. *Engineering Analysis with Boundary Elements 50* (2015), 1–7.

[28] Larsson, E., Åhlander, K., and Hall, A. Multi-dimensional option pricing using radial basis functions and the generalized Fourier transform. *Journal of Computational and Applied Mathematics 222*, 1 (2008), 175–192.

[29] Larsson, E., Lehto, E., Heryudono, A., and Fornberg, B. Stable computation of differentiation matrices and scattered node stencils based on Gaussian radial basis functions. *SIAM Journal on Scientific Computing 35*, 4 (2013), A2096–A2119.

[30] Merton, R. C. Theory of rational option pricing. *Bell J. Econom. Man. Sci. 4* (1973), 141–183.

[31] Milovanović, S., and von Sydow, L. Radial basis function generated finite differences for option pricing problems. *Computers & Mathematics with Applications 75*, 4 (2018), 1462–1481.

[32] Pettersson, U., Larsson, E., Marcusson, G., and Persson, J. Improved radial basis function methods for multi-dimensional option pricing. *Journal of Computational and Applied Mathematics 222*, 1 (2008), 82–93.

[33] Safdari-Vaighani, A., Heryudono, A., and Larsson, E. A radial basis function partition of unity collocation method for convection–diffusion equations arising in financial applications. *Journal of Scientific Computing* (2015), 1–27.

[34] Salmi, S., Toivanen, J., and von Sydow, L. An IMEX-scheme for pricing options under stochastic volatility models with jumps. *SIAM Journal on Scientific Computing 36*, 5 (2014), B817–B834.

[35] Shcherbakov, V. Radial basis function partition of unity operator splitting method for pricing multi-asset American options. *BIT Numerical Mathematics* (2016), 1–23.

[36] Shcherbakov, V., and Larsson, E. Radial basis function partition of unity methods for pricing vanilla basket options. *Computers & Mathematics with Applications 71*, 1 (2016), 185–200.

[37] TOLSTYKH, A. I. On using RBF-based differencing formulas for unstructured and mixed structured–unstructured grid calculations. In *Proceedings of the 16th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation, Lausanne, Switzerland* (2000), p. 6.

[38] VLASIUK, O., MICHAELS, T., FLYER, N., AND FORNBERG, B. Fast high-dimensional node generation with variable density. *arXiv preprint arXiv:1710.05011* (2017).

[39] VON SYDOW, L., JOSEF HÖÖK, L., LARSSON, E., LINDSTRÖM, E., MILOVANOVIĆ, S., PERSSON, J., SHCHERBAKOV, V., SHPOLYANSKIY, Y., SIRÉN, S., TOIVANEN, J., ET AL. BENCHOP — the BENCHmarking project in option pricing. *International Journal of Computer Mathematics 92*, 12 (2015), 2361–2379.

[40] VON SYDOW, L., MILOVANOVIĆ, S., LARSSON, E., IN 'T HOUT, K., WIKTORSSON, M., OOSTERLEE, C. W., SHCHERBAKOV, V., WYNS, M., LEITAO, A., JAIN, S., HAENTJENS, T., AND WALDÉN, J. BENCHOP–SLV: The BENCHmarking project in option pricing — stochastic and local volatility problems. Submited to *International Journal of Computer Mathematics*, July 2018.

[41] WENDLAND, H. Fast evaluation of radial basis functions: Methods based on partition of unity. In *Approximation Theory X: Wavelets, Splines, and Applications* (2002), Citeseer.

[42] WRIGHT, G. B., AND FORNBERG, B. Scattered node compact finite difference-type formulas generated from radial basis functions. *Journal of Computational Physics 212*, 1 (2006), 99–123.