Forward and backward option pricing

Basel Aiesh, Ludvig Backlund, Perry Hansler

Project in Computational Science: Report

January 2016

Forward and backward option pricing

Ludvig Backlund, Perry Hanser, Basel Aiesh

January 2016

Abstract

Option pricing is an important part of financial mathematics. Not only does it cover most relevant models throughout the discipline, it is still to this day heavily researched to find more efficient pricing methods for more varied and complex derivatives. In this project two approaches for option pricing are analyzed and compared. The backward approach is applied by looking at the payoff at time of expiry and then stepping back through time to find the current value. The forward approach involves evolving the probability density function according to the underlying model dynamics from the day of the pricing until time of expiry. Then the option price is found according to the Feynman-Kac theorem. Forward mode is especially promising when having to evaluate multiple payoffs for the same underlying asset, since the probability density has to be evaluated only once, followed by multiple numerical integrations. This is not the case in backward mode. In this project simulations are performed in the Black-Scholes framework, evaluating European call options for one underlying asset, comparing four different methods in both backward and forward mode. It is found that the COS method does not provide additional computational efficiency in forward mode compared to backward mode. The localized RBF methods RBF-FD and RBF-PUM do have some computational advantages when flexibility with respect to the payoff function is required, while the adaptive FD method in addition to this shows promise when pricing only one option.

Contents

1 Introduction

An option is a financial derivative that gives the holder the right, but not the obligation, to buy or sell an underlying asset. The ability to price options efficiently and accurately has wide applications in terms of hedging and managing risk in stock portfolios, which has led to extensive research into the numerical methods used to price these options. Today there are a myriad of different types of options available on the financial markets, but in this project we choose only to treat the basic European call option on one underlying asset. The European call option is a contract that allows the holder to buy the underlying asset at a fixed price K at a certain point T in the future, often referred to as the strike price and the time of maturity. A Black-Scholes market model [1] is assumed where the price-dynamics of a deterministic bond B_t and a stochastic asset S_t is as

$$\begin{aligned} dB_t &= rB_t dt, \\ dS_t &= \mu S_t dt + \sigma S_t dW_t, \end{aligned} \tag{1}$$

where r is the risk-free interest rate, μ is the drift of the asset, σ is the volatility of the asset and W_t denotes a Wiener process. Equation (1) implies that the underlying asset follows geometric Brownian motion, meaning that the stock will have a log-normal distribution. At time of maturity T the value of a European call option is described by a so-called payoff function: $\phi(s) = \max(s - K, 0)$, where s is the price of the underlying asset and K is the strike price. The value of the option today u_0 is then described by

$$u_0 = e^{-rT} E_Q(\phi(s_T, K)), \tag{2}$$

where E_Q is the expected value assuming a risk-neutral measure Q. Computing u_0 can be accomplished both by applying a Monte Carlo approach or applying a deterministic approach by solving the Black-Scholes equation

$$\frac{\partial u}{\partial t} + \frac{1}{2}\sigma^2 s^2 \frac{\partial^2 u}{\partial s^2} + rs\frac{\partial u}{\partial s} - ru = 0, \quad s \in R^+, t > 0. \tag{3}$$

The deterministic approach has proven to be more efficient when solving for only one or a few underlying assets [8], and is therefore more suitable for the vanilla European call options we are considering in this project. In the backward approach we are using $\phi(s_T, K)$ as final condition and solving Equation (3) backwards in time. This results in a solution for all initial values s_0 of s, but only for one specific value of K. If the payoff function is changed the entire computation would have to be redone. However, in the forward approach, we are using the same approach as in [2] and solving the Fokker-Planck equation instead,

$$\begin{gathered} \frac{\partial}{\partial t}p(s,t) + \frac{\partial}{\partial s}\left[\mu(s,t)p(s,t)\right] - \frac{1}{2}\frac{\partial^2}{\partial s^2}\left[\sigma^2(s,t)p(s,t)\right] = 0, \quad s \in R^+, t > 0, \\ p(s,0) = \delta(s_0 - s). \end{gathered} \tag{4}$$

This allows for a greater flexibility with regards to the payoff function. Solving Equation (4) and proceeding forward in time returns the probability density function of the underlying asset at time of maturity. This, according to the Feynman-Kac theorem [9], allows us to multiply by a payoff function of our choice and integrate over the domain to obtain the option price.

In the project four different numerical methods are adapted to solve in forward mode rather than backward mode. These methods are then optimized to meet a set of error tolerances and are compared both against each other and with their backward counterparts, ditto with respect to computational efficiency.

2 The Fokker-Planck equation

Given a stochastic process S_t that is the solution to a stochastic differential equation $dS_t = \mu(S_t, t)dt + \sigma(S_t, t)dW$, where μ is the expectation and σ is the variance, the probability density function of the stochastic process is found as the solution to the Fokker-Planck equation, which is written as

$$\frac{\partial}{\partial t} p(s,t) = -\frac{\partial}{\partial s}\left[\mu(s,t)p(s,t)\right] + \frac{1}{2}\frac{\partial^2}{\partial s^2}\left[\sigma^2(s,t)p(s,t)\right], \tag{5}$$

where $p(s,t)$ is the probability density function of the stochastic process S_t, $\mu(s,t)$ is the drift and $\sigma(s,t)$ is the diffusion. Performing the component-wise differentiation on the right hand side of Equation (5) results in

$$\begin{aligned} -\frac{\partial}{\partial s}\left[\mu(s,t)p(s,t)\right] &= -\frac{\partial \mu}{\partial s}p - \mu\frac{\partial p}{\partial s}, \\ \frac{1}{2}\frac{\partial^2}{\partial s^2}\left[\sigma^2(s,t)p(s,t)\right] &= \frac{1}{2}\left[\frac{\partial^2\sigma^2}{\partial s^2}p + 2\frac{\partial \sigma^2}{\partial s}\frac{\partial p}{\partial s} + \sigma^2\frac{\partial^2 p}{\partial s^2}\right]. \end{aligned} \tag{6}$$

In the framework of the Black-Scholes model the underlying asset process is log-normally distributed, therefore letting $\mu(s,t) = rs$ and $\sigma(s,t) = gs$ where r and g are constants, and substituting Equation (6) in Equation (5) gives

$$\frac{\partial}{\partial t} p(s,t) = -rp - rs\frac{\partial p}{\partial s} + \frac{1}{2}\left[2g^2p + 4g^2s\frac{\partial p}{\partial s} + g^2s^2\frac{\partial^2 p}{\partial s^2}\right], \tag{7}$$

which after simplification gives the resulting equation

$$\frac{\partial}{\partial t} p(s,t) = (g^2 - r)p + (2g^2s - rs)\frac{\partial p}{\partial s} + \frac{1}{2}g^2s^2\frac{\partial p^2}{\partial s^2}. \tag{8}$$

This is the Fokker-Planck equation that is solved in order to obtain the probability density function. From the solution of Equation (8) the option price is evaluated via a rewritten Equation (2), known as the Feynman-Kac formula:

$$u_0(K,T) = e^{-rT}\int_{s\in R^+} \phi(s,K)p(s,T)ds, \tag{9}$$

where ϕ is the payoff function and T is time of maturity.

3 Numerical methods

3.1 Radial Basis Functions

Two of the examined numerical methods are radial basis function (RBF) based methods. RBFs are mesh free and based on scattered nodes, they can therefore be advantageous to use on irregular domains [3]. Given N scattered nodes $(x_1, \dots, x_N) \in \Omega$ and the function values $u(x_1), \dots, u(x_N)$, the global RBF approximation takes the form

$$\mathcal{J}(x) = \sum_{j=1}^{N} \lambda_j \phi_j(||x - x_j||), \quad x \in \Omega \tag{10}$$

where $||\cdot||$ is the Euclidean norm, λ_j is an unknown coefficient and $\phi_j(x)$ is a real valued radial basis function with center x_j. Due to the geometrical similarities with the density function the RBF $\phi(x)$ is in our case chosen as a Gaussian function

$$\phi(r) = e^{-(\varepsilon r)^2}, \tag{11}$$

where ε is the so called shape parameter which determines the width of the RBF and therefore has a high impact on the accuracy of the approximation. Hence, it should be chosen carefully. In order to find λ we enforce the interpolation condition

$$u(x_j) = \mathcal{J}(x_j). \tag{12}$$

Thereby, we obtain a linear system

$$A\bar{\lambda} = \bar{u}, \tag{13}$$

where A is the interpolation matrix containing the elements $A_{ij} = \phi(||x_i - x_j||)$, $\bar{\lambda} = [\lambda_1, ..., \lambda_N]^T$ and $\bar{u} = [u(x_1), ..., u(x_N)]^T$. The problems considered in this project are time-dependent, meaning that the RBF approximation in our case is of the form

$$\mathcal{J}(x,t) = \sum_{j=1}^{N} \lambda_j(t) \phi_j(||x - x_j||), \quad x \in \Omega, t > 0. \tag{14}$$

Solving problems using global RBFs results in a dense interpolation matrix, meaning that solving the system of equation (13) will be computationally expensive. To bypass this problem the localized methods RBF-FD and RBF-PUM are introduced.

3.2 RBF-FD

The RBF finite difference (RBF-FD) method, as any finite difference method, uses a stencil to write finite difference approximations to derivatives at grid points. However, it employs the RBF method to estimate the weight on each member point in the stencil. The advantage of such an approach is that it is more flexible than the standard finite difference method, e.g., scattered nodes can be easily used and stencils of different sizes can be easily designed. Moreover, RBF-FD helps to overcome the high computational cost associated with the global RBF method.

The RBF-FD method is defined as follows; Given a set of scattered nodes $X = \{x_1, x_2, ...x_N\}$ in the computational domain Ω, let $X_i = \{x_1^{(i)}, x_2^{(i)}, ..., x_M^{(i)}\}, X_i \subset X$ be a stencil of nodes surrounding each center node x_i. Suppose that $u(x)$ is a smooth function with values $u(x_1), u(x_2), .., u(x_N)$ at the nodes and that we would like to recover this function in the whole computational domain. Let us introduce "internal" indexing for a subset $X_i; x_i = x_j^{(i)}$, $u(x_i) = u_i = u_j^{(i)}$ where i denotes global index in the node set X and j denotes the local index in the stencil X_i, $j \in \{1, \ldots, M\}$. For each stencil X_i we construct an interpolant

$$u^{(i)}(x) = \sum_{j=1}^{M} \lambda_j{}^{(i)} \phi(||x - x_j{}^{(i)}||), \tag{15}$$

where ϕ is some radial basis function and $\lambda_j^{(i)}$ are the corresponding weights. By enforcing the interpolation conditions $u^{(i)}(x_j{}^{(i)}) = u_j{}^{(i)}$ the weights can be obtained by solving the linear system of equations that the interpolation conditions generate:

$$A^{(i)}\lambda^{(i)} = u^{(i)}. \tag{16}$$

Defining $B^{(i)} = (A^{(i)})^{-1}$ with elements $b_{jk}^{(i)}$, leads to $da^{(i)} = B^{(i)}u^{(i)}$ or

$$\lambda^{(i)} = \sum_{k=1}^{M} b_{jk}^{(i)} u_k^{(i)}. \tag{17}$$

For any linear operator L we construct the approximate formula

$$[Lu]_i \approx \sum_{j=1}^{M} \lambda_j{}^{(i)} [L\phi(||x - x_j{}^{(i)}||)]_i, \tag{18}$$

where the notation $[Lu]_i = Lu|_{x=x_i}$ is used. By plugging in Equation (17) into Equation (18) we get

$$[Lu]_i \approx \sum_{j=1}^{M} \sum_{k=1}^{M} b_{jk}^{(i)} u_k^{(i)} [L\phi(||x - x_j{}^{(i)}||)]_i = \sum_{j=1}^{M} \sum_{k=1}^{M} c_{jk}^{(i)} u_k^{(i)}, \tag{19}$$

where the coefficients $c_{jk}^{(i)}$ depend only on L and the coordinates of the nodes of the i-th stencil. This should be done for each stencil X_i in the computational domain Ω. The obtained coefficients from each stencil are then assembled on the diagonal of the M-diagonal global matrix W so that each row contains values from only one stencil. This matrix will be well conditioned and sparse and hence the global system of equations will be easier to solve.

3.3 RBF-PUM

The RBF Partition of Unity Method exploits the idea that was suggested by Babuška and Melenk in [10]. In order to sparsify the linear system in Equation (13) the computational domain is subdivided into M overlapping partitions and a local RBF interpolation is constructed within each partition. Then the local interpolates are combined by the partition of unity weight functions $w_i(x)$, which are subordinated to $\{\Omega_i\}_{i=1}^{M} \supseteq \Omega$ and

$$\sum_{i=1}^{M} w_i(x) = 1. \tag{20}$$

Thus, making the relation between the interpolant on the entire domain $\mathcal{J}_u$ and the local interpolants $\mathcal{J}_u^i$ as

$$\mathcal{J}_u = \sum_{i=1}^{M} w_i(x)\mathcal{J}_u^i = \sum_{i}^{M} w_i(x) \sum_{j=1}^{N_i} \lambda_j^i \phi(|x - x_j|), \quad x \in \Omega, \tag{21}$$

the partition of unity function $w_i(x)$ is defined as

$$w_i(x) = \frac{\varphi_i(x)}{\sum_{k=1}^{M} \varphi_k(x)}, \tag{22}$$

where $\varphi_i(x)$ is chosen to be a C^2 Wendland function compactly supported on Ω_i [7]:

$$\varphi(r) = \begin{cases} (1-r)^4(4r+1), & \text{if } 0 \leq r \leq 1, \\ 0, & \text{if } r > 1. \end{cases} \tag{23}$$

The subdomains Ω_i are chosen as circular patches, meaning that the Wendland functions need to be normalized:

$$\varphi(x) = \phi\left(\frac{||x - c_i||}{r_i}\right), \tag{24}$$

where c_i and r_i is the center point and radius respectively for the domain.

3.4 Time integration scheme used with the RBF-methods

The time integration scheme used together with the RBF-FD method and the RBF-PUM is the backward differentiation formula of second order BDF-2, which is an implicit method that requires solutions from the two previous time levels to compute the solution at the next time level. This means that to compute the solution at the first time step, another method needs to be used. In our case it is BDF-1, also known as implicit Euler method. Since two different methods for the time integration are involved, two LU matrix factorisations have to be performed. However, using the BDF-2 scheme as described in [11], we can avoid double matrix factorisation. The time-interval $[0, T]$ is discretized into M non-uniform steps of length $k_n = t_n + 1 - t_n$, then

$$(I - \beta_0^n L)p_I^1 = p_I^0, \tag{25}$$

and

$$(I - \beta_0^n L)p_I^n = \beta_1^n p_I^{n-1} - \beta_2^n p_I^{n-2}, \quad n = 2, .., M. \tag{26}$$

Where I is the identity matrix, L is the discretized spatial operator $\mathcal{L}$, V_I is the solution in the interior and β are coefficients chosen as

$$\beta_0^n = k^n \frac{1 + w_n}{1 + 2w_n}, \quad \beta_1^n = k^n \frac{(1 + w_n)^2}{1 + 2w_n}, \quad \beta_2^n = k^n \frac{w_n^2}{1 + 2w_n}, \tag{27}$$

where

$$w_n = k^n / k^{n-1}, \quad n = 2, ..., M. \tag{28}$$

3.5 COS

When the characteristic function of a stochastic process is known, we can use transform techniques to evaluate the probability density of this process. We expand the probability density function of our process $f(y|x)$ on the basis of cosine functions, which span the Lebesgue space $L_2([a, b])$,

$$f(y|x) = \frac{F_0}{2} + \sum_{k=1}^{\infty} F_k \cdot \cos \frac{k\pi(x - a)}{b - a}., \tag{29}$$

where the coefficients F_k are given by the formula

$$F_k = \frac{2}{b - a} Re \left[\phi_{GBM}(\frac{k\pi}{b - a}) \cdot e^{-\frac{ik\pi a}{b-a}} \right]. \tag{30}$$

The characteristic function ϕ of a random variable X can be given by: [13]

$$\phi(u) = E\left[e^{iuX}\right]. \tag{31}$$

In the Black-Scholes model the asset follows geometric Brownian motion: $dX_t = rX_t dt + \sigma X_t dW_t$. Using Ito's Lemma we find the characteristic function

$$\phi_{GBM}(u) = e^{iu(\ln(S_0)+(r-\frac{\sigma^2}{2})T)-\frac{\sigma^2 u^2}{2}T} \tag{32}$$

The option value can now be calculated by the Feynman-Kac formula (9) using the obtained probability density. Since we are not able to numerically calculate an integral over an infinite interval, we have to truncate the domain and reduce the integration interval from R to $[a, b]$, where the limits a and b are carefully chosen according to [12] to make sure that the traction error is sufficiently small.

Also, in order to facilitate numerical computation we truncate the infinite series in Equation (29) after N_C number of summands:

$$f(y|x) = \frac{F_0}{2} + \sum_{k=1}^{N_C} F_k \cdot \cos\frac{k\pi(x-a)}{b-a}. \tag{33}$$

N_C is chosen with respect to a preferred error tolerance for the truncation error

In short, we use the characteristic function of the log-normal distribution to obtain the coefficient of the cosine expansion of the density function, then numerically integrating the product of the density function and the payoff and discounting it back in time to the present date according to the Feynman-Kac formula, we obtain the option value.

3.6 Adaptive Finite Differences

Consider a problem of the form

$$\frac{\partial u}{\partial t} + \mathcal{L}u = 0, \tag{34}$$

where $\mathcal{L}$ is a one-dimensional spatial differential operator. Using central second order finite differences on an equidistant grid for the spatial derivatives we can approximate the solution of Equation (5). The resulting approximate solution u_h can then be presented in the form

$$u_h = u + h^2 c(s), \tag{35}$$

where u is the exact solution, h is the step size and c is some coefficient. Truncating the error to these lower order terms we see that $u_{2h} = u + (2h)^2 c(s)$. The local truncation error is defined as $\tau = \mathcal{L}_h u - \mathcal{L}u$, using this with Equation (35) we get

$$\tau_h = \mathcal{L}_h u_h - \mathcal{L}u - h^2 \mathcal{L}_h c(x) \quad , \quad \tau_{2h} = \mathcal{L}_{2h} u_h - \mathcal{L}u - h^2 \mathcal{L}_{2h} c(x), \tag{36}$$

where the operator $\mathcal{L}_{2h}$ is acting on every other element of u_h. Subtracting the expressions in Equation (36) and introducing $\delta_h = \mathcal{L}_h u_h$ and $\delta_{2h} = \mathcal{L}_{2h} u_h$ we get

$$\tau_{2h} - \tau_h = \delta_{2h} - \delta_h - h^2(\mathcal{L}_{2h} - \mathcal{L}_h)c(x) = \delta_2 h - \delta_h + \mathcal{O}(h^4). \tag{37}$$

Neglecting higher order terms and defining $\tau_h = h^2 \eta(s)$ we get

$$\eta(s) = \frac{\delta_{2h} - \delta_h}{3h^2} \quad , \quad \tau(s) = \frac{\delta_{2h} - \delta_h}{3}. \tag{38}$$

Using this $\eta(s)$ can be estimated by obtaining the solution u_h. If we define some tolerance as ε and require that $|\tau_h| = |h^2 \eta(s)| < \varepsilon$ this can be obtained by computing a new solution using the spatial discretization defined by

$$h(s) = \bar{h} \left(\frac{\varepsilon}{|\tau_{\bar{h}}(s)|} \right)^{1/2}. \tag{39}$$

To ensure that not to large steps are taken, a parameter d is introduced and assigned a small value. Equation (39) is rewritten as

$$h(x) = \bar{h} \left(\frac{\varepsilon}{|\tau_{\bar{h}}(s)| + \varepsilon \cdot d} \right)^{1/2}. \tag{40}$$

Since we are working with a time dependent PDE, τ_h will vary in time. To account for this we compute new computational grids at $t = 0$, $t = T/3$ and $t = 2T/3$, where T is the time of maturity. The time integration scheme used with the adaptive FD method is discontinuous Galerkin [14].

3.7 Approximating the forward initial condition

In order to solve for a density function the Fokker-Planck equation is subject to an initial condition

$$p(x, 0) = \delta(x), \tag{41}$$

where $\delta(x)$ is the Dirac delta function,

$$\int_{-\infty}^{\infty} f(x)\delta(x - x_0) = f(x_0). \tag{42}$$

Here x_0 represents the initial price of the underlying stock. Using the Dirac delta function as initial condition is quite problematic since it is a generalised function that is hard to approximate accurately. To circumvent this issue at the first time step of the time integration scheme, the probability density function is approximated using Hermite polynomials as introduced in [4] by Ait-Sahalia. An illustrative example of the approximation can be seen in Figure 1. This function is somewhat smoother and hence much more suitable to be used for computations, compared to the Dirac delta function. From here on the size of the first time step using Ait-Sahalia approximation will be referred to as AS-Δt.

4 Numerical experiments

4.1 Forward solution sensitivity to volatility and Ait-Sahalia time step

From the numerical experiments it is noticed that the solution of the Fokker-Planck equation is sensitive to the size of the Ait-Sahalia time step, thus the Ait-Sahalia time step size needs to be optimized with respect to the accuracy of the numerical solution. Fortunately, in the one-dimensional case there exists an analytical solution for the European option pricing problem, which is used to benchmark against and calibrate the Ait-Sahalia step size. As it turns out the optimal Ait-Sahalia step size in its turn is sensitive to the variance of the underlying stock dynamics σ. The relative error with respect to the Ait-Sahalia step size and volatility is plotted in Figure 2. This sensitivity to σ is not inherent for the tested numerical methods, see for example how the RBF-PUM behaves

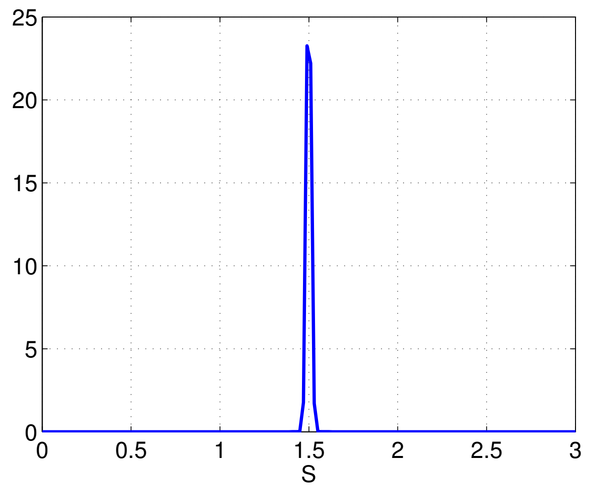

Figure 1: Approximation of the density function at the first time step using Ait-Sahalia.

in backward mode in [6], but rather comes from the Ait-Sahalia time step which does not return good enough approximations when σ is large or very small. In our tests, for comparison reasons, σ is fixed to 0.15. It is possible to decrease the error dependency of σ by adjusting the value of the initial time step, generally a smaller AS-Δt means higher accuracy but there is always a value larger than zero that minimizes the error as we see in the upcoming sections.

4.2 RBF Methods

4.2.1 Optimizing the Ait-Sahalia time step

As S_0 increases, the diffusion part of Equation (5) increases, meaning that the resulting probability density function becomes wider, as visualized in Figure 3. The altered shape of the density function leads to a change in the optimal value of the Ait-Sahalia time step AS-Δt. This is because a wider density function is better approximated by a wider initial condition which is highly affected by the value of AS-Δt. Figure 4 shows how the error changes when the only varying variable is AS-Δt. As can be seen the solution is most accurate when AS-$\Delta t = 0.007$ and AS-$\Delta t = 0.22$ for RBF-FD and RBF-PUM respectively. These are the values we use when benchmarking against the other methods.

4.2.2 Optimizing grid size and shape parameters

Both the number of spatial nodes N and the shape parameter ε affect the precision of the methods. This means that the minimum value of N paired with the optimal value of the shape parameter needs to be found for each error

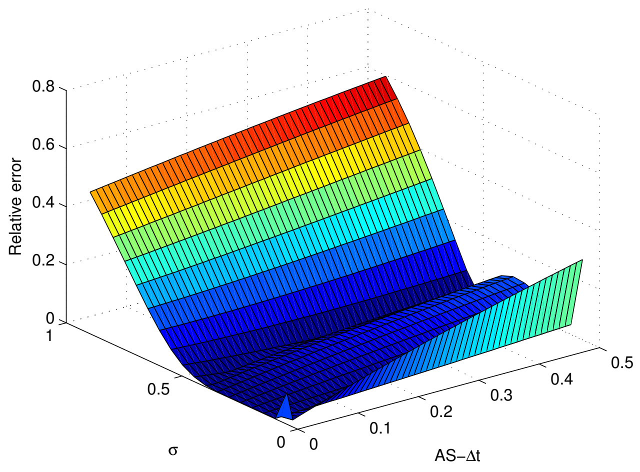

Figure 2: The error of the option price for the RBF-FD method against AS-Δt and σ.

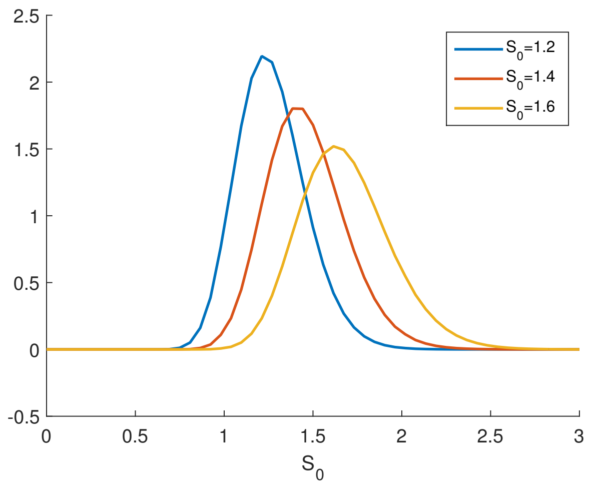

Figure 3: The resulting probability density function when varying only the initial value S_0.

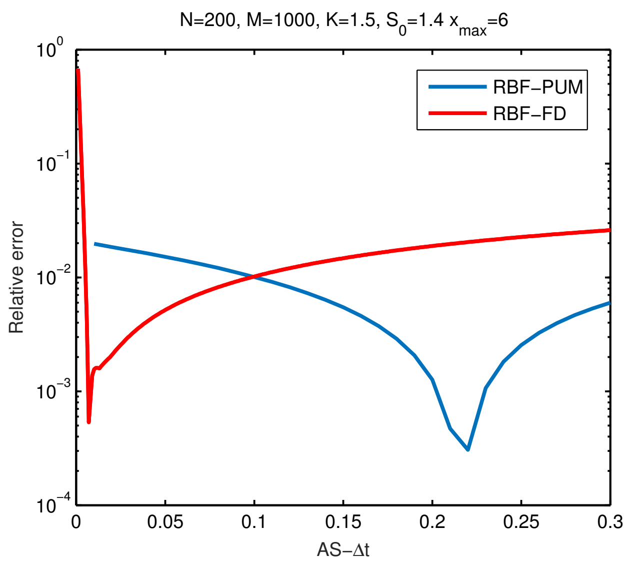

Figure 4: The error as the value of AS-Δt is varied, all other parameters fixed.

	RBF-FD		RBF-PUM	
Error tol.	N	ε	N	ε
1e-2	14	4.92	14	1.9
1e-3	27	7.57	36	3.7
1e-4	66	6.47	50	4.3
1e-5	122	1.21	90	8.8

Table 1: Lowest value of N paired with the optimal value of ε that returns a solution within each specified error tolerance.

tolerance. The number of time steps is set to 500 in order to ensure that it is the spatial resolution that acts as the 'bottleneck' for the precision. For each N the most precise value of ε is found, illustrated in Figure 5 and Figure 6. Starting with a low value of N and testing for a stable value set of ε, plots like these can be acquired. The lowest value of N that produces a solution within each error tolerance is noted and later used when measuring the method's execution times. These values paired with the optimal values of ε are displayed in Table 1.

4.2.3 Stencil size for RBF-FD

The RBF-FD method divides the spatial domain into a set of stencils where each stencil has a center node. Beware that the stencils are overlapping, meaning that all nodes in the domain are center nodes. The number of nodes in the stencils n corresponds to what order of approximation is used for the derivatives. It comes natural to choose the number of stencil nodes as an odd number,

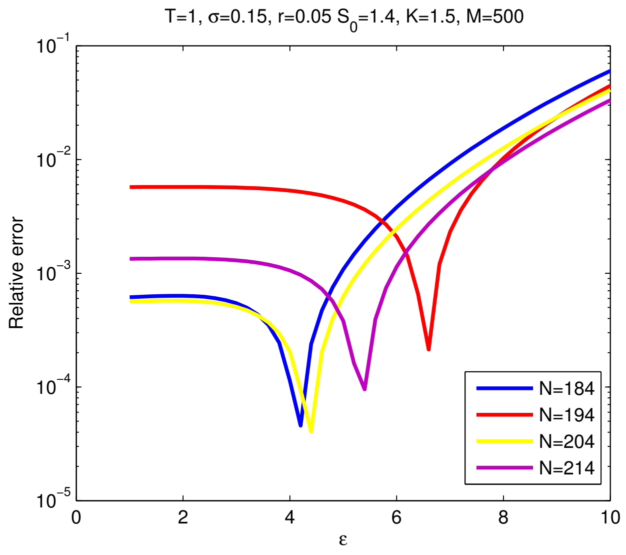

Figure 5: The relative error of the RBF-FD against ε for different resolutions in the spatial dimension.

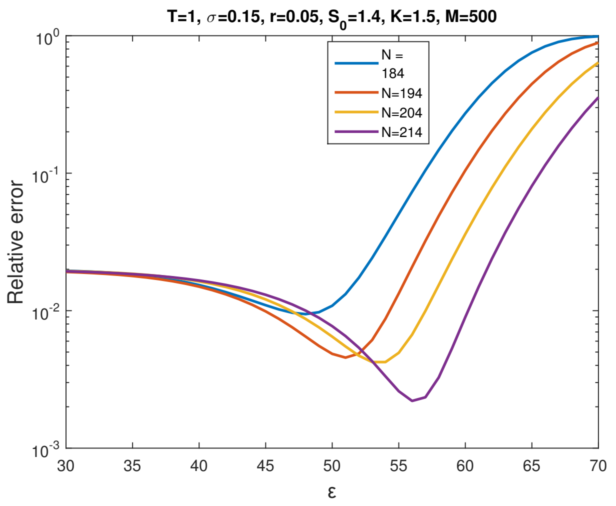

Figure 6: The relative error of the RBF-PUM against ε for different resolutions in the spatial dimension.

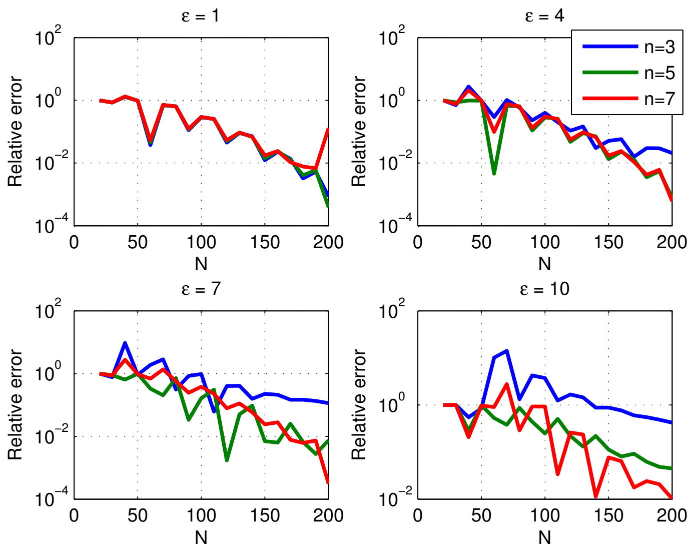

Figure 7: Relative error as a function of the number of spatial nodes N for different shape parameters ε and stencl sizes n.

in the benchmarking tests it was set to n=5.

Figure 7 displays the error as a function of the number of spatial nodes N for different ε and n. We see that when ε increases a larger stencil size can be advantageous for finding a lower N that meet a certain error tolerance. As a clarifying example, consider the subplot with ε=7 and assume we found $N \simeq 170$ as the optimal number of nodes to reach error tolerance 1e-2 for n=7. By decreasing the stencil size to n=5 the same error tolerance can be satisfied for $N \simeq 120$ and thus improving the calculation time.

4.2.4 Number of partitions for RBF-PUM

Increasing the number of partitions results in a more sparse interpolation matrix. This makes the process of solving the system of equation (13) more efficient, but comes at the cost of accuracy. Figure 8 shows how the relative error of the option price alters when increasing the number of partitions and Figure 9 shows how the computational time is affected, the number of space- and time steps are set to a high value in order to demonstrate how these parameters affect the solution. When operating at the number of space- and time steps used together with the RBF-PUM in this report, the computational efficiency gained from increasing the number of partitions is very small compared to the effect of changing the number of spatial nodes The effect will however most likely be more apparent when working with more than one underlying asset. The number

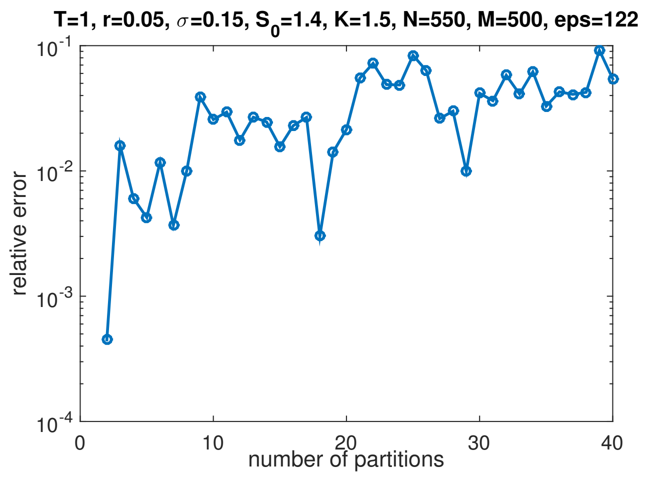

Figure 8: The relative error as the number of partitions is altered, all other parameters fixed.

of partitions is set to four when timing the method.

4.3 COS

The error is analyzed as a function of the computational time it takes to evaluate one option. Figure 10 shows a very steep rise near the 10^{-7} treshold.

Next, the effect of space discretization on the error of the price is evaluated. This is shown in Figure 11, we notice a strongly periodic aspect. We see that even though the general trend tends to lower relative error in a certain periodic manner, there are a few outliers at particularly low discretizations, reaching errors less than 10^{-8}. All parameters are kept the same as above in this experiment.

Lastly, the effect of the number of cosine coefficients on the error is analyzed. The analysis can be seen in Figure 12, extending the number of coefficients beyond the range of the figure has been tested but has provided negligible improvement. In this range, it can be seen that the same behaviour of some coefficients dipping below the general trend as seen when looking at discretization. It has been shown that involving more coefficients is useful when dealing with more complicated payoffs [12].

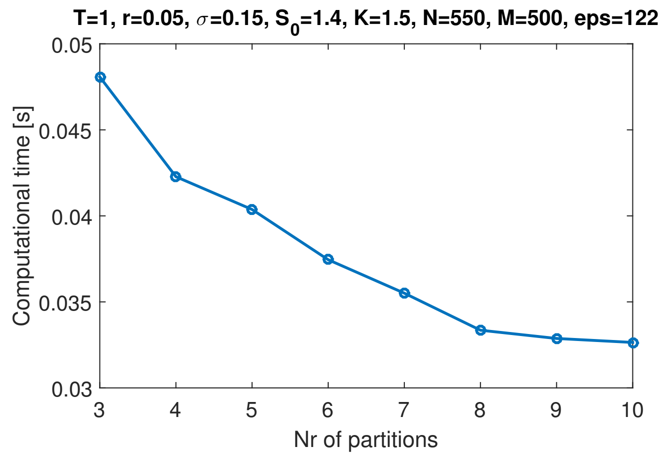

Figure 9: The computational time as the number of partitions is altered, all other parameters fixed.

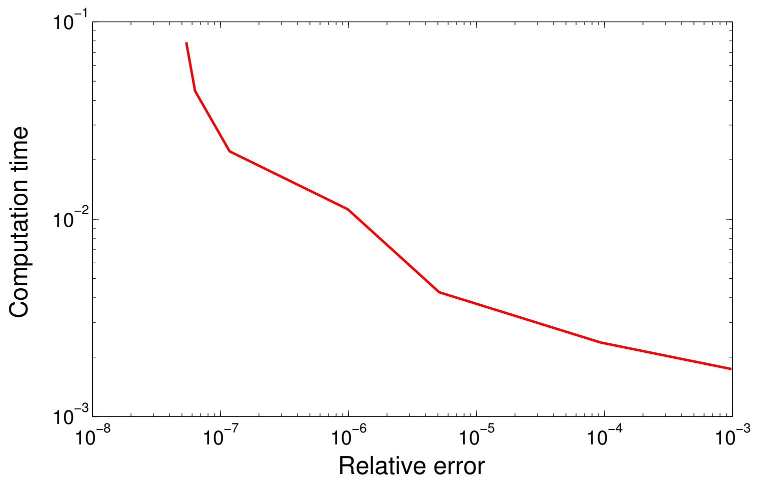

Figure 10: Computational time for one option price evaluation against the relative error

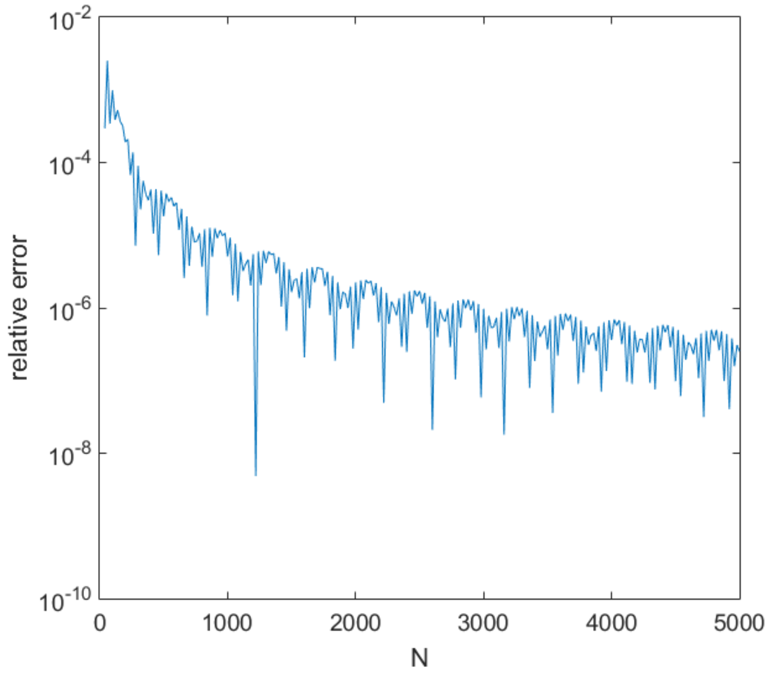

Figure 11: Error of the evaluated option against the number of points used to discretize the stock axis. The numbers of points used to discretize were varied from 20 to 5000 in steps of 20

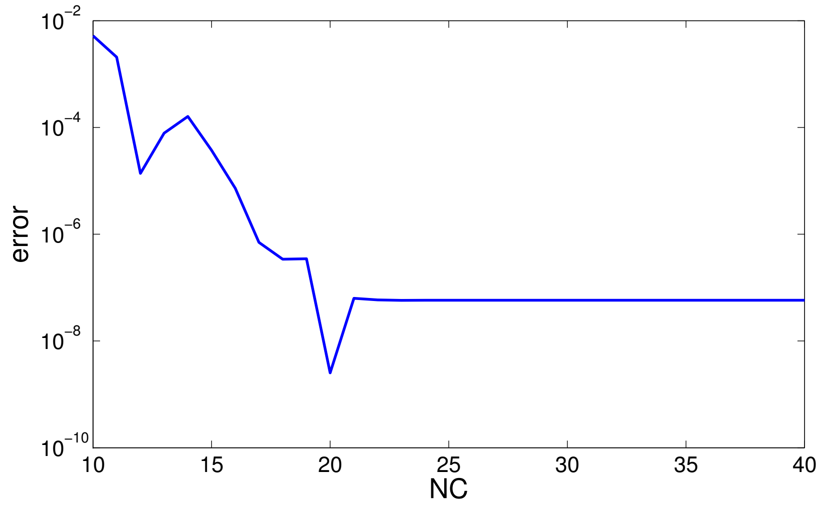

Figure 12: Error analysis of a European call option for different number of cosine coefficients N_C.

4.4 Adaptive FD

4.4.1 Optimizing the Ait-Sahalia time step

All parameters are set to the values used in the benchmark tests and only the value of AS-Δt, the initial time step size used in the Ait-Sahalia approximation, is altered. Figure 13 shows how the relative error varies as this parameter is changed. Under these circumstances the optimal value of proves to be AS-Δt is 0.1581.

4.4.2 Optimizing the adaptive error tolerance

The adaptive FD method remodels the spatial grid, both with respect to the number of nodes and how they are distributed, according to the adaptive tolerance described in Equation (40). In the tests we use an initial spatial grid-size of N=41. Lowering the adaptive tolerance will lead to a better adaptation of the spatial grid, meaning that the precision of the solution increases. This, however, comes at a trade-off with computational efficiency, thus for each benchmark tolerance the highest possible adaptive tolerance needs to be found. Figure 14 shows how the error of both the forward and backward adaptive FD method increases as the adaptive tolerance is increased. The forward method has a smaller error for a given adaptive tolerance than the backward, this most likely is due to that the shape of the density function is more localised than the smoothed pay-off function produced by the backward solution. The forward method converges to a relative error of approximately 1e-5, where the Ait-Sahalia approximations starts to act as a bottleneck for the precision. Starting at an adaptive tolerance

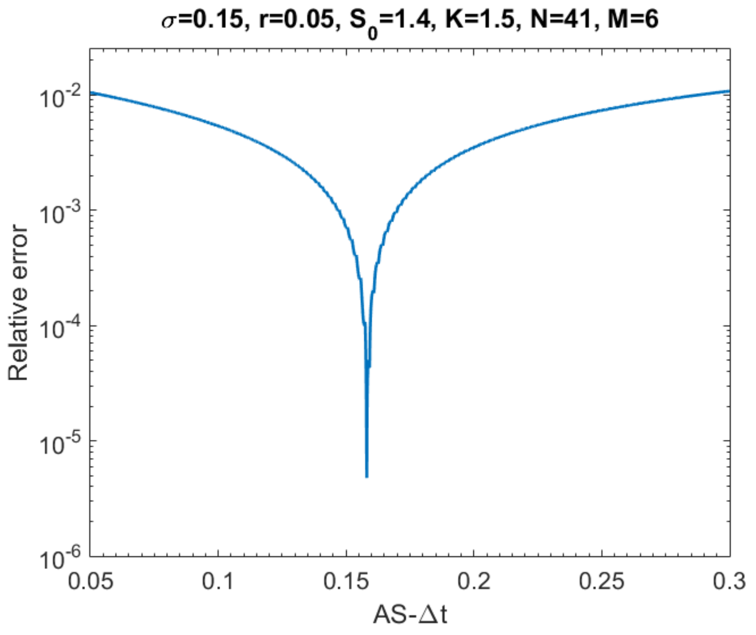

Figure 13: The relative error as a function of the initial time step approximation AS-Δt. The most precise value is found to be $AS - \Delta t = 0.1581$.

of 1e-6 and testing 10 logarithmically spaced values per decade an optimal error tolerance for each benchmark tolerance was found, these values are presented in Table 2.

5 Computational efficiency

To compare the computational efficiency of the different numerical methods the execution-times for a set of different error tolerances were measured. All problems were solved on the spatial domain $x \in [0, 6]$ using strike $K = 1.5$, initial stock value $S_0 = 1.4$, volatility $\sigma = 0.15$, risk-free rate $r = 0.05$ and time of

	Adaptive tolerance	
Error tol.	Forward	Backward
1e-2	9.617e-2	2.868e-3
1e-3	1.477e-2	2.868e-3
1e-4	1.477e-2	1.661e-5
1e-5	2.183e-4	5.151e-6

Table 2: The highest value on the adaptive tolerance that allows to be within each specified error tolerance for the adaptive FD method.

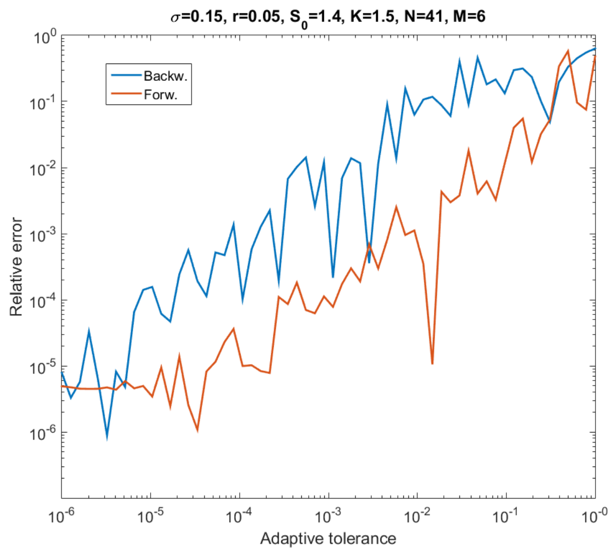

Figure 14: The relative error as a function of the adaptive error tolerance for the forward- and backward-version of the adaptive FD method.

maturity $T = 1$. The code was written in Matlab and the benchmark tests were made on a laptop using an Intel core i7-4510u CPU clocked at 2.60 Ghz. The execution times presented are average times over 1000 runs.

The execution times for all methods for a set of different error tolerances are presented in Table 3, and in Table 4 the spatial- and time resolutions used when benchmarking the RBF-methods are presented. There is not much difference in execution time when comparing the backward and forward version of the RBF methods, although the backward versions need less spatial nodes and therefore are slightly faster. The COS method in backward mode is by far the fastest of all methods and also scales best when lowering the error tolerance. The forward version of the COS is faster than all other forward versions although it experiences a considerable slowdown compared to its backward ditto. The adaptive FD method is the only method that experiences a speedup in forward mode.

In Table 5 the computation times in milliseconds for the different methods evaluating 100 payoffs are presented. In this scenario considerable computational gains are seen for the RBF-FD, RBF-PUM and adaptive FD methods, where they are around 30, 80 and 150 times faster respectively. The COS method is still faster in its backward version compared to the forward one despite the advantage of only having to redo the numerical integration.

	RBF-FD		RBF-PUM		COS		Ad. FD	
Error tol.	Backw.	Forw.	Backw.	Forw.	Backw.	Forw.	Backw.	Forw.
1e-2	1.6	1.5	3.9	2.8	0.1	1.5	2.4	2.4
1e-3	1.6	2.3	4.0	4.3	0.1	1.5	2.4	2.6
1e-4	2.8	8.3	4.2	4.5	0.15	2	3.5	2.6
1e-5	9.6	16	4.5	5.1	0.2	3	4.7	4.3

Table 3: Benhmarking forward vs. backward mode for all methods when pricing one vanilla European call option, time in milliseconds.

	RBF-FD		RBF-PUM	
Error tol.	Backward	Forward	Backward	Forward
1e-2	13×8	14×5	13×6	14×4
1e-3	14×8	27×19	13×8	36×6
1e-4	27×40	66×423	18×10	50×8
1e-5	58×392	122×500	36×10	90×10

Table 4: The spatial- and time resolutions used when benchmarking the RBF-methods, denoted as $N \times M$, where N is the number of spatial nodes and M is the number of time steps.

	RBF-FD		RBF-PUM		COS		Ad. FD	
Error tol.	Backw.	Forw.	Backw.	Forw.	Backw.	Forw.	Backw.	Forw.
1e-4	280	10	420	5.2	6.0	9.0	470	3.1

Table 5: Computational time in milliseconds for the different methods evaluating 100 payoffs.

6 Discussion and conclusion

Our tests have shown that some of the tested methods are more suitable for forward implementation than others. The RBF methods have shown a slight slow down in forward mode compared to backward mode for all tested error tolerances except for 1e-2. This slow down is evident but not considerable, and most likely more extensive testing is needed in order to absolutely determine that the RBF methods work faster in backward mode. A suggestion for potential improvement in forward mode is to refine the resolution of the ε vector that we looped over when optimizing N in 3.1.3, the resolution used was 0.01. Another aspect for improvement could be, for each error tolerance, to optimize the stencil size for RBF-FD and the number of partitions for RBF-PUM.

The results however lead us to believe that the RBF methods are indeed suitable for forward implementation, especially considering the advantage like the one demonstrated in Table 5. In a situation where robustness is prioritized one should prefer the backward method, since it is more robust due to that it does not need to approximate the initial condition.

The COS method has proven to be much more efficient in backward mode than forward mode. This is mostly because the backward mode is faster than most numerical integration techniques as such. The reached accuracy is relatively immense and the time taken in backward mode is not vastly increased. In this case, even when dealing with many payoffs for the same density function, the forward method does not beat the backward method in computational efficiency.

When dealing with forward mode, certain combinations of discretization density and number of cosine coefficients bring the accuracy down to considerable regimes, further research could be done in the relations between these two parameters for forward COS method option pricing.

The adaptive FD method has shown to be the most improved method when comparing its forward version to the backward one. The forward version demands a lower number of spatial steps compared to the backward, this is most likely due to a more favorable shape of the solution, but more tests would be needed to confirm this. The forward version do carry some extra overhead since the initial condition needs to be re-approximated as the grid is refined. This means that using the same adaptive tolerance the execution time is slower for the forward version. Our tests, though, imply that the lower resolution of the spatial grid outweighs this factor to make the forward version more efficient.

To summarize we can draw the conclusion that, as earlier mentioned, there are most definitely cases where a forward implementation is to be preferred compared to a backward. The flexibility due to the payoff function is the most obvious advantage, but as for example as the case of the adaptive FD method indicates, there might also be more straightforward computational gains. The forward version does however introduce a source of instability due to the necessity of approximating the initial condition which, at least for the approximation method used in this report, limits the robustness with respect to large/small values of σ. Note that the COS method does not need this approximation, but on the other hand the backward version of the COS is still faster than the forward one.

Bibliography

[1] F. Black and M. Scholes. The pricing of options and corporate liabilities. *J. Polit. Econ., 81:637–654, 1973.*

[2] J. Amani Rad, L.J. Höök, E. Larsson, L. von Sydow. Forward option pricing using Gaussian RBFs. *Manuscript in preparation, 2015.*

[3] G. E. Fasshauer, Meshfree Approximation Methods with Matlab, *Interdisciplinary Mathematical Sciences: Volume 6, ISBN 978-981-270-633-1.*

[4] Y. Ait-Sahalia. Maximum likelihood estimation of discretely sampled diffusions: A closed-form approximation approach. *Econometrica, 70(1):223–262, 2002.*

[5] I.Tolstykh, D.A.Shirobokov, On using radial basis functions in a finite difference mode with applications to elasticity problems, *Comput. Mech. 33 (2003) 68–79.*

[6] V. Shcherbakov, E. Larsson. Radial basis function partition of unity methods for pricing vanilla basket options. *Computers Mathematics with Applications, Volume 71, Issue 1, 2016, Pages 185-200.*

[7] H.Wendland, Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Adv. Comput. Math. 4 (4) (1995) 389396, doi:10.1007/BF02123482.*

[8] L. von Sydow , L.J. Höök , E. Larsson, E. Lindström, S. Milovanović, J. Persson, V. Shcherbakov, Y. Shpolyanskiy, S. Sirén, J. Toivanen, J. Walden,M. Wiktorsson, J. Levesley, J. Li, C.W. Oosterlee, M.J. Ruijter, A. Toropov, Y. Zhao. BENCHOP - the BENCHmarking project in Option Pricing. *International Journal of Computer Mathematics, Volume 92, Issue 12, 2015, pages 2361-2379*

[9] M. Kac. On Distributions of Certain Wiener Functionals. *Transactions of the American Mathematical Society 65 1949 (1), 1–13*

[10] I. Babuška, J. M. Melenk. The Partition of Unity Method. *Numerical methods in engineering, Volume 40, Issue 4 28 feb 1997 , pages 727-758.*

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

[12] F. Fang, C.W. Oosterlee. A novel pricing method for European options based on Fourier-cosine series expansions. *SIAM Journal on Scientific Computing, 2009, Vol. 31, No. 2, pages 826-848*

[13] A Hirsa. Computational methods in finance, *2013*

[14] L. von Sydow. On discontinuous Galerkin for time integration in option pricing problems with adaptive finite differences in space. *Numerical Analysis and Applied Mathematics: ICNAAM 2013, AIP Conference Proceedings, vol. 1558, American Institute of Physics (AIP), Melville, NY, 2013, pages 2373–2376.*