Computers and Mathematics with Applications 75 (2018) 1462–1481

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Radial Basis Function generated Finite Differences for option pricing problems

Slobodan Milovanović, Lina von Sydow *

Division of Scientific Computing, Department of Information Technology, Uppsala University, SE-751 05 Uppsala, Sweden

ARTICLE INFO

Article history:
Received 21 April 2017
Received in revised form 30 September 2017
Accepted 12 November 2017
Available online 1 December 2017

Keywords:
RBF-FD
Finite Differences
Radial basis function approximation
PDEs
Multi-asset option pricing
High-dimensional problems

ABSTRACT

In this paper we present a numerical method to price options based on Radial Basis Function generated Finite Differences (RBF-FD) in space and the Backward Differentiation Formula of order 2 (BDF-2) in time. We use Gaussian RBFs that depend on a shape parameter ε. The choice of this parameter is crucial for the performance of the method. We chose ε as const $\cdot\, h^{-1}$ and we derive suitable values of the constant for different stencil sizes in 1D and 2D. This constant is independent of the problem parameters such as the volatilities of the underlying assets and the interest rate in the market. In the literature on option pricing with RBF-FD, a constant value of the shape parameter is used. We show that this always leads to ill-conditioning for decreasing h, whereas our proposed method avoids such ill-conditioning. We present numerical results for problems in 1D, 2D, and 3D demonstrating the useful features of our method such as discretization sparsity, flexibility in node placement, and easy dimensional extendability, which provide high computational efficiency and accuracy.

1. Introduction

Calibration and pricing of financial instruments is something that is going on daily in the financial industry. In most cases there are no analytical solutions available and therefore numerical methods have to be used for this purpose. Hence, accurate and efficient methods for this kind of problems are of utmost importance. In this paper our focus is on developing methods for pricing of multi-asset options, i.e. when an option is issued on several underlying assets. This leads to a high-dimensional problem which is numerically very challenging to solve.

We start by considering the Black–Scholes–Merton model with a risk free asset B and a risky asset S that follow the dynamics

$$\begin{aligned} \mathrm{d}B(t) &= rB(t)\mathrm{d}t, \\ \mathrm{d}S(t) &= \mu S(t)\mathrm{d}t + \sigma S(t)\mathrm{d}W(t), \end{aligned} \tag{1}$$

where t is time, r is the interest rate, μ is the drift and σ is the volatility of S, and W is the Wiener process. A European type option issued on S, maturing at time T, with a payoff function $g(S(T))$ can be priced from

$$u(S(t), t) = e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}\left[g(S(T))\right], \tag{2}$$

* Corresponding author.
E-mail addresses: slobodan.milovanovic@it.uu.se (S. Milovanović), lina.von.sydow@it.uu.se (L. von Sydow).

https://doi.org/10.1016/j.camwa.2017.11.015
0898-1221/

where $\mathbb{E}^Q[\cdot]$ denotes the expected value under the risk-neutral measure Q. In [1] and [2] it was independently shown that the price of an option can also be obtained by solving

$$\begin{aligned} \frac{\partial u}{\partial t} + rs\frac{\partial u}{\partial s} + \frac{1}{2}s^2\sigma^2\frac{\partial^2 u}{\partial s^2} - ru &= 0, \\ u(s,T) &= g(s). \end{aligned} \tag{3}$$

Eq. (3) is a parabolic partial differential equation (PDE), which can be solved backward in time using e.g. Finite Differences (FD) in space [3–6].

Next, we turn to multi-asset options that depend on D underlying assets $S_d(t)$, $d = 1, \ldots, D$. The multi-dimensional analogue to (1) is

$$\begin{aligned} \mathrm{d}B(t) &= rB(t)\mathrm{d}t, \\ \mathrm{d}S_1(t) &= \mu_1 S_1(t)\mathrm{d}t + \sigma_1 S_1(t)\mathrm{d}W_1(t), \\ \mathrm{d}S_2(t) &= \mu_2 S_2(t)\mathrm{d}t + \sigma_2 S_2(t)\mathrm{d}W_2(t), \\ &\vdots \\ \mathrm{d}S_D(t) &= \mu_D S_D(t)\mathrm{d}t + \sigma_D S_D(t)\mathrm{d}W_D(t), \end{aligned} \tag{4}$$

where 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 high-dimensional setting, an option issued on the assets (4) with payoff function $g(S_1(T), \ldots, S_D(T))$ can be priced from

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

The corresponding high-dimensional Black–Scholes–Merton equation reads

$$\begin{aligned} \frac{\partial u}{\partial t} + r\sum_i^D s_i\frac{\partial u}{\partial s_i} + \frac{1}{2}\sum_{i,j}^D \rho_{i,j}\sigma_i\sigma_j s_i s_j\frac{\partial^2 u}{\partial s_i \partial s_j} - ru &\equiv \frac{\partial u}{\partial t} + \mathcal{L}u = 0, \\ u(s_1, s_2, \ldots, s_D, T) &= g(s_1, s_2, \ldots, s_D). \end{aligned} \tag{5}$$

Since FD methods are generally discretized on tensor product grids of 1D Cartesian grids, the number of degrees of freedom grows exponentially in the number of dimensions D—the so-called *curse of dimensionality*. Traditionally, Monte-Carlo methods have been the only way to price options in dimensions larger than approximately 5.

Global radial basis functions (RBFs) approximation methods are mesh-free, meaning that they are flexible with respect to the geometry of the computational domain. Hence, we are no longer restricted to Cartesian grids, and we can more freely place nodes where they are needed for accuracy reasons. Moreover, the methods are not more complicated for high-dimensional problems than in lower dimensions, since the only geometrical property that is used is the pairwise distance between points. Finally, for smooth functions, approximations with smooth RBFs can give spectral convergence. When RBFs are used, the space is discretized using N nodes $\mathbf{s}^{(i)}$ and the solution is approximated by

$$u(\mathbf{s}, t) \approx \sum_{i=1}^N \lambda_i(t)\phi(\|\mathbf{s} - \mathbf{s}^{(i)}\|), \ k = 1, 2, \ldots, N,$$

where $\phi(r)$ is a radial basis function. Possible radial basis functions are eg. Gaussian ($e^{-(\varepsilon r)^2}$), inverse quadratic ($1/(1+(\varepsilon r)^2)$), multiquadric ($\sqrt{1+(\varepsilon r)^2}$) and inverse multiquadric ($1/\sqrt{1+(\varepsilon r)^2}$), where ε is a shape parameter that determines the width of the basis function, see e.g. [7].

To sparsify the linear systems of equations, various techniques to localize the RBFs are possible such as RBF Partition of Unity Methods (RBF-PUM) [8–11], where the computational domain is partitioned into subdomains. In this paper we consider an even more localized strategy, RBF generated Finite Difference (RBF-FD) [12–18]. In RBF-FD methods, the finite difference weights in the computational stencils are computed from RBFs rather than from the monomials $\{1, x, x^2, \ldots\}$ which are used in standard finite differences.

We would like to have a solution method for (5) that yields the same type of sparsity structure as FD but is as easy to employ in higher dimensions as the RBF method. By deriving finite difference approximations based on radial basis function approximations we expect to achieve these properties. RBF-FD for option pricing has previously been used in e.g. [19–23] for various types of options. In all these references, 1D and in some cases 2D problems are considered with uniform node layouts and (very) small stencils, which severely restricts the potential usefulness of the method. Moreover, no analysis or suggestions on the choice of the shape parameter ε is provided which is important for the method to be used in practice.

In this paper we propose an RBF-FD method for option pricing that can be used with non-uniform node layouts on suitably shaped domains and in any number of dimensions. We provide numerical examples in up to three dimensions. Also, we analyze our method with respect to accuracy and choice of shape parameter ε. From this analysis and numerical experiments, we propose a way of choosing this important parameter.

The article is organized as follows. In Section 2 we describe the spatial and temporal discretization as well as the treatment of the open boundary problem for the American options. The spatial error is analyzed for a 1D problem in Section 3 and numerical experiments are presented in Section 4. Finally in Section 5 we summarize and draw some conclusions.

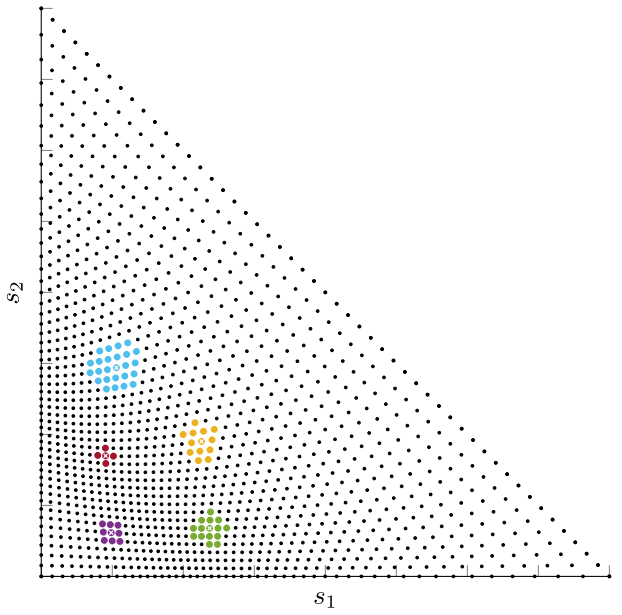


Fig. 1. Examples of nearest neighbor based stencils used for approximating the differential operator on a non-uniform node layout. The central node of each displayed stencil is denoted by a white cross mark, while the nodes that belong to each of the displayed stencils are shown in the same color.

2. Discretization

In this section we present the spatial and temporal discretization used, as well as how we solve the open boundary problem that occurs when pricing American options.

2.1. Spatial discretization using RBF-FD

For each node $\mathbf{s}^{(i)} = (s_1^{(i)}, s_2^{(i)}, \ldots, s_D^{(i)})$ in space, we define a subset $\mathcal{S}^{(i)} = \{\mathbf{s}_k^{(i)}\}_{k=1}^n$ consisting of $n-1$ neighboring nodes and $\mathbf{s}^{(i)}$ itself, and consider it as a stencil. In Fig. 1 we show an example of a 2D node layout with stencils of different sizes.

The differential operator $\mathcal{L}$ defined in (5) is approximated in every node point $\mathbf{s}^{(i)}$, yielding

$$\mathcal{L}u(\mathbf{s}^{(i)}) \approx \sum_{k=1}^{n} w_k^{(i)} u_k^{(i)} \equiv Lu(\mathbf{s}^{(i)}), \quad i = 1, \ldots, N, \tag{6}$$

where $u_k^{(i)} = u(\mathbf{s}_k^{(i)})$. The weights $w_k^{(i)}$ are calculated by enforcing (6) to be exact for RBFs centered at each of the nodes in $\mathcal{S}^{(i)}$ yielding

$$\begin{pmatrix} \phi(\|\mathbf{s}_1^{(i)} - \mathbf{s}_1^{(i)}\|) & \ldots & \phi(\|\mathbf{s}_1^{(i)} - \mathbf{s}_n^{(i)}\|) \\ \vdots & \ddots & \vdots \\ \phi(\|\mathbf{s}_n^{(i)} - \mathbf{s}_1^{(i)}\|) & \ldots & \phi(\|\mathbf{s}_n^{(i)} - \mathbf{s}_n^{(i)}\|) \end{pmatrix} \begin{pmatrix} w_1^{(i)} \\ \vdots \\ w_n^{(i)} \end{pmatrix} = \begin{pmatrix} \mathcal{L}\phi(\|\mathbf{s}^{(i)} - \mathbf{s}_1^{(i)}\|) \\ \vdots \\ \mathcal{L}\phi(\|\mathbf{s}^{(i)} - \mathbf{s}_n^{(i)}\|) \end{pmatrix}. \tag{7}$$

Note that standard FD-discretizations only allow for 1D approximations on a Cartesian grid. Hence, high-dimensional operators are discretized separately in each spatial direction. For the RBF-FD approximations, different dimensions are not important in this sense, only distances between nodes. This means that (7) gives a D-dimensional discretization operator. From the theory on RBF interpolation, (7) forms a nonsingular system. This means that for $i = 1, \ldots, N$, a unique set of weights can be computed and assembled row-wise into a differentiation matrix W. Since $n \ll N$ the resulting matrix is sparse. Due to this sparsity structure, the memory cost is $\mathcal{O}(N)$ compared to $\mathcal{O}(N^2)$ for global approximation with RBFs.

In [13,24,25] it was shown that better accuracy is achieved if the interpolant is able to represent a constant, i.e. (6) is changed to

$$\mathcal{L}u(\mathbf{s}^{(i)}) \approx \sum_{k=1}^{n} \tilde{w}_k^{(i)} u_k^{(i)} + \tilde{w}_{n+1}^{(i)} \equiv \tilde{L}u(\mathbf{s}^{(i)}), \quad i = 1, \ldots, N, \tag{8}$$

and $\sum_{k=1}^{n} \tilde{w}_k^{(i)} = 0$. This means that instead of solving (7), we solve

$$\begin{pmatrix} \phi(\|\mathbf{s}_1^{(i)} - \mathbf{s}_1^{(i)}\|) & \dots & \phi(\|\mathbf{s}_1^{(i)} - \mathbf{s}_n^{(i)}\|) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ \phi(\|\mathbf{s}_n^{(i)} - \mathbf{s}_1^{(i)}\|) & \dots & \phi(\|\mathbf{s}_n^{(i)} - \mathbf{s}_n^{(i)}\|) & 1 \\ 1 & \dots & 1 & 0 \end{pmatrix} \begin{pmatrix} \tilde{w}_1^{(i)} \\ \vdots \\ \tilde{w}_n^{(i)} \\ \tilde{w}_{n+1}^{(i)} \end{pmatrix} = \begin{pmatrix} \mathcal{L}\phi(\|\mathbf{s}^{(i)} - \mathbf{s}_1^{(i)}\|) \\ \vdots \\ \mathcal{L}\phi(\|\mathbf{s}^{(i)} - \mathbf{s}_n^{(i)}\|) \\ 0 \end{pmatrix}, \tag{9}$$

where $\tilde{w}_{n+1}^{(i)}$ is ignored after the system is solved and then we put the obtained weights $\tilde{w}_k^{(i)}$ with $k = 1, \dots, n$ at the corresponding places of row i in the differentiation matrix W.

Moreover, in [26] and [27], it is demonstrated that for overcoming the convergence saturation it is useful to add even higher degree polynomials. Nevertheless, in the numerical experiments of Section 4 we show that the saturation levels corresponding to approximation $\tilde{L}$ are below the desired error tolerances, and therefore there is no need here to further increase the computational load of the method by adding higher degree polynomial terms to the interpolation matrix.

In this article we employ Gaussian basis function $\phi(r) = e^{-\varepsilon^2 r^2}$. In [28], Gaussian basis functions, inverse quadric, and multiquadric basis functions performed similarly for an option pricing problem. We perform analysis and numerical studies in order to establish whether to use approximation (6) or (8), as well as a relation between ε and the spatial discretization parameter h that we propose.

An approach known as RBF-GA [29], was used to efficiently mitigate the dependency of the RBF-FD method on the shape parameter ε in a comprehensive comparison of numerical methods for option pricing called BENCHOP [30]. These stable algorithms are usually a few dozens of times more computationally intensive than the standard approach in a stable regime. Therefore, in the sequel of this paper, we focus on developing stable RBF-FD methods that do not need to employ such computationally expensive treatments.

2.2. Temporal discretization

The spatial discretization of (5) described in the previous section leads to the following systems of ODEs

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} + W\mathbf{u} = 0, \tag{10}$$

where $\mathbf{u} = (u_1, \dots, u_N)^T$ and u_i, $i = 1, \dots, N$, is the approximation of $u(\mathbf{s}^{(i)})$ from the spatial discretization. For the time discretization of (10), we use the Backward Differentiation Formula of order 2 (BDF-2) [31]. Since BDF-2 requires the solution at two previous time steps, we employ backward Euler (BDF-1) for the first time step. In order to have a constant coefficient matrix, we use non-equidistant time steps as described in [32]. This is done by discretizing the time interval with M steps of length $\Delta t^\ell = t^\ell - t^{\ell-1}$, where $\ell = 1, \dots, M$. Then we let $\omega_\ell = \Delta t^\ell / \Delta t^{\ell-1}$ for $\ell = 2, \dots, M$ and arrive at

$$\mathbf{u}^1 - \mathbf{u}^0 = \Delta t^1 W \mathbf{u}^1, \tag{11}$$

$$\mathbf{u}^\ell - \beta_1^\ell \mathbf{u}^{\ell-1} - \beta_2^\ell \mathbf{u}^{\ell-2} = \beta_0^\ell W \mathbf{u}^\ell, \quad \ell = 2, \dots, M, \tag{12}$$

where

$$\beta_0^\ell = \Delta t^\ell \frac{1+\omega_\ell}{1+2\omega_\ell}, \quad \beta_1^\ell = \frac{(1+\omega_\ell)^2}{1+2\omega_\ell}, \quad \beta_2^\ell = \frac{\omega_\ell^2}{1+2\omega_\ell}. \tag{13}$$

We compute the values for ω_ℓ using the recursive condition $\beta_0^\ell = \beta_0^{\ell-1}$, which keeps the coefficient matrix constant throughout all time steps. Since our time interval has the length T, we chose the initial time step length Δt^1 from

$$\sum_{\ell=1}^{M} \Delta t^\ell = T = \Delta t^1 (1 + \sum_{\ell=2}^{M} \prod_{\ell'=2}^{\ell} \omega^\ell).$$

Finally, we start the integration by setting $\mathbf{u}^0 = g(\mathbf{s})$, where $\mathbf{s} = \left(\mathbf{s}^{(1)}, \dots, \mathbf{s}^{(N)}\right)^T$.

2.3. American options

American options can be exercised for all $t \le T$, as opposed to European options that can only be exercised at $t = T$. This leads to the following linear complementarity problem (LCP) for American options

$$\begin{aligned} &\frac{d\mathbf{u}}{dt} + W\mathbf{u} \ge 0, \\ &\mathbf{u} \ge g(\mathbf{s}), \\ &(\mathbf{u} - g(\mathbf{s})) \left(\frac{d\mathbf{u}}{\partial t} + W\mathbf{u} \right) = 0. \end{aligned} \tag{14}$$

We use the operator splitting method to solve (14), see [33–35]. Moreover, as for the European options case, we want to keep the system matrix constant throughout the time stepping and therefore we combine the operator splitting procedure with the previously presented scheme with varying time stepping. Therefore, our varying time stepping BDF-1/BDF-2 scheme applied to (14) looks like

$$\mathbf{u}^1 - \mathbf{u}^0 = \Delta t^1 W \mathbf{u}^1 + \Delta t^1 \boldsymbol{\lambda}^0, \tag{15}$$

$$\begin{aligned} &\frac{1}{\Delta t^1}\left(\mathbf{u}^1 - \hat{\mathbf{u}}^1\right) - \left(\boldsymbol{\lambda}^1 - \boldsymbol{\lambda}^0\right) = 0, \\ &\left(\boldsymbol{\lambda}^1\right)^T \left(\mathbf{u}^1 - g(\mathbf{s})\right) = 0, \quad \mathbf{u}^1 \geq g(\mathbf{s}) \text{ and } \boldsymbol{\lambda}^1 \geq 0 \end{aligned} \tag{16}$$

and similarly

$$\mathbf{u}^\ell - \beta_1^\ell \mathbf{u}^{\ell-1} - \beta_2^\ell \mathbf{u}^{\ell-2} = \beta_0^\ell W \mathbf{u}^\ell + \beta_0^\ell \boldsymbol{\lambda}^{\ell-1}, \quad \ell = 2, \ldots, M, \tag{17}$$

$$\begin{aligned} &\frac{1}{\beta_0^\ell}\left(\mathbf{u}^\ell - \hat{\mathbf{u}}^\ell\right) - \left(\boldsymbol{\lambda}^\ell - \boldsymbol{\lambda}^{\ell-1}\right) = 0, \\ &\left(\boldsymbol{\lambda}^\ell\right)^T \left(\mathbf{u}^\ell - g(\mathbf{s})\right) = 0, \quad \mathbf{u}^\ell \geq g(\mathbf{s}) \text{ and } \boldsymbol{\lambda}^\ell \geq 0. \end{aligned} \tag{18}$$

Hence, we first solve for $\hat{\mathbf{u}}^\ell$ using (15) and (17) and then update for $\mathbf{u}^\ell$ (and $\boldsymbol{\lambda}^\ell$) using (16) and (18), respectively.

3. Spatial error analysis

In this section we perform a spatial error analysis for a 1D European option pricing problem (3), approximated using (6) or (8) with Gaussian radial basis function $\phi(r) = e^{-\varepsilon^2 r^2}$. For the purpose of analysis, we consider a three-point equidistant stencil ($n = 3$) with nodes $s^{(i)} - h$, $s^{(i)}$, and $s^{(i)} + h$. We study the local truncation errors τ^I, τ^{II}, $\tilde{\tau}^I$ and $\tilde{\tau}^{II}$ in the approximations (6) and (8) of

$$\mathcal{L}^I u = \frac{\partial u}{\partial s} \tag{19}$$

and

$$\mathcal{L}^{II} u = \frac{\partial^2 u}{\partial s^2} \tag{20}$$

respectively. This gives (for details, see Appendix)

$$\begin{aligned} \tau^I &= \frac{h^2}{6}\frac{\partial^3 u}{\partial s^3} + \frac{h^4}{120}\frac{\partial^5 u}{\partial s^5} + \varepsilon^2 h^2 \frac{\partial u}{\partial s} + (\varepsilon^2 h^2)\frac{h^2}{6}\frac{\partial^3 u}{\partial s^3} - \frac{1}{6}(\varepsilon^2 h^2)^2 \frac{\partial u}{\partial s} + \mathcal{O}(\text{h.o.}), \\ \tilde{\tau}^I &= \tau^I \\ \tau^{II} &= \frac{h^2}{12}\frac{\partial^4 u}{\partial s^4} + \frac{h^4}{360}\frac{\partial^6 u}{\partial s^6} + \frac{1}{3h^2}(\varepsilon^2 h^2)^2 u - \frac{17}{h^2}(\varepsilon^2 h^2)^3 u + \varepsilon^2 h^2 \frac{\partial^2 u}{\partial s^2} + \\ &\quad \frac{1}{6}(\varepsilon^2 h^2)^2 \frac{\partial^2 u}{\partial s^2} + (\varepsilon^2 h^2)\frac{h^2}{12}\frac{\partial^4 u}{\partial s^4} + \mathcal{O}(\text{h.o.}), \\ \tilde{\tau}^{II} &= \frac{h^2}{12}\frac{\partial^4 u}{\partial s^4} + \frac{h^4}{360}\frac{\partial^6 u}{\partial s^6} + \frac{5}{6}\varepsilon^2 h^2 \frac{\partial^2 u}{\partial s^2} + \frac{5}{6}(\varepsilon^2 h^2)\frac{h^2}{12}\frac{\partial^4 u}{\partial s^4} + \frac{1}{36}(\varepsilon^2 h^2)^2 \frac{\partial^2 u}{\partial s^2} + \mathcal{O}(\text{h.o.}) \end{aligned} \tag{21}$$

where h.o. stands for higher order terms. Combining the spatial local truncation errors for the different operators in (3) together gives

$$\begin{aligned} \tau &= rs\left(\frac{h^2}{6}\frac{\partial^3 u}{\partial s^3} + \frac{h^4}{120}\frac{\partial^5 u}{\partial s^5} + \varepsilon^2 h^2 \frac{\partial u}{\partial s} + (\varepsilon^2 h^2)\frac{h^2}{6}\frac{\partial^3 u}{\partial s^3} - \frac{1}{6}(\varepsilon^2 h^2)^2 \frac{\partial u}{\partial s}\right) + \\ &\quad \frac{1}{2}s^2\sigma^2\left(\frac{h^2}{12}\frac{\partial^4 u}{\partial s^4} + \frac{h^4}{360}\frac{\partial^6 u}{\partial s^6} + \frac{1}{3h^2}(\varepsilon^2 h^2)^2 u - \frac{17}{h^2}(\varepsilon^2 h^2)^3 u + \right. \\ &\quad \left. \varepsilon^2 h^2 \frac{\partial^2 u}{\partial s^2} + \frac{1}{6}(\varepsilon^2 h^2)^2 \frac{\partial^2 u}{\partial s^2} + (\varepsilon^2 h^2)\frac{h^2}{12}\frac{\partial^4 u}{\partial s^4}\right) + \mathcal{O}(\text{h.o.}), \end{aligned} \tag{22}$$

and

$$\begin{aligned}\tilde{\tau} &= rs\left(\frac{h^2}{6}\frac{\partial^3 u}{\partial s^3}+\frac{h^4}{120}\frac{\partial^5 u}{\partial s^5}+\varepsilon^2h^2\frac{\partial u}{\partial s}+(\varepsilon^2h^2)\frac{h^2}{6}\frac{\partial^3 u}{\partial s^3}-\frac{1}{6}(\varepsilon^2h^2)^2\frac{\partial u}{\partial s}\right)+\\ &\frac{1}{2}s^2\sigma^2\left(\frac{h^2}{12}\frac{\partial^4 u}{\partial s^4}+\frac{h^4}{360}\frac{\partial^6 u}{\partial s^6}+\frac{5}{6}\varepsilon^2h^2\frac{\partial^2 u}{\partial s^2}+\frac{5}{6}(\varepsilon^2h^2)\frac{h^2}{12}\frac{\partial^4 u}{\partial s^4}+\right.\\ &\left.\frac{1}{36}(\varepsilon^2h^2)^2\frac{\partial^2 u}{\partial s^2}\right)+\mathcal{O}(\text{h.o.}),\end{aligned} \tag{23}$$

for (6) and (8) respectively.

Just as it is pointed out in [36], it is possible to derive optimal values of ε in each node by minimizing the local truncation errors in (22) and (23). This gives values of ε that depend on the solution u and its derivatives. Here we take another approach and study two different ways to define ε:

$$\varepsilon = \delta, \tag{24}$$

and

$$\varepsilon = \gamma/h, \tag{25}$$

where γ and δ are constants. In Section 4.1.1 we numerically compute suitable values of γ and δ. Inserting (24) and (25) into (22) and (23) gives

$$\begin{aligned}\tau_\delta &= rs\left(\frac{h^2}{6}\frac{\partial^3 u}{\partial s^3}+\frac{h^4}{120}\frac{\partial^5 u}{\partial s^5}+\delta^2h^2\frac{\partial u}{\partial s}+\delta^2\frac{h^4}{6}\frac{\partial^3 u}{\partial s^3}-\frac{1}{6}\delta^4h^4\frac{\partial u}{\partial s}\right)+\\ &\frac{1}{2}s^2\sigma^2\left(\frac{h^2}{12}\frac{\partial^4 u}{\partial s^4}+\frac{h^4}{360}\frac{\partial^6 u}{\partial s^6}+\delta^4\frac{h^2}{3}u-\delta^6 17h^4u+\delta^4\frac{h^4}{6}\frac{\partial^2 u}{\partial s^2}+\right.\\ &\left.\delta^2\frac{h^4}{12}\frac{\partial^4 u}{\partial s^4}+\delta^2h^2\frac{\partial^2 u}{\partial s^2}+\delta^2\frac{h^4}{12}\frac{\partial^4 u}{\partial s^4}\right)+\mathcal{O}(\text{h.o.}),\end{aligned} \tag{26}$$

$$\begin{aligned}\tau_\gamma &= rs\left(\frac{h^2}{6}\frac{\partial^3 u}{\partial s^3}+\frac{h^4}{120}\frac{\partial^5 u}{\partial s^5}+\gamma^2\frac{\partial u}{\partial s}+\gamma^2\frac{h^2}{6}\frac{\partial^3 u}{\partial s^3}-\frac{1}{6}\gamma^4\frac{\partial u}{\partial s}\right)+\\ &\frac{1}{2}s^2\sigma^2\left(\frac{h^2}{12}\frac{\partial^4 u}{\partial s^4}+\frac{h^4}{360}\frac{\partial^6 u}{\partial s^6}+\frac{\gamma^4}{3h^2}u-\frac{17\gamma^6}{h^2}u+\gamma^2\frac{\partial^2 u}{\partial s^2}+\right.\\ &\left.\frac{1}{6}\gamma^4\frac{\partial^2 u}{\partial s^2}+\gamma^2\frac{h^2}{12}\frac{\partial^4 u}{\partial s^4}\right)+\mathcal{O}(\text{h.o.}),\end{aligned} \tag{27}$$

$$\begin{aligned}\tilde{\tau}_\delta &= rs\left(\frac{h^2}{6}\frac{\partial^3 u}{\partial s^3}+\frac{h^4}{120}\frac{\partial^5 u}{\partial s^5}+\delta^2h^2\frac{\partial u}{\partial s}+\delta^2\frac{h^4}{6}\frac{\partial^3 u}{\partial s^3}-\frac{1}{6}\delta^4h^4\frac{\partial u}{\partial s}\right)+\\ &\frac{1}{2}s^2\sigma^2\left(\frac{h^2}{12}\frac{\partial^4 u}{\partial s^4}+\frac{h^4}{360}\frac{\partial^6 u}{\partial s^6}+\frac{5}{6}\delta^2\frac{\partial^2 u}{\partial s^2}+\frac{5}{6}\delta^2\frac{h^2}{12}\frac{\partial^4 u}{\partial s^4}+\frac{1}{36}\delta^4\frac{\partial^2 u}{\partial s^2}\right)+\mathcal{O}(\text{h.o.}),\end{aligned} \tag{28}$$

and

$$\begin{aligned}\tilde{\tau}_\gamma &= rs\left(\frac{h^2}{6}\frac{\partial^3 u}{\partial s^3}+\frac{h^4}{120}\frac{\partial^5 u}{\partial s^5}+\gamma^2\frac{\partial u}{\partial s}+\gamma^2\frac{h^2}{6}\frac{\partial^3 u}{\partial s^3}-\frac{1}{6}\gamma^4\frac{\partial u}{\partial s}\right)+\\ &\frac{1}{2}s^2\sigma^2\left(\frac{h^2}{12}\frac{\partial^4 u}{\partial s^4}+\frac{h^4}{360}\frac{\partial^6 u}{\partial s^6}+\frac{5}{6}\gamma^2\frac{\partial^2 u}{\partial s^2}+\frac{5}{6}\gamma^2\frac{h^2}{12}\frac{\partial^4 u}{\partial s^4}+\frac{1}{36}\gamma^4\frac{\partial^2 u}{\partial s^2}\right)+\mathcal{O}(\text{h.o.}),\end{aligned} \tag{29}$$

respectively. Next, using a numerical example we demonstrate that a constant ε as in (24) yields ill-conditioning problems and that the terms $\frac{\gamma^4}{3h^2}u-\frac{17\gamma^6}{h^2}u$ in (27) cause trouble for large u and small h.

In Figs. 2 and 3 we present results for a one-dimensional European call option problem with parameters $r = 0.03$, $\sigma = 0.15$, $K = 1$, and $T = 1$. In Fig. 2 we show the error

$$\Delta u_{\max} = \max_{x\in\left[\frac{1}{3}K,\frac{5}{3}K\right]}|u-u_{BS}|$$

in the solution as a function of the spatial discretization parameter h for both approximations (6) and (8) and the two different ε (24) and (25). Here u_{BS} is the analytical solution from the Black–Scholes formula

$$u_{BS}(s,t) = \mathcal{N}(\nu)s-\mathcal{N}(\mu)Ke^{-r(T-t)}, \tag{30}$$

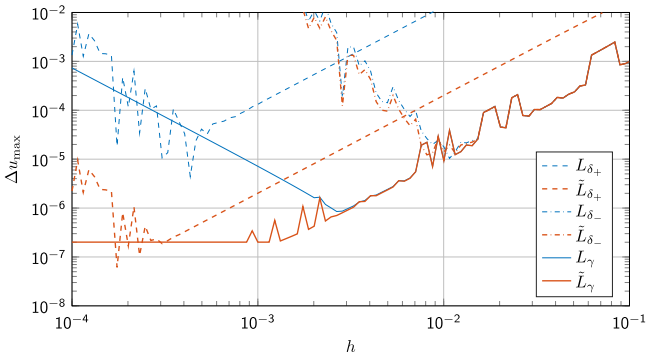


Fig. 2. Error $\Delta u_{\max}$ in the solution as a function of the spatial discretization parameter h for the approximations that are defined in Eq. (6) denoted by L (thinner blue lines) and defined in Eq. (8) denoted by $\tilde{L}$ (thicker red lines). Dashed and dash-dotted lines are for constant ε as defined in (24), with $\delta_+ = 4.7434$ (dashed lines) and $\delta_- = 0.0474$, (dash-dotted lines), and solid lines are for ε proportional to h^{-1} as defined in (25) with $\gamma = 0.0015$, which is in the legend denoted by the corresponding subscripts.

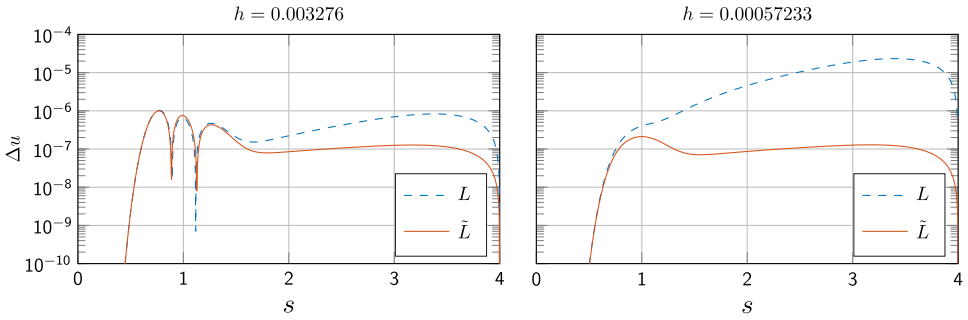


Fig. 3. Absolute error $\Delta u_{\max}$ in the solution as a function of s for the approximations defined in (6) (blue dashed line) and (8) (red solid line) for two different h.

where $\mathcal{N}(x)$ denotes the standard normal cumulative distribution function

$$\mathcal{N}(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{z^2}{2}} dz$$

and

$$\nu = \frac{1}{\sigma\sqrt{T-t}} \left[\ln\left(\frac{s}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t) \right],$$
$$\mu = \nu - \sigma\sqrt{T-t}.$$

We use $\gamma = 0.0015$ and two values of δ: $\delta_- = 0.0474$ and $\delta_+ = 4.7434$. In Section 4.1.1 we verify that these are reasonable choices for (6) and (8). From Fig. 2 we conclude that using a constant ε as defined in (24) yields linear systems of Eqs. (7) and (9) that are highly ill-conditioned for small h. This means that the weights w and $\tilde{w}$ become inaccurate in these cases, and hence also the computed solution. Moreover, by using an ε that is proportional to h^{-1} as in (25) together with the approximation (6), the error in the solution starts to grow for small h which is explained by the terms $\frac{\gamma^4}{3h^2}u - \frac{17\gamma^6}{h^2}u + \mathcal{O}(\frac{\gamma^8}{h^2}u)$ in (27). For large u (which occurs for large values of the underlying asset) and small h, this error can be quite substantial in Fig. 3. We note that for the larger spatial discretization parameter $h = 0.003276$, the error in the solution for the approximation defined in (6) is larger than the error for the approximation defined in (8) only for s larger than approximately 1.5, i.e. a bit away from the area around K that we are most interested in. However, for $h = 0.00057233$ the error for the approximation in (6) is larger already for $s \approx 0.8$, i.e. in the domain of interest.

From the theoretical analysis presented above, that is also corroborated by numerical experiments, we conclude that the approximation defined in (8) is much better to use compared to the approximation in (6), at least for $n = 3$. Furthermore, a constant ε as in (24) is not a good choice due to the ill-conditioning of the linear system of Eqs. (7) or (9), for small h. For this reason we use the approximation defined in (8) together with ε defined by (25) in the sequel of this paper. For this combination we do not run into problems with ill-conditioning or obtain an error that increases with decreasing h. Note however, that the convergence saturates at a low error level. This is predicted by the theory presented here and is also corroborated by previous research, see e.g. [26] and [27].

Table 1
Volatilities and correlations for the different problems.

Problem	Volatilities	Correlations
1D	$\sigma_1 = 0.15$	–
2D	$\sigma_1 = \sigma_2 = 0.15$	$\rho_{1,2} = 0.5$
3D	$\sigma_1 = \sigma_2 = \sigma_3 = 0.15$	$\rho_{1,2} = \rho_{1,3} = \rho_{2,3} = 0.5$

In Section 4.1.1 we show how to establish suitable values of γ in (25).

4. Numerical experiments

We consider pricing of European and American basket options in up to three dimensions. The experiments were run serially using MATLAB on one of the 304 nodes with two 10-core Intel Xeon V4 CPUs of the `rackham` cluster at Uppsala Multidisciplinary Center for Advanced Computational Science (UPPMAX), Uppsala University.

The parameters used in all problems are $T = 1, K = 1, r = 0.03$. In Table 1 we present the values used for the volatilities and the correlations between the Wiener processes.

The payoff functions for the call and put options are given by

$$g(s_1, \ldots, s_D) = \left(\frac{1}{D} \sum_{k=1}^{D} s_k - K \right)^+,$$

$$g(s_1, \ldots, s_D) = \left(K - \frac{1}{D} \sum_{k=1}^{D} s_k \right)^+,$$

respectively. The boundary conditions used for European call options are

$$\begin{aligned} u(0, \ldots, 0, t) &= 0, \\ u(s_1^*, \ldots, s_D^*, t) &= \frac{1}{D} \sum_{k=1}^{D} s_k^* - e^{-r(T-t)} K, \end{aligned} \tag{31}$$

and for American put options

$$u(s_1^*, \ldots, s_D^*, t) = 0, \tag{32}$$

where $\mathbf{s} = (s_1^*, \ldots, s_D^*)$ denotes the far-field boundary. For the American put option it is not necessary to impose a boundary condition in $\mathbf{s} = (0, \ldots, 0)$, since this boundary point lies in the early exercise region of the option. In Sections 4.1–4.3 the computational domain and the different types of node layouts used are shown for the different option pricing problems.

The linear systems of equations arising in each time step in (11), (12), (15), and (17) are solved using MATLAB's built-in GMRES solver [37] with an ILU preconditioner. Since the coefficient matrix is the same for all time steps, the ILU preconditioner is computed once prior to the time stepping. The L and U factors are computed using no fill-in. We set the tolerance for convergence to 10^{-6} and the initial guess in each time step to the solution in the previous time step.

The error $\Delta u_{\max}$ presented in the experiments is measured as the maximum value of the absolute difference between the numerical solution and the reference solution, in a part of the domain close to the strike defined by $\left[\frac{1}{3}K, \frac{5}{3}K\right]^D$. For the 1D European call option the reference solution is computed analytically from the Black–Scholes formula (30). For the 2D problems the reference solutions are computed on a very fine grid using standard centered second-order FD.

4.1. A one-dimensional option pricing problem

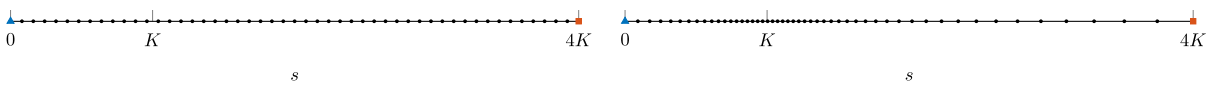
In order to establish how to determine suitable shape parameters ε defined in (25), we start by studying pricing of a one-dimensional European call option. We consider both a uniform and a non-uniform node layout, see Fig. 4. The reason for introducing a non-uniform node layout is that we can cluster nodes where we are most interested in having an accurate solution. In general, we are most interested in having an accurate solution in the neighborhood of $s = K$, which is also where the truncation error is largest due to large derivatives in the solution, see (29).

The non-uniform node distribution is generated as introduced in [38]. We start with N equidistant nodes $x^{(1)} < \cdots < x^{(i)} < \cdots < x^{(N)}$ constructed by

$$x^{(i)} = \operatorname{arcsinh}\left(-\frac{K}{c}\right) + (i-1)\Delta x, \quad i = 1, \ldots, N, \tag{33}$$

where c is a positive real constant which specifies how dense the node distribution becomes around the strike price K,

$$\Delta x = \frac{1}{N}\left[\operatorname{arcsinh}\left(\frac{s_{\max} - K}{c}\right) - \operatorname{arcsinh}\left(-\frac{K}{c}\right)\right],$$



(a) Uniform node layout.

(b) Non-uniform node layout.

Fig. 4. Uniform and non-uniform node layouts in 1D, the boundary conditions are employed in the blue triangle node ($u = 0$) and in the red square node (the far-field boundary condition).

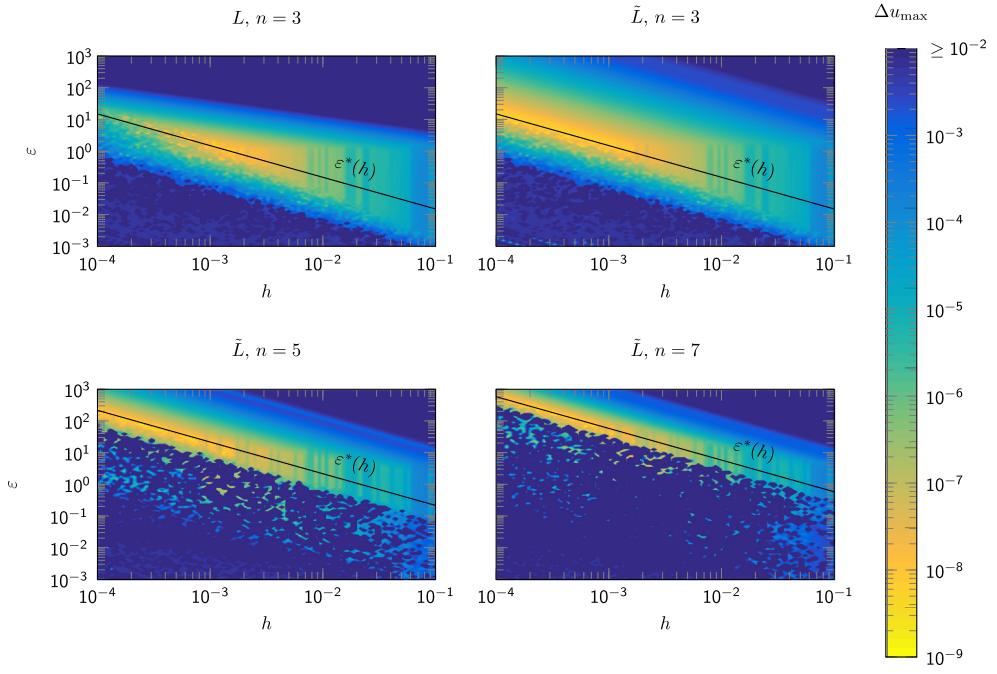


Fig. 5. Error $\Delta u_{\max}$ in the solution as a function of h and ε. Upper-left corner: approximation L defined in (6) and $n = 3$, upper-right corner: approximation $\tilde{L}$ defined in (8) and $n = 3$, lower-left corner: approximation $\tilde{L}$ defined in (8) and $n = 5$, and lower-right corner: approximation $\tilde{L}$ defined in (8) and $n = 7$.

and $s_{\max}$ denotes the far-field boundary. Then, the non-uniform node distribution s is generated pointwise as

$$s^{(i)} = K + c \cdot \sinh(x^{(i)}), \quad i = 1, \ldots, N. \tag{34}$$

The density tuning parameter used for the non-uniform node layouts in 1D is $c = 0.4$. An example of a non-uniform node layout using this tuning parameter and $N = 51$ is shown in Fig. 4b.

We employ the far-field boundary condition in $s_{\max} = 4K$, which is in most practical cases far enough in order to keep the negative boundary effects from affecting the solution.

4.1.1. Selection of shape parameter ε

The choice of the shape parameter ε is crucial for the performance of the method. In this section we study how to select this parameter. To start with, we study the error in the solution as a function of ε and h for $n = 3, 5$, and 7. In all experiments we use $\Delta t \approx h/2$ since numerical experiments show that the spatial error dominates for these values. For larger values of Δt, the error from the temporal discretization affects the total error in the solution.

In Fig. 5 we display the error in the solution as a function of ε and h together with $\varepsilon^*(h)$ defined by (25) with γ_n^* for $n = 3$, 5, and 7 where $\gamma_3^* = 0.0015$, $\gamma_5^* = 0.0225$, $\gamma_7^* = 0.0585$. All errors $\geq 10^{-2}$ are presented as 10^{-2} in this figure in order to have a more detailed resolution in the region of interest. The values of γ_n^* are chosen such that $\varepsilon^*(h)$ always is close to the minimal error in the solution as a linear function of h.

We see that for a constant h, the error in the solution decreases with decreasing ε until it saturates and stays constant for a while. For $n = 3$ this agrees well with the theory in Section 3, see Eq. (23) for the local truncation error. For an even smaller ε, the linear system of equations in (9) becomes highly ill-conditioned which means that the parameters $\tilde{w}$ become inaccurate and hence also the computed solution u. We note that for larger n, we need to use a larger ε for a given h in order to have a well-conditioned system for the parameters $\tilde{w}$.

From Fig. 5 we conclude that for every constant value of ε we eventually run into problems with ill-conditioning when we decrease h. Also, the range in h where we have a small error $\Delta u_{\max}$, becomes more narrow when we increase the stencil size

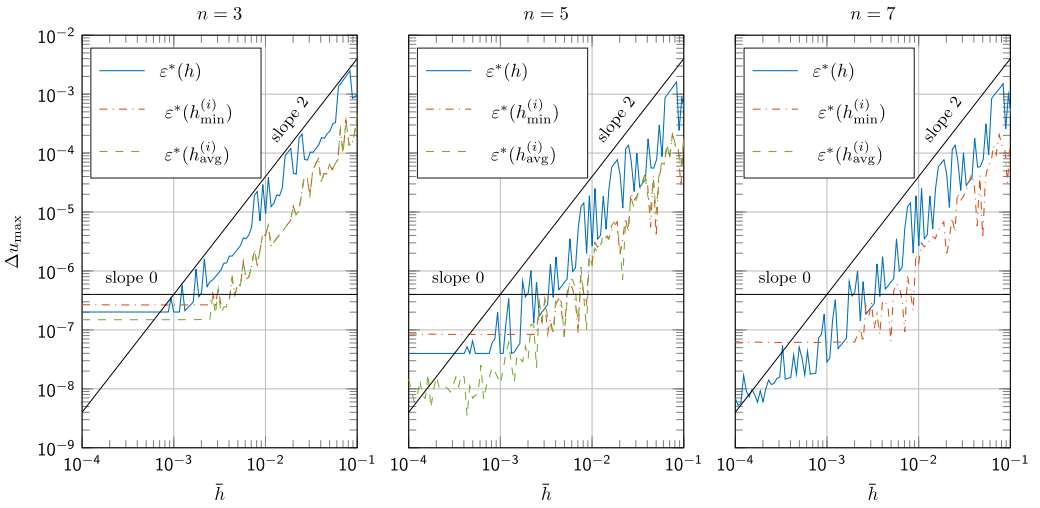


Fig. 6. Error $\Delta u_{\max}$ in the computed solution as a function of $\bar{h}$ for $n = 3$ (left), 5 (middle), and 7 (right), and each of those for uniform node layouts (blue solid lines) and non-uniform node layouts using $\varepsilon^*(h^{(i)}_{\min})$ (red dash-dotted lines), and $\varepsilon^*(h^{(i)}_{\text{avg}})$ (green dashed lines). The two solid black lines mark slopes 2 (right) and 0 (left). Note that for $n = 7$ and $\varepsilon^*(h^{(i)}_{\text{avg}})$, we run into problems with ill-conditioning for small h.

n. We note that for the approximation L in (6) and $n = 3$ (upper-left corner), $\varepsilon^*(h)$ is a suitable choice. However, as predicted by the theory in Section 3, for small values of h the error starts to increase with decreasing h which is not a good property of the method. For the approximation $\tilde{L}$ in (8) the error decreases with decreasing h until the error saturates and remains constant. Still, the error for all $n = 3, 5$, and 7 (upper-right, lower-left, and lower-right corner respectively) has a saturation level well below 10^{-6}. We consider the behavior of this approximation together with our choice of $\varepsilon^*(h)$ as a suitable method for our purposes. We have numerically verified that $\varepsilon^*(h)$ is not strongly dependent of the problem parameters r, σ, K and T and hence is a good choice independent of these parameters.

4.1.2. Convergence and computational time

Next, we study and compare the convergence behavior of our proposed method for different n and different node layouts. For the non-uniform node layouts we define $h^{(i)}_{\min}$ and $h^{(i)}_{\text{avg}}$ as the minimal and average distances between all pair of nodes in $\mathcal{S}^{(i)}$ and use $\varepsilon^*(h^{(i)}_{\min})$ and $\varepsilon^*(h^{(i)}_{\text{avg}})$ in the computation of the weights for $\mathcal{S}^{(i)}$. We also define the average distance between the nodes in the node layout $\bar{h} = s_{\max}/(N-1)$. This way we can compare the performance of the methods using different types of node layouts. We note that for uniform node layouts $\bar{h} = h$. In Fig. 6 we show the error in the computed solution as a function of $\bar{h}$ for $n = 3, 5$, and 7 for both uniform and non-uniform node layouts. Again, we use $\Delta t \approx \bar{h}/2$ in order to have a spatial error that dominates over the temporal error.

From Fig. 6, first of all we conclude that for all nine combinations and $\bar{h}$ larger than approximately $1.5 \cdot 10^{-3}$, the method is second-order accurate. Below this value of $\bar{h}$, the error stays constant at a saturation level in the order of $[10^{-8}, 5 \cdot 10^{-7}]$ depending on what computational node layout and stencil size we are considering. For $n = 3$ and uniform layouts this agrees with (29), where the terms that are of order $\mathcal{O}(h^2)$ dominate for large h and the terms that are of order $\mathcal{O}(\gamma^2)$ (which is a constant) dominate for small h. The reason why we cannot expect a higher order convergence than 2 for $n = 5$ and 7 is the discontinuity in the first derivative of the initial condition.

However, for ε chosen as a constant (24), we observe convergence orders of $n - 1$. The errors in these cases are larger than the error in Fig. 6, i.e. they originate from an error term that is larger than the dominating terms in (29). Hence, this is not the type of convergence behavior that we are interested in when looking for methods with higher order convergence. We will study high-order approximations using RBF-FD in a forthcoming paper by smoothing of the initial condition, [39]. Finally, we see from Fig. 6 that the smallest error is obtained by using $\varepsilon^*(h^{(i)}_{\text{avg}})$ and $n = 5$. However, for $n = 7$ we run into problems with ill-conditioning when using $\varepsilon^*(h^{(i)}_{\text{avg}})$. We note that for large n, in this respect it is safer to use $\varepsilon^*(h^{(i)}_{\min})$ when using non-uniform node layouts.

In the left-hand side plot of Fig. 7 we present the error in the solution as a function of $\bar{h}$ for $n = 5$, using a uniform node layout with $\varepsilon^*(h)$ and a constant $\varepsilon = 43.81$ (heuristically chosen as a good value from numerical results), as well as a non-uniform node layout with $\varepsilon^*(h^{(i)}_{\min})$ and $\varepsilon^*(h^{(i)}_{\text{avg}})$. We also show the error from a standard centered second-order FD method. In the right-hand side plot of Fig. 7 we display the computational time as a function of the error for the same methods.

Fig. 7 shows that the error in the solution using the RBF-FD methods with $\varepsilon^*(h)$, $\varepsilon^*(h^{(i)}_{\min})$ and $\varepsilon^*(h^{(i)}_{\text{avg}})$ are in most cases smaller than the error using FD. However, the error from using FD never saturates but keeps decreasing as $\mathcal{O}(h^2)$, while the

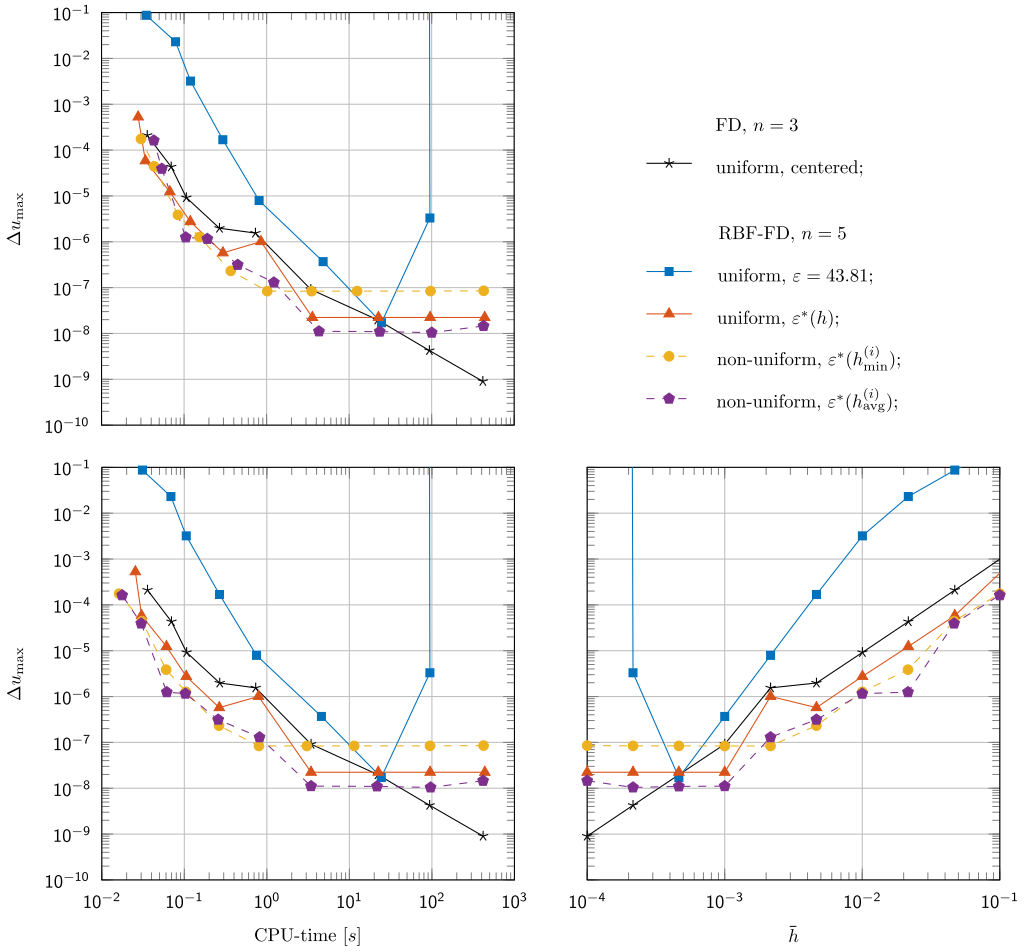


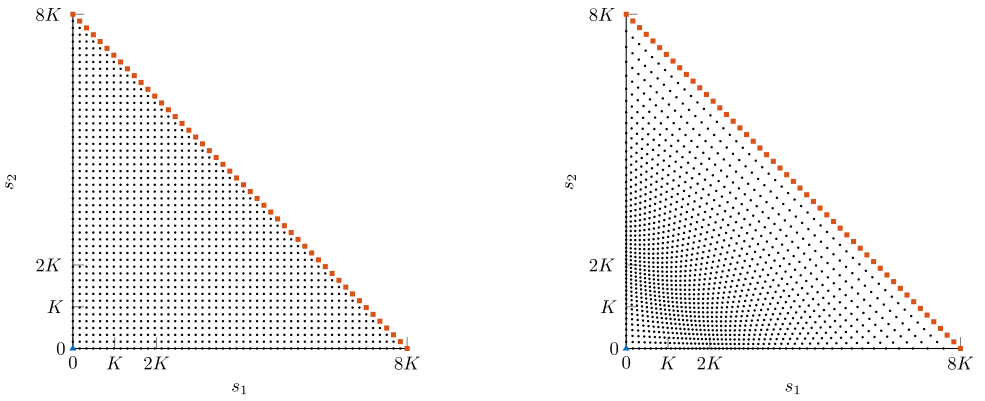
Fig. 7. Comparison of RBF-FD with the standard FD method for the 1D European call option defined by parameters given in Table 1 with respect to CPU-time (on the left) and spatial convergence (on the right). The upper-left figure shows CPU-time including the computation of RBF-FD weights, while the lower-left figure shows CPU-time not counting computation of the RBF-FD weights.

RBF-FD methods with $\varepsilon^*(h)$, $\varepsilon^*(h^{(i)}_{\min})$ and $\varepsilon^*(h^{(i)}_{\text{avg}})$ saturate but at a very small error level. RBF-FD with $\varepsilon = 43.81$ is the least or close to the least favorable method in the whole range of node distances. We observe the fourth order convergence for large $\bar{h}$ as we described above, but note that this error is larger than the error for the other methods. We also see that when we refine the node layout, we soon run into problems with ill-conditioning. Numerical experiments show that for every choice of constant ε we have problems with ill-conditioning when refining the node layout. This can be seen in e.g. Fig. 5. From this we conclude that our proposed method using ε as in (25) outperforms the usage of a constant value of ε which was used in previous papers on option pricing with RBF-FD, [19–23].

For the non-uniform node layouts each node has different weights while they are the same in the uniform node layouts. Hence, the initial time to compute the weights is larger for non-uniform node layouts than for uniform ones. This can be seen in the plots of Fig. 7, for a given $\bar{h}$ the errors for the non-equidistant node layouts are in general smaller than for the uniform node layout. Still, the computational time to reach a certain error is almost comparable for uniform and non-uniform node layouts (top left plot). However, when the time to compute the weights is excluded in the CPU time (bottom left plot), the non-uniform node layouts are slightly faster than the uniform ones to reach a certain error in the computed solution. Also, all approximations using ε^* defined in (25) are superior to FD for $\Delta u_{\max} \in [10^{-4}, 10^{-7}]$ and the superiority is even more pronounced when the weights are already computed. Note that the weights for the approximations of (19) and (20) are only determined by the node layout. Hence, these can be computed once for each node layout and then used to price many different options.

4.2. A two-dimensional option pricing problem

Next, we study a two-dimensional problem. It should be noted that unlike with classical grid based methods (e.g. standard FD-methods) we do not need to use a rectangular domain which is standard practice. Instead, just the lower-triangular



(a) Uniform node layout.

(b) Non-uniform node layout.

Fig. 8. Uniform and non-uniform computational node layouts in 2D, the boundary conditions are employed in the blue triangle node ($u = 0$) and in the red square nodes (the far-field boundary condition).

half of the rectangle can be used, see Fig. 8. This reduces the number of computational nodes by a factor 2 and hence the computational complexity significantly. The uniform and non-uniform node layouts are defined as analogues to the one-dimensional case. The non-uniform node layout is generated by using the one-dimensional node layouts in (33) and (34) along the axes s_1 and s_2, and then uniformly placing the internal points in the diagonal direction. The number of nodes along each diagonal is increased by one for each diagonal since this way a uniform node layout is aligned in the directions parallel to the axes and can be more easily compared to a Cartesian grid. We also implemented uniform staggered node layouts, but numerical experiments show that they are equal or inferior to the non-staggered ones. For the 2D problem, the far-field boundary is located at $s_1 + s_2 = s_{\max} = 8K$.

The nearest neighbors for constructing the stencils are efficiently determined using the k-D tree algorithm, [40]. In Fig. 1 we show examples of stencils of different sizes. In the numerical experiments we used $c = 4/3$ in (33) and (34) for non-uniform node layouts.

We have numerically computed suitable values of shape-parameters $\varepsilon^*(h)$ following the strategy in Section 4.1.1 for $n = 9$, 13, and 25 in (25) which gives $\gamma_9^* = 0.022$, $\gamma_{13}^* = 0.029$, and $\gamma_{25}^* = 0.090$. Note that here h determines the distance between nodes in the two-dimensional space. Due to problems with ill-conditioning when using $\varepsilon^*(h_{\text{avg}}^{(i)})$ for the 2D problems, we only use $\varepsilon^*(h_{\min}^{(i)})$ here.

In the comparisons between RBF-FD and FD, we use the full rectangle $[0, 8K] \times [0, 8K]$ as computational domain for FD. Also, we define $\bar{h} = s_{\max}/(\sqrt{\lceil 2N - \sqrt{2N} \rceil} - 1)$ for RBF-FD because that would be a characteristic distance between the nodes if they would be filling out the chosen computational domain uniformly and $\bar{h} = s_{\max}/(\sqrt{N} - 1)$ for FD since we are using an equidistant Cartesian grid in that case. Again, we use $\Delta t \approx \bar{h}/2$ with the same argument as in the 1D case.

4.2.1. A European call option

In this section we consider a two-dimensional European call option with parameters given in Table 1. Fig. 9 shows the error $\Delta u_{\max}$ as a function of $\bar{h}$ and CPU-time using $n = 9$, 13, and 25 for both uniform and non-uniform node layouts together with standard centered second-order FD.

From the right-hand side plot in Fig. 9, we see that for large $\bar{h}$, the error is smallest for RBF-FD with non-uniform node layouts and $n = 25$. When $\bar{h}$ is below $5 \cdot 10^{-2}$, this error saturates around $5 \cdot 10^{-5}$. Eventually this method runs into problems with ill-conditioning when we continue to decrease $\bar{h}$.

For smaller $\bar{h}$, RBF-FD with non-uniform node layouts and $n = 13$ has the smallest error until also this method saturates. We note that all RBF-FD methods eventually saturate while the FD method has an error that continues to decrease with decreasing $\bar{h}$. However, we consider an error below 10^{-4} to be small enough. When it comes to computational times (top left plot of Fig. 9), we note that for the larger errors, FD is the best choice. This is due to the fact that for RBF-FD, the computation of the weights takes a large part of the CPU-time when we have fewer nodes and hence also time steps. For smaller errors RBF-FD and FD perform similarly when it comes to CPU-time to reach a certain error when the time to compute the weights is included. However, if the weights are precomputed, the CPU-times for RBF-FD are much smaller than for FD (bottom left plot). This is useful when we want to price many options using the same node layout, see Section 4.1.2. Also, in general it is more efficient to use a non-uniform layout in the sense that the CPU-time to reach a certain error is smaller. Finally, when going to higher dimensions it is important to have high accuracy while keeping demand on the memory as low as possible. From the bottom right plot of Fig. 9 we conclude that the non-uniform node layouts are in general better to use compared to the uniform ones also in this respect.

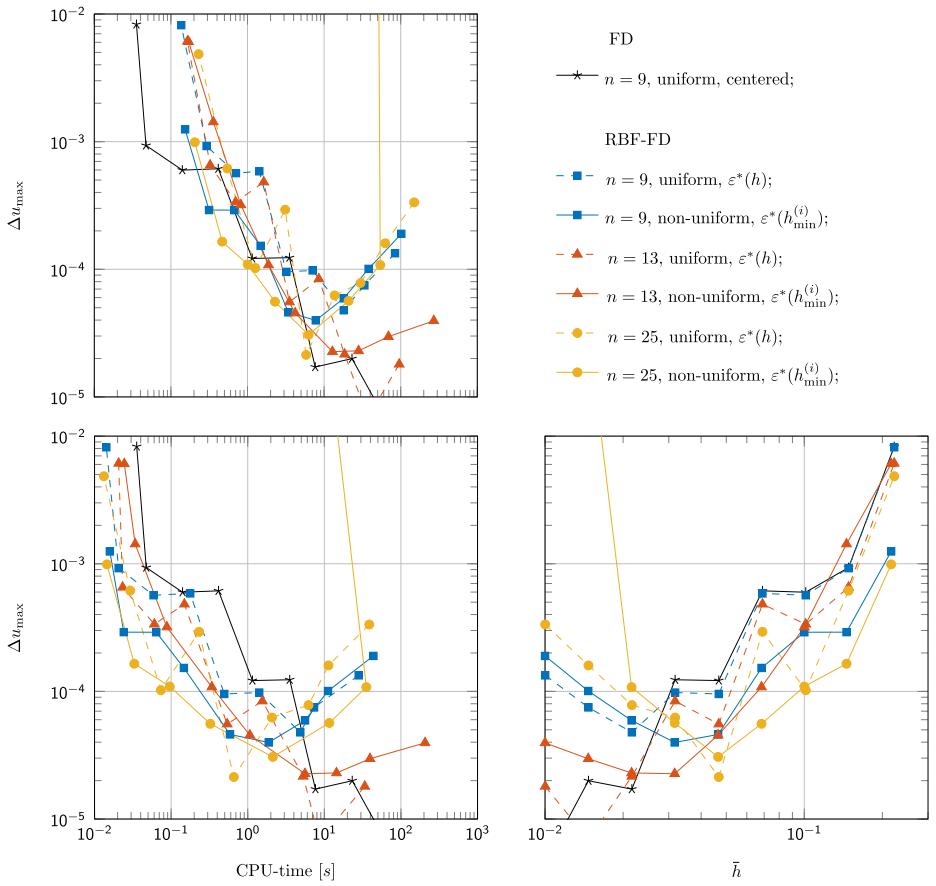


Fig. 9. Comparison of RBF-FD with the standard FD method for the 2D European call basket option (defined by parameters given in Table 1), with respect to CPU-time (on the left) and spatial convergence (on the right). The upper-left figure shows CPU-time including the computation of RBF-FD weights, while the lower-left figure shows CPU-time not counting computation of the RBF-FD weights.

4.2.2. An American put option

Next we turn our attention to an American put option with the same parameters as for the European call option. In Fig. 10 we display the error $\Delta u_{\max}$ as a function of $\bar{h}$ and the same error as a function of CPU-time.

From the plots of Fig. 10 we see that we can draw similar conclusions for the American put option as for the European call option.

4.3. A three-dimensional option pricing problem

Here we present the numerical solution of a three-dimensional European call option to demonstrate the computational strength and flexibility of our method.

The parameters used are presented in Table 1 and the uniform node layout used in Fig. 11. If we let N_s present the number of nodes we use to discretize space in one dimension, we note that in the 3D case we only need to solve the problem using approximately $N_s^3/6$ nodes as opposed to N_s^3 nodes for FD. In the D-dimensional case, this generalizes to $N_s^D/D!$ for RBF-FD compared to N_s^D for FD. In Fig. 12 we present the numerical solution. It took 1.55 seconds of CPU-time to compute the solution using $N = 4960$ nodes, $n = 15$, and $\varepsilon = 0.05$.

5. Conclusions

In this paper we study numerical pricing of European and American type multi-asset options using an RBF-FD method in space (together with operator splitting for American options) and BDF-2 in time. In contrast to previous papers using RBF-FD to price options we add a constant in the approximation of the spatial operator as in (8). Also, instead of using a constant ε as in previous papers on the topic, we use an ε that is proportional to h^{-1} as in (25) and we numerically determine suitable values of γ for $n = 3, 5$, and 7 in 1D and $n = 9$, 13, and 25 in 2D. By using our proposed parameters, the RBF-FD method to price options is robust and reliable, and we avoid problems with ill-conditioning.

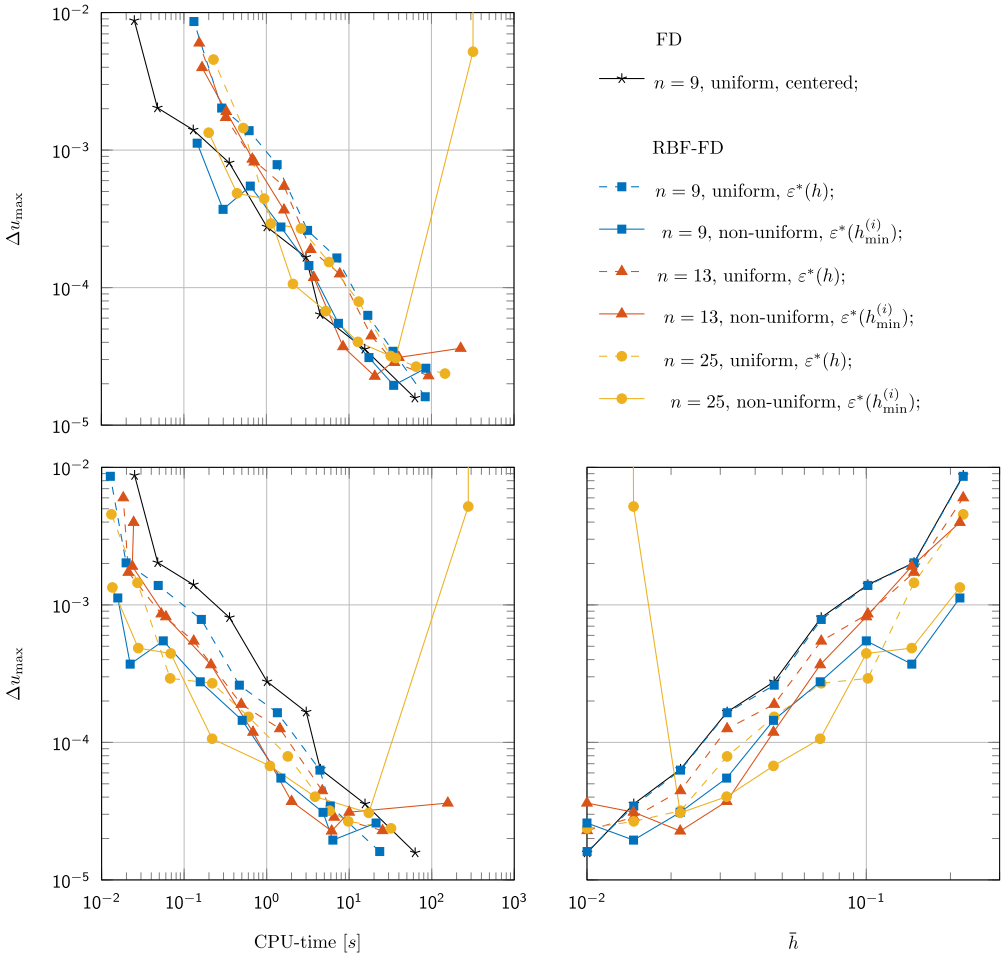


Fig. 10. Comparison of RBF-FD with the standard FD method for the 2D American put basket option (defined by parameters given in Table 1), with respect to CPU-time (on the left) and spatial convergence (on the right). The upper-left figure shows CPU-time including the computation of RBF-FD weights, while the lower-left figure shows CPU-time not counting computation of the RBF-FD weights.

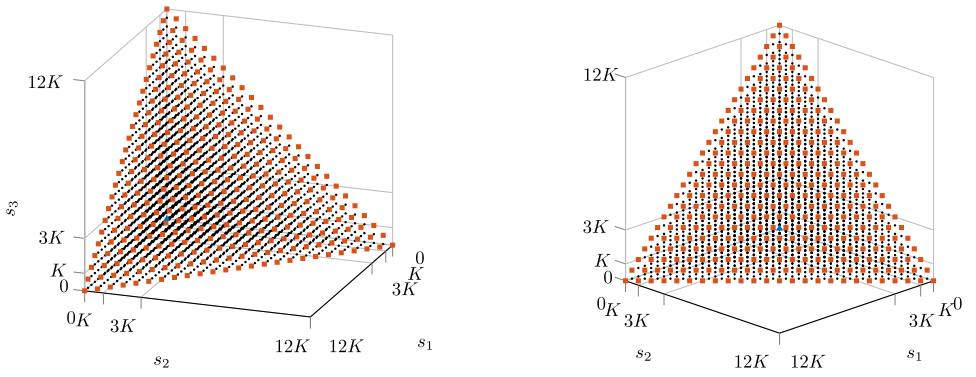


Fig. 11. Uniform node layout in 3D from two angles, the boundary conditions are employed in the blue triangle node ($u = 0$) and in the red square nodes (the far-field boundary condition).

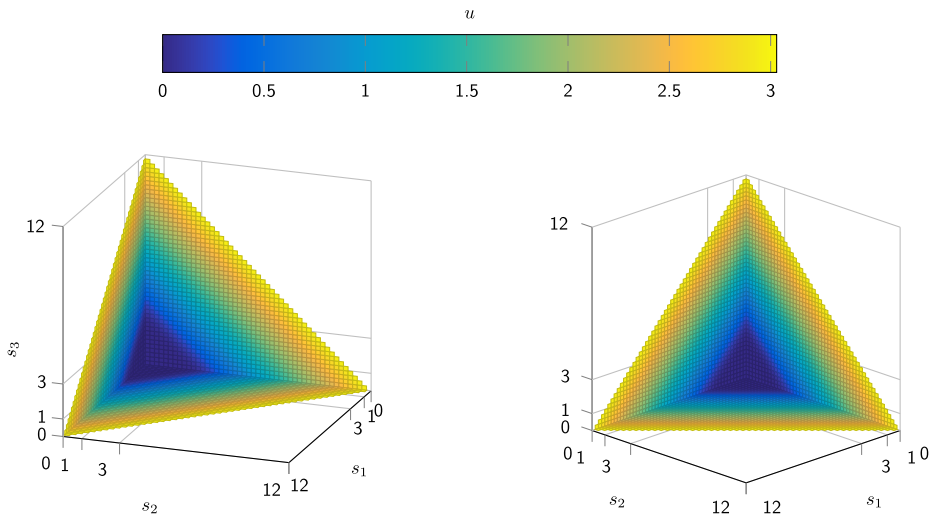


Fig. 12. Solution in 3D from two angles.

By introducing a non-uniform node layout we further exploit the possibilities of RBF-FD. As opposed to standard FD that rely on logically Cartesian grids, we can use irregular node distributions in domains that are not of (hyper-) rectangular shape. This way we can refine the node layout in areas where we are most interested in having an accurate solution and only have computational nodes in areas of interest, e.g. refining around the strike and using a triangular computational domain instead of a rectangular one in our 2D case. We numerically verify that in many cases a non-uniform node layout is the best choice, especially if a high accuracy in the solution is required and this can be even further improved by developing a truly adaptive node placement. Moreover, our proposed RBF-FD method performs better than standard second order FD for most error tolerances in the final solution.

We present numerical results in 1D, 2D, and 3D that demonstrate the good properties of our method. Due to the sparsity of the discretization matrix and the possibility of local node refinement, we believe that our suggested method has the potential to work well also in higher dimensions.

Acknowledgments

The computations are performed on resources provided by SNIC through Uppsala Multidisciplinary Center for Advanced Computational Science (UPPMAX) under Project SNIC 2015/6-159.

The authors are grateful for valuable discussions with Elisabeth Larsson from our department and Erik Lehto from the Royal Institute of Technology.

Appendix. Derivation of truncation errors

To start with, we study approximation (6) of the first-order derivative, i.e. $\mathcal{L}^I u = \frac{\partial u}{\partial s}$ which gives

$$\begin{pmatrix} 1 & e^{-\varepsilon^2 h^2} & e^{-4\varepsilon^2 h^2} \\ e^{-\varepsilon^2 h^2} & 1 & e^{-\varepsilon^2 h^2} \\ e^{-4\varepsilon^2 h^2} & e^{-\varepsilon^2 h^2} & 1 \end{pmatrix} \begin{pmatrix} w_1^I \\ w_2^I \\ w_3^I \end{pmatrix} = \begin{pmatrix} -2\varepsilon^2 h e^{-\varepsilon^2 h^2} \\ 0 \\ 2\varepsilon^2 h e^{-\varepsilon^2 h^2} \end{pmatrix}. \tag{A.1}$$

Since the nodes are equidistant we get that the weights will be the same for all $s^{(i)}$ and the super-index on the weights w_k^I indicates here that they are derived for the operator $\mathcal{L}^I$. The linear system of Eqs. (A.1) has the solution

$$\begin{pmatrix} w_1^I \\ w_2^I \\ w_3^I \end{pmatrix} = \begin{pmatrix} -\dfrac{2\varepsilon^2 e^{3\varepsilon^2 h^2}}{-1 + e^{4\varepsilon^2 h^2}} \\ 0 \\ \dfrac{2\varepsilon^2 e^{3\varepsilon^2 h^2}}{-1 + e^{4\varepsilon^2 h^2}} \end{pmatrix}.$$

Taylor expanding both the numerator and denominator in w_1^l for small ε^2h^2 gives

$$w_1^l = -\frac{1}{2h}\left(\frac{1+3\varepsilon^2h^2+\frac{(3\varepsilon^2h^2)^2}{2}+\frac{(3\varepsilon^2h^2)^3}{6}+\frac{(3\varepsilon^2h^2)^4}{24}+\mathcal{O}((\varepsilon^2h^2)^4)}{1+2\varepsilon^2h^2+\frac{8}{3}(\varepsilon^2h^2)^2+\frac{8}{3}(\varepsilon^2h^2)^2+\mathcal{O}((\varepsilon^2h^2)^4)}\right).$$

Defining

$$\begin{aligned} C &= \left(\frac{1+3\varepsilon^2h^2+\frac{(3\varepsilon^2h^2)^2}{2}+\frac{(3\varepsilon^2h^2)^3}{6}+\frac{(3\varepsilon^2h^2)^4}{24}+\mathcal{O}((\varepsilon^2h^2)^4)}{1+2\varepsilon^2h^2+\frac{8}{3}(\varepsilon^2h^2)^2+\frac{8}{3}(\varepsilon^2h^2)^2+\mathcal{O}((\varepsilon^2h^2)^4)}\right) \\ &= c_0+c_1\varepsilon^2h^2+c_2(\varepsilon^2h^2)^2+c_3(\varepsilon^2h^2)^3+\mathcal{O}((\varepsilon^2h^2)^4), \end{aligned}$$

gives

$$\begin{aligned} &1+3\varepsilon^2h^2+\frac{(3\varepsilon^2h^2)^2}{2}+\frac{(3\varepsilon^2h^2)^3}{6}+\frac{(3\varepsilon^2h^2)^4}{24}+\mathcal{O}((\varepsilon^2h^2)^4)= \\ &\left(1+2\varepsilon^2h^2+\frac{8}{3}(\varepsilon^2h^2)^2+\frac{8}{3}(\varepsilon^2h^2)^2+\mathcal{O}((\varepsilon^2h^2)^4)\right)\times \\ &\left(c_0+c_1\varepsilon^2h^2+c_2(\varepsilon^2h^2)^2+c_3(\varepsilon^2h^2)^3+\mathcal{O}((\varepsilon^2h^2)^4)\right). \end{aligned} \tag{A.2}$$

Identifying terms in (A.2) leads to

$$\begin{aligned} c_0 &= 1, \\ 2c_0+c_1 &= 3, \\ \frac{8}{3}c_0+2c_1+c_2 &= 9/2, \\ \frac{8}{3}c_0+\frac{8}{3}c_1+2c_2+c_3 &= 27/6, \end{aligned}$$

with the solution $c_0=1,\ c_1=1,\ c_2=-1/6,\ c_3=-1/2$. Summing this up we get that

$$w_1^l = -\frac{1}{2h}\left(1+\varepsilon^2h^2-\frac{1}{6}(\varepsilon^2h^2)^2-\frac{1}{2}(\varepsilon^2h^2)^3+\mathcal{O}((\varepsilon^2h^2)^4)\right).$$

Proceeding in the same way for w_3^l, we get

$$w_3^l = \frac{1}{2h}\left(1+\varepsilon^2h^2-\frac{1}{6}(\varepsilon^2h^2)^2-\frac{1}{2}(\varepsilon^2h^2)^3+\mathcal{O}((\varepsilon^2h^2)^4)\right).$$

Hence, our approximation L^lu of the operator $\mathcal{L}^lu=\frac{\partial u}{\partial s}$ is defined by

$$\begin{aligned} L^lu &= \frac{1}{2h}\left(1+\varepsilon^2h^2-\frac{1}{6}(\varepsilon^2h^2)^2-\frac{1}{2}(\varepsilon^2h^2)^3+\mathcal{O}((\varepsilon^2h^2)^4)\right)u(s^{(i)}+h) \\ &- \frac{1}{2h}\left(1+\varepsilon^2h^2-\frac{1}{6}(\varepsilon^2h^2)^2-\frac{1}{2}(\varepsilon^2h^2)^3+\mathcal{O}((\varepsilon^2h^2)^4)\right)u(s^{(i)}-h). \end{aligned} \tag{A.3}$$

A Taylor expansion of u around $s^{(i)}$, denoting $u(s^{(i)})=u$, gives

$$\begin{aligned} u(s^{(i)}+h) &= u+h\frac{\partial u}{\partial s}+\frac{h^2}{2}\frac{\partial^2u}{\partial s^2}+\frac{h^3}{6}\frac{\partial^3u}{\partial s^3}+\frac{h^4}{24}\frac{\partial^4u}{\partial s^4}+\frac{h^5}{120}\frac{\partial^5u}{\partial s^5}+\mathcal{O}(h^6), \\ u(s^{(i)}-h) &= u-h\frac{\partial u}{\partial s}+\frac{h^2}{2}\frac{\partial^2u}{\partial s^2}-\frac{h^3}{6}\frac{\partial^3u}{\partial s^3}+\frac{h^4}{24}\frac{\partial^4u}{\partial s^4}-\frac{h^5}{120}\frac{\partial^5u}{\partial s^5}+\mathcal{O}(h^6), \end{aligned} \tag{A.4}$$

and insertion in (A.3) leads to

$$\begin{aligned} L^I u &= \frac{1}{2h}\left(1+\varepsilon^2h^2-\frac{1}{6}(\varepsilon^2h^2)^2-\frac{1}{2}(\varepsilon^2h^2)^3+\mathcal{O}((\varepsilon^2h^2)^4)\right)\times \\ &\quad \left(u+h\frac{\partial u}{\partial s}+\frac{h^2}{2}\frac{\partial^2 u}{\partial s^2}+\frac{h^3}{6}\frac{\partial^3 u}{\partial s^3}+\frac{h^4}{24}\frac{\partial^4 u}{\partial s^4}+\frac{h^5}{120}\frac{\partial^5 u}{\partial s^5}+\mathcal{O}(h^6)\right) \\ &- \frac{1}{2h}\left(1+\varepsilon^2h^2-\frac{1}{6}(\varepsilon^2h^2)^2-\frac{1}{2}(\varepsilon^2h^2)^3+\mathcal{O}((\varepsilon^2h^2)^4)\right)\times \\ &\quad \left(u-h\frac{\partial u}{\partial s}+\frac{h^2}{2}\frac{\partial^2 u}{\partial s^2}-\frac{h^3}{6}\frac{\partial^3 u}{\partial s^3}+\frac{h^4}{24}\frac{\partial^4 u}{\partial s^4}-\frac{h^5}{120}\frac{\partial^5 u}{\partial s^5}+\mathcal{O}(h^6)\right) \\ &= \frac{\partial u}{\partial s}+\frac{h^2}{6}\frac{\partial^3 u}{\partial s^3}+\frac{h^4}{120}\frac{\partial^5 u}{\partial s^5}+\varepsilon^2h^2\frac{\partial u}{\partial s}+(\varepsilon^2h^2)\frac{h^2}{6}\frac{\partial^3 u}{\partial s^3}-\frac{1}{6}(\varepsilon^2h^2)^2\frac{\partial u}{\partial s} \\ &\quad +\mathcal{O}(\text{h.o.}), \end{aligned} \tag{A.5}$$

where h.o. stands for higher order terms. From Eq. (A.5) we draw the conclusion that the local truncation error τ^I is defined by

$$\tau^I = \frac{h^2}{6}\frac{\partial^3 u}{\partial s^3}+\frac{h^4}{120}\frac{\partial^5 u}{\partial s^5}+\varepsilon^2h^2\frac{\partial u}{\partial s}+(\varepsilon^2h^2)\frac{h^2}{6}\frac{\partial^3 u}{\partial s^3}-\frac{1}{6}(\varepsilon^2h^2)^2\frac{\partial u}{\partial s}+\mathcal{O}(\text{h.o.}). \tag{A.6}$$

We recognize the first two terms as the leading terms in the local truncation error from a second order standard finite difference

$$\frac{\partial u}{\partial s}\approx\frac{u(s^{(i)}+h)-u(s^{(i)}-h)}{2h}.$$

When (6) is used to approximate $\mathcal{L}^{II}u=\frac{\partial^2 u}{\partial s^2}$, we get the following linear system of equations

$$\begin{pmatrix} 1 & e^{-\varepsilon^2h^2} & e^{-4\varepsilon^2h^2} \\ e^{-\varepsilon^2h^2} & 1 & e^{-\varepsilon^2h^2} \\ e^{-4\varepsilon^2h^2} & e^{-\varepsilon^2h^2} & 1 \end{pmatrix}\begin{pmatrix} w_1^{II} \\ w_2^{II} \\ w_3^{II} \end{pmatrix}=\begin{pmatrix} 4\varepsilon^4h^2e^{-\varepsilon^2h^2}-2\varepsilon^2e^{-\varepsilon^2h^2} \\ -2\varepsilon^2 \\ 4\varepsilon^4h^2e^{-\varepsilon^2h^2}-2\varepsilon^2e^{-\varepsilon^2h^2} \end{pmatrix} \tag{A.7}$$

with solution

$$\begin{pmatrix} w_1^{II} \\ w_2^{II} \\ w_3^{II} \end{pmatrix}=\begin{pmatrix} \dfrac{4\varepsilon^4h^2e^{3\varepsilon^2h^2}}{\left(-1+e^{2\varepsilon^2h^2}\right)^2} \\ -\dfrac{2\varepsilon^4(1-2e^{2\varepsilon^2h^2}+e^{4\varepsilon^2h^2}+4\varepsilon^2h^2e^{4\varepsilon^2h^2})}{\left(-1+e^{2\varepsilon^2h^2}\right)^2} \\ \dfrac{4\varepsilon^4h^2e^{3\varepsilon^2h^2}}{\left(-1+e^{2\varepsilon^2h^2}\right)^2} \end{pmatrix}.$$

Taylor expansions and identification of terms lead to

$$\begin{aligned} w_1^{II} &= \frac{1}{h^2}\left(1+\varepsilon^2h^2+\frac{1}{6}(\varepsilon^2h^2)^2-\frac{1}{6}(\varepsilon^2h^2)^3+\mathcal{O}((\varepsilon^2h^2)^4)\right), \\ w_2^{II} &= -\frac{2}{h^2}\left(1+\varepsilon^2h^2+\frac{25}{3}(\varepsilon^2h^2)^3+\mathcal{O}((\varepsilon^2h^2)^4)\right), \\ w_3^{II} &= \frac{1}{h^2}\left(1+\varepsilon^2h^2+\frac{1}{6}(\varepsilon^2h^2)^2-\frac{1}{6}(\varepsilon^2h^2)^3+\mathcal{O}((\varepsilon^2h^2)^4)\right), \end{aligned}$$

which together with (A.4) gives

$$\begin{aligned} W^{II}u &= \frac{1}{h^2}\left(1+\varepsilon^2h^2+\frac{1}{6}(\varepsilon^2h^2)^2-\frac{1}{6}(\varepsilon^2h^2)^3+\mathcal{O}((\varepsilon^2h^2)^4)\right)\times \\ &\quad \left(u+h\frac{\partial u}{\partial s}+\frac{h^2}{2}\frac{\partial^2 u}{\partial s^2}+\frac{h^3}{6}\frac{\partial^3 u}{\partial s^3}+\frac{h^4}{24}\frac{\partial^4 u}{\partial s^4}+\frac{h^5}{120}\frac{\partial^5 u}{\partial s^5}+\frac{h^6}{720}\frac{\partial^6 u}{\partial s^6}+\mathcal{O}(h^7)\right) \\ &\quad -\frac{2}{h^2}\left(1+\varepsilon^2h^2+\frac{25}{3}(\varepsilon^2h^2)^3+\mathcal{O}((\varepsilon^2h^2)^4)\right)\times u \end{aligned}$$

$$
\begin{aligned}
&+\frac{1}{h^2}\left(1+\varepsilon^2h^2+\frac{1}{6}(\varepsilon^2h^2)^2-\frac{1}{6}(\varepsilon^2h^2)^3+\mathcal{O}((\varepsilon^2h^2)^4)\right)\times\\
&\left(u-h\frac{\partial u}{\partial s}+\frac{h^2}{2}\frac{\partial^2u}{\partial s^2}-\frac{h^3}{6}\frac{\partial^3u}{\partial s^3}+\frac{h^4}{24}\frac{\partial^4u}{\partial s^4}-\frac{h^5}{120}\frac{\partial^5u}{\partial s^5}+\frac{h^6}{720}\frac{\partial^6u}{\partial s^6}+\mathcal{O}(h^7)\right)\\
&=\frac{\partial^2u}{\partial s^2}+\frac{h^2}{12}\frac{\partial^4u}{\partial s^4}+\frac{h^4}{360}\frac{\partial^6u}{\partial s^6}+\frac{1}{3h^2}(\varepsilon^2h^2)^2u-\frac{17}{h^2}(\varepsilon^2h^2)^3u\\
&+(\varepsilon^2h^2)\frac{\partial^2u}{\partial s^2}+\frac{1}{6}(\varepsilon^2h^2)^2\frac{\partial^2u}{\partial s^2}+(\varepsilon^2h^2)\frac{h^2}{12}\frac{\partial^4u}{\partial s^4}+\mathcal{O}(\text{h.o}),
\end{aligned}
\tag{A.8}
$$

i.e. the local truncation error for the operator $\mathcal{L}^{II}$ is given by

$$
\begin{aligned}
\tau^{II}&=\frac{h^2}{12}\frac{\partial^4u}{\partial s^4}+\frac{h^4}{360}\frac{\partial^6u}{\partial s^6}+\frac{1}{3h^2}(\varepsilon^2h^2)^2u-\frac{17}{h^2}(\varepsilon^2h^2)^3u\\
&+\varepsilon^2h^2\frac{\partial^2u}{\partial s^2}+\frac{1}{6}(\varepsilon^2h^2)^2\frac{\partial^2u}{\partial s^2}+(\varepsilon^2h^2)\frac{h^2}{12}\frac{\partial^4u}{\partial s^4}+\mathcal{O}(\text{h.o.}).
\end{aligned}
\tag{A.9}
$$

Again, we recognize the first two terms as the leading term in the truncation error for standard second order finite differences

$$
\frac{\partial^2u}{\partial s^2}\approx\frac{u(s_i+h)-2u(s_i)+u(s_i-h)}{h^2}.
$$

We also note that the next two terms come from the fact that $w_2^{II}\neq-2w_{1,3}^{II}$.

Next, we turn our attention to the approximation (8) applied to $\mathcal{L}^Iu$

$$
\begin{pmatrix}1&e^{-\varepsilon^2h^2}&e^{-4\varepsilon^2h^2}&1\\e^{-\varepsilon^2h^2}&1&e^{-\varepsilon^2h^2}&1\\e^{-4\varepsilon^2h^2}&e^{-\varepsilon^2h^2}&1&1\\1&1&1&0\end{pmatrix}\begin{pmatrix}\tilde{w}_1^I\\\tilde{w}_2^I\\\tilde{w}_3^I\\\tilde{w}_4^I\end{pmatrix}=\begin{pmatrix}-2\varepsilon^2he^{-\varepsilon^2h^2}\\0\\2\varepsilon^2he^{-\varepsilon^2h^2}\\0\end{pmatrix},
\tag{A.10}
$$

which has solution

$$
\begin{pmatrix}\tilde{w}_1^I\\\tilde{w}_2^I\\\tilde{w}_3^I\\\tilde{w}_4^I\end{pmatrix}=\begin{pmatrix}-\dfrac{2\varepsilon^2e^{3\varepsilon^2h^2}}{-1+e^{4\varepsilon^2h^2}}\\0\\\dfrac{2\varepsilon^2e^{3\varepsilon^2h^2}}{-1+e^{4\varepsilon^2h^2}}\\0\end{pmatrix}.
$$

This means that the approximation (8) of $\mathcal{L}^I$ becomes the same as the approximation defined in (6). If we now consider approximation (8) of $\mathcal{L}^{II}$ we get

$$
\begin{pmatrix}1&e^{-\varepsilon^2h^2}&e^{-4\varepsilon^2h^2}&1\\e^{-\varepsilon^2h^2}&1&e^{-\varepsilon^2h^2}&1\\e^{-4\varepsilon^2h^2}&e^{-\varepsilon^2h^2}&1&1\\1&1&1&0\end{pmatrix}\begin{pmatrix}\tilde{w}_1^{II}\\\tilde{w}_2^{II}\\\tilde{w}_3^{II}\\\tilde{w}_4^{II}\end{pmatrix}=\begin{pmatrix}4\varepsilon^4h^2e^{-\varepsilon^2h^2}-2\varepsilon^2e^{-\varepsilon^2h^2}\\-2\varepsilon^2\\4\varepsilon^4h^2e^{-\varepsilon^2h^2}-2\varepsilon^2e^{-\varepsilon^2h^2}\\0\end{pmatrix},
$$

with solution

$$
\begin{pmatrix}\tilde{w}_1^{II}\\\tilde{w}_2^{II}\\\tilde{w}_3^{II}\\\tilde{w}_4^{II}\end{pmatrix}=\begin{pmatrix}\dfrac{2\varepsilon^2e^{3\varepsilon^2h^2}(e^{\varepsilon^2h^2}+2\varepsilon^2h^2-1)}{3e^{4\varepsilon^2h^2}-4e^{3\varepsilon^2h^2}+1}\\\dfrac{-4\varepsilon^2e^{3\varepsilon^2h^2}(e^{\varepsilon^2h^2}+2\varepsilon^2h^2-1)}{3e^{4\varepsilon^2h^2}-4e^{3\varepsilon^2h^2}+1}\\\dfrac{2\varepsilon^2e^{3\varepsilon^2h^2}(e^{\varepsilon^2h^2}+2\varepsilon^2h^2-1)}{3e^{4\varepsilon^2h^2}-4e^{3\varepsilon^2h^2}+1}\\\dfrac{2(\varepsilon^2e^{\varepsilon^2h^2}-\varepsilon^2e^{2\varepsilon^2h^2}-\varepsilon^2e^{3\varepsilon^2h^2}+\varepsilon^2+4\varepsilon^4h^2e^{2\varepsilon^2h^2})}{(e^{\varepsilon^2h^2}-1)(2e^{\varepsilon^2h^2}+3e^{2\varepsilon^2h^2}+1)}\end{pmatrix}.
$$

Taylor expanding the numerator and denominator in $\tilde{w}_1^{II}$ gives

$$\begin{aligned}\tilde{w}_1^{II} &= \varepsilon^2 \frac{6\varepsilon^2h^2 + 19(\varepsilon^2h^2)^2 + \frac{91}{3}(\varepsilon^2h^2)^3 + \frac{391}{12}(\varepsilon^2h^2)^4 + \mathcal{O}((\varepsilon^2h^2)^5)}{6(\varepsilon^2h^2)^2 + 14(\varepsilon^2h^2)^3 + \frac{37}{2}(\varepsilon^2h^2)^4 + \frac{35}{2}(\varepsilon^2h^2)^5 + \mathcal{O}((\varepsilon^2h^2)^6)} \\ &= \frac{1}{h^2} \cdot \frac{1 + \frac{19}{6}\varepsilon^2h^2 + \frac{91}{18}(\varepsilon^2h^2)^2 + \frac{391}{72}(\varepsilon^2h^2)^3 + \mathcal{O}((\varepsilon^2h^2)^4)}{1 + \frac{14}{6}\varepsilon^2h^2 + \frac{37}{12}(\varepsilon^2h^2)^2 + \frac{35}{12}(\varepsilon^2h^2)^3 + \mathcal{O}((\varepsilon^2h^2)^4)} \\ &= \frac{1}{h^2}\left(1 + \frac{5}{6}\varepsilon^2h^2 + \frac{1}{36}(\varepsilon^2h^2)^2 - \frac{13}{18}(\varepsilon^2h^2)^3 + \mathcal{O}((\varepsilon^2h^2)^4)\right).\end{aligned}$$

Since $\tilde{w}_2^{II} = -2\tilde{w}_1^{II} = -2\tilde{w}_3^{II}$ we get

$$\begin{aligned}\tilde{W}^{II}u = &\frac{1}{h^2}\left(1 + \frac{5}{6}\varepsilon^2h^2 + \frac{1}{36}(\varepsilon^2h^2)^2 - \frac{13}{18}(\varepsilon^2h^2)^3 + \mathcal{O}((\varepsilon^2h^2)^4)\right) \times \\ &\left(u + h\frac{\partial u}{\partial s} + \frac{h^2}{2}\frac{\partial^2 u}{\partial s^2} + \frac{h^3}{6}\frac{\partial^3 u}{\partial s^3} + \frac{h^4}{24}\frac{\partial^4 u}{\partial s^4} + \frac{h^5}{120}\frac{\partial^5 u}{\partial s^5} + \frac{h^6}{720}\frac{\partial^6 u}{\partial s^6} + \mathcal{O}(h^7)\right) \\ &- \frac{2}{h^2}\left(1 + \frac{5}{6}\varepsilon^2h^2 + \frac{1}{36}(\varepsilon^2h^2)^2 - \frac{13}{18}(\varepsilon^2h^2)^3 + \mathcal{O}((\varepsilon^2h^2)^4)\right) \times u \\ &+ \frac{1}{h^2}\left(1 + \frac{5}{6}\varepsilon^2h^2 + \frac{1}{36}(\varepsilon^2h^2)^2 - \frac{13}{18}(\varepsilon^2h^2)^3 + \mathcal{O}((\varepsilon^2h^2)^4)\right) \times \\ &\left(u - h\frac{\partial u}{\partial s} + \frac{h^2}{2}\frac{\partial^2 u}{\partial s^2} - \frac{h^3}{6}\frac{\partial^3 u}{\partial s^3} + \frac{h^4}{24}\frac{\partial^4 u}{\partial s^4} - \frac{h^5}{120}\frac{\partial^5 u}{\partial s^5} + \frac{h^6}{720}\frac{\partial^6 u}{\partial s^6} + \mathcal{O}(h^7)\right) \\ &= \frac{\partial^2 u}{\partial s^2} + \frac{h^2}{12}\frac{\partial^4 u}{\partial s^4} + \frac{h^4}{360}\frac{\partial^6 u}{\partial s^6} + \frac{5}{6}\varepsilon^2h^2\frac{\partial^2 u}{\partial s^2} + \frac{5}{6}\varepsilon^2h^2\frac{h^2}{12}\frac{\partial^4 u}{\partial s^4} + \frac{1}{36}(\varepsilon^2h^2)^2\frac{\partial^2 u}{\partial s^2} \\ &+ \mathcal{O}(\text{h.o}),\end{aligned} \tag{A.11}$$

i.e. the local discretization error $\tilde{\tau}^{II}$ is defined by

$$\begin{aligned}\tilde{\tau}^{II} = &\frac{h^2}{12}\frac{\partial^4 u}{\partial s^4} + \frac{h^4}{360}\frac{\partial^6 u}{\partial s^6} + \frac{5}{6}\varepsilon^2h^2\frac{\partial^2 u}{\partial s^2} + \frac{5}{6}(\varepsilon^2h^2)\frac{h^2}{12}\frac{\partial^4 u}{\partial s^4} \\ &+ \frac{1}{36}(\varepsilon^2h^2)^2\frac{\partial^2 u}{\partial s^2} + \mathcal{O}(\text{h.o.}).\end{aligned} \tag{A.12}$$

References

[1] F. Black, M. Scholes, The pricing of options and corporate liabilities, J. Polit. Econ. 81 (1973) 637–654.
[2] R.C. Merton, Theory of rational option pricing, Bell J. Econ. Manage. Sci. 4 (1973) 141–183.
[3] C.R.D. Tavella, Pricing Financial Instruments: The Finite Difference Method, John Wiley & Sons, Chichester, 2000.
[4] J. Persson, L. von Sydow, Pricing European multi-asset options using a space–time adaptive FD-method, Comput. Vis. Sci. 10 (4) (2007) 173–183.
[5] P. Lötstedt, J. Persson, L. von Sydow, J. Tysk, Space–time adaptive finite difference method for European multi-asset options, Comput. Math. Appl. 53 (8) (2007) 1159–1180.
[6] G. Linde, J. Persson, L. von Sydow, A highly accurate adaptive finite difference solver for the Black–Scholes equation, Int. J. Comput. Math. 86 (12) (2009) 2104–2121.
[7] G.E. Fasshauer, Meshfree Approximation Methods with MATLAB, Vol. 6, World Scientific, 2007.
[8] H. Wendland, Fast evaluation of radial basis functions: Methods based on partition of unity, in: Approximation Theory X: Wavelets, Splines, and Applications, Citeseer, 2002.
[9] A. Safdari-Vaighani, A. Heryudono, E. Larsson, A radial basis function partition of unity collocation method for convection–diffusion equations arising in financial applications, J. Sci. Comput. (2015) 1–27.
[10] V. Shcherbakov, E. Larsson, Radial basis function partition of unity methods for pricing vanilla basket options, Comput. Math. Appl. 71 (1) (2016) 185–200.
[11] V. Shcherbakov, Radial basis function partition of unity operator splitting method for pricing multi-asset American options, BIT Numer. Math. (2016) 1–23.
[12] A.I. Tolstykh, 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.
[13] G.B. Wright, B. Fornberg, Scattered node compact finite difference-type formulas generated from radial basis functions, J. Comput. Phys. 212 (1) (2006) 99–123.
[14] G. Chandhini, Y. Sanyasiraju, Local RBF-FD solutions for steady convection–diffusion problems, Internat. J. Numer. Methods Engrg. 72 (3) (2007) 352–378.
[15] Y. Shan, C. Shu, Z. Lu, Application of local MQ-DQ method to solve 3D incompressible viscous flows with curved boundary, Comput. Model. Eng. Sci. 25 (2) (2008) 99.
[16] D. Stevens, H. Power, M. Lees, H. Morvan, The use of PDE centres in the local RBF Hermitian method for 3D convective-diffusion problems, J. Comput. Phys. 228 (12) (2009) 4606–4624.
[17] N. Flyer, E. Lehto, S. Blaise, G.B. Wright, A. St-Cyr, A guide to RBF-generated finite differences for nonlinear transport: Shallow water simulations on a sphere, J. Comput. Phys. 231 (11) (2012) 4078–4095.

[18] B. Fornberg, N. Flyer, Solving PDEs with radial basis functions, Acta Numer. 24 (2015) 215–258.
[19] M.K. Kadalbajoo, A. Kumar, L.P. Tripathi, Application of radial basis function with L-stable Padé time marching scheme for pricing exotic option, Comput. Math. Appl. 66 (4) (2013) 500–511.
[20] M.K. Kadalbajoo, A. Kumar, L.P. Tripathi, Application of the local radial basis function-based finite difference method for pricing American options, Int. J. Comput. Math. 92 (8) (2015) 1608–1624.
[21] A. Kumar, L.P. Tripathi, M.K. Kadalbajoo, A numerical study of Asian option with radial basis functions based finite differences method, Eng. Anal. Bound. Elem. 50 (2015) 1–7.
[22] M.K. Kadalbajoo, A. Kumar, L.P. Tripathi, An efficient numerical method for pricing option under jump diffusion model, Int. J. Adv. Eng. Sci. Appl. Math. 7 (3) (2015) 114–123.
[23] A. Golbabai, E. Mohebianfar, A new stable local radial basis function approach for option pricing, Comput. Econ. (2016) 1–18.
[24] B. Fornberg, T. Driscoll, G. Wright, R. Charles, Observations on the behavior of radial basis function approximations near boundaries, Comput. Math. Appl. 43 (3) (2002) 473–490.
[25] B. Fornberg, E. Lehto, Stabilization of RBF-generated finite difference methods for convective PDEs, J. Comput. Phys. 230 (6) (2011) 2270–2285.
[26] N. Flyer, B. Fornberg, V. Bayona, G.A. Barnett, On the role of polynomials in RBF-FD approximations: I. interpolation and accuracy, J. Comput. Phys. 321 (2016) 21–38.
[27] E. Larsson, E. Lehto, A. Heryudono, B. Fornberg, Stable computation of differentiation matrices and scattered node stencils based on Gaussian radial basis functions, SIAM J. Sci. Comput. 35 (4) (2013) A2096–A2119.
[28] X. Wang, Finite Differences Based on Radial Basis Functions to Price Options, Vol. 2014, Uppsala University, 2014, p. 36 u.U.D.M. project report.
[29] B. Fornberg, E. Lehto, C. Powell, Stable calculation of Gaussian-based RBF-FD stencils, Comput. Math. Appl. 65 (4) (2013) 627–637.
[30] L. von Sydow, L. Josef Höök, E. Larsson, E. Lindström, S. Milovanović, J. Persson, V. Shcherbakov, Y. Shpolyanskiy, S. Sirén, J. Toivanen, et al., BENCHOP–the BENCHmarking project in option pricing, Int. J. Comput. Math. 92 (12) (2015) 2361–2379.
[31] E. Hairer, S. Nørsett, G. Wanner, Solving Ordinary Differential Equations I: Nonstiff Problems, Solving Ordinary Differential Equations, Springer, 2008.
[32] E. Larsson, K. Åhlander, A. Hall, Multi-dimensional option pricing using radial basis functions and the generalized Fourier transform, J. Comput. Appl. Math. 222 (1) (2008) 175–192.
[33] S. Ikonen, J. Toivanen, Operator splitting methods for American option pricing, Appl. Math. Lett. 17 (2004) 809–814.
[34] S. Ikonen, J. Toivanen, Operator splitting methods for pricing American options under stochastic volatility, Numer. Math. 113 (2) (2009) 299–324.
[35] S. Salmi, J. Toivanen, L. von Sydow, An IMEX-scheme for pricing options under stochastic volatility models with jumps, SIAM J. Sci. Comput. 36 (5) (2014) B817–B834.
[36] V. Bayona, M. Moscoso, M. Carretero, M. Kindelan, RBF-FD formulas and convergence properties, J. Comput. Phys. 229 (22) (2010) 8281–8295.
[37] Y. Saad, M.H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (3) (1986) 856–869.
[38] K. In't Hout, S. Foulon, ADI finite difference schemes for option pricing in the Heston model with correlation, Int. J. Numer. Anal. Model. 7 (2) (2010) 303–320.
[39] H.-O. Kreiss, V. Thomée, O. Widlund, Smoothing of initial data and rates of convergence for parabolic difference equations, Comm. Pure Appl. Math. 23 (2) (1970) 241–259.
[40] J.L. Bentley, Multidimensional binary search trees used for associative searching, Commun. ACM 18 (9) (1975) 509–517.