Pricing Derivatives under Multiple Stochastic Factors by Localized Radial Basis Function Methods

Slobodan Milovanović
Uppsala University
slobodan.milovanovic@it.uu.se

Victor Shcherbakov
Uppsala University
victor.shcherbakov@it.uu.se

August 14, 2018

Abstract

We propose two localized Radial Basis Function (RBF) methods, the Radial Basis Function Partition of Unity method (RBF–PUM) and the Radial Basis Function generated Finite Differences method (RBF–FD), for solving financial derivative pricing problems arising from market models with multiple stochastic factors. We demonstrate the useful features of the proposed methods, such as high accuracy, sparsity of the differentiation matrices, mesh-free nature and multi-dimensional extendability, and show how to apply these methods for solving time-dependent higher-dimensional PDEs in finance. We test these methods on several problems that incorporate stochastic asset, volatility, and interest rate dynamics by conducting numerical experiments. The results illustrate the capability of both methods to solve the problems to a sufficient accuracy within reasonable time. Both methods exhibit similar orders of convergence, which can be further improved by a more elaborate choice of the method parameters. Finally, we discuss the parallelization potentials of the proposed methods and report the speedup on the example of RBF–FD.

1 Introduction

Pricing derivatives and calibration of financial models are computationally very intensive tasks, which require a certain precision. Therefore, models that should be

used are the ones which allow for the most accurate representation of market features, such as volatility smiles or skews and fat tails of the return distributions. Often, such market models involve multiple stochastic factors, e.g., stochastic volatility and interest rate. Although, each additional stochastic factor gives rise to an extra dimension. Thus, having several stochastic factors leads to multi-dimensional problems. However, thanks to the development of modern computational hardware and numerical methods, multi-factor asset pricing models rapidly gain in popularity. In this paper we suggest a numerical approach for pricing derivatives based on radial basis function (RBF) methods.

RBF methods are mesh-free numerical methods that exhibit a high, and for smooth problems, even exponential convergence order of the approximate solutions [1, 2, 3, 4]. These methods are attractive from a computational point of view for higher-dimensional problems due to the simplicity of constructing interpolants using the radial basis. Moreover, RBF methods are able to achieve the desired accuracy using fewer computational nodes than, e.g., finite difference methods (FD) [5], thus, reducing the storage requirements.

RBF methods were adapted for financial applications in the late 1990s and early 2000s by several authors [6, 7, 8, 9]. However, they used a global RBF method with the basis functions globally supported over the entire computational domain. That approach resulted in a dense coefficient matrix of the system of linear equations that needs to be solved, therefore, preventing the method to be extended to higher dimensions because of the increase in computational complexity. To overcome this issue, several authors suggested to use localized RBF methods, such as the radial basis function partition of unity method (RBF–PUM) [5, 10, 11] and the radial basis function generated finite difference method (RBF–FD) [12, 13, 14, 15, 16, 17]. These method modifications allow for a significant sparsification of the coefficient matrix, yielding a higher computational efficiency. In fact, it was shown in [18] that RBF–PUM is more efficient than any other deterministic numerical method that relies on spatial discretization for higher-dimensional option pricing problems, e.g., for the Heston model, while RBF–FD at its current state of development is able to closely follow that result.

Thereby, we see a great potential of the localized RBF methods to be useful for pricing financial derivatives under models with several stochastic factors. In this paper, we demonstrate the performance of the RBF–PUM and RBF–FD methods on a set of pricing problems with multiple stochastic factors. The formulations of these problems are inspired by the BENCHOP project on stochastic and local volatility [19]. For the problems with available semi-analytical solutions, we compute reference values and present convergence tests and computational performance

comparisons, while for the other problems we provide solutions in the form of tables with values. Moreover, we discuss the parallelization potentials of our implementations and demonstrate the gains. Besides the positive highlights, we also draw attention to the limitations of the proposed methods and do not claim that these methods are suitable for solving problems with a very high dimensionality, e.g., larger than 10. For those cases, most often, Monte Carlo methods are still the only way to obtain the solutions.

Nevertheless, problems of moderate dimensionality, such as a problem of pricing a basket option on five assets, e.g., an option on the DAX index after dimension reduction [20], can be solved by both RBF–PUM and RBF–FD in a few seconds on an ordinary laptop [21]. Also, RBF–PUM has been successfully applied to a problem of valuing quanto CDS under a model with four stochastic factors [22].

The remainder of this paper is structured as follows. In Section 2, we define four models with multiple stochastic factors under which we price a European call option. In Section 3, we develop the RBF–PUM and RBF–FD approaches that we use for estimating the option values numerically. In Section 4, we present and discuss the numerical results. Finally, in Section 5 we conclude the discussion on our findings.

2 Models with Multiple Stochastic Factors

Models with multiple stochastic factors allow for much better reproduction of market features than the standard Black–Scholes formulation, which is known for missing out on some important market features, such as fat tails of return distributions, as well as volatility smiles and skews. Therefore, various models with local volatilities, local stochastic volatilities, stochastic volatilities of volatilities, stochastic interest rates, and combinations of the above, have become increasingly popular since they are able to capture such market phenomena.

In this section, we present four models with multiple stochastic factors that are often used for pricing options. Most of these model choices were inspired by [19]. In our setting we focus on European call options. However, the approach is not limited to this particular type of the payoff functions and can be easily extended to exotic payoffs as it is shown in [19].

We consider the models given a probability space $(\Omega, \mathcal{F}, \mathbb{Q})$ satisfying the standard assumptions, where Ω is the sample space, $(\mathcal{F}_t, t \geq 0)$ is the filtration under which the dynamics of the stochastic factors are adapted, and $\mathbb{Q}$ is the risk neutral probability measure.

2.1 The Quadratic Local Stochastic Volatility Model

The interest for local volatility models was triggered by the work of Dupire [23]. Ever since, such models have become increasingly popular among practitioners, because they allow for capturing the volatility features that are observed in the market data, such as smiles and skews. The first reference to the quadratic local stochastic volatility (QLSV) relates to the paper by Andersen [24], where he applied the model for option pricing.

The dynamics of the QLSV models read as follows

$$\mathrm{d}S_t = rS_t\mathrm{d}t + \sqrt{V_t}f(S_t)\mathrm{d}W_t^s, \quad S_0 = s, \tag{2.1}$$

$$\mathrm{d}V_t = \kappa(\eta - V_t)\mathrm{d}t + \sigma\sqrt{V_t}\mathrm{d}W_t^v, \quad V_0 = v, \tag{2.2}$$

where S_t is the stochastic 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_t^s and W_t^v are correlated Wiener processes with constant correlation ρ, i.e. $\langle \mathrm{d}W_t^s, \mathrm{d}W_t^v \rangle = \rho \mathrm{d}t$, and $f(s)$ is a quadratic volatility function that takes the form

$$f(s) = \frac{1}{2}\alpha s^2 + \beta s + \gamma, \tag{2.3}$$

with constant parameters α, β, and γ. The standard Heston model [25] is a particular case of the QLSV model with $\alpha = 0$, $\beta = 1$, and $\gamma = 0$.

By applying the Itô lemma and the Feynman–Kac theorem, a pricing partial differential equation (PDE) for the QLSV model can be derived

$$-\frac{\partial u}{\partial t} = \frac{1}{2}vf(s)^2\frac{\partial^2 u}{\partial s^2} + \rho\sigma v f(s)\frac{\partial^2 u}{\partial s \partial v} + \frac{1}{2}\sigma^2 v\frac{\partial^2 u}{\partial v^2} + rs\frac{\partial u}{\partial s} + \kappa(\eta - v)\frac{\partial u}{\partial v} - ru, \tag{2.4}$$

subject to the terminal condition

$$u(T, s, v) = \max(s - K, 0), \tag{2.5}$$

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

We set the following values for the model parameters

- Set 1: $\alpha = 0, \beta = 1, \gamma = 0$,
- Set 2: $\alpha = 2, \beta = 0, \gamma = 0$,

while the other parameters are identical for both testing sets and selected as

$$K = 1, T = 1, r = 0, \kappa = 2.58, \eta = 0.043, \sigma = 1, \rho = -0.36.$$

The Feller condition [26] is violated for both testing sets. We choose three evaluation points at which we measure and report the option value results

$$S_0 = 0.75, 1.00, 1.25; \qquad V_0 = 0.114.$$

2.2 The SABR Model

The stochastic alpha, beta, rho (SABR) model was developed by Hagan et al. [27] to attempt to capture the volatility smile in derivative markets. The model is an extension of the constant elasticity of variance (CEV) model with an assumption of the volatility being stochastic. The stochastic asset and volatility processes are defined by the following dynamics

$$\mathrm{d}S_t = V_t S_t^{\beta} \mathrm{d}W_t^s, \quad S_0 = s, \tag{2.6}$$
$$\mathrm{d}V_t = \sigma V_t \mathrm{d}W_t^v, \quad V_0 = v, \tag{2.7}$$

where β is the elasticity parameter and the other parameters are defined as in (2.1)–(2.2).

By applying the Itô lemma and the Feynman–Kac theorem we can derive a pricing PDE for the SABR model that takes the form

$$-\frac{\partial u}{\partial t} = \frac{1}{2} v^2 s^{2\beta} \frac{\partial^2 u}{\partial s^2} + \rho\sigma v^2 s^{\beta} \frac{\partial^2 u}{\partial s \partial v} + \frac{1}{2}\sigma^2 v^2 \frac{\partial^2 u}{\partial v^2} - ru, \tag{2.8}$$

subject to the terminal condition

$$u(T, s, v) = \max(s - K, 0). \tag{2.9}$$

For valuing the option we assume the model parameters to have the following values

$$K = 1, T = 1, r = 0, \sigma = 0.4, \beta = 0.5.$$

Here, we consider two test cases: zero-correlation between the asset and volatility for which there exists a semi-analytical solution [28] and nonzero-correlation between the asset and volatility for which some attempts to derive an approximate semi-analytical solution have been made [29] but the results are still rather poor:

- Set 1: $\rho = 0$,

- Set 2: $\rho = -0.5$.

The three evaluation points at which we measure and report the computed option values are

$$S = 0.75, 1.00, 1.25; \qquad V_0 = 0.200.$$

The SABR model is typically used for valuing interest rate derivatives.

2.3 The Heston–Hull–White Model

The Heston–Hull–White (HHW) model is an extension of the Heston stochastic volatility model [25] that is augmented with a stochastic interest rate that follows the Hull–White process [30]. This is an important extension since the market interest rates are non-constant. Another notable property of the Hull–White model is that the interest rates can go negative, which nowadays happens in some economies, e.g. in 2015 the central bank of Sweden (Sveriges Riksbank) adopted a negative interest rate strategy for interbank lending to boost the economy of the country [31].

The three stochastic model factors are defined by the following dynamics

$$\mathrm{d}S_t = R_t S_t \mathrm{d}t + \sqrt{V_t} S_t \mathrm{d}W_t^s, \quad S_0 = s, \tag{2.10}$$

$$\mathrm{d}V_t = \kappa(\eta - V_t)\mathrm{d}t + \sigma_v \sqrt{V_t} \mathrm{d}W_t^v, \quad V_0 = v, \tag{2.11}$$

$$\mathrm{d}R_t = a(b - R_t)\mathrm{d}t + \sigma_r \mathrm{d}W_t^r, \quad R_0 = r, \tag{2.12}$$

where R_t is the stochastic interest rate, a is the speed of mean reversion of the interest rate process, b is its mean reversion level, σ_r is its volatility, W_t^s, W_t^v, and W_t^r are correlated Wiener processes with constant correlation ρ_{ij}, i.e. $\langle \mathrm{d}W_t^i, \mathrm{d}W_t^j \rangle = \rho_{ij}\mathrm{d}t$, $i, j \in \{s, v, r\}$, and the other parameters are defined as in (2.1)–(2.2).

We can apply the Itô lemma and the Feynman–Kac theorem to derive a pricing PDE for the HHW model that takes the following form

$$\begin{aligned} -\frac{\partial u}{\partial t} =& \frac{1}{2} v s^2 \frac{\partial^2 u}{\partial s^2} + \frac{1}{2}\sigma_v^2 v \frac{\partial^2 u}{\partial v^2} + \frac{1}{2}\sigma_r^2 \frac{\partial^2 u}{\partial r^2} + \\ & \rho_{sv}\sigma_v v s \frac{\partial^2 u}{\partial s \partial v} + \rho_{sr}\sigma_r \sqrt{v} s \frac{\partial^2 u}{\partial s \partial r} + \rho_{vr}\sigma_v \sigma_r \sqrt{v} \frac{\partial^2 u}{\partial v \partial r} + \\ & r s \frac{\partial u}{\partial s} + \kappa(\eta - v)\frac{\partial u}{\partial v} + a(b - r)\frac{\partial u}{\partial r} - ru, \end{aligned} \tag{2.13}$$

subject to the terminal condition

$$u(T, s, v, r) = \max(s - K, 0). \tag{2.14}$$

The property of the interest rate to take negative values introduces a numerical issue when solving the pricing PDE. Some methods such as FD methods encountered and reported spurious oscillations in the numerical solution [32]. To alleviate the spurious oscillations an upwind FD scheme can be constructed and used when $r < 0$.

The model parameter values that we use for the numerical experiments are

$$K = 1, T = 1, \kappa = 0.50, \eta = 0.04, \sigma_v = 0.25, \sigma_r = 0.09, \rho_{sv} = -0.9,$$
$$\rho_{sr} = 0.6, \rho_{vr} = -0.7, a = 0.08, b = 0.1,$$

and the three evaluation points are

$$S = 0.75, 1.00, 1.25; \qquad V_0 = 0.040; \qquad R_0 = 0.100.$$

2.4 The Heston–Cox–Ingersoll–Ross Model

The Cox–Ingersoll–Ross (CIR) interest rate model [33] is another model for describing the stochastic nature of interest rates. We combine the CIR stochastic interest rate model with the Heston stochastic volatility model to obtain a three-factor structure that we call the Heston–Cox–Ingersoll–Ross (HCIR) model. A notable property and the difference of the CIR model compared to the Hull–White model is that the interest rates cannot be negative.

The dynamics of the stochastic processes read as follows

$$\mathrm{d}S_t = R_t S_t \mathrm{d}t + \sqrt{V_t} S_t \mathrm{d}W_t^s, \quad S_0 = s, \tag{2.15}$$
$$\mathrm{d}V_t = \kappa(\eta - V_t)\mathrm{d}t + \sigma_v \sqrt{V_t}\mathrm{d}W_t^v, \quad V_0 = v, \tag{2.16}$$
$$\mathrm{d}R_t = a(b - R_t)\mathrm{d}t + \sigma_r \sqrt{R_t}\mathrm{d}W_t^r, \quad R_0 = r. \tag{2.17}$$

In order to derive a pricing PDE we commit to the standard procedure of applying the Itô lemma and the Feynman–Kac theorem. The obtained PDE takes the form

$$\begin{aligned} -\frac{\partial u}{\partial t} =& \frac{1}{2} v s^2 \frac{\partial^2 u}{\partial s^2} + \frac{1}{2}\sigma_v^2 v \frac{\partial^2 u}{\partial v^2} + \frac{1}{2}\sigma_r^2 r \frac{\partial^2 u}{\partial r^2} + \\ & \rho_{sv}\sigma_v v s \frac{\partial^2 u}{\partial s \partial v} + \rho_{sr}\sigma_r \sqrt{v}\sqrt{r} s \frac{\partial^2 u}{\partial s \partial r} + \rho_{vr}\sigma_v \sigma_r \sqrt{v}\sqrt{r} \frac{\partial^2 u}{\partial v \partial r} + \\ & r s \frac{\partial u}{\partial s} + \kappa(\eta - v)\frac{\partial u}{\partial v} + a(b - r)\frac{\partial u}{\partial r} - ru, \end{aligned} \tag{2.18}$$

subject to the terminal condition

$$u(T, s, v, r) = \max(s - K, 0). \tag{2.19}$$

Here we use the same parameter values as for the HHW model, that is,

$$K = 1, T = 1, \kappa = 0.50, \eta = 0.04, \sigma_v = 0.25, \sigma_r = 0.09, \rho_{sv} = -0.9,$$
$$\rho_{sr} = 0.6, \rho_{vr} = -0.7, a = 0.08, b = 0.1,$$

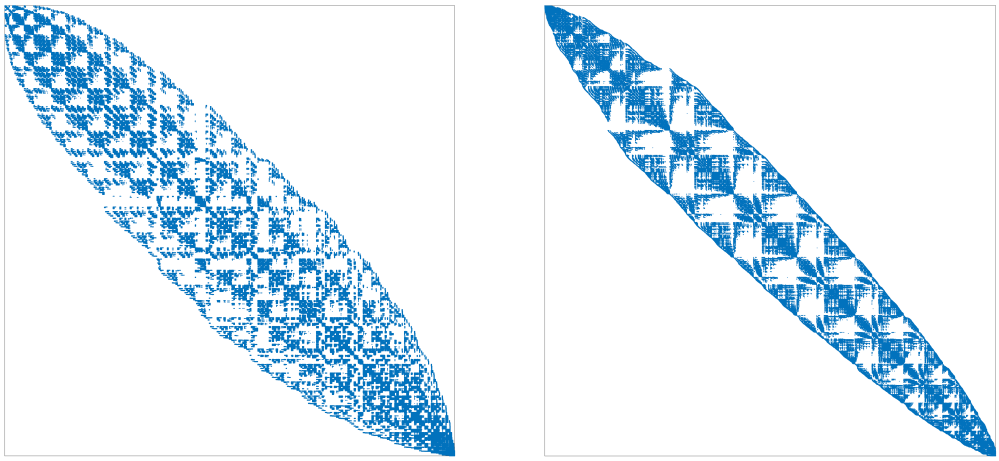
and the three evaluation points are

$$S = 0.75, 1.00, 1.25; \qquad V_0 = 0.040; \qquad R_0 = 0.100.$$

3 Localized Radial Basis Function Methods

RBF methods were first used to approximate solutions of partial differential equations by Kansa [2, 34] in the early 1990s. Since then, the methods have gained in popularity and have been applied to various types of problems of mathematical physics [35, 36], glaciology [37, 38], quantum physics [39], and computational finance [6, 8, 9]. RBF methods exhibit several very attractive properties, such as a high order convergence of the approximated solution [4, 40, 41] and flexibility with respect to the computational domain geometry [37, 41]. However, the main issue with the standard global formulation [42] is a dense coefficient matrix. In this paper, we address this issue by utilizing the mentioned localized approaches. It has been shown in the BENCHOP project [18], that localized RBF approximations make efficient numerical methods for partial differential equations and that they have a great potential for solving multi-dimensional problems. Here, we develop two localized RBF techniques, namely RBF–PUM and RBF–FD.

Since the problems (2.4), (2.8), (2.13), and (2.18) are stated in infinite domains, in order to perform numerical simulations, we need to truncate the domains and select appropriate boundary conditions at the boundaries imposed by the truncations. We denote the truncated domain as $\hat{\Omega}$. For the introduced two-factor models we define $\hat{\Omega} := [s_{\min}, s_{\max}] \times [v_{\min}, v_{\max}]$, while for the three-factor models $\hat{\Omega} := [s_{\min}, s_{\max}] \times [v_{\min}, v_{\max}] \times [r_{\min}, r_{\max}]$.



(a) RBF–PUM (b) RBF–FD

Figure 1: The sparsity structures of the differentiation matrices obtained from the discretization of the HHW problem.

For convenience, we rewrite equations (2.4), (2.8), (2.13), and (2.18) in a shorthand form and augment them with the boundary conditions

$$\frac{\partial u}{\partial t} + \mathcal{L}u(t, \underline{x}) = 0, \quad \underline{x} \in \hat{\Omega}, \tag{3.1}$$

$$\mathcal{B}u(t, \underline{x}) = f(t, \underline{x}), \quad \underline{x} \in \partial\hat{\Omega}, \tag{3.2}$$

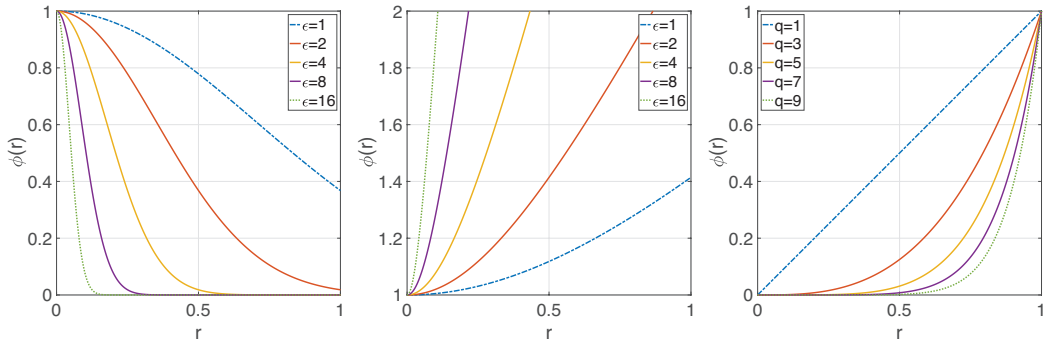
where $\mathcal{L}$ is the corresponding differential operator and $\mathcal{B}$ is the boundary differential operator, while $f(t, \underline{x})$ is the forcing function and $\underline{x} = [s, v]$ or $\underline{x} = [s, v, r]$ depending on the dimensionality of the model.

In Figure 1, we demonstrate the sparse structure of the discretized differential operator for the HHW model, that follows from the approximations by our methods after the reverse Cuthill–McKee reordering [43, 44] to reduce the bandwidth.

In Table 1 and Figure 2 we present a set of typical choices of RBFs used to develop numerical methods and illustrate how the shape parameter affects the scaling of the functions.

Table 1: Commonly used radial basis functions, where $\varepsilon \in \mathbb{R}^+$ is the shape parameter and $q \in \{2m-1, m \in \mathbb{N}\}$ is the polyharmonic spline degree.

RBF	$\phi(r)$
Gaussian (GA)	$\exp\left(-\varepsilon^2 r^2\right)$
Multiquadric (MQ)	$\sqrt{1+\varepsilon^2 r^2}$
Inverse Multiquadric (IMQ)	$1/\sqrt{1+\varepsilon^2 r^2}$
Inverse Quadratic (IQ)	$1/(1+\varepsilon^2 r^2)$
Polyharmonic Spline (PHS)	r^q



(a) Gaussian (b) Multiquadric (c) Polyharmonic spline

Figure 2: Commonly used basis functions with respect to the value of the shape parameter ε or the polyharmonic spline degree q.

In the following parts of this section, we specify the details of the RBF methods and discuss their properties. Additionally, we present the main implementation steps in the form of pseudocode algorithms in the appendices of this paper.

3.1 RBF–PUM

In order to construct an RBF–PUM approximation we start by defining an open cover $\{\Omega_j\}_{j=1}^{P}$ of the computational domain $\hat{\Omega} \subset \mathbb{R}^d$ such that

$$\hat{\Omega} \subseteq \bigcup_{j=1}^{P} \Omega_j. \tag{3.3}$$

We select the patches Ω_j to be of a spherical form. Inside each patch a local RBF approximation of the solution u is defined as

$$\tilde{u}_j(t,\underline{x}) = \sum_{i=1}^{n_j} \lambda_i^j(t)\phi(\|\underline{x}-\underline{x}_i^j\|), \tag{3.4}$$

where n_j is the number of computational nodes belonging to the patch Ω_j, $\phi(\|\underline{x}-\underline{x}_i^j\|)$ is the i-th basis function centred at $\underline{x}_i^j$, which is the i-th local node in the j-th patch Ω_j, and $\lambda_i^j(t)$ are the unknown coefficients.

In addition to the patches, we construct partition of unity weight functions $w_j(\underline{x})$, $j=1,\ldots,P$, subordinated to the open cover, such that

$$\sum_{j=1}^{P} w_j(\underline{x}) = 1, \quad \forall \underline{x} \in \hat{\Omega}. \tag{3.5}$$

Functions $w_j(\underline{x})$ can be obtained, e.g., by Shepard's method, [45], from compactly supported generating functions $\varphi_j(\underline{x})$

$$w_j(\underline{x}) = \frac{\varphi_j(\underline{x})}{\sum_{i=1}^{P}\varphi_i(\underline{x})}, \quad j=1,\ldots,P, \quad \forall \underline{x} \in \hat{\Omega}. \tag{3.6}$$

The generating functions $\varphi_j(\underline{x})$ must fulfil some smoothness requirements. For instance, for the problems considered in this paper they should be at least $C^2(\mathbb{R}^d)$. To proceed, as a suitable candidate for $\varphi_j(\underline{x})$ we choose compactly supported Wendland functions, [46]

$$\varphi(r) = (4r+1)(1-r)_+^4, \quad r \in \mathbb{R}, \tag{3.7}$$

with $\mathrm{supp}(\varphi(r)) = \mathbb{B}^d(0,1)$, where $\mathbb{B}^d(0,1)$ is a unit d-dimensional ball centred at the origin. In order to map the generating function to the patch Ω_j with centre $\underline{c}_j$ and radius ρ_j, we shift and scale it as

$$\varphi_j(\underline{x}) = \varphi_j\left(\frac{\|\underline{x}-\underline{c}_j\|}{\rho_j}\right), \quad \forall \underline{x} \in \hat{\Omega}. \tag{3.8}$$

Further we combine the local RBF approximations with the partitions of unity weight functions and obtain a global RBF–PUM solution $\tilde{u}(\underline{x})$ as

$$\tilde{u}(t,\underline{x}) = \sum_{j=1}^{P} w_j(\underline{x})\tilde{u}_j(t,\underline{x}). \tag{3.9}$$

In order to attack the problem (3.1)–(3.2) numerically by RBF–PUM we need to scatter N nodes $\mathbf{x} = \{\underline{x}_1, \ldots, \underline{x}_N\}$ in $\hat{\Omega}$. Without loss of generality, we assume that the first N_I nodes belong to the interior of $\hat{\Omega}$ and the remaining $N_B = N - N_I$ nodes belong to its boundary. We seek the solution to (3.1)–(3.2) in the form of (3.9). Collocating (3.9) at the nodes we obtain the following linear system of equations

$$\begin{bmatrix} A \\ 0 \end{bmatrix} \frac{\mathrm{d}\vec{\lambda}}{\mathrm{d}t} + \begin{bmatrix} L \\ B \end{bmatrix} \vec{\lambda} := \begin{bmatrix} A_{II} & A_{BI} \\ 0 & 0 \end{bmatrix} \frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} \vec{\lambda}_I \\ \vec{\lambda}_B \end{bmatrix} + \begin{bmatrix} L_{II} & L_{BI} \\ B_{IB} & B_{BB} \end{bmatrix} \begin{bmatrix} \vec{\lambda}_I \\ \vec{\lambda}_B \end{bmatrix} = \begin{bmatrix} 0 \\ \vec{f}_B \end{bmatrix}, \quad (3.10)$$

where L and B are the discrete representation of the differential operator and boundary differential operator, respectively, A is the interpolation matrix, $\vec{\lambda}_I = [\lambda_1, \ldots, \lambda_{N_I}]^T$, $\vec{\lambda}_B = [\lambda_{N_I+1}, \ldots, \lambda_N]^T$, and $\vec{f}_B(t) = [f(t, \underline{x}_{N_I+1}), \ldots, f(t, \underline{x}_N)]^T$.

However, using the interpolation relation between $\vec{u}$ and $\vec{\lambda}$ on (3.9) at the nodes we get the following system of equations

$$\vec{u} = \mathcal{A}\vec{\lambda}, \qquad \mathcal{A} := \begin{bmatrix} A_{II} & A_{BI} \\ A_{IB} & A_{BB}, \end{bmatrix} \quad (3.11)$$

which can be used to express the coefficients $\vec{\lambda}$ in terms of the function values $\vec{u}$ as

$$\vec{\lambda} = \mathcal{A}^{-1}\vec{u}. \quad (3.12)$$

For the smooth basis functions from Table 1 the matrix $\mathcal{A}$ is non-singular, i.e., $\mathcal{A}^{-1}$ exists [47]. Then using the relationship (3.12) we can transform the problem (3.10) of finding the coefficients $\vec{\lambda}$ to a problem of finding the function values directly. It has been shown [48, 49, 50] that for smooth RBFs the magnitude of the coefficients $\vec{\lambda}$ becomes unbounded as $\varepsilon \to 0$, while the values $\vec{u}$ remain well-behaved. Therefore, we prefer to express the problem in terms of the nodal function values $\tilde{u}$.

$$\begin{bmatrix} A_{II} & A_{BI} \\ 0 & 0 \end{bmatrix} \mathcal{A}^{-1} \frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} \vec{u}_I \\ \vec{u}_B \end{bmatrix} + \begin{bmatrix} L_{II} & L_{BI} \\ B_{IB} & B_{BB} \end{bmatrix} \mathcal{A}^{-1} \begin{bmatrix} \vec{u}_I \\ \vec{u}_B \end{bmatrix} = \begin{bmatrix} 0 \\ \vec{f}_B \end{bmatrix}, \quad (3.13)$$

where $\vec{u}_I(t) = [\tilde{u}(t, \underline{x}_1), \ldots, \tilde{u}(t, \underline{x}_{N_I})]^T$ and $\vec{u}_B(t) = [\tilde{u}(t, \underline{x}_{N_I+1}), \ldots, \tilde{u}(t, \underline{x}_N)]^T$.

The formulation (3.13) is a system of ordinary differential equations, which can, in principle, be numerically solved by any suitable method. We refer to Section 3.3 for the time integration details. Moreover, the algorithm for the RBF–PUM solver is presented in Appendix B.

The RBF–PUM approximation in the given form allows to maintain accuracy similar to that of the global method while significantly reducing the computational

effort [5, 37] thanks to the sparse structure of the coefficient matrix that, additionally, enables the use of sparse operations in environments that support such functionality, e.g., in MATLAB. Moreover, RBF–PUM is well suited for parallel implementation since the computation of the local differentiation matrices is parallelizable.

3.2 RBF–FD

In order to construct an RBF–FD approximation we start by scattering N nodes accross the computational domain $\hat{\Omega} \subset \mathbb{R}^d$. For each node $\underline{x}_j$, we define subsets $\{\Omega_j\}_{j=1}^N$ consisting of $n_j - 1$ neighboring nodes and $\underline{x}_j$ itself, and consider them as stencils of size n_j.

The differential operator $\mathcal{L}$ defined in (3.1) is approximated in every node $\underline{x}_j$ as

$$\mathcal{L}u(\underline{x}_j) \approx \sum_{i=1}^{n_j} w_i^j u_i^j \equiv Wu(\underline{x}_j), \quad j = 1, \ldots, N, \tag{3.14}$$

where $u_i^j = u(\underline{x}_i^j)$, and $\underline{x}_i^j$ is a locally indexed node as introduced before.

When it comes to the standard RBF–FD method introduced in [51, 52], the weights w_i^j are calculated by enforcing (3.14) to be exact for RBFs centered at each of the nodes in Ω_j yielding

$$\begin{bmatrix} \phi(\|\underline{x}_1^j - \underline{x}_1^j\|) & \ldots & \phi(\|\underline{x}_1^j - \underline{x}_{n_j}^j\|) \\ \vdots & \ddots & \vdots \\ \phi(\|\underline{x}_{n_j}^j - \underline{x}_1^j\|) & \ldots & \phi(\|\underline{x}_{n_j}^j - \underline{x}_{n_j}^j\|) \end{bmatrix} \begin{bmatrix} w_1^j \\ \vdots \\ w_{n_j}^j \end{bmatrix} = \begin{bmatrix} \mathcal{L}\phi(\|\underline{x}^j - \underline{x}_1^j\|) \\ \vdots \\ \mathcal{L}\phi(\|\underline{x}^j - \underline{x}_{n_j}^j\|) \end{bmatrix}. \tag{3.15}$$

From the theory on RBF interpolation, (3.15) forms a nonsingular system which means that a unique set of weights can be computed and assembled into a differentiation matrix L. Since $n_j \ll N$ the resulting differentiation matrix is sparse, making the memory cost $\mathcal{O}(N)$ compared to $\mathcal{O}(N^2)$ for global approximations with RBFs. Moreover, besides the freedom of node placement, the method is also featured with freedom of choice of the stencil size n_j for each node $\underline{x}_j$ in the computational domain $\hat{\Omega}$, which can be used to control approximation accuracy in different parts of the domain.

Many different types of RBFs have been considered for approximating similar differential operators in recent history and some of them are listed in Table 1 and shown in Figure 2. The main issue following this kind of approximations is that the linear systems of equations that need to be solved in order to obtain the weights w_i^j

are most often ill-conditioned. The shape parameter ε, featured in most of the RBFs needs to be picked very carefully and sometimes even that may not be enough to ensure a stable approximation.

In several works in the past decade [12, 36, 53, 54, 55, 56, 57], it was demonstrated that the stencil approximation can be stabilized by adding low-order polynomials together with RBFs into the presented interpolation. Still, the problem of choosing the optimal shape parameter, which is thoroughly examined for option pricing problems in [12], remains unsolved for the general case. Nevertheless, recent developments [58, 59], showed that the RBF–FD approximation can be greatly improved by using high-order polynomials together with RBFs. It appears that, in this setting, the polynomial degree, instead of the RBF, dictates the rate of convergence. This finding suggests choosing piecewise smooth polyharmonic splines (PHS) as RBFs since they are not featured with a shape parameter. Still, it seems that RBFs do contribute to reduction of approximation errors and, therefore, are necessary for having both stable and accurate approximation.

Therefore, the linear system that we solve to get the differentiation weights for each node in our problems is of the following form

$$\begin{bmatrix} A & P^T \\ P & 0 \end{bmatrix} \begin{bmatrix} \underline{w}_j \\ \underline{\gamma}_j \end{bmatrix} = \begin{bmatrix} \mathcal{L}\phi(\|\underline{x}^j - \underline{x}_1^j\|) \\ \vdots \\ \mathcal{L}\phi(\|\underline{x}^j - \underline{x}_{n_j}^j\|) \\ \mathcal{L}p_1(\underline{x}_j) \\ \vdots \\ \mathcal{L}p_{m_j}(\underline{x}_j) \end{bmatrix}, \tag{3.16}$$

where A is the RBF matrix and $\underline{w}_j$ is the vector of differentiation weights, both shown on the left-hand side of equation (3.15). P is the matrix of size $m_j \times n_j$ that contains all monomials up to order q_p (corresponding to m_j monomial terms) that are evaluated in each node $\underline{x}_i^j$ of the stencil Ω_j and 0 is a zero square matrix of size $m_j \times m_j$. Furthermore, $\underline{\gamma}_j$ is a vector of dummy weights that should be discarded and $\{p_1, p_2, \ldots, p_{m_j}\}$ is an 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 q_p.

It is worth noting that compared to standard FD discretizations, where differential operators are approximated only on one-dimensional Cartesian grids, which means that high-dimensional operators need to be discretized separately in each direction, in the RBF–FD approximations, dimensionality does not play a role in the difficulty of this problem. Moreover, when it comes to the boundary nodes and the

ones that are close to it, the nearest neighbor based stencils automatically deform, therefore requiring no special treatment for computing the differentiation weights in those areas. The only information that is required for the approximation of differential operators are the Euclidian distances between the nodes. This means that (3.15) represents a way to approximate a differential operator in any number of dimensions. Furthermore, it is true that the FD weights can be directly derived and that for the RBF–FD case one has to solve a small linear system for each node to obtain them, but we need to stress that this task (corresponding to the `for` loop at lines 4–11 of Algorithm **??** in Appendix B) is perfectly parallelizable and that this extra cost can be justified by the great features of the method presented so far. Finally, if a special boundary condition needs to be imposed, that can be done by simply replacing the operator $\mathcal{L}$ with an appropriate boundary operator $\mathcal{B}$ when approximating the weights for the nodes that belong to the boundary.

After the weights are computed and stored in the differentiation matrix, approximation of (3.1) and (3.2) looks as the following semi-discrete equations

$$\begin{bmatrix} E_{II} & 0_{IB} \\ 0_{BI} & 0_{BB} \end{bmatrix} \frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} \vec{u}_I \\ \vec{u}_B \end{bmatrix} = \begin{bmatrix} L_{II} & L_{IB} \\ B_{BI} & B_{BB} \end{bmatrix} \begin{bmatrix} \vec{u}_I \\ \vec{u}_B \end{bmatrix}, \tag{3.17}$$

which can be integrated in time as described in the following subsection. Moreover, the algorithm for the RBF–FD solver is presented in Appendix B.

3.3 Integration in Time

For the time discretization we select the unconditionally stable second order backward differentiation formula (BDF-2) [60, p. 401]. The BDF-2 scheme involves three time levels. To initiate the method, often the BDF-1 scheme (Euler backward) is used for the first time step. Thus, two different matrices need to be factorized. In order to avoid this we use the BDF-2 scheme as described in [61].

We split the time interval $[0, T]$ into N_t non-uniform steps of length $k^n = t^{n+1} - t^n$, $n = 1, \ldots, N_t$ and define the BDF-2 weights as

$$\beta_0^n = k^n \frac{1 + \omega_n}{1 + 2\omega_n}, \quad \beta_1^n = \frac{(1 + \omega_n)^2}{1 + 2\omega_n}, \quad \beta_2^n = \frac{\omega_n^2}{1 + 2\omega_n}, \tag{3.18}$$

where $\omega_n = k^n / k^{n-1}$, $n = 2, \ldots, N_t$. In [61] it is shown how the time steps can be chosen in such a way that $\beta_0^n \equiv \beta_0$. Therefore, the coefficient matrix is the same in all time steps and only one matrix factorization is needed.

Applying the BDF-2 scheme to (3.13) or (3.17) we arrive at a fully discrete system of equation that reads as follows

$$\begin{bmatrix} E_{II} + \beta_0 L_{II} & \beta_0 L_{IB} \\ \beta_0 B_{IB} & E_{BB} + \beta_0 B_{BB} \end{bmatrix} \begin{bmatrix} \vec{u}_I^n \\ \vec{u}_B^n \end{bmatrix} = \begin{bmatrix} \vec{f}_I^n \\ \vec{f}_B^n \end{bmatrix}, \tag{3.19}$$

where E_{II} and E_{BB} are the identity matrices of the appropriate size, and

$$\begin{aligned} \vec{f}_I^n &= \beta_1^n \vec{u}_I^{n-1} - \beta_2^n \vec{u}_I^{n-2}, \\ \vec{f}_B^n &= [f(t^n, \underline{x}_{N_I+1}), \ldots, f(t^n, \underline{x}_N)]^T. \end{aligned} \tag{3.20}$$

To solve this system we exploit the iterative GMRES method with incomplete LU factorization as preconditioner. The matrix in (3.19) is sparse and strongly diagonally dominant. Therefore, GMRES converges in a few iterations.

4 Numerical Experiments

In this section we perform numerical experiments with the presented methods in order to determine the price of a European call option under the assumption of multiple stochastic factors. The design of the experiments is based on [19]. We evaluate the option price for three predetermined asset spot prices that correspond to out-of-the-money, at-the-money, and in-the-money positions. In some cases for some of the selected models there exist semi-analytical solutions. For example, for the Heston model which corresponds to Set 1 of the parameters under the QLSV model, and for the SABR model with zero-correlation between the asset and volatility, i.e., parameter Set 1. Thus, we obtain these values and use them as our reference solutions to study the convergence properties as well as the computational performance of the two localized RBF methods. We test the convergence on node sets with a slightly different number of nodes between the methods, because Cartesian node layouts are not optimal for RBF–PUM [62], and for some node densities we may encounter an inconsistency in the results as we refine. Typically, the difference does not exceed one order of magnitude and may with the same likelihood give better and worse approximations. Thus, the errors that we obtain on some repeatedly refined node sets may not align on a straight line when presented using log-log plots. Therefore, we pick points which give a representative picture of the overall convergence rate of RBF–PUM. The values in the Tables are obtained on identical node sets for both RBF–FD and RBF–PUM, since there we do not know the exact reference solutions.

We measure the error in the pointwise l_∞-norm

$$\|\vec{u} - \vec{u}_{\text{ref}}\|_\infty,$$

where $\vec{u}_{\text{ref}}$ is the reference solution and is equal to

- QLSV, Set 1: $\vec{u}_{\text{ref}} = [0.009085, 0.090467, 0.285148]$,
- SABR, Set 1: $\vec{u}_{\text{ref}} = [0.009545, 0.080717, 0.264368]$.

For the plots demonstrating the computational performance, we executed MATLAB implementations of the discussed methods on a laptop equipped with a 2.3 GHz Intel Core i7 CPU and 16 GB of RAM. Furthermore, for all the comparative figures we use the serial versions of our codes, while the parallelization potentials are presented in the last part of this section.

Additionally, we report tables with option values for the problems where the reference solutions are not readily available.

As we mention in Section 3, the infinite domain Ω has to be truncated in order to perform the numerical simulations. We use the following truncated domains

$$\hat{\Omega} := [s_{\min}, s_{\max}] \times [v_{\min}, v_{\max}] = [0, 2] \times [0.001, 1]$$

for the models with two stochastic factors,

$$\hat{\Omega} := [s_{\min}, s_{\max}] \times [v_{\min}, v_{\max}] \times [r_{\min}, r_{\max}] = [0, 4] \times [0.005, 2] \times [-1, 1]$$

for the Heston–Hull–White model, and

$$\hat{\Omega} := [s_{\min}, s_{\max}] \times [v_{\min}, v_{\max}] \times [r_{\min}, r_{\max}] = [0, 4] \times [0.005, 2] \times [0, 2]$$

for the Heston–Cox–Ingersoll–Ross model. We avoid using zero values for $v_{\min}$ because this leads to a singularity in the PDE formulation.

We select the following boundary conditions at the truncated boundaries:

$$u|_{s=s_{\min}} = 0, \qquad \left.\frac{\partial^2 u}{\partial s^2}\right|_{s=s_{\max}} = 0 \ \text{ or } \ u|_{s=s_{\max}} = s - Ke^{-rt}, \tag{4.1}$$

$$\left.\frac{\partial u}{\partial v}\right|_{v=v_{\min}} = \left.\frac{\partial u}{\partial v}\right|_{v=v_{\max}} = 0, \tag{4.2}$$

$$\left.\frac{\partial u}{\partial r}\right|_{r=r_{\min}} = \left.\frac{\partial u}{\partial r}\right|_{r=r_{\max}} = 0. \tag{4.3}$$

However, we do not necessarily assign these conditions directly on the solution at the boundary. In case of RBF–PUM, we approximate the boundary conditions by

enforcing them inside the differential operator and deriving a reduced form boundary operator $\mathcal{B}$, which is applied at the boundary nodes, while for RBF–FD we assign only the Dirichlet boundary conditions directly at the nodes and for the other boundary nodes we construct the weights in the same way as for the nodes in the inner parts of the computational domain. Our experience shows that this approach works better and provides a more stable localized RBF approximation. We indicate two types of boundary conditions for the far-field boundary in the asset direction. The vanishing second derivative is a correct choice, but for some sets of parameter values we might encounter numerical issues with this approach for the Heston–Hull–White model due to the possibility of the interest rate to be negative. Such a situation is challenging for RBF methods, since, e.g., for RBF–PUM no upwind scheme can be constructed. Therefore, as an alternative we suggest to use a Dirichlet boundary condition, which is asymptotically correct for the Black–Scholes model but biased for the considered models. Although, at infinity this bias vanishes. Thus, to diminish the impact of the biased Dirichlet boundary condition the point of truncation $s_{\max}$ must be moved farther away from the origin.

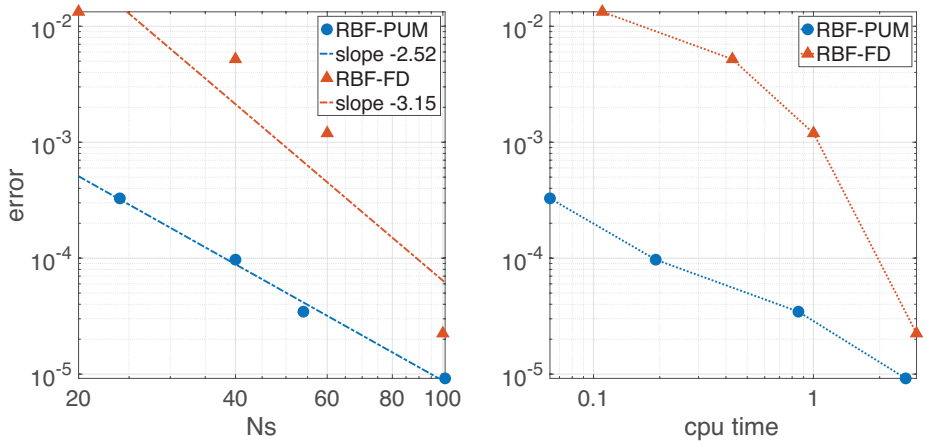
Although, our methods are featured with freedom of node placement, to discretize the computational domain, we use Cartesian nodes that are uniformly distributed in all directions. The number of nodes is defined as $N_s \times N_v$ for the two-factor problems and $N_s \times N_v \times N_r$ for the three factors models, where N_s, N_v, and N_r denote the number of nodes in the asset, volatility, and interest rate directions, respectively. We assign the following values to N_s, N_v, and N_r

$$N_s = 2N_v = 2N_r.$$

In the following parts of this section we present and discuss the results obtained by the localized RBF methods for each of the problems defined in Section 2. The parameters of our methods used to solve the problems can be found in Appendix A.

4.1 The Quadratic Local Stochastic Volatility Model

In Figure 3 we present the convergence and the computational performance of the two methods for the Heston problem (i.e., QLSV with $\alpha = 0, \beta = 1, \gamma = 0$).



(a) Convergence (b) Computational performance

Figure 3: Numerical results of the localized RBF methods in terms of convergence and computational performance measured in seconds of the execution time for the Heston problem.

It can be seen that both of the methods converge with similar orders, although RBF–PUM performs more efficiently than RBF–FD on the chosen interval. Moreover, both of the methods consume similar computational time for the same number of the discretization nodes, but the accuracy of RBF–PUM dominates. Nevertheless, RBF–FD has a potential to reach the accuracy of RBF–PUM at finer node sets due to a higher order of convergence.

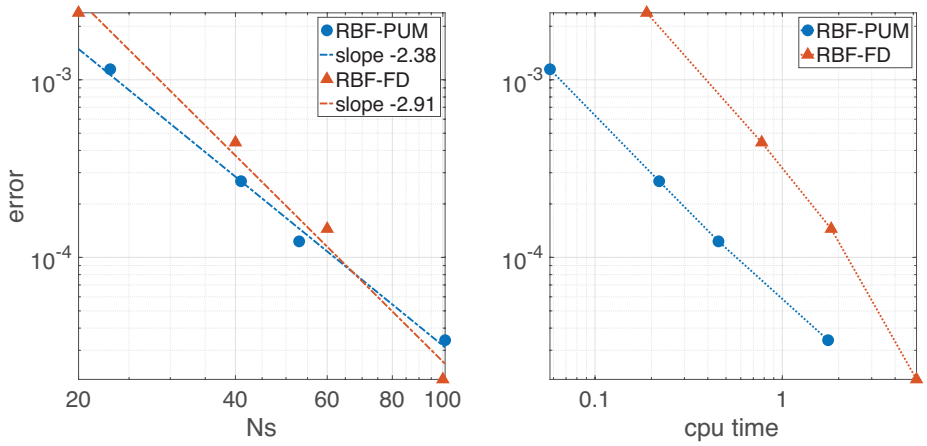
Since we do not have analytical solutions of the QLSV problem with $\alpha = 2, \beta = 0$, $\gamma = 0$, we present the option values for different node set resolutions in Table 2. Based on these results we see that both of the methods manage to lock in three digits after the decimal point, although there is a slight difference in the obtained values for at-the-money and in-the-money prices between the methods. The maximum absolute difference between the methods on the finest node set is below $6 \cdot 10^{-4}$.

Table 2: The values of the European call option under the QLSV model with $\alpha = 2, \beta = 0, \gamma = 0$.

RBF–PUM				RBF–FD			
N_s	$S_0 = 0.75$	$S_0 = 1.00$	$S_0 = 1.25$	N_s	$S_0 = 0.75$	$S_0 = 1.00$	$S_0 = 1.25$
20	0.005851	0.089819	0.291363	20	0.007359	0.087353	0.291477
40	0.005434	0.089478	0.291170	40	0.005493	0.088851	0.290970
60	0.005407	0.089465	0.291152	60	0.005324	0.088929	0.290882
100	0.005258	0.089453	0.291114	100	0.005282	0.088922	0.290836

4.2 The SABR Model

In Figure 4 we show the convergence and the computational performance of our methods for the SABR problem with $\rho = 0$. Again, we observe that both methods converge with a similar order and demonstrate a more similar accuracy. Nevertheless, RBF–PUM is significantly faster than RBF–FD when it comes to reaching a particular error level. Although, this gap in efficiency may vanish after the algorithm parallelization, because the main overhead comes from the matrix assembly, which is more time-consuming for RBF–FD.



(a) Convergence (b) Computational performance

Figure 4: Numerical results of the localized RBF methods in terms of convergence and computational performance measured in seconds of the execution time for the SABR problem with $\rho = 0$.

Since we do not have available reference values for the SABR problem with $\rho = -0.5$, we present the option values for different node set resolutions in Table 3. The results shows that both methods manage to converge to the same solutions within a $2 \cdot 10^{-4}$ maximum absolute difference for the finest node set.

Table 3: The values of the European call option under the SABR model with the correlation $\rho = -0.5$.

RBF–PUM				RBF–FD			
N_s	$S_0 = 0.75$	$S_0 = 1.00$	$S_0 = 1.25$	N_s	$S_0 = 0.75$	$S_0 = 1.00$	$S_0 = 1.25$
20	0.007379	0.083762	0.269809	20	0.007028	0.082064	0.269984
40	0.006017	0.080824	0.268928	40	0.005641	0.080085	0.268992
60	0.005573	0.080305	0.268649	60	0.005318	0.080049	0.268442
100	0.005412	0.080054	0.268513	100	0.005319	0.079928	0.268478

4.3 The Heston–Hull–White Model

The results for this most numerically challenging problem from the list are shown in Table 4. The table shows that both methods manage to converge to the same solutions within a $8 \cdot 10^{-3}$ maximum absolute difference at the highest node densities. It can be observed that the methods tend to converge to slightly different values, which is potentially caused by the numerical issues associated with the negative interest rate values and boundary conditions (4.1). The computed option values for the out-of-the-money spot price are significantly different, since that value is much smaller compared to the others.

Table 4: The values of the European call option under the HHW model.

RBF–PUM				RBF–FD			
N_s	$S_0 = 0.75$	$S_0 = 1.00$	$S_0 = 1.25$	N_s	$S_0 = 0.75$	$S_0 = 1.00$	$S_0 = 1.25$
20	0.005332	0.109525	0.346429	20	0.017820	0.128679	0.357461
30	0.006946	0.117208	0.350123	30	0.009746	0.123169	0.356759
40	0.008453	0.125490	0.354327	40	0.005824	0.122055	0.359996
50	0.008590	0.127896	0.354546	50	0.003351	0.120412	0.360258

4.4 The Heston–Cox–Ingersoll–Ross Model

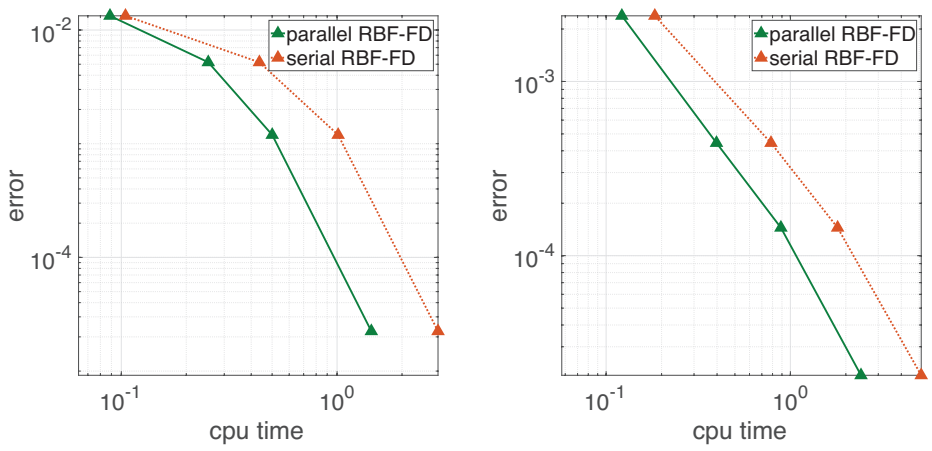
Finally, we present the results for the HCIR problem in Table 5. The option values are similar to the values obtained for the HHW problem, which verifies the trustworthiness of our solvers, since they are evaluated on the same parameter sets, and the only difference between these problems is in the chosen stochastic model for the dynamics of the interest rate R_t. The table shows that both methods manage to converge to the same solutions again within a $8 \cdot 10^{-3}$ maximum absolute difference at the highest node density. Here, again we see a slight difference in the converged values. As it can be seen in the previously presented experiment, there is a significant relative difference between the option values out-of-the-money, since that value is small.

Table 5: The values of the European call option under the HCIR model.

RBF–PUM				RBF–FD			
N_s	$S_0 = 0.75$	$S_0 = 1.00$	$S_0 = 1.25$	N_s	$S_0 = 0.75$	$S_0 = 1.00$	$S_0 = 1.25$
20	0.003478	0.108315	0.346736	20	0.019821	0.130269	0.353338
30	0.004580	0.117966	0.351828	30	0.009327	0.122469	0.357499
40	0.005405	0.119717	0.351985	40	0.004659	0.120718	0.360338
50	0.006225	0.122347	0.352823	50	0.002748	0.120701	0.360891

4.5 Parallelization

As mentioned above, both of the methods have parallelization potentials. We pick RBF–FD to demonstrate the improvements in the computational performance when the calculation of the differentiation weights is parallelized. For this purpose we use the `parfor` functionality available in the Parallel Computing Toolbox by MATLAB, which scatters the computational load across n_w available parallel workers. On our machine we had $n_w = 4$. In Figure 5 we present the computational performance of the parallel implementations compared with the computational performance of the serial codes for the Heston and SABR problems. The red dotted line is the same one seen in the right-hand side panels of Figures 3 and 4.



(a) The Heston problem (b) The SABR problem

Figure 5: Computational performance of parallel and serial RBF–FD solvers measured in seconds of the execution time.

RBF–PUM can be parallelized in a similar way as RBF–FD by running in parallel the operations inside the patches.

The results indicate that by parallelization the execution time of RBF–FD can be decreased by several times. It shows a decrease from 5.13 to 2.41 seconds of run time for the SABR problem on the finest node set. Moreover, the figures suggest that the computational savings increase as a more accurate solution is required.

Finally, these speedups may be much greater if the computational load is scattered on a high performance computing cluster that has more than 4 employable parallel workers. Nevertheless, we need to stress that the parallel RBF–FD solver still contains a non-parallelized time integration part that is based on the BDF-2 method with GMRES for the iterative solution of the linear systems. The operations of matrix-vector multiplications within GMRES can also be parallelized [63, 64, 65], but for that level of performance optimization it would be more suitable to use a compiled programming language and more advanced hardware, which is out of the scope of this study.

5 Conclusions

In this paper we study the usage of localized RBF methods, namely RBF–PUM and RBF–FD, for numerical pricing of financial derivatives under models with multiple stochastic factors.

We choose advanced pricing models that are of interest for the financial industry. In these models the volatilities, interest rates, or both, are modeled as stochastic processes, in addition to the stochastic asset dynamics, therefore spawning challenging time-dependent partial differential equations in two or three spatial dimensions as mathematical problems that need to be solved (in most cases possible only numerically), in order to obtain the option prices. As an appropriate approach to solving these problems, we demonstrate in detail the best properties of the two localized RBF methods, such as sparsity and high convergence orders, and show how numerical solvers for the posed problems can be designed and implemented. Moreover, we test and compare our solvers in a fair setting. We demonstrate the convergence properties of the numerical methods and their computational performance. Furthermore, we discuss the parallelization potentials and gains of the presented methods.

The results demonstrate the capability of both methods to solve the posed problems to an adequate accuracy in reasonable time (a few seconds). Both methods show similar orders of convergence, which can be further optimized by more elaborate choices of the parameters, such as the sizes of patch overlaps for RBF–PUM and the stencil sizes for RBF–FD. In our tests, RBF–FD shows slightly higher orders of convergence and is able to achieve the accuracy of RBF–PUM at the finer node layouts. Moreover, in the case of serial algorithms, RBF–PUM appears to be faster than RBF–FD in some cases, but after parallelization this gap vanishes due to the distribution of the task of weights computation over multiple parallel workers.

Overall, both methods possess very similar performance features and it is difficult to advice on which method should be selected for a particular application. Nevertheless, from the experiments it is clear that both methods are suitable for problems with multiple factors and have high potential to tackle high-dimensional PDEs that arise from the problems of mathematical finance.

Acknowledgement

The authors would like to thank Elisabeth Larsson and Lina von Sydow for fruitful discussions and for proofreading the manuscript.

Declaration of Interest

The authors report no conflicts of interest. The authors alone are responsible for the content and writing of the paper.

References

[1] A. H. D. Cheng, M. A. Golberg, E. J. Kansa, and G. Zammito. Exponential convergence and hc multiquadric collocation method for partial differential equations. *Numer. Meth. Part. D. E.*, 19(5):571–594, 2003.

[2] E. J. Kansa. Multiquadrics—A scattered data approximation scheme with applications to computational fluid-dynamics—I surface approximations and partial derivative estimates. *Comput. Math. Appl.*, 19(8-9):127–145, 1990.

[3] W. R. Madych and S. A. Nelson. Bounds on multivariate polynomials and exponential error estimates for multiquadric interpolation. *J. Approx. Theory*, 70(1):94–114, 1992.

[4] C. Rieger and B. Zwicknagl. Sampling inequalities for infinitely smooth functions, with applications to interpolation and machine learning. *Adv. Comput. Math.*, 32(1):103–129, 2010.

[5] V. Shcherbakov and E. Larsson. Radial basis function partition of unity methods for pricing vanilla basket options. *Comput. Math. Appl.*, 71(1):185–200, 2016.

[6] G. E. Fasshauer, A. Q. M. Khaliq, and D. A. Voss. Using meshfree approximation for multi-asset American option problems. *J. Chinese Inst. Engrs.*, 27(4):563–571, 2004.

[7] Y. C. Hon. A quasi-radial basis functions method for American options pricing. *Comput. Math. Appl.*, 43(3-5):513–524, 2002.

[8] Y. C. Hon and X. Z. Mao. A radial basis function method for solving options pricing model. *Financial Engineering*, 8(1):31–49, 1999.

[9] U. Pettersson, E. Larsson, G. Marcusson, and J. Persson. Improved radial basis function methods for multi-dimensional option pricing. *J. Comput. Appl. Math.*, 222(1):82–93, 2008.

[10] A. Safdari-Vaighani, A. Heryudono, and E. Larsson. A radial basis function partition of unity collocation method for convection-diffusion equations. *J. Sci. Comput.*, 64(2):341–367, 2015.

[11] V. Shcherbakov. Radial basis function partition of unity operator splitting method for pricing multi-asset American options. *BIT*, 56(4):1401–1423, 2016.

[12] S. Milovanović and L. von Sydow. Radial basis function generated finite differences for option pricing problems. *Comput. Math. Appl.*, 2017.

[13] A. Golbabai and E. Mohebianfar. A new stable local radial basis function approach for option pricing. *Comput. Econ.*, 49(2):271–288, 2017.

[14] M. K. Kadalbajoo, A. Kumar, and L. P. Tripathi. An efficient numerical method for pricing option under jump diffusion model. *Int. J. Adv. Eng. Sci. Appl. Math.*, 7(3):114–123, 2015.

[15] A. Kumar, L. P. Tripathi, and M. K. Kadalbajoo. A numerical study of asian option with radial basis functions based finite differences method. *Eng. Anal. Bound. Elem.*, 50:1–7, 2015.

[16] M. K. Kadalbajoo, A. Kumar, and L. P. Tripathi. Application of the local radial basis function-based finite difference method for pricing american options. *Int. J. Comput. Math.*, 92(8):1608–1624, 2015.

[17] M. K. Kadalbajoo, A. Kumar, and L. P. Tripathi. Application of radial basis function with L-stable Padé time marching scheme for pricing exotic option. *Comput. Math. Appl.*, 66(4):500–511, 2013.

[18] L. von Sydow, L. J. Höök, E. Lindström, S. Milovanović, J. Persson, V. Shcherbakov, Y. Shpolyanskiy, S. Sirén, J. Toivanen, J. Waldén, M. Wiktorsson, J. Levesley, J. Li, C. W. Oosterlee, M. J. Ruijter, A. Toropov, and Y. Zhao. BENCHOP—The BENCHmarking project in Option Pricing. *Int. J. Comput. Math.*, 92(12):2361–2379, 2015.

[19] E. Larsson, S. Milovanović, V. Shcherbakov, L. von Sydow, and et al. BENCHOP—The BENCHmarking project in Option Pricing: Stochastic and local volatility. In preparation,

2017.
[20] C. Reisinger and G. Wittum. Efficient hierarchical approximation of high-dimensional option pricing problems. *SIAM J. Sci. Comput.*, 29(1):440–458, 2007.
[21] E. Larsson, S. Milovanović, V. Shcherbakov, L. von Sydow, and et al. BENCHOP—The BENCHmarking project in Option Pricing: Basket options. In preparation, 2017.
[22] A. Itkin, V. Shcherbakov, and A. Veygman. Influence of jump-at-default in IR and FX on Quanto CDS prices. *arXiv:1711.07133*, 2017.
[23] B. Dupire. Pricing with a smile. *Risk Magazine*, 4(1):18–20, 1994.
[24] L. Andersen. Option pricing with quadratic volatility: a revisit. *Financ. Stoch.*, 15(2):191–219, 2011.
[25] S. L. Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Rev. Financ. Stud.*, 6(2):327–343, 1993.
[26] W. Feller. Two singular diffusion problems. *Ann. of Math.*, 54(1):173–182, 1951.
[27] P. S. Hagan, D. Kumar, A. S. Lesniewski, and D. E. Woodward. Managing smile risk. *Wilmott Magazine*, pages 84–108, 2002.
[28] A. Antonov, M. Konikov, and M. Spector. SABR spreads its wings. *Risk*, August:58–63, 2013.
[29] O. Islah. Solving SABR in exact form and unifying it with LIBOR market model. Available at SSRN: http://ssrn.com/abstract=1489428, 2009.
[30] J. Hull and A. White. Pricing interest-rate derivative securities. *Rev. Financial Stud.*, 3:573–592, 1990.
[31] Sweden's Riksbank. http://www.riksbank.se/en/. November 2017.
[32] K. J. in 't Hout and T. Haentjens. Alternating direction implicit finite difference schemes for the Heston–Hull–White partial differential equation. *J. Comput. Financ.*, 16:83–110, 2012.
[33] J. C. Cox, J. E. Ingersoll, and S. A. Ross. A theory of the term structure of interest rates. *Econometrica*, 53(2):385–407, 1985.
[34] E. J. Kansa. Multiquadrics—A scattered data approximation scheme with applications to computational fluid-dynamics—II solutions to parabolic, hyperbolic and elliptic partial differential equations. *Comput. Math. Appl.*, 19(8-9):147–161, 1990.
[35] A. J. M. Ferreira, G. E. Fasshauer, R. C. Batra, and J. D. Rodrigues. Static deformations and vibration analysis of composite and sandwich plates using a layerwise theory and RBF-PS discretizations with optimal shape parameter. *Compos. Struct.*, 86(4):328–343, 2008.
[36] N. Flyer, E. Lehto, S. Blaise, G. B. Wright, and A. St-Cyr. A guide to RBF-generated finite differences for nonlinear transport: shallow water simulations on a sphere. *J. Comput. Phys.*, 231(11):4078–4095, 2012.
[37] J. Ahlkrona and V. Shcherbakov. A meshfree approach to non-Newtonian free surface ice flow: Application to the Haut Glacier d'Arolla. *J. Comput. Phys.*, 330:633–649, 2017.
[38] G. Cheng and V. Shcherbakov. Anisotropic radial basis function methods for continental size ice sheet simulations. *arXiv:1711.09947*, 2017.
[39] K. Kormann and E. Larsson. A Galerkin radial basis function method for the Schrödinger equation. *SIAM J. Sci. Comput.*, 35(6):2832–2855, 2013.
[40] C. Rieger and B. Zwicknagl. Improved exponential convergence rates by oversampling near the boundary. *Constr. Approx.*, 39(2):323–341, 2014.
[41] E. Larsson, V. Shcherbakov, and A. Heryudono. A least squares radial basis function partition of unity method for solving PDEs. *SIAM J. Sci. Comput.*, 39(6):2538–2563, 2017.
[42] G. E. Fasshauer. *Meshfree approximation methods with MATLAB*, volume 6 of *Interdisciplinary Mathematical Sciences*. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2007.

[43] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *24th National Conference ACM*, pages 157–172, 1969.

[44] A. George and J. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, 1981.

[45] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference*, ACM '68, pages 517–524, New York, NY, USA, 1968. ACM.

[46] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Adv. Comput. Math.*, 4(4):389–396, 1995.

[47] C. A. Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constr. Approx.*, 2(1):11–22, 1986.

[48] T. A. Driscoll and B. Fornberg. Interpolation in the limit of increasingly flat radial basis functions. *Comput. Math. Appl.*, 43(3-5):413–422, 2002.

[49] E. Larsson and B. Fornberg. Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions. *Comput. Math. Appl.*, 49(1):103–130, 2005.

[50] R. Schaback. Multivariate interpolation by polynomials and radial basis functions. *Constr. Approx.*, 21(3):293–317, 2005.

[51] 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*, volume 228, pages 4606–4624, Lausanne, 2000.

[52] G. B. Wright and B. Fornberg. Scattered node compact finite difference-type formulas generated from radial basis functions. *J. Comput. Phys.*, 212(1):99–123, 2006.

[53] O. Davydov and D. T. Oanh. Adaptive meshless centres and rbf stencils for poisson equation. *J. Comput. Phys.*, 230(2):287–304, 2011.

[54] B. Fornberg and E. Lehto. Stabilization of rbf-generated finite difference methods for convective pdes. *J. Comput. Phys.*, 230(6):2270–2285, 2011.

[55] E. Larsson, E. Lehto, A. Heryudono, and B. Fornberg. Stable computation of differentiation matrices and scattered node stencils based on Gaussian radial basis functions. *SIAM J. Sci. Comput.*, 35(4):2096–2119, 2013.

[56] B. Fornberg, E. Lehto, and C. Powell. Stable calculation of Gaussian-based RBF-FD stencils. *Comput. Math. Appl.*, 65(4):627–637, 2013.

[57] N. Flyer, G. A. A Barnett, and L. J. Wicker. Enhancing finite differences with radial basis functions: experiments on the navier–stokes equations. *J. Comput. Phys.*, 316:39–62, 2016.

[58] N. Flyer, B. Fornberg, V. Bayona, and G. A. Barnett. On the role of polynomials in RBF-FD approximations: I. Interpolation and accuracy. *J. Comput. Phys.*, 321:21–38, 2016.

[59] V. Bayona, N. Flyer, B. Fornberg, and G. A. Barnett. On the role of polynomials in RBF-FD approximations: II. Numerical solution of elliptic PDEs. *J. Comput. Phys.*, 332:257–273, 2017.

[60] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff problems*. Springer-Verlag, Berlin, second edition, 2000.

[61] E. Larsson, K. Åhlander, and A. Hall. Multi-dimensional option pricing using radial basis functions and the generalized Fourier transform. *J. Comput. Appl. Math.*, 222(1):175–192, 2008.

[62] B. Fornberg, N. Flyer, and J. M. Russell. Comparisons between pseudospectral and radial basis function derivative approximations. *IMA J. Numer. Anal.*, 30(1):149–172, 2010.

[63] I. Tominec. Parallel localized radial basis function methods for the shallow water equations

on the sphere. Master's thesis, Technische Universität München, 2017.
[64] M. Tillenius, E. Larsson, E. Lehto, and N. Flyer. A scalable RBF–FD method for atmospheric flow. *J. Comput. Phys.*, 298:406–422, 2015.
[65] Evan F Bollig, Natasha Flyer, and Gordon Erlebacher. Solution to PDEs using radial basis function finite-differences (RBF-FD) on multiple GPUs. *Journal of Computational Physics*, 231(21):7133–7151, 2012.

A Method Parameters

Below are the values of the various method parameters that were used to obtain the presented results.

- Values for GMRES: We use the `nofill` setting for the incomplete LU factorization to produce the preconditioner for the iterative solver. We set the tolerance to 10^{-8} for all our experiments. Once the residual drops below the selected tolerance, we acknowledge the iteration as converged. To speed up the convergence we use the values from the previous time step as the initial value for the next iteration.

- Values for RBF-PUM: We use multiquadric RBFs for all the experiments. The shape parameter ε is initially chosen as $\varepsilon = 0.17/h - 0.8$ and then slightly adjusted for each model individually. Here $h = (s_{\max} - s_{\min})/(N_s - 1)$. The number of patches is chosen in such a way to have approximately 130 computational nodes in the interior patches. The radius of the patches is computed as $\sqrt{2}H(1+\delta)$, where H is the half spacing distance between the patch centers and $\delta = 0.2$. The number of partitions along the spacial dimensions is chosen according to the ratio $P_s = 2P_v = 2P_r$.

- Values for RBF-FD: For the approximations of weights in the QLSV and SABR problems, we use PHSs of degree $q = 5$ augmented with monomials of degree $q_p = 5$ which span a polynomial space of size $m_j = 21$. The stencil sizes are chosen to be $n_j := 3 \cdot m_j = 63$. As the HHW and HCIR problems require more stability, the degree of monomials and PHS are decreased to $q = 3$ and $q_p = 3$, which gives $m_j = 20$ and we chose $n_j := 5 \cdot m_j = 100$.

B Algorithm for RBF-PUM

Algorithm 1 Valuing options by RBF-PUM

Define a set of computational nodes $\mathbf{x}$.
Define a domain partitioning $\{\Omega_j\}_{j=1}^{P}$.
Construct the partition of unity weights w_j using (3.6).
Construct the matrix as in (3.19) by running the following **for** loop.
for each patch from 1 to P **do**:
    Compute distances between local nodes in a patch.
    Compute local differentiation matrices.
    Construct a local differential operator.
    Insert the local differential operator into the global differential operator in the position specified by the node set indexing.
end for
Initialise $\vec{u}^0 = \text{Payoff}$.
Compute incomplete LU factors, iL and iU, for the matrix in (3.19).
Define the tolerance tol for GMRES.
for each time step from 2 to N_t **do**:
    Define rhs according to (3.20).
    Solve for $\vec{u}^n$ using GMRES $\vec{u}^n = \text{gmres}(\text{rhs}, \text{iL}, \text{iU}, \vec{u}^{n-1}, \text{tol})$.
    Update $\vec{u}^{n-2} = \vec{u}^{n-1}$ and $\vec{u}^{n-1} = \vec{u}^n$.
end for

C Algorithm for RBF-FD

Algorithm 2 Valuing options by RBF-FD

Define a set of computational nodes $\mathbf{x}$.
Choose the polynomial degree q_p and the stencil sizes n_j.
Construct the matrix as in (3.19) by running the following **for** loop.
for each node from 1 to N **do**:
    Estimate its $n_j - 1$ nearest neighbors using k-d tree algorithm.
    Compute local distance matrix.
    Compute local RBF and monomial matrices as in (3.16).
    Compute the right-hand side vector as in (3.16).
    Solve the linear system (3.16) to obtain the weights for that node.
    Store the weights in the global differential operator at the position specified by the node set indexing.
end for
Initialise $\vec{u}^0 = \text{Payoff}$
Compute incomplete LU factors, iL and iU, for the matrix in (3.19).
Define the tolerance tol for GMRES.
for each time step from 2 to N_t **do**:
    Define rhs according to (3.20).
    Solve for $\vec{u}^n$ using GMRES $\vec{u}^n = \text{gmres}(\text{rhs}, \text{iL}, \text{iU}, \vec{u}^{n-1}, \text{tol})$.
    Update $\vec{u}^{n-2} = \vec{u}^{n-1}$ and $\vec{u}^{n-1} = \vec{u}^n$.
end for