

# A Krylov subspace method for option pricing

Jitse Niesen\* and Will M. Wright†

March 30, 2011

---

## Abstract

We consider the pricing of financial contracts that are based on two or three underlyings and are modelled using time dependent linear parabolic partial differential equations (PDEs). To provide accurate and efficient numerical approximations to the financial contract's value, we decompose the numerical solution into two parts. The first part involves the spatial discretization, using finite difference methods of the governing PDE. From this a large system of ordinary differential equations (ODEs) with a special affine structure is obtained. On the second part, we develop highly efficient numerical methods to approximate the exact solution of the system of ODEs with a high level of accuracy. We compare our approach, which is based on Krylov subspace methods, with the Crank–Nicolson and Alternating Direct Implicit (ADI) methods on Rainbow and Basket options, European call options using Heston stochastic volatility, and power reverse dual currency (PRDC) swaps with Bermudan and knockout features.

---

## 1 Introduction

In realistic financial applications, pricing derivative securities analytically is rarely possible. Traders making markets in complex structured products require fast pricing to obtain a competitive advantage. Typically, these practitioners value derivative securities using numerical techniques, which can broadly be broken down into one of two types: first, the simulation of many realizations of a stochastic process or, second the numerical solution of parabolic partial differential equations (PDEs). Simulation-based or Monte Carlo methods are most suitable for high-dimensional problems because the convergence is independent of the dimension; they do not suffer from the curse of dimensionality. For a comprehensive discussion on Monte Carlo-based methods we recommend the books by Glasserman (2004) and Joshi (2008). PDE-based methods are most suitable if the dimension of the PDE is low; generally the figure quoted in the literature is around five, but the actual dimension at which the crossover takes place depends strongly on the problem and implementation. Several excellent books have been written focusing on financial applications of the numerical solution of PDEs, for example Duffy (2006), Tavella and Randall (2000) as well as Wilmott (2006).

In this paper, we shall focus our attention on the PDE approach. In particular, we develop a highly efficient numerical method that approximates linear parabolic PDEs; resulting in large computational improvements in the pricing of derivative securities. The standard approach to compute numerical approximations to PDEs is first to discretize the PDE in space, resulting in a large system of ordinary differential equations (ODEs), which must then be solved numerically. This is sometimes referred to as the method of lines. As the dimension of the PDE increases, the number of ODEs generally increases exponentially; this is the curse of dimensionality. The aim is to compute numerical approximations to the PDE as efficiently as possible. To the best of our knowledge, the technique proposed in this paper has not been used previously in financial applications.

In most financial applications the spatial domain of interest is infinite, but after a suitable truncation the domain is usually rectangular. For this reason, the PDEs are discretized using finite differences rather than finite elements, which are most often used for more complicated domains. In this paper, we focus our interest on the efficient numerical solution of the large system of ODEs that result from the discretization of

---

\*University of Leeds, [jitse@maths.leeds.ac.uk](mailto:jitse@maths.leeds.ac.uk).

†University of Melbourne and La Trobe University, [w.wright@latrobe.edu.au](mailto:w.wright@latrobe.edu.au).

the PDEs. In mathematical finance, one commonly used method is the fully implicit second-order accurate Crank–Nicolson method, which is A-stable but not L-stable.<sup>1</sup> The fully implicit nature of the Crank–Nicolson method renders it computationally inefficient as the number of ODEs increases. A popular alternative is the alternating direction implicit (ADI) methods, considered to be the-state-of-the-art in financial modelling; see Andersen and Piterbarg (2010). The most popular ADI methods were developed by Craig and Sneyd (1988) and by Hundsdorfer and Verwer (2003); their popularity stems from the fact that they can handle PDEs with mixed spatial derivatives explicitly and the remaining derivatives separately, but implicitly, while maintaining overall unconditional stability and second-order accuracy, for PDEs with less than four spatial dimensions. For PDEs with four or more spatial dimensions the method of Hundsdorfer–Verwer scheme remains unconditionally stable and second-order accurate; see in 't Hout and Welfert (2009). In in 't Hout and Foulon (2010) it has been pointed out that the Craig–Sneyd scheme can exhibit undesirable convergence behaviour for non-smooth payoffs, which are commonplace in financial applications.

In recent years, there has been renewed interest in a class of numerical methods known as exponential integrators. For an introduction to exponential integrators, we suggest the review articles by Minchev and Wright (2005) or Hochbruck and Ostermann (2010). The idea behind exponential integrators is to split the differential equation into a part that can be computed exactly and a part that needs to be computed numerically: in that sense, they are related to splitting schemes. One common application of exponential integrators is when the ODEs can be split into a linear and nonlinear part. The exact solution of a linear ODEs is well known to be the product of the matrix exponential and the initial condition. These exponential integrators solve exactly the linear part of the problem and then make corrections to cope with the nonlinear terms. If the system of ODEs is very large, then it may be impossible to compute the exact solution of the linear part exactly, but efficient and accurate approximations can be computed using Krylov subspace methods. This problem has been dealt with by Sidje (1998). We shall use his results extensively in this paper. For an introduction to Krylov subspace methods we suggest the book by Saad (2003). In financial engineering, almost all the PDEs that are studied are linear, with reasonably simple boundary conditions. After the spatial discretization, the system of ODEs is generally linear or affine, depending on the type of boundary conditions used. Neither the Crank–Nicolson nor the ADI methods take full advantage of the fact that the system of ODEs has this special structure. The main aim of this paper is to show that exponential integrators, which take advantage of this special structure, can provide highly efficient numerical algorithms for pricing various financial contracts.

In summary, we are able to price rainbow and basket option contracts based on three underlyings, to within one basis point accuracy, ten times faster than the best ADI methods. Also we can price 30 year PRDC swaps, using a three factor model, to around one basis point of accuracy in only a few seconds using MATLAB on a notebook computer. Our approach has inbuilt adaptivity, removing a significant amount of the trial and error required to convince practitioners that the derivative prices provided are sufficiently accurate. Interested users can download MATLAB and C++ implementations of the code.

In Section 2, we set up the PDE framework used in this paper. Finite difference approximations of the PDEs are discussed in Section 3. Exponential integrators used to solve the ODEs are developed in Section 4. In Sections 5, basket and rainbow options are discussed. European options with Heston's stochastic volatility are outlined in Section 6. We apply the exponential integrators to the cross-currency interest rate PRDC swaps in Section 7. Extensive numerical experiments are conducted in each relevant section. Finally, Section 8, outlines our findings and discusses future work.

## 2 Model

We set-up the model following Dang et al. (2010b). The details are included here for the readers convenience. The price  $u(x_1, x_2, \dots, x_n, \tau)$  of various financial derivatives satisfy the parabolic PDE

$$\frac{\partial u}{\partial \tau} = \sum_{\ell, m=1}^n c_{\ell m} \frac{\partial^2 u}{\partial x_\ell \partial x_m} + \sum_{\ell=1}^n c_\ell \frac{\partial u}{\partial x_\ell} + c_0 u, \quad (1)$$

<sup>1</sup>A method is A-stable if its stability domain satisfies  $S \supset \mathbf{C}^- = \{z; \operatorname{Re} z \leq 0\}$ , where  $R(z)$  is the stability function, while it is L-stable if it is A-stable and  $\lim_{z \rightarrow \infty} R(z) = 0$ . The stability function  $R(z)$  is one step of the numerical method, with stepsize  $h$ , on the problem  $y' = \lambda y$ , with  $z = h\lambda$ ; the stability function satisfies  $y_{n+1} = R(z)y_n$ .

in the domain  $\Omega \times (0, T]$  subject to the initial condition

$$u(x_1, x_2, \dots, x_n, 0) = u_0(x_1, x_2, \dots, x_n), \quad (\partial\Omega \cup \Omega) \times \{0\},$$

where  $\Omega \subset \mathbf{R}^n$  is the spatial domain,  $\partial\Omega$  is the boundary of  $\Omega$ ,  $(0, T]$  with  $T > 0$  is the time domain and the coefficients  $c_{\ell m}$  and  $c_\ell$ , for  $\ell, m = 1, \dots, n$  are functions of  $x_1, x_2, \dots, x_n, \tau$ . To ensure, that the PDE in Equation (1) is well-posed, we require that the coefficient matrix  $[c_{\ell m}]_{1 \leq \ell, m, \leq n}$  is positive semi-definite. Generally, in financial applications,  $\Omega$  is an infinite domain. To compute approximations numerically, one must truncate  $\Omega$  to a finite domain. We assume that the domain of interest is the rectangular domain  $\Omega = [L_{x_1}, U_{x_1}] \times [L_{x_2}, U_{x_2}] \times \dots \times [L_{x_n}, U_{x_n}] \subset \mathbf{R}^n$ . Note that if the domain has been truncated, it must be sufficiently large so that the boundary conditions on the truncated sides do not affect the solution  $u$  in the region of interest much. This can often be achieved in a probabilistic way by requiring that the underlying stochastic processes have very small probability of reaching the boundary. In this paper, we only consider examples where  $n \leq 3$ . The main results of this paper are valid for larger values of  $n$ , except that sparse grids rather than finite difference grid are required; see Section 3. In this section, we assume linear boundary conditions hold, this is an unnecessary requirement, various types of boundary conditions are valid. In fact, in Section 6, we consider an alternative choice. Linear boundary conditions require that  $u(x_1, x_2, x_3, \tau)$  is linear in  $x_\ell$  at the boundaries  $x_\ell = L_{x_\ell}$  and  $x_\ell = U_{x_\ell}$ , for  $\ell = 1, \dots, n$ . Therefore,

$$\left. \frac{\partial^2 u}{\partial x_\ell^2} \right|_{L_{x_\ell}} = \left. \frac{\partial^2 u}{\partial x_\ell^2} \right|_{U_{x_\ell}} = 0.$$

This significantly simplifies the parabolic PDE in Equation (1) when we are on an open face of the boundary, on an edge of the boundary or at a corner of the boundary. When we are at a corner of the boundary  $\partial\Omega$ , so either  $x_1 = L_{x_1}$  or  $x_1 = U_{x_1}$  and either  $x_2 = L_{x_2}$  or  $x_2 = U_{x_2}$  and either  $x_3 = L_{x_3}$  or  $x_3 = U_{x_3}$ , then the parabolic PDE at the corner is given by

$$\frac{\partial u}{\partial \tau} = \sum_{\ell, m=1, \ell \neq m}^n c_{\ell m} \frac{\partial^2 u}{\partial x_\ell \partial x_m} + \sum_{\ell=1}^n c_\ell \frac{\partial u}{\partial x_\ell} + c_0 u.$$

In the situation of most interest here when  $n = 3$ , similar equations for the edges and the faces can also be derived for these equations; see Dang et al. (2010b).

### 3 Spatial Discretization

We now discuss the spatial discretization of the PDE in Equation (1), when  $n = 3$ . For most financial applications, the value of  $u$  is required in a smaller region  $\tilde{\Omega} \subset \Omega$  of the domain than the approximation is computed, in some cases  $\tilde{\Omega}$  can even be a single point. For this reason, it is useful to concentrate the grid within the region of most importance. Therefore, we employ a variable grid in each of the three spatial directions. Now, for  $\ell = 1, \dots, 3$ , let  $x_{\ell,1} = L_{x_\ell}$  and  $x_{\ell, N_\ell} = U_{x_\ell}$ , split each direction into  $N_\ell - 1$  (possibly uneven) pieces. This results in a total of  $N_1 N_2 N_3$  grid points. Define the length between the spatial grid points  $\Delta x_{\ell, i} = x_{\ell, i+1} - x_{\ell, i}$ .

The first partial derivative  $\frac{\partial u}{\partial x_1}$  can be approximated, at the point  $(x_{1,i}, x_{2,j}, x_{3,k})$ , by

$$\left( \frac{\partial u}{\partial x_1} \right)_{i,j,k} \approx \beta_{1,-1,i} u_{i-1,j,k} + \beta_{1,0,i} u_{i,j,k} + \beta_{1,1,i} u_{i+1,j,k},$$

where the coefficients  $\beta_{1,-1,1}$ ,  $\beta_{1,-1,1}$  and  $\beta_{2,-1,1}$  are

$$\beta_{1,-1,1} = 0, \quad \beta_{1,0,1} = -\frac{1}{\Delta x_{1,1}}, \quad \beta_{1,1,1} = \frac{1}{\Delta x_{1,1}}.$$

This results in a first-order accurate approximation. Given that the domain is truncated so the effects of the boundary conditions are minimal in the region of interest, using a first-order approximation on the boundary

should have little overall effect on the numerical approximation. The coefficients  $\beta_{1,-1,i}$ ,  $\beta_{1,0,i}$  and  $\beta_{1,1,i}$ , for  $i = 2, \dots, N_1 - 1$  are

$$\beta_{1,-1,i} = \frac{-\Delta x_{1,i}}{\Delta x_{1,i-1}(\Delta x_{1,i-1} + \Delta x_{1,i})}, \quad \beta_{1,0,i} = \frac{\Delta x_{1,i} - \Delta x_{1,i-1}}{\Delta x_{1,i-1}\Delta x_{1,i}}, \quad \beta_{1,1,i} = \frac{\Delta x_{1,i-1}}{\Delta x_{1,i}(\Delta x_{1,i-1} + \Delta x_{1,i})}.$$

This results in second-order accurate approximations for all grid points not on the boundary. Finally, the coefficients  $\beta_{1,-1,N_1}$ ,  $\beta_{1,0,N_1}$  and  $\beta_{1,1,N_1}$  are

$$\beta_{1,-1,N_1} = -\frac{1}{\Delta x_{1,N_1-1}}, \quad \beta_{1,0,N_1} = \frac{1}{\Delta x_{1,N_1-1}}, \quad \beta_{1,1,N_1} = 0.$$

Again, this is a first-order approximation on the boundary, but has little overall effect on accuracy. Similar expressions hold for  $\frac{\partial u}{\partial x_2}$  and  $\frac{\partial u}{\partial x_3}$ . The second partial derivative  $\frac{\partial^2 u}{\partial x_2^2}$  can be approximated, to second-order accuracy, at the point  $(x_{1,i}, x_{2,j}, x_{3,k})$ , not on the boundary given that Equation (2) holds, by

$$\left( \frac{\partial^2 u}{\partial x_2^2} \right)_{i,j,k} \approx \gamma_{2,-1,i} u_{i,j-1,k} + \gamma_{2,0,i} u_{i,j,k} + \gamma_{2,1,i} u_{i,j+1,k},$$

where the coefficients  $\gamma_{2,-1,i}$ ,  $\gamma_{2,0,i}$  and  $\gamma_{2,1,i}$ , for  $i = 2, \dots, N_2 - 1$ , are defined

$$\gamma_{2,-1,i} = \frac{2}{\Delta x_{1,i-1}(\Delta x_{1,i-1} + \Delta x_{1,i})}, \quad \gamma_{2,0,i} = \frac{-2}{\Delta x_{1,i-1}\Delta x_{1,i}}, \quad \gamma_{2,1,i} = \frac{2}{\Delta x_{1,i}(\Delta x_{1,i-1} + \Delta x_{1,i})}.$$

Similar expressions hold for  $\frac{\partial^2 u}{\partial x_1^2}$  and  $\frac{\partial^2 u}{\partial x_3^2}$ . The mixed partial derivative  $\frac{\partial^2 u}{\partial x_1 \partial x_3}$  is approximated, at the point  $(x_{1,i}, x_{2,j}, x_{3,k})$ , by

$$\left( \frac{\partial^2 u}{\partial x_1 \partial x_3} \right)_{i,j,k} \approx \sum_{l,m=-1}^1 \beta_{1,l,i} \beta_{3,m,k} u_{i+l,j,k+m}.$$

This results in a nine point stencil when the grid points are not equally spaced in each direction and a four point stencil when they are equally spaced in each direction. The approximation is of first order on the boundary and second order everywhere else in the domain.

We want to be able to express each of the partial derivatives in Equation (1) compactly, at each point on the spatial grid. To do this, we must decide on a particular ordering of the grid points. Given we have  $N_1 N_2 N_3$  grid points, let the vector  $u \in \mathbf{R}^{N_1 N_2 N_3}$  denote the value of our financial contract at each of the points in our grid, where the value of  $u$  at the grid point  $(x_{1,i}, x_{2,j}, x_{3,k})$  is given by element number  $i + (j-1)N_1 + (k-1)N_1 N_2$ . Let  $T_{N_1}$ , an  $N_1 \times N_1$  tridiagonal matrix, be the non-uniform first-order difference operator

$$T_{N_1} = \begin{bmatrix} \beta_{1,0,1} & \beta_{1,1,1} & 0 & \cdots & 0 & 0 & 0 \\ \beta_{1,-1,2} & \beta_{1,0,2} & \beta_{1,1,2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & & & \\ 0 & 0 & 0 & \cdots & \beta_{1,-1,N_1-1} & \beta_{1,0,N_1-1} & \beta_{1,1,N_1-1} \\ 0 & 0 & 0 & \cdots & 0 & \beta_{1,-1,N_1} & \beta_{1,0,N_1} \end{bmatrix}.$$

Now, the partial derivative  $\frac{\partial u}{\partial x_1}$ , at the grid point  $(x_{1,i}, x_{2,j}, x_{3,k})$  can be approximated by element number  $i + (j-1)N_1 + (k-1)N_1 N_2$  in the product of an  $(N_1 N_2 N_3) \times (N_1 N_2 N_3)$  matrix and the  $N_1 N_2 N_3$  vector  $u$  as follows

$$\frac{\partial u}{\partial x_1} \approx (I_{N_3} \otimes I_{N_2} \otimes T_{N_1}) u, \quad (2)$$

where  $I_{N_2}$  and  $I_{N_3}$  are the  $N_2 \times N_2$  and  $N_3 \times N_3$  identity matrices and  $\otimes$  represents the tensor or Kronecker product. Similar expressions hold for  $\frac{\partial u}{\partial x_2}$  and  $\frac{\partial u}{\partial x_3}$ . Let  $S_{N_2}$ , an  $N_2 \times N_2$  tridiagonal matrix, be the non-uniform second order difference operator

$$S_{N_2} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \gamma_{1,-1,2} & \gamma_{1,0,2} & \gamma_{1,1,2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & & & \\ 0 & 0 & 0 & \cdots & \gamma_{1,-1,N_1-1} & \gamma_{1,0,N_1-1} & \gamma_{1,1,N_1-1} \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix}.$$

The first and last rows are zero because linear boundary conditions were assumed in Equation (2). Now, the partial derivative  $\frac{\partial^2 u}{\partial x_2^2}$ , at the grid point  $(x_{1,i}, x_{2,j}, x_{3,k})$  can be approximated by element number  $i + (j - 1)N_1 + (k - 1)N_1N_2$  in the product of an  $(N_1N_2N_3) \times (N_1N_2N_3)$  matrix and the  $N_1N_2N_3$  vector  $u$  as follows

$$\frac{\partial^2 u}{\partial x_2^2} \approx (I_{N_3} \otimes S_{N_2} \otimes I_{N_1})u. \quad (3)$$

Similar expressions hold for  $\frac{\partial^2 u}{\partial x_1^2}$  and  $\frac{\partial^2 u}{\partial x_3^2}$ . The beauty of the tensor notation lies in the mixed derivative terms. The partial derivative  $\frac{\partial^2 u}{\partial x_1 \partial x_3}$ , at the grid point  $(x_{1,i}, x_{2,j}, x_{3,k})$  can be approximated by element number  $i + (j - 1)N_1 + (k - 1)N_1N_2$  in the product of an  $(N_1N_2N_3) \times (N_1N_2N_3)$  matrix and the  $N_1N_2N_3$  vector  $u$  as follows

$$\frac{\partial^2 u}{\partial x_1 \partial x_3} \approx (T_{N_3} \otimes I_{N_2} \otimes T_{N_1})u. \quad (4)$$

Similar expressions hold for  $\frac{\partial^2 u}{\partial x_1 \partial x_2}$  and  $\frac{\partial^2 u}{\partial x_2 \partial x_3}$ . Equations (2), (3), and (4) and similar expressions are all that are needed for a complete spatial discretization of Equation (1).

After we have performed the spatial discretization for each of the partial derivatives on the right-hand side of Equation (1), we are left with a system  $N = N_1N_2N_3$  ODEs of the form

$$\frac{du(\tau)}{d\tau} = A(\tau)u(\tau) + b(\tau), \quad u(\tau_0) = u_0. \quad (5)$$

The vector  $b : \mathbf{R} \rightarrow \mathbf{R}^N$  collects the boundary terms, which do not naturally fit into  $A(\tau)$ . The boundary conditions chosen lead to a free boundary, so  $b(\tau) = 0$ . Generally, this need not be the case for other spatial discretizations. In many applications in finance, the coefficients  $c_{\ell,m}$  and  $c_\ell$  for  $\ell, m = 1, \dots, n$  in the PDE of Equation (1) are independent of time. In this case, the discretization matrix is constant leading to the system of ODEs

$$\frac{du(\tau)}{d\tau} = Au(\tau) + b(\tau), \quad u(\tau_0) = u_0. \quad (6)$$

Our main aim is how to integrate ODEs like (5) most efficiently, on-going work is concentrating on (6).

This is a good time to discuss three important points: first, even though we have chosen to adopt the linear boundary conditions which lead to each boundary being a free boundary, it is also very common, in financial applications, to fix the boundary conditions; see for example Tavella and Randall (2000). In this case, each of the partial derivatives given by Equations (2), (3) and (4) are slightly modified by the addition of a, possibly time dependent, vector, which collects the values at the boundary. Second, we are interested in derivatives which depend on three state variables. Financial contracts often depend on more than three state variables, consider the case when we have  $n$  state variables, in this case we have  $N_1N_2 \dots, N_n$ , ODEs to solve numerically. This quickly becomes, for large  $n$ , computationally challenging. The use of sparse grids (see for example the paper Leentvaar and Oosterlee (2008)) can lead to possibility of approximating financial contracts with more than three state variables using the PDE approach. Even when sparse grids are used to perform the spatial discretization the resulting system of ODEs are of the form (5) or (6). Therefore, the results presented in Sections 4 are still relevant to that situation. Finally, given that the domain is rectangular the finite element methods do not seem to provide any real advantage, we are currently considering whether it is possible to use spectral methods efficiently.

Before completing this section, we discuss the type of non-uniform meshes that will be used in our experiments. We follow the approach outlined by Tavella and Randall (2000) as well as Kluge (2002). Let integer  $N_\ell \geq 1$  and constants  $c_\ell > 0$  and  $K_\ell > 0$  for  $\ell = 1, \dots, 3$ . Now define the mesh in the  $x_\ell$ -direction. Let equidistant points  $\xi_{\ell,1} < \xi_{\ell,2} < \dots < \xi_{\ell,N_\ell}$  satisfy

$$\xi_{\ell,i} = \sinh^{-1} \left( \frac{L_{x_\ell} - K_\ell}{c_\ell} \right) + i\Delta\xi, \quad 1 \leq i \leq N_\ell,$$

where

$$\Delta\xi = \frac{1}{N_\ell} \left[ \sinh^{-1} \left( \frac{U_{x_\ell} - K_\ell}{c_\ell} \right) - \sinh^{-1} \left( \frac{L_{x_\ell} - K_\ell}{c_\ell} \right) \right].$$

Then a non-uniform mesh  $L_{x_\ell} = x_{\ell,1} < x_{\ell,2} < \dots < x_{\ell,N_\ell} = U_{x_\ell}$  is defined through the transformation

$$x_{\ell,i} = K_\ell + c_\ell \sinh(\xi_{\ell,i}), \quad 1 \leq i \leq N_\ell.$$

The parameter  $c_\ell$  control the fraction of mesh points  $x_\ell$  that lie in the neighbourhood of the strike  $K_\ell$ . In particular,  $\Delta x_{\ell,i} \approx c_\ell \Delta \xi$ , whenever  $x_{\ell,i} \approx K_\ell$ . We note that Andersen and Piterbarg (2010) have proposed alternative approaches to grid selection.

## 4 ODEs

Our main aim now is the efficient numerical solution of the large system of ODEs given in Equations (5) and (6). The most commonly used numerical method to solve these ODEs in computational finance seems to be the method proposed by Crank and Nicolson (1947), which for readers familiar with Runge–Kutta methods is the trapezoidal rule developed by Runge (1895). The numerical solution of ODEs is one of the most highly developed areas of applied mathematics, with sophisticated numerical algorithms and corresponding implementations; the reader is referred to the book by Butcher (2008) as well as the two volumes by Hairer et al. (1993) and Hairer and Wanner (1996) for an extensive overview of this active research area.

Generally, ODEs come in two flavours, nonstiff and stiff. In computational finance, we are almost always dealing with stiff ODEs, given that the ODEs result from a spatial discretization of a PDE. Usually, stiff ODEs require implicit methods for their efficient numerical solution because the eigenvalues of  $A(\tau)$  have large negative real parts, and the stability region of the implicit methods contain these eigenvalues, so efficiency rather than stability controls the size of the integration steps.

ADI methods are implicit methods which split the ODE in Equation (6) into pieces and approximate each piece separately; see Craig and Sneyd (1988), Hundsdorfer and Verwer (2003) as well as in 't Hout and Foulon (2010). The advantage then is that it is computationally more efficient to solve each piece separately and combine the results, than to solve the ODE in its entirety. ADI methods use the splitting  $A = A_0 + A_1 + \dots + A_n$  where  $A_0$  contains all the mixed terms and  $A_\ell$  (for  $\ell = 1, \dots, n$ ) are the terms involving partial derivatives in the  $x_\ell$  direction. The mixed term is treated explicitly and the other terms are treated implicitly. in 't Hout and Welfert (2007) have shown that certain ADI methods are A-stable, despite the explicit step; this is generally regarded as a necessary condition for efficient solution of stiff ODEs. It turns out that the ADI methods are actually a special case (albeit a highly efficient special case) of the additive Runge–Kutta methods, which were introduced by Cooper and Sayfy (1980, 1983). Order conditions for the additive Runge–Kutta methods are discussed in the paper Murua (1999).

In recent years, an alternative approach to implicit methods, known as exponential integrators, has resurfaced. These methods are designed to take advantage of the simple nature of the system of ODEs given in Equations (5) and (6). An essential role in the exponential integrators is played by the so-called  $\varphi$ -functions. For scalar arguments, these functions are defined by the integral representation

$$\varphi_0(z) = e^z, \quad \varphi_\ell(z) = \frac{1}{(\ell-1)!} \int_0^1 e^{(1-\theta)z} \theta^{\ell-1} d\theta, \quad \ell = 1, 2, \dots, z \in \mathbf{C}.$$

The first few  $\varphi$ -functions are

$$\varphi_1(z) = \frac{e^z - 1}{z}, \quad \varphi_2(z) = \frac{e^z - 1 - z}{z^2}, \quad \varphi_3(z) = \frac{e^z - 1 - z - \frac{1}{2}z^2}{z^3}.$$

The  $\varphi$ -functions satisfy the recurrence relation

$$\varphi_\ell(z) = z\varphi_{\ell+1}(z) + \frac{1}{\ell!}, \quad \ell = 1, 2, \dots$$

The definition can then be extended to matrices instead of scalars using any of the available definitions of matrix functions, such as those based on the Jordan canonical form; see Higham (2008).

The  $\varphi$ -functions are important in the solution of Equations (5) and (6) because they provide the exact solution of linear differential equations with polynomial inhomogeneity, as illustrated by the following lemma.

**Lemma 4.1** *Let  $A \in \mathbf{C}^{N \times N}$  and  $b_1, \dots, b_p \in \mathbf{C}^N$ , then the non-autonomous linear initial value problem*

$$\frac{du(\tau)}{d\tau} = Au(\tau) + \sum_{j=0}^{p-1} \frac{\tau^j}{j!} b_{j+1}, \quad u(\tau_0) = u_0, \quad (7)$$

has the solution

$$u(\tau_0 + h) = \varphi_0(hA)u_0 + \sum_{j=0}^{p-1} \sum_{\ell=0}^j \frac{\tau_0^{j-\ell}}{(j-\ell)!} h^{\ell+1} \varphi_{\ell+1}(hA)b_{j+1}. \quad (8)$$

The proof of the above lemma can be found in Niesen and Wright (2011). In computational finance, the boundary effects are generally small so one can approximate  $b(\tau)$  by a polynomial, in which case the exact solution is known. Our aim is to evaluate Equation (8) as accurately and efficiently as possible. The reader may have noted that computing the matrix functions  $\varphi_0(hA), \dots, \varphi_p(hA)$  is likely to be an order  $N^3$  operation. In the discretization, described in Section 3, there are  $N_1 N_2 N_3$  ODEs. For example, if we split each of the space dimensions into  $10^2$  intervals, we end up with  $10^6$  ODEs, and thus it would require in the order of  $p(10^6)^3$  operations to compute Equation (8), which is intractable, so we need to consider alternatives.

The first step is to combine the matrix functions into one. This is achieved by the following lemma, which shows that it is possible to evaluate Equation (8) using only the product of one  $(N+p) \times (N+p)$  matrix exponential and an  $N+p$  vector.

**Lemma 4.2** *Let  $A \in \mathbf{C}^{N \times N}$ ,  $B = [b_p, \dots, b_2, b_1] \in \mathbf{C}^{N \times p}$ ,  $s = \left[ \frac{\tau_0^{p-1}}{(p-1)!}, \dots, \tau_0, 1 \right]^T \in \mathbf{C}^p$ ,*

$$\tilde{A} = \begin{bmatrix} A & B \\ 0 & K \end{bmatrix} \in \mathbf{C}^{(N+p) \times (N+p)}, \quad K = \begin{bmatrix} 0 & I_{p-1} \\ 0 & 0 \end{bmatrix} \in \mathbf{C}^{p \times p}, \quad v = \begin{bmatrix} u_0 \\ s \end{bmatrix} \in \mathbf{C}^{N+p},$$

then, for  $h \in \mathbf{C}$ , the first  $N$  elements of the matrix vector product  $\varphi_0(h\tilde{A})v$  is

$$u(\tau_0 + h) = \varphi_0(hA)u_0 + \sum_{j=0}^{p-1} \sum_{\ell=0}^j \frac{\tau_0^{j-\ell}}{(j-\ell)!} h^{\ell+1} \varphi_{\ell+1}(hA)b_{j+1}.$$

This lemma was proved in a slightly less general form recently by Al-Mohy and Higham (2009).

This reduces the cost by a factor  $p$ , but it is still cubic in  $N$ . We can achieve a far greater reduction by realizing that one need not compute the matrix function itself, but only its action on a vector. That is, one need only compute  $\varphi_0(h\tilde{A})v$  directly, instead of first computing  $\varphi_0(h\tilde{A})$  and then multiplying it by  $v$ . Giles and Glasserman (2006) have proposed a similar matrix-free approach which improves the time needed to compute Greeks in the LIBOR market model using the adjoint approach.

One outstanding question we must address before continuing is why we compute these matrix functions rather than solving linear systems of the form  $(I - hA)x = v$ , which are the building blocks of the implicit solvers such as the Crank–Nicolson and ADI methods. The answer lies in the paper by Hochbruck and Lubich (1997), where they show that the convergence of the Krylov subspace approximations to exponential type expressions is faster than that of corresponding Krylov methods for the solution of linear equations.

The algorithm proposed in this paper follows closely the work of Sidje (1998), who considered the cases when  $b(\tau) = 0$  and  $b(\tau) = b_0$ , and Niesen and Wright (2011), who considered the general case where  $b(\tau)$  is a polynomial. However, the method described here differs from this earlier work in that here we first combine the  $\varphi$  functions using Lemma 4.2 and then project on the Krylov subspace, while Sidje (1998) and Niesen and Wright (2011) first project on the Krylov subspace and then appeal to Lemma 4.2. Al-Mohy and Higham (2010) found that the latter method is very sensitive to round-off errors, especially as  $p$  increases; the method described here suffers significantly less from round-off errors.

We now give further details of our Krylov subspace approach to the computation of  $\varphi_0(h\tilde{A})v$ . The Krylov subspace is the subspace of  $\mathbf{R}^{N+p}$  with basis  $\{v, \tilde{A}v, \tilde{A}^2v, \dots, \tilde{A}^{m-1}v\}$ ; here  $m \in \mathbf{N}$  denotes the dimension of the Krylov subspace. However, this basis is unstable, so we apply the Gram–Schmidt algorithm to find an orthonormal basis  $\{v_1, \dots, v_m\}$ . Let  $V_m$  denote the  $(N+p) \times m$  matrix with these orthonormal vectors as

columns. Then  $V_m$  is the orthogonal projection of  $\mathbf{R}^{N+p}$  onto the Krylov subspace. Now define the  $m$ -by- $m$  matrix

$$H_m = V_m^T \tilde{A} V_m,$$

which can be thought of as the projection of the matrix  $\tilde{A}$  to the Krylov subspace, expressed in the basis  $\{v_1, \dots, v_m\}$ . The matrices  $V_m$  and  $H_m$  are computed using the Arnoldi iteration (Algorithm 1). The Lanczos iteration is not applicable here because the extended matrix  $\tilde{A}$  is not symmetric even when the matrix  $A$  is symmetric.

---

**Algorithm 1** The Arnoldi iteration

---

```

 $h_{1,1} = \|v\|$ ;  $v_1 = v/h_{1,1}$ 
for  $j = 1, \dots, m - 1$  do
   $w = Av_j$ 
  for  $i = 1, \dots, j$  do
     $h_{i,j} = v_i^T w$ ;  $w = w - h_{i,j}v_i$ 
  end for
   $h_{j+1,j} = \|w\|$ ;  $v_{j+1} = w/h_{j+1,j}$ 
end for

```

---

Now, consider the  $(N + p)$ -by- $(N + p)$  matrix  $V_m H_m V_m^T$ . This is the projection of the action of  $\tilde{A}$  on the Krylov subspace in the standard basis of  $\mathbf{R}^{N+p}$ . The idea behind Krylov subspace methods is to use the matrix  $V_m H_m V_m^T$  instead of  $\tilde{A}$ . Thus, we approximate  $\varphi_0(h\tilde{A})v$  by  $\varphi_0(hV_m H_m V_m^T)v$ . Using the identities  $V_m^T V_m = I_m$  and  $V_m V_m^T v = v$ , it follows that

$$\varphi_0(h\tilde{A})v \approx \varphi_0(hV_m H_m V_m^T)v = V_m \varphi_0(hH_m) V_m^T v = \beta V_m \varphi_0(hH_m) e_1, \quad (9)$$

where  $\beta = \|v\|$  and  $e_1$  is the first vector in the standard basis. The advantage of this formulation is that the matrix  $H_m$  has size  $m \times m$ . Typically,  $m$  is small and generally, much smaller than the size of  $\tilde{A}$ . Thus, it is much cheaper to evaluate  $\varphi_0(hH_m)$  than  $\varphi_0(h\tilde{A})$ . A typical algorithm for evaluating  $\varphi_0(hH_m)$  is the scaling and squaring algorithm implemented in MATLAB; see Higham (2005) for the details of the algorithm or a recent improvement discussed by Al-Mohy and Higham (2009).

This concludes the description of the algorithm, except for one detail: how to choose the dimension  $m$  and the step size  $h$ ? The ODE solver will only be efficient if these are chosen properly. Sidje (1998) choose  $m = 30$  and then varies  $h$  in order to ensure that the error, as estimated using a relation proved by (Saad, 1992, Thm. 5.1), stays under a user-defined tolerance. Niesen and Wright (2011) varied both  $m$  and  $h$  and use the additional freedom to keep the computational cost as low as possible. At every step of the algorithm, two options are considered: either to change  $m$  in order to ensure that the error estimate stays under the user-defined tolerance, or to change  $h$ . The option with the lowest computational cost is then picked. The algorithm proposed here follows that strategy.

Algorithm 2 outlines the overall algorithm for solving the ODE (7). For further details on the error estimate and the choice of  $h$  and  $m$ , the reader is referred to Niesen and Wright (2011).

The algorithm described above is implemented in a MATLAB function called `expmvp`. A call of the `expmvp` function has the form

```
[u, stats] = expmvp(T, A, b, tol, m)
```

There are three mandatory input arguments and one mandatory output argument; the other arguments are optional. The first input argument is `T`, the final time,  $T$ , for the differential equation (7). Generally, this is chosen to be  $T = 1$  as  $A$  can be rescaled to take into account different final times. The second argument is the  $n$ -by- $n$  matrix argument. Finally, `b` is an  $n$ -by- $(p + 1)$  matrix with columns representing the vectors  $b_0, b_1, \dots, b_p$  to be multiplied by the corresponding  $\varphi$ -functions. There is one mandatory output argument: `u`, the numerical approximation to the solution (8) at the final time  $T$ . There are two optional input arguments. The first one is `tol`, which is a measure of how accurate the numerical approximation is. The default tolerance is  $10^{-7}$ . The final input argument is `m`, the initial choice for the dimension of the Krylov subspace. The initial

---

**Algorithm 2** Algorithm for integrating the ODE (7) from time  $\tau = \tau_0$  to  $\tau = \tau_{\text{end}}$ .

---

```

 $\tau = t_0; k = 0$ 
Choose initial values for  $h$  and  $m$ 
Compute  $\tilde{A}$  as in Lemma 4.2
repeat
  repeat
    Compute  $H_m$  and  $V_m$  using Algorithm 1
    Compute approximation to  $\varphi_0(h\tilde{A})u_k$  given by (9)
    Compute error estimate using (Saad, 1992, Thm. 5.1)
    Choose whether it is better to change  $m$  or  $h$ 
    Compute new value for  $m$  or  $h$ 
  until error estimate is small enough
   $u_{k+1} =$  computed approximation to  $\varphi_0(h\tilde{A})y_k$ 
   $\tau = \tau + h; k = k + 1$ 
until  $\tau = \tau_{\text{end}}$ 
return  $y_k$ 

```

---

choice is  $m = 10$  by default. There is also one optional output argument: `stats`, for providing the user with various statistics of the computation. It is a vector with four entries: `stats(1)` is the number of steps needed to complete the integration, `stats(2)` is the number of rejected steps, `stats(3)` is the number of matrix-vector products, and `stats(4)` is the number of matrix exponentials computed.

## 5 European Options

Consider three different stocks and let their value at time  $t$  be  $s_i(t)$ , for  $i = 1, \dots, 3$ . We assume that the underlying stock prices, in the risk-neutral measure, satisfy the log-normal diffusion processes

$$\frac{ds_i(t)}{s_i(t)} = rdt + \sigma_i dW_i(t), \quad i = 1, \dots, 3,$$

where  $W_i(t)$ , for  $i = 1, \dots, 3$  are correlated Brownian motions satisfying  $dW_i(t)dW_j(t) = \rho_{ij}dt$ , with  $-1 \leq \rho_{ij} \leq 1$  and  $\rho_{ii} = 1$ . The riskless interest rate  $r$  is constant as are the volatilities  $\sigma_i$  of the stochastic differential equations. The PDE that governs the price of an option, which derives its value from the three assets  $u = u(s_1, s_2, s_3, \tau)$ , where at each time  $\tau = T - t$ , with option expiry  $T$  is

$$\frac{\partial u}{\partial \tau} = \frac{1}{2} \sum_{i,j=1}^3 \rho_{ij} \sigma_i \sigma_j s_i s_j \frac{\partial^2 u}{\partial s_i \partial s_j} + r \sum_{i=1}^3 s_i \frac{\partial u}{\partial s_i} - ru. \quad (10)$$

The PDE is defined on the infinite domain  $0 \leq s_i < \infty$ , for all  $s_i = 1, \dots, 3$  and  $\tau \in (0, T]$ . For practical implementations, the domain is truncated; we follow the approach outlined in Section 2, where the computational domain is truncated to  $0 \leq s_i \leq U_{s_i}$ . We follow Andersen and Piterbarg (2010) when choosing  $U_{s_i}$ : use the transformation  $x_i(t) = \log s_i(t)$ , then the mean of  $x_i(T)$  is  $\bar{x}_i = x_i(0) + r - \frac{1}{2}\sigma^2 T$  and the standard deviation  $\bar{\sigma} = \sigma_i \sqrt{T}$ . A suitably truncated domain is then  $[\bar{x} - \alpha \sigma \sqrt{T}, \bar{x} + \alpha \sigma \sqrt{T}]$ , where  $\alpha$  is the number of standard deviations from the mean. The initial condition used is particular to the option being valued. Once the pricing PDE in Equation (10) has been discretized, the system of ODEs is of the form in Equation (6), since the coefficients are time independent. We consider two rather standard exotic option contracts, whose payoff is the relevant initial condition. The first is a call on minimum rainbow option, which has a payoff

$$u(s_1, s_2, s_3, 0) = \max(\min(s_1, s_2, s_3) - K, 0), \quad (11)$$

where  $K$  is the options strike price. Closed-form solutions of rainbow put and call options are known and can be found in Johnson (1987). The second is a basket call option, which is typically a call on a weighted

sum of stocks in this case three stocks, the payoff is

$$u(s_1, s_2, s_3, 0) = \max \left( \sum_{i=1}^3 w_i s_i - K, 0 \right), \quad (12)$$

where  $w_i$  for  $i = 1, \dots, 3$  is the weight placed on each of the three stocks. The rainbow is considered to be more computationally challenging as the topology of the payoff is less regular. We chose the parameters given in Dang et al. (2010a). The strike price is  $K = 100$  and the initial spot prices are equal to the strike:  $s_1(0) = s_2(0) = s_3(0) = K$ . The remaining problem parameters are  $r = 0.04$ ,  $\sigma_1 = 0.3$ ,  $\sigma_2 = 0.35$ ,  $\sigma_3 = 0.4$ , and  $\rho_{12} = \rho_{13} = \rho_{23} = 0.5$ . The option expiry is set to  $T = 1$ . The weights in the basket option are  $w_1 = w_2 = w_3 = \frac{1}{3}$ . The reference solution for the call-on-minimum rainbow is 4.4450 and for the basket option is 13.2449. We use the domain  $\Omega = [0, 300] \times [0, 300] \times [0, 300]$ , as was proposed by Leentvaar and Oosterlee (2008). We point out that this domain contains  $\alpha = 2.85$  standard deviations of  $x(T)$  when  $\sigma = 0.4$ ; see the discussion under Equation (10).

In our first experiment in this section, we compared the following five methods on this problem:

- the Crank–Nicholson method (represented in the plots by a red line);
- an ADI method due to Douglas and Rachford (1956) with  $\theta = \frac{1}{2}$  (green line);
- an ADI method due to Hundsdorfer and Verwer (2003), also with  $\theta = \frac{1}{2}$  (blue line);
- the `expmvp` function (black line);
- the single-precision exponential integrator of Al-Mohy and Higham (2010) (cyan line).

It is worth noting a couple of important points with regards to discretization errors in the numerical solution of PDEs before we report our results. Assume that we have discretized the PDE, even if we solve the system of ODEs extremely accurately we still have a spatial error due to the finite difference discretization. Balancing the errors incurred in the spatial and time discretizations is very important for an efficient numerical approximation. There is very little point in over solving one part of the problem; for example, using a very fine spatial discretization followed by only a few time-steps to approximate the ODEs (given the number of ODEs is very large) is pointless. Below, we shall state clearly whether the errors are in approximations to the ODEs or in the PDE. We discretize the truncated domain of the PDE using 31 grid points in each direction, resulting in a system of 29,791 ODEs, and then repeat the experiments with 61 grid points in each direction (226,981 ODEs). We plot the error in the ODEs for the five methods in Figure 1.

The first three methods all use a fixed step size. We divide the time interval in  $2^n$  equal steps, for  $n = 4, 5, \dots, 10$ . The methods require the use of column reordering and row scaling in the LU decomposition as implemented in MATLAB. A call to this function takes the form `[L,U,P,Q,R] = lu(X)`. Using pivoting significantly improves the computational efficiency of the Crank–Nicolson scheme. Dropping the output arguments `Q` and `R` slightly improves the performance of the ADI methods. We suggest the paper in 't Hout and Foulon (2010) for further information on the most commonly used ADI methods in an options-pricing setting. The Krylov method as discussed in this paper is an adaptive method, so we specify the error tolerances instead of the number of stepsizes. We chose five equally-spaced points in log-space between  $10^{-1}$  and  $10^{-7}$  as the error that we would be happy to accept in the solution of the ODEs. Finally, the Al-Mohy–Higham method uses  $2^n$  steps, with  $n = 0, 1, \dots, 8$ .

We computed the error, in the ODEs, for each of our five numerical approximations by comparing them with an “exact” solution on the smaller domain consisting of all grid points in the  $9 \times 9 \times 9$  cube that has the spot price in the centre. The “exact” solution, of the ODEs, was computed using two independent methods: the double-precision version of Al-Mohy–Higham method and the `phipm` method from Niesen and Wright (2011) with accuracy requirements set to  $2^{-53}$ . Both methods were compared on the  $9 \times 9 \times 9$  cube to ensure that the solution at each point on the cube was accurate to within  $100 \cdot 2^{-53}$ . Given that we are interested in accuracy levels of around  $10^{-4}$  these solutions can be considered as exact solutions of the ODEs on the  $9 \times 9 \times 9$  cube.<sup>2</sup> Both the ADI methods exhibit the correct order of convergence, first-order for the Douglas method and

<sup>2</sup>All experiments use a MacBookPro with 2.66 GHz Intel Core 2 Duo processor and 4GB 1067 MHz DDR3 memory. We used

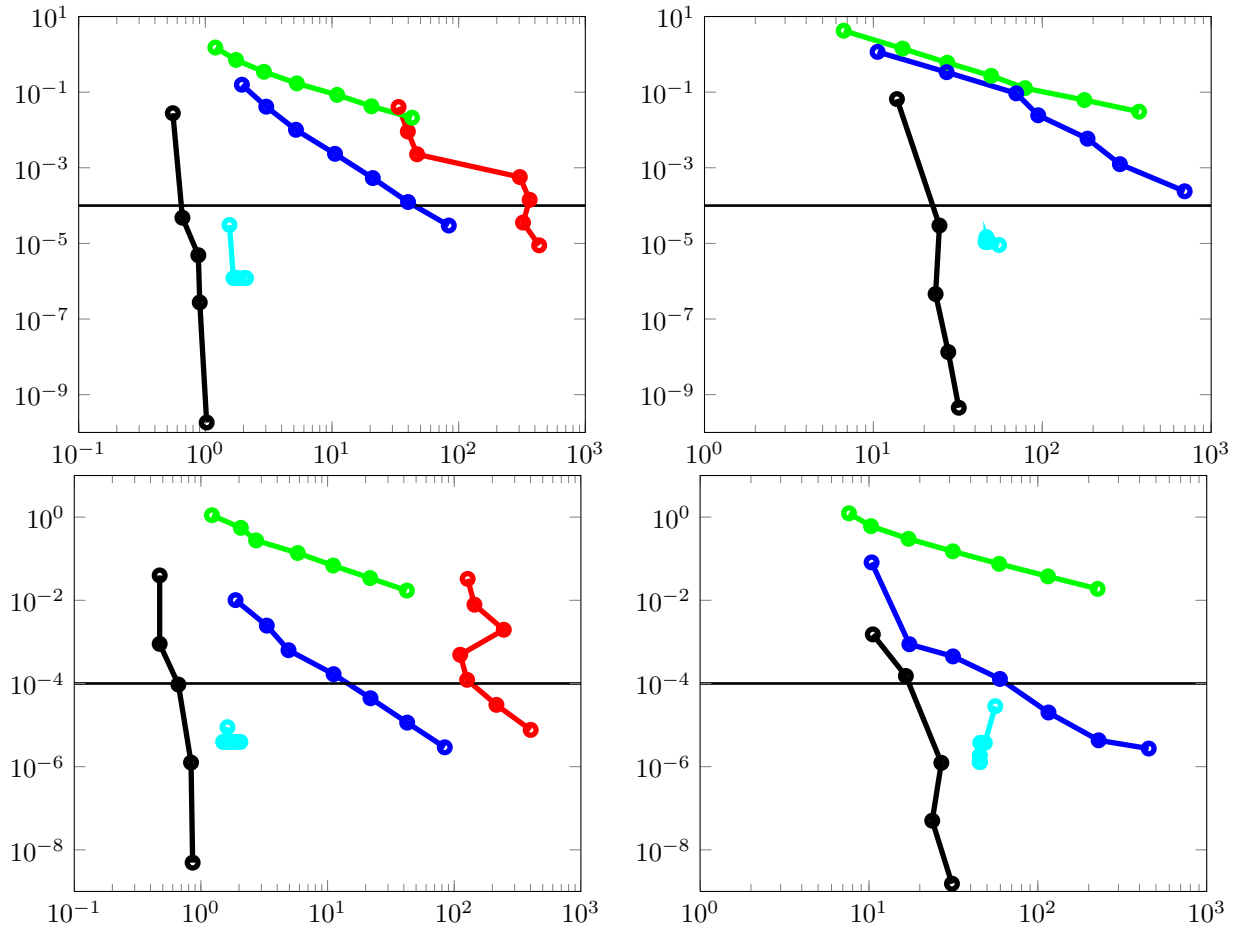


Figure 1: Logarithmic plots of the error, in the ODEs, (on the vertical axis) versus CPU time (in seconds, on the horizontal axis) for five numerical methods. The domain  $\Omega = [0, 300] \times [0, 300] \times [0, 300]$  is discretized using 31 grid points in each direction in the left column, while 61 points are used in the right column. The top two plots show the rainbow option, equation (11). The bottom two plots show the basket option, equation (12). The error is the Euclidean norm of the absolute error in the ODEs on the  $9 \times 9 \times 9$  cube of points surrounding the point  $(100, 100, 100)$ . The black line represents the Krylov integrator, the blue line represents the Hundsdorfer–Verwer, the green line Douglas, the red line Crank–Nicolson (left plot only) and the cyan line AI-Mohy–Higham.

second-order for the Hundsdorfer–Verwer method. Because of the computational requirements, the Crank–Nicolson method was only run on the smaller grid: even on the smaller grid it exhibited extremely erratic behaviour and is significantly more computational expensive than the Hundsdorfer–Verwer method. The AI-Mohy–Higham scheme computes approximations to single-precision, so all approximations are clustered around  $2^{-23}$  as expected. The Krylov integrator exhibits an extremely higher order of convergence given that it directly approximates the exact solution of the ODEs. In most financial applications, accuracy to  $10^{-4}$  or one basis point is desired. In this case the Krylov method is by far the most efficient. For the rainbow option, with 61 grid points in each spatial direction the difference between the Hundsdorfer–Verwer method and the Krylov solver at an accuracy level  $10^{-4}$  in the ODEs, is almost two orders of magnitude. This is an extremely favourable result for the Krylov exponential integrator.

MATLAB Version 7.11 (R2010b) for all experiments, which computes the matrix exponential as described in Higham (2005). We have noticed that the results described in this section depend on the specifications of the computer but the overall nature of the numerical experiments remains the same.

In MATLAB the standard stiff solver is `ode15s`. We did not include this integrator in our comparisons as it is significantly less efficient than any of the other solvers that we have used. We also tried to solve the linear systems in the Crank–Nicolson and ADI methods with the built-in Krylov solvers `gmres` and `bicgstabl`. For the ADI methods this proved to be less efficient than solving the full system directly, whereas for the Crank–Nicolson method it was generally more efficient except that the convergence behaviour was extremely erratic. This situation may be improved by computing suitable preconditioners or initial approximations. It takes Hundsdorfer–Verwer approximately 700 seconds to solve the ODEs for the rainbow option to an accuracy of one basis point on the fine, 61 points in each direction, spatial discretization; for the basket options 60 seconds is needed. Traders with large books of such options need to compute prices quickly, so such fine grids are usually not recommended. For PDEs with dimension three or more it is more efficient to use sparse grids; we are currently investigating the use of sparse grids and exponential integrators as a way to efficiently solve high dimensional PDEs.

In our second experiment, we discretized the PDE in Equation (10) for pricing the rainbow option, Equation (11), and the basket option, Equation (12), with  $\{11, 21, 41, 61, 81, 101\}$  grid points in each spatial direction. This will show how the numerical solution of the ODEs converges to the true price as the spatial discretization becomes finer. The resulting system of ODEs, for each spatial discretization, are solved using the Douglas, Hundsdorfer–Verwer and Krylov exponential integrator to an accuracy level of at least one basis point, where the “exact” solution, of the ODEs, is computed, as above, using two independent methods. The spot price is then compared to the actual price of the option, 4.4450 for the rainbow and 13.2449 for the basket. This determines the effects of the spatial discretization error and provides the computational times needed to achieve the required accuracy. We used the in-built `tic` and `toc` functions in MATLAB as an estimate of computational time. As an estimate of the stiffness of the system of ODEs, for each of the six discretizations, we have computed the real part of the smallest eigenvalue, using the `eigs` function in MATLAB. The eigenvalues for increasing number of grid points are  $\{-24.21, -109.80, -467.91, -1275.68, -1862.50, -2924.16\}$ ; resulting in moderately stiff systems of ODEs. In Table 1 we present results for the rainbow option and in Table 2 we present results for the basket option.

Table 1: Discretize the PDE, Equation (10), for pricing the rainbow option, Equation (11), with  $\{11, 21, 41, 61, 81, 101\}$  grid points in each spatial direction. Determine the number of steps required for the Douglas and Hundsdorfer–Verwer method and the tolerance for the Krylov method to solve the resulting system of ODEs to an accuracy level of at least  $10^{-4}$ . Then compare the spot price with the exact solution 13.2449 to compute the error. The CPU time measured, using the `tic` and `toc` MATLAB functions, to solve the system of ODEs to  $10^{-4}$  is recorded.

points	Douglas			Hundsdorfer–Verwer			Krylov exponential		
	steps	error	time	steps	error	time	tol	error	time
11	10191	$1.3 \times 10^{-1}$	6.6	104	$1.3 \times 10^{-1}$	0.2	$2.8 \times 10^{-2}$	$1.3 \times 10^{-1}$	0.1
21	11325	$2.7 \times 10^{-2}$	60.5	113	$2.7 \times 10^{-2}$	1.4	$5.0 \times 10^{-2}$	$2.7 \times 10^{-2}$	0.1
41	12282	$4.9 \times 10^{-3}$	771.6	144	$4.9 \times 10^{-3}$	18.8	$2.9 \times 10^{-1}$	$5.0 \times 10^{-3}$	1.8
61	13756	$2.5 \times 10^{-5}$	2980.6	252	$1.2 \times 10^{-4}$	112.9	$3.1 \times 10^{-1}$	$2.4 \times 10^{-4}$	15.8
81	9852	$2.8 \times 10^{-4}$	5381.5	261	$2.5 \times 10^{-4}$	290.0	$2.3 \times 10^{-3}$	$3.8 \times 10^{-4}$	44.8
101	11133	$2.7 \times 10^{-3}$	18534.5	359	$2.7 \times 10^{-3}$	1204.8	$2.3 \times 10^{-3}$	$3.9 \times 10^{-6}$	137.0

There are several interesting results to take from Tables 1 and 2. The most striking finding is the differences in computational times recorded for each of the three methods. The difference between the two ADI schemes is significant and leads to the natural conclusion that if an ADI is used it should be the Hundsdorfer–Verwer method and not Douglas. The Krylov method is more efficient than the best ADI for all choices of discretizations.

The results for the basket option and those for the rainbow option are surprisingly different. For the basket option, the Hundsdorfer–Verwer method and the Krylov method show comparable performance, whereas for the rainbow option the Krylov method is significantly more efficient. The matrix  $A$  for both the rainbow and the basket option are the same; the difference is the payoff, which leads to different initial conditions for the ODEs. We have noticed in various experiments that as the option payoff becomes more complicated the Krylov methods perform significantly better than ADI methods. Investigating the efficiency differences

Table 2: Discretize the PDE, Equation (10), for pricing the basket option, Equation (12), with  $\{11, 21, 41, 61, 81, 101\}$  grid points in each spatial direction. Determine the number of steps required for the Douglas and Hundsdorfer–Verwer method and the tolerance for the Krylov method to solve the resulting system of ODEs to an accuracy level of at least  $10^{-4}$ . Then compare the spot price with the exact solution 4.4450 to compute the error. The CPU time measured, using the `tic` and `toc` MATLAB functions, to solve the system of ODEs to  $10^{-4}$  is recorded.

points	Douglas			Hundsdorfer–Verwer			Krylov exponential		
	steps	error	time	steps	error	time	tol	error	time
11	7543	$7.4 \times 10^{-2}$	4.8	36	$7.4 \times 10^{-2}$	0.06	$2.8 \times 10^{-3}$	$7.4 \times 10^{-2}$	0.3
21	7452	$1.2 \times 10^{-2}$	46.8	35	$1.2 \times 10^{-2}$	0.5	$5.0 \times 10^{-2}$	$1.2 \times 10^{-2}$	0.1
41	7408	$3.9 \times 10^{-3}$	465.3	39	$4.0 \times 10^{-3}$	5.8	$8.5 \times 10^{-2}$	$4.0 \times 10^{-3}$	2.1
61	7419	$1.5 \times 10^{-3}$	1609.1	43	$1.6 \times 10^{-3}$	21.8	$8.2 \times 10^{-2}$	$1.6 \times 10^{-3}$	15.1
81	7442	$9.8 \times 10^{-4}$	4061.3	36	$7.8 \times 10^{-4}$	46.7	$7.5 \times 10^{-2}$	$7.9 \times 10^{-4}$	44.9
101	7459	$4.6 \times 10^{-4}$	12535.6	48	$5.3 \times 10^{-4}$	184.2	$1.2 \times 10^{-2}$	$5.2 \times 10^{-4}$	176.4

between the Krylov exponential integrators and the ADI methods for various PDEs and associated option contracts is a topic for further work.

The second finding is that the number of steps required to reach an accuracy level of  $10^{-4}$ , in the ODEs, remains almost constant for all discretizations, but varies significantly between methods and options. The Douglas method requires around 10,000 steps to reach an accuracy level of  $10^{-4}$ , in the ODEs, for the rainbow option, whereas the Hundsdorfer–Verwer method requires around 35 steps to value the basket option to the same level of accuracy. Given that the ADI methods are not adaptive, determining the number of steps required and the number of grid points in each direction must be done using trial and error. This is a major disadvantage of the ADI and for that matter the Crank–Nicolson methods. Once the number of grid points has been chosen the Krylov method will solve the ODEs at least to the required accuracy level without the need for trial and error. The Krylov method often over-solves the problem resulting in more accurate approximations than required: the tolerance needed to achieve an accuracy level of  $10^{-4}$  in the ODEs is larger than  $10^{-4}$ . Determining more reliable adaptivity in the Krylov methods is an active area of research.

## 6 European Options with Heston Stochastic Volatility

Let the underlying stock price  $s(t)$  and its variance  $v(t)$  satisfy the following SDEs

$$\begin{aligned} ds(t) &= (r - q)s(t)dt + \sqrt{v(t)}s(t)dW_1(t), \\ dv(t) &= \kappa(\nu - v(t))dt + \sigma\sqrt{v(t)}dW_2(t), \end{aligned}$$

where  $W_1(t)$  and  $W_2(t)$  are correlated Brownian motions satisfying  $dW_1(t)dW_2(t) = \rho dt$ . The variance process is a mean reverting process with rate  $\kappa$  and level  $\nu$ . The risk neutral interest rate is  $r$  and the dividends paid are  $q$ . The PDE, derived by Heston (1993), which governs the price of an option is  $u = u(s, v, \tau)$ , where at each time  $\tau = T - t$ , with option expiry  $T$  is

$$\frac{\partial u}{\partial \tau} = \frac{1}{2}vs^2\frac{\partial^2 u}{\partial s^2} + \rho\lambda vs\frac{\partial^2 u}{\partial s\partial v} + \frac{1}{2}\lambda^2 v\frac{\partial^2 u}{\partial v^2} + (r - q)s\frac{\partial u}{\partial s} + \kappa(\eta - v)\frac{\partial u}{\partial v} - ru. \quad (13)$$

The PDE is defined on the infinite domain  $s > 0$  and  $v \geq 0$ , while  $\tau$  ranges from 0 to  $T$ . Here,  $\rho \in [-1, 1]$ . For practical implementations the domain is truncated, we follow the approach outlined in Section 2, where the computational domain is  $\tilde{\Omega} = [0, U_s] \times [0, U_v]$ . The initial condition used is particular to the option being valued in our experiments we choose the European call option

$$u(s, v, 0) = \max(s - K_s, 0),$$

where  $K_s \geq 0$  denotes the strike price. We now need to specify appropriate boundary conditions for  $0 < \tau \leq T$ . The most common choice of boundary conditions, following Heston (1993), in 't Hout and Foulon (2010) and

Winkler et al. (2001), are

$$\begin{aligned} u(0, v, \tau) &= 0, \\ u(s, U_v, \tau) &= se^{-q\tau}, \\ \frac{\partial u(s, v, \tau)}{\partial s} &= e^{-q\tau}, \\ \frac{\partial u(s, 0, \tau)}{\partial \tau} &= (r - q)s \frac{\partial u(s, 0, \tau)}{\partial s} + \kappa\eta \frac{\partial u(s, 0, \tau)}{\partial v} - ru(s, 0, \tau). \end{aligned}$$

The last equation assumes that the boundary when  $v = 0$  is a free boundary. Therefore, it is included in the solution grid. Note that the boundary and initial conditions are inconsistent or non-matching, that is the boundary and initial conditions at  $\tau = 0$  do not agree. Note also that these boundary conditions are different than those described in Section 2 so the finite difference discretization outlined in Section 3 must also be modified. We set-up the grid exactly as is outlined by in 't Hout and Foulon (2010). The two main modifications to Section 3 are; second-order upwind scheme is used to discretize  $\frac{\partial u}{\partial v}$  when  $v > 1$ ; the boundary conditions result in ODEs of the form given in Equation (6) with  $b(\tau) = b_0 + b_1 e^{-q\tau}$ . To use the Krylov method we must approximate  $b(\tau)$  by a polynomial, we approximate  $e^{-q\tau}$  by a fifth degree Taylor-series expansion, about  $\tau = 0$ . In our experiments, we only approximate  $b(\tau)$  by the Taylor-series for the Krylov method, all other methods use the exact form of  $b(\tau)$ , this also allows us to gauge the effects of this approximation.

We also follow in 't Hout and Foulon (2010) in the setup of the experiments. Our computational domain is  $\Omega = [0, 8K] \times [0, 5]$ , which is discretized as explained in Section 3 with  $K_s = 100$ ,  $K_v = 0$ ,  $c_s = \frac{K}{5}$  and  $c_v = 1100$ . The domain where we compare solutions is  $\tilde{\Omega} = [\frac{1}{2}K, \frac{3}{2}K] \times [0, 1]$ . The parameters for the PDE in Equation (13) are given in Table 3.

Table 3: Parameters for European call options using Heston's stochastic volatility model.

	Case 1	Case 2	Case 3	Case 4
$\kappa$	1.5	3	0.6067	2.5
$\nu$	0.04	0.12	0.0707	0.06
$\sigma$	0.3	0.04	0.2928	0.5
$\rho$	-0.9	0.6	-0.7571	-0.1
$r$	0.025	0.01	0.03	0.0507
$q$	0	0.04	0	0.0469
$T$	1	1	3	0.25

Equation (13) is a two-dimensional PDE, so it is easier to solve than the three-dimensional equation considered in Section 5. Thus, we can use a finer spatial discretization. We chose 101 grid points in the  $s$ -direction and 50 grid points in the  $v$ -direction, resulting in 4,900 ODEs. We solved these using the same five methods as in Section 5, but also included the standard inbuilt MATLAB stiff solver `ode15s`, as the system of equations is of moderate size. The `ode15s` solver is a variable-stepsize, variable-order Backwards Differentiation Formulae (BDF) method, with maximum order five; see Shampine and Reichelt (1997) for more implementation details. We ran the Crank–Nicolson, Douglas, and Hundsdorfer–Verwer methods with  $2^7, 2^8, \dots, 2^{15}$  steps of equal length. The Krylov and `ode15s` solvers are both adaptive methods, so we specified the error tolerances instead of the number of steps. We choose six equally-spaced points in log-space between  $10^{-3}$  and  $10^{-8}$ . For `ode15s` the relative and absolute errors are set equal. Finally, for the Al-Mohy–Higham method, with single-precision, we chose the number of steps to be  $2^n$ , for  $n = 0, 1, \dots, 8$ . The “exact” solution, of the ODEs, that we used to compare the numerical solutions on the  $\tilde{\Omega}$  domain was computed with two different methods, first the Al-Mohy–Higham method with double precision and the Hundsdorfer–Verwer method with very small stepsize, so that the solutions agreed to a level of precision of  $10^{-12}$ . The smallest eigenvalue for each of the four cases are  $\{-50769.20, -50784.50, -50758.69, -50782.21\}$ : the system of ODEs are stiff. The results are presented in Figure 2.

There are several interesting observations that can be made about these experiments. In Cases 1 and 3, where there are no dividends ( $q = 0$ ), the Crank–Nicolson method performs slightly more efficiently than

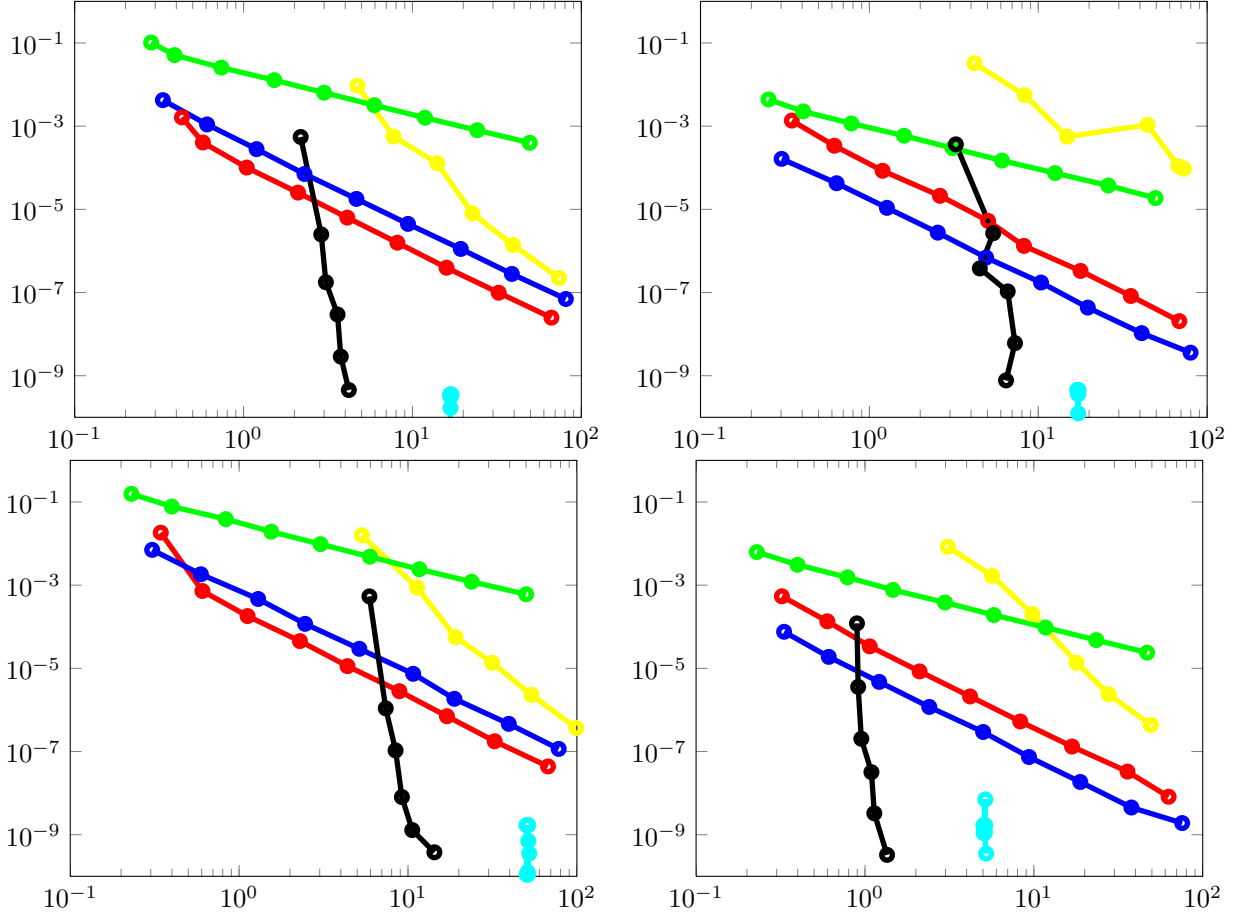


Figure 2: Plots in a log-log scale of the Euclidean norm of the absolute error (on the domain  $\tilde{\Omega} = [\frac{1}{2}K, \frac{3}{2}K] \times [0, 1]$ ) against CPU time for the system of ODEs from the discretized Heston PDE (13). The plots correspond to: Case 1 (top left); Case 2 (top right); Case 3 (bottom left); Case 4 (bottom right). The integrators used are: Douglas (green); Hundsdorfer-Verwer (blue); Crank-Nicolson (red); `ode15s` (yellow); Al-Mohy-Higham (cyan); Krylov (black).

the Hundsdorfer-Verwer method. On the other hand, when  $q \neq 0$  (Cases 2 and 4), the Hundsdorfer-Verwer method slightly outperforms Crank-Nicolson. The Douglas method converges with order one and the Hundsdorfer-Verwer and Crank-Nicolson methods converge with order two as expected. No significant loss of accuracy due to boundary effects can be noticed for the Crank-Nicolson method using the presented number of integration steps, though a loss of accuracy can be identified when fewer steps are used in the time integration. For tolerances stricter than approximately  $10^{-5}$ , the Krylov solver is the most efficient; significantly more efficient for very strict tolerances. We also point out that approximating  $b(\tau)$  by an order-five Taylor-series does not have any effect on the convergence of the Krylov method. The `ode15s` method does not perform well on any of the examples and even fails to reach the required level of accuracy. It does achieve in three of the four cases the order-five behaviour that is expected. The Crank-Nicolson or Hundsdorfer-Verwer methods may be most accurate for accuracy levels of around  $10^{-4}$ , in the ODEs, but they lack adaptivity. Thus, trial and error is needed to determine whether the required accuracy level has been met. It is for this reason that it is surprising that little research effort has been made to equip ADI methods with error estimates or develop higher order ADI methods that take advantage of the special structure of the ODEs. As in Section 5, it is likely that as the option contracts become more exotic the Krylov methods will respond more favourably than the ADI methods.

## 7 Power Reverse Dual Currency Swaps

Power reverse dual currency (PRDC) swaps, particularly popular with Japanese investors, are the most highly traded cross-currency exotic interest rate derivatives. The large interest in these products, makes accurate pricing and sensitivities essential for risk management purposes. PRDC swaps are long dated options, 30 to 40 year contracts being typical, where a coupon on the exchange rate is exchanged for floating LIBOR payments, at set times during the contract. Often the contract offers the payer of the coupon, generally the bank, the right to cancel the contract at a set of the payment dates. Such contracts are known as cancellable Bermudan contracts. For further information on PRDC swaps see the articles Sippel and Ohkoshi (2002); Risk Cover Story (2003). Efficient pricing of PRDC swaps using Monte Carlo simulations in a cross-currency LIBOR market setting has recently been reported by Beveridge et al. (2010). In this section, we use the PDE approach following closely the work of Dang et al. (2010b).

Bermudan contracts are typical in interest rates books and can be viewed as a discretization of American options. By this we mean that there are a set of pre-specified tenor dates  $T_0, \dots, T_{N+1}$ , where the option can be exercised by holder of the optionality. The cashflow at time  $T_n$ , for  $n = 1, \dots, N$ , of the PRDC swaps, from the point of view of the issuer of the contract (usually the bank) who is also the holder of the optionality, has the value

$$u(T_n) = N_n \tau_n (L_r(T_{n-1}, T_n) - C(T_n)). \quad (14)$$

The contract notional is  $N_n$  and  $\tau_n = T_n - T_{n-1}$  are the tenors. The simply compounded domestic  $L_r(T_{n-1}, T_n)$  and foreign  $L_f(T_{n-1}, T_n)$  LIBOR rates, defined over the period  $[T_{n-1}, T_n]$  are computed by

$$L_r(T_{n-1}, T_n) = \frac{1}{\tau_n} \left( \frac{1}{P_r(T_{n-1}, T_n)} - 1 \right), \quad L_f(T_{n-1}, T_n) = \frac{1}{\tau_n} \left( \frac{1}{P_f(T_{n-1}, T_n)} - 1 \right),$$

where  $P_r(t, T)$  and  $P_f(t, T)$  are the time- $t$  value of the domestic and foreign zero coupon bonds, which pay 1 in their corresponding currency at time  $T$ . Given the domestic and foreign short interest rates  $r(t)$  and  $f(t)$ , the zero coupon bonds are given by

$$P_r(t, T) = \exp \left( - \int_t^T r(s) ds \right), \quad P_f(t, T) = \exp \left( - \int_t^T f(s) ds \right).$$

The values of these bonds can be stripped from liquid traded bonds. The coupon  $C(T_n)$ , paid to the buyer of the contract (usually the Japanese investor), can generally be represented as a call option on the exchange rate and has the form

$$C(T_n) = Q_n \max(s(T_n) - K_n, 0), \quad Q_n = \frac{c_d}{c_f} F(T_0, T_n), \quad K_n = \frac{c_f}{F(T_0, T_n)},$$

where  $N_n$  is the options notional,  $K_n$  is the strike,  $c_d$  and  $c_f$  are called the domestic and foreign coupons respectively and  $F(T_0, T_n)$  is the forward exchange rate, at time  $T_0$ , given by the expression

$$F(t, T_n) = s(t) \frac{P_f(t, T_n)}{P_r(t, T_n)}.$$

To value the PRDC swaps we follow closely Piterbarg (2005), who outlines a cross-currency framework, where the domestic and foreign short interest rates  $r(t)$  and  $f(t)$  are modelled by mean reverting stochastic processes (often called the Hull–White or extended Vasicek model) and the exchange rate  $s(t)$  is modelled as a lognormal stochastic process

$$\begin{aligned} dr(t) &= (\theta_r(t) - \mu_r(t)r(t)) dt - \sigma_r(t)dW_r(t), \\ df(t) &= (\theta_f(t) - \mu_f(t)f(t) - \rho_{fs}(t)\sigma_f(t)\gamma(t, s(t))) dt - \sigma_f(t)dW_f(t), \\ ds(t) &= (r(t) - f(t)) s(t)dt + \gamma(t, s(t))s(t)dW_s(t), \end{aligned}$$

where  $W_r(t)$ ,  $W_f(t)$  and  $W_s(t)$  are correlated Brownian motions, satisfying

$$dW_r(t)dW_f(t) = \rho_{rf}(t)dt, \quad dW_r(t)dW_s(t) = \rho_{rs}(t)dt, \quad dW_f(t)dW_s(t) = \rho_{fs}(t)dt.$$

The deterministic functions  $\theta_r(t)$  and  $\theta_f(t)$  are the corresponding mean reverting levels,  $\sigma_r(t)$  and  $\sigma_f(t)$  are the corresponding volatility functions and  $\mu_r(t)$  and  $\mu_f(t)$  the mean reversion rate. In order to calibrate this model to market data the local volatility function  $\gamma(t, s(t))$  has the following parametric form

$$\gamma(t, s(t)) = \nu(t) \left( \frac{s(t)}{F(0, t)} \right)^{\beta(t)-1}, \quad (15)$$

where the functions  $\nu(t)$  and  $\beta(t)$  are defined to be piecewise constant between the tenor dates, that is

$$\nu(t) = \sum_{n=1}^N \nu_n \mathbf{1}_{(T_{n-1}, T_n]}(t), \quad \beta(t) = \sum_{n=1}^N \beta_n \mathbf{1}_{(T_{n-1}, T_n]}(t).$$

Note that the term  $\rho_{fs}(t)\sigma_f(t)\gamma(t, s(t))$  in the stochastic process for  $f(t)$  comes from changing the measure from the foreign risk-neutral measure to the domestic risk-neutral measure, see the book (Musielka and Rutkowski, 2005, Chapter 4) for more details.

Let  $u = u(r, f, s, \tau)$  be the domestic value of the option, at time  $\tau = T_N - t \in T_N - [T_n, T_{n-1}]$ , for  $n = N, \dots, 1$ , then given an initial condition (terminal payoff) at time  $T_N - T_n$  the option price satisfies the following PDE

$$\begin{aligned} \frac{\partial u}{\partial \tau} &= (\theta_r(\tau) - \mu_r(\tau)r) \frac{\partial u}{\partial r} + (\theta_f(\tau) - \mu_f(\tau)f + \rho_{fs}\sigma_f(\tau)\gamma(\tau, s)) \frac{\partial u}{\partial f} + (r - f)s \frac{\partial u}{\partial s} \\ &+ \frac{1}{2}\sigma_r^2(\tau) \frac{\partial^2 u}{\partial r^2} + \frac{1}{2}\sigma_f^2(\tau) \frac{\partial^2 u}{\partial f^2} + \frac{1}{2}\gamma^2(\tau, s)s^2 \frac{\partial^2 u}{\partial s^2} \\ &+ \rho_{rf}\sigma_r(\tau)\sigma_f(\tau) \frac{\partial^2 u}{\partial r \partial f} + \rho_{rs}\sigma_r(\tau)\gamma(\tau, s)s \frac{\partial^2 u}{\partial r \partial s} + \rho_{fs}\sigma_f(\tau)\gamma(\tau, s)s \frac{\partial^2 u}{\partial f \partial s} - ru. \end{aligned} \quad (16)$$

The discounted option price in the domestic risk-neutral measure is a martingale and must therefore have zero drift. Differentiate the discounted option price using Itô's Lemma setting the drift term to zero yields Equation (16).

The numerical solution of Equation (16) was not considered by Piterbarg (2005, Page 3), except to say that it is “most efficiently solved by utilizing a level-splitting scheme such as an ADI scheme.” In the recent papers by Dang et al. (2010a,b) the PRDC swaps have been extensively studied, in particular PRDC with European, Bermudan and knockout features. In our experiments, we followed closely the problem specifications first outlined in Piterbarg (2005). We chose the Japanese Yen (JPY) as the domestic currency and the US dollar (USD) as the foreign currency. The initial spot domestic and foreign interest rates are  $r(T_0) = 0.02$  and  $f(T_0) = 0.05$  respectively and the initial spot FX rate is  $s(T_0) = 105.0$ . Domestic and foreign zero coupon bonds were  $P_r(T_0, T_n) = \exp(-r(T_0) T_n)$  and  $P_f(T_0, T_n) = \exp(-f(T_0) T_n)$  respectively. The volatility parameters were constant over all tenor dates, with values  $\sigma_r = 0.007$  and  $\sigma_f = 0.012$ . The mean reverting rates are also constant over all tenor dates:  $\mu_r = 0.0$  and  $\mu_f = 0.05$ . From (Brigo and Mercurio, 2006, Page 73), the time-dependent mean reverting levels were

$$\theta_r(\tau) = \sigma_r^2 \tau, \quad \theta_f(\tau) = f(T_0)\mu_f + \frac{\sigma_f^2}{2\mu_f}(1 - e^{-2\mu_f \tau}).$$

The correlation coefficients were also time independent; they are  $\rho_{rf} = 0.25$ ,  $\rho_{rs} = -0.15$  and  $\rho_{fs} = -0.15$ . Local volatility parameters were assumed to be piecewise constant, with values given in Table 4.

Table 4: Local volatility parameters  $\nu(t)$  and  $\beta(t)$  for values of  $t \in (0, 30]$ .

$t$	(0, 0.5]	(0.5, 1]	(1, 3]	(3, 5]	(5, 7]	(7, 10]	(10, 15]	(15, 20]	(20, 25]	(25, 30]
$\nu(t)$	0.0903	0.0887	0.0842	0.0899	0.1018	0.1330	0.1818	0.1673	0.1351	0.1351
$\beta(t)$	-2.0000	-1.7200	-1.1500	-0.6500	-0.5000	-0.2400	0.1000	0.3800	0.3800	0.3800

The contract has the following arrangements:

- The tenor structure is  $T_i$ , for  $i = 0, \dots, 30$ , with  $\tau_i = T_i - T_{i-1} = 1$ .
- Comparing with Table 4 this implies that there is no cash exchanged between the parties at time 0.5 despite the change in local volatility at that point.
- Pay annual PRDC coupon and receive annual domestic LIBOR payments, as specified in (14).
- The domestic and foreign coupons provide three different types of leverage: low  $c_d = 0.0225$  and  $c_f = 0.0450$ ; medium  $c_d = 0.0436$  and  $c_f = 0.0625$ ; high  $c_d = 0.0810$  and  $c_f = 0.0900$ .
- The barrier option is an up and out FX-linked barrier option with three different leverage levels: low  $B_u = 110$ ; medium  $B_u = 120$ ; high  $B_u = 130$ .

The truncated domain chosen was  $\tilde{\Omega} = [L_s, U_s] \times [L_r, U_r] \times [L_f, U_f] = [5, 305] \times [0, 0.06] \times [0, 0.15]$ . This domain does not include  $s = 0$  because the local volatility function (15) blows up there. The barriers are halfway between adjacent grid points for the grid spacing we choose in our experiments; see Dang et al. (2010b) for more details.

Given the above assumptions, it follows that the only terms in the pricing PDE, Equation (16), that are time-dependent are the coefficients of  $\frac{\partial u}{\partial r}$  and  $\frac{\partial u}{\partial f}$ , where the time dependence comes from the mean reverting levels,  $\theta_r(t)$  and  $\theta_f(t)$ . For large  $\tau$ , the time dependence in  $\theta_f(t)$  becomes negligible. In this paper, we will follow an approach used often in practice and approximate  $\theta_r(t)$  and  $\theta_f(t)$  by the value at the midpoint of the integration interval, that is  $\theta_f(T_n + \frac{1}{2})$ , for  $t \in (T_n, T_{n+1}]$  with  $n = 1, \dots, N$ . For the intervals  $(0, 0.5]$  and  $(0.5, 1]$  approximate by  $\theta_f(\frac{1}{4})$  and  $\theta_f(\frac{3}{4})$  respectively. This means that after we discretize the PDE, Equation (16), we arrive at a system of ODEs which is of the form (6) on each integration interval. Thus, we can use the approach described in Section 4.

An alternative would be to retain the time dependence of the mean reverting levels  $\theta_r(t)$  and  $\theta_f(t)$ . This leads to a system of ODEs with variable coefficients of the form (5) after discretization. Lemma 4.1 does not apply to such an equation. Instead, the solution is given by the exponential of an infinite sum of expressions involving the matrix  $A(t)$  evaluated at different times. This infinite sum is known as the *Magnus expansion*, following the pioneering work of Magnus (1954). The Magnus expansion can be combined with ideas summarized in the review article Blanes et al. (2009) to construct a numerical scheme for the pricing of PRDC swaps. We intend to investigate this approach further in the near future.

We have one further technicality to address before we can report the experiments. The PRDC cashflows, given in Equation (14), can be split into the coupon payment and the funding leg. The coupon payment is paid at the time its value is realized, whereas the funding leg is valued at time  $T_{n-1}$  but paid at time  $T_n$ , for  $n = 1, \dots, 29$ . Following the approach proposed in Dang et al. (2010b), we split the cashflows into two parts and compute the time- $T_{n-1}$  value of the funding leg as  $N_n(1 - P_r(T_{n-1}, T_n))$ . For completeness we include Algorithm 3; a similar algorithm also appears in (Dang et al., 2010b, Algorithm 1).

---

**Algorithm 3** Computing the PRDC swaps.

---

```

 $u(T_N) = 0$ 
for  $n = N, \dots, 1$  do
   $u(T_n^-) = u(T_n) - \tau_n C(T_n)$ 
  Solve PDE, Equation (16), over the interval  $[T_{n-1}, T_n]$ , with  $u(T_n^-)$  as initial condition, to obtain  $u(T_{n-1}^+)$ .
   $u(T_{n-1}) = u(T_{n-1}^+) + (1 - P_r(T_{n-1}, T_n))$ .
   $u(T_{n-1}) = u(T_{n-1}) 1(s(T_n) > B_u)$ .
end for

```

---

Given that the matrix  $A$  changes over each tenor, we find the smallest eigenvalue over all tenors. For the coarse mesh it is  $-91.83$  and for the fine mesh it is  $-255.67$ . These ODEs are only mildly stiff. Our results are summarized in Figure 3. For Douglas and Hundsdorfer–Verwer, we used 1, 2, 4, 8, 16, 32, 64 steps per integration period. For single-precision Al-Mohy–Higham, we used 1 step per integration interval. For Krylov, we used the tolerances  $10^{-1}, 10^{-1.5}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$ .

What is most surprising about these results is that the accuracy obtained for a given computational cost is the same, for a given method, whether the option is European, Bermudan or knockout with a low, medium

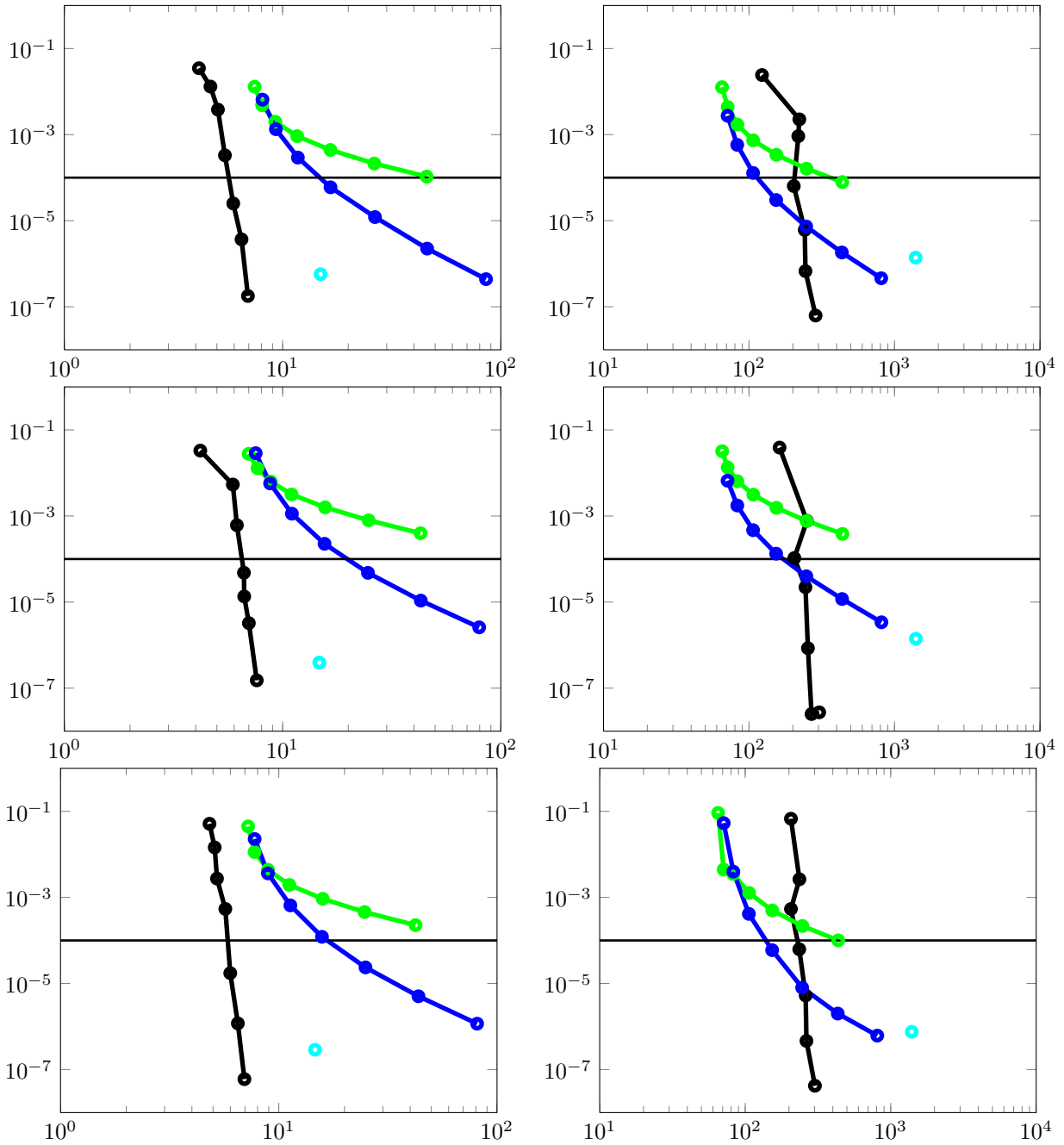


Figure 3: Plots in a log-log scale of the error, in the ODEs, of the PRDC swap against CPU time as computed by solving the PDE (16). The top row pertains to the swap with low leverage and European features, the middle row to the swap with medium leverage and Bermudan features, and the bottom row to the swap with high leverage and knockout features. The plots in the left column use 31 grid points in every direction and the plots in the right column use 61 grid points. The integrators used are: Douglas (green); Hundsdorfer–Verwer (blue); Al-Mohy–Higham (cyan); Krylov (black).

or high level of leverage. The exponential Krylov method is clearly favourable for the course mesh; for the fine mesh the Hundsdorfer–Verwer ADI method and the Krylov solver are comparable. As the mesh is refined further the problem becomes more stiff and the ADI method is favoured. However, we note that the

computational time for the  $61 \times 61 \times 61$  mesh is already unreasonable for large books; sparse grids may ease this computational bottleneck.

## 8 Conclusion and future work

Parabolic PDEs arising in financial engineering are generally linear. In this paper, we applied exponential integrators based on Krylov methods to linear parabolic PDEs arising from various financial option contracts. The Krylov exponential integrators are designed to provide very efficient numerical approximations to linear PDEs. We demonstrated using the European basket and rainbow call option contracts that the Krylov methods provide very large performance improvements over Crank–Nicolson and the current state-of-the-art ADI methods. We also tested our method on Heston’s stochastic volatility model and PRDC swaps with very favourable results. Comparing all the numerical results leads to the possible conclusion that the exponential methods perform more favourably for systems of ODEs with moderate stiffness. We intend to look into these issues more thoroughly in future work. A MATLAB and C++ implementation of the Krylov exponential integrator can be downloaded from the first authors website. All our experiments have been performed in MATLAB but given the promising results we intend to implement our method in CUDA for simulations on GPUs.

There are three main areas that we will focus our future work on. First, we are investigating the use of Magnus series methods to extend the results presented in this paper to PDEs where the coefficients are time-dependent. Early investigations in this direction are promising. Second, we are investigating for what choices of PDE and option contract that the Krylov methods the most efficient. It seems that as the option contract becomes more involved the Krylov methods perform much better relative to the ADI methods. Finally, we are investigating the use of sparse grids in combination with exponential integrators to solve higher dimension PDE problems.

## References

- AL-MOHY, A. H. AND HIGHAM, N. J. 2009. A new scaling and squaring algorithm for the matrix exponential. *SIAM J. Matrix Anal. Appl.* 31, 3, 970–989.
- AL-MOHY, A. H. AND HIGHAM, N. J. 2010. Computing the action of the matrix exponential, with an application to exponential integrators. Tech. Rep. 30, Manchester Institute for Mathematical Sciences, Manchester, England.
- ANDERSEN, L. B. AND PITERBARG, V. V. 2010. *Interest Rate Modeling*. Volume I. Atlantic Financial Press, London. Volume I: Foundations and Vanilla Models.
- BEVERIDGE, C. J., JOSHI, M. S., AND WRIGHT, W. M. 2010. Efficient pricing and Greeks in the cross-currency LIBOR market model. *SSRN eLibrary*, 1–35.
- BLANES, S., CASAS, F., OTEO, J. A., AND ROS, J. 2009. The Magnus expansion and some of its applications. *Phys. Rep.* 470, 5-6, 151–238.
- BRIGO, D. AND MERCURIO, F. 2006. *Interest rate models—theory and practice*, Second ed. Springer Finance. Springer-Verlag, Berlin. With smile, inflation and credit.
- BUTCHER, J. C. 2008. *Numerical methods for ordinary differential equations*, Second ed. John Wiley & Sons Ltd., Chichester.
- COOPER, G. J. AND SAYFY, A. 1980. Additive methods for the numerical solution of ordinary differential equations. *Math. Comp.* 35, 152, 1159–1172.
- COOPER, G. J. AND SAYFY, A. 1983. Additive Runge-Kutta methods for stiff ordinary differential equations. *Math. Comp.* 40, 161, 207–218.

- CRAIG, I. J. D. AND SNEYD, A. D. 1988. An alternating-direction implicit scheme for parabolic equations with mixed derivatives. *Comput. Math. Appl.* 16, 4, 341–350.
- CRANK, J. AND NICOLSON, P. 1947. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Proc. Cambridge Philos. Soc.* 43, 50–67.
- DANG, D. M., CHRISTARA, C. C., AND JACKSON, K. R. 2010a. GPU pricing of a cross-currency interest rate derivatives under a FX volatility skew model. *SSRN eLibrary*, 1–17.
- DANG, D. M., CHRISTARA, C. C., AND JACKSON, K. R. 2010b. Parallel implementation on GPUs of ADI finite difference methods for parabolic PDEs with applications to finance. *SSRN eLibrary*, 1–30.
- DOUGLAS, JR., J. AND RACHFORD, JR., H. H. 1956. On the numerical solution of heat conduction problems in two and three space variables. *Trans. Amer. Math. Soc.* 82, 421–439.
- DUFFY, D. J. 2006. *Finite difference methods in financial engineering*. Wiley Finance Series. John Wiley & Sons Ltd., Chichester. A partial differential equation approach.
- GILES, M. AND GLASSERMAN, P. 2006. Smoking adjoints: fast Monte Carlo Greeks. *Risk January*, 92–96.
- GLASSERMAN, P. 2004. *Monte Carlo methods in financial engineering*. Applications of Mathematics (New York), vol. 53. Springer-Verlag, New York. Stochastic Modelling and Applied Probability.
- HAIRER, E., NØRSETT, S. P., AND WANNER, G. 1993. *Solving ordinary differential equations. I*, Second ed. Springer Series in Computational Mathematics, vol. 8. Springer-Verlag, Berlin. Nonstiff problems.
- HAIRER, E. AND WANNER, G. 1996. *Solving ordinary differential equations. II*, Second ed. Springer Series in Computational Mathematics, vol. 14. Springer-Verlag, Berlin. Stiff and differential-algebraic problems.
- HESTON, S. L. 1993. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies* 6, 2, 327–43.
- HIGHAM, N. J. 2005. The scaling and squaring method for the matrix exponential revisited. *SIAM J. Matrix Anal. Appl.* 26, 4, 1179–1193 (electronic).
- HIGHAM, N. J. 2008. *Functions of matrices*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA. Theory and computation.
- HOCHBRUCK, M. AND LUBICH, C. 1997. On Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.* 34, 5, 1911–1925.
- HOCHBRUCK, M. AND OSTERMANN, A. 2010. Exponential integrators. *Acta Numer.* 19, 209–286.
- HUNDSORFER, W. AND VERWER, J. 2003. *Numerical solution of time-dependent advection-diffusion-reaction equations*. Springer Series in Computational Mathematics, vol. 33. Springer-Verlag, Berlin.
- IN 'T HOUT, K. J. AND FOULON, S. 2010. ADI finite difference schemes for option pricing in the Heston model with correlation. *Int. J. Numer. Anal. Mod.* 7, 2, 303–320.
- IN 'T HOUT, K. J. AND WELFERT, B. D. 2007. Stability of ADI schemes applied to convection-diffusion equations with mixed derivative terms. *Appl. Numer. Math.* 57, 1, 19–35.
- IN 'T HOUT, K. J. AND WELFERT, B. D. 2009. Unconditional stability of second-order ADI schemes applied to multi-dimensional diffusion equations with mixed derivative terms. *Appl. Numer. Math.* 59, 4, 677–692.
- JOHNSON, H. 1987. Options on the maximum or the minimum of several assets. *J. Finan. Quant. Anal.* 22, 3, 277–283.
- JOSHI, M. S. 2008. *The concepts and practice of mathematical finance*, Second ed. Mathematics, Finance and Risk. Cambridge University Press, Cambridge.

- KLUGE, T. 2002. Pricing derivatives in stochastic volatility models using the finite difference method. M.S. thesis, TU Chemnitz.
- LEENTVAAR, C. C. W. AND OOSTERLEE, C. W. 2008. On coordinate transformation and grid stretching for sparse grid pricing of basket options. *J. Comput. Appl. Math.* 222, 1, 193–209.
- MAGNUS, W. 1954. On the exponential solution of differential equations for a linear operator. *Comm. Pure Appl. Math.* 7, 649–673.
- MINCHEV, B. AND WRIGHT, W. M. 2005. A review of exponential integrators for semilinear problems. Tech. Rep. 2, Department of Mathematical Sciences, NTNU, Norway.
- MURUA, A. 1999. Formal series and numerical integrators. I. Systems of ODEs and symplectic integrators. *Appl. Numer. Math.* 29, 2, 221–251.
- MUSIELA, M. AND RUTKOWSKI, M. 2005. *Martingale methods in financial modelling*, Second ed. Stochastic Modelling and Applied Probability, vol. 36. Springer-Verlag, Berlin.
- NIESEN, J. AND WRIGHT, W. M. 2011. A Krylov subspace algorithm for evaluating the  $\varphi$ -functions. *ACM Trans. Math. Soft.*, To appear.
- PITERBARG, V. V. 2005. A multi-currency model with FX volatility skew. *SSRN eLibrary*, 1–25.
- RISK COVER STORY. 2003. The problem with power-reverse duals. *Risk* 16, 10.
- RUNGE, C. 1895. Ueber die numerische Auflösung von Differentialgleichungen. *Math. Ann.* 46, 2, 167–178.
- SAAD, Y. 1992. Analysis of some Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.* 29, 1, 209–228.
- SAAD, Y. 2003. *Iterative methods for sparse linear systems*, Second ed. Society for Industrial and Applied Mathematics, Philadelphia, PA.
- SHAMPINE, L. F. AND REICHEL, M. W. 1997. The MATLAB ODE suite. *SIAM J. Sci. Comput.* 18, 1, 1–22.
- SIDJE, R. 1998. EXPOKIT: Software package for computing matrix exponentials. *ACM Transactions on Mathematical Software* 24, 1, 130–156.
- SIPPEL, J. AND OHKOSHI, S. 2002. All power to PRDC notes. *Risk* 15, 11, 31–33.
- TAVELLA, D. AND RANDALL, C. 2000. *Pricing financial instruments*. John Wiley & Sons Ltd., New York.
- WILMOTT, P. 2006. *Paul Wilmott on quantitative finance*, Second ed. John Wiley & Sons Ltd., Chichester.
- WINKLER, G., APEL, T., AND WYSTUP, U. 2001. Valuation of options in Heston’s stochastic volatility model using finite element methods. In *Foreign Exchange Risk*. Risk Publications, London, 278–313.