Proceedings of the World Congress on Engineering 2007 Vol II
WCE 2007, July 2 - 4, 2007, London, U.K.

Solving financial differential equations using differentiation matrices

Robert Piché * Juho Kanniainen †

Abstract—**This paper illustrates the use of the differentiation matrix technique for solving differential equations in finance. The technique provides a compact and unified formulation for a variety of discretisation and time-stepping algorithms for solving problems in one and two dimensions. Using differentiation matrix models, we compare time-stepping algorithms for option pricing computations and present numerical results that show the advantage of the L-stable Alexander method over the Crank-Nicolson method. We also compare the efficiency of the spectral collocation and finite difference methods, and give numerical results that show spectral methods to be competitive for problems with smooth terminal conditions.**

Keywords: finite difference method, spectral collocation, Runge-Kutta method, numerical option pricing, optimal shutdown problem

1 Introduction

Matrix notation is widely used in applied mathematics because of its conciseness and flexibility. The advent of matrix-based scientific computing programming languages such as Matlab and Fortran 90 allows algorithms to be directly coded using matrix notation, making software development faster and less prone to errors. The resulting code also gains in efficiency because of the mostly transparent use of highly optimised linear algebra libraries.

In the theory of spectral (or pseudospectral) methods for numerical solution of partial differential equations, matrix notation is systematically exploited through the use of *differentiation matrices* [6]. A differentiation matrix is a matrix $\mathbf{D}^{(d)}$ such that the values at distinct nodes $\mathbf{x}$ of the dth derivative of a univariate scalar function f are approximated by

$$f^{(d)}(\mathbf{x}) \approx \mathbf{D}^{(d)} f(\mathbf{x}). \tag{1}$$

Partial derivatives of multivariate functions on rectangular domains are approximated using Kronecker tensor products of differentiation matrices. Differential equations and boundary conditions are discretised by replacing derivative operators with differentiation matrices, yielding a numerical model that has the same symbolic form as the original boundary value problem.

The matrix notation is also useful in the further numerical treatment of the discretised model. In particular, it is straightforward to derive the model's jacobian matrix, which is needed by Newton methods and by implicit time-integration schemes.

The differentiation matrix formalism can also be used to with finite difference methods [5]. Differentiation matrix models for finite difference methods and for spectral methods have the same symbolic form: the details of the discretisation, of the data structures and of the linear algebra computations are hidden in the matrix notation. This makes it easier to compare different discretisation schemes for a given problem using the same high-level code.

Generally, spectral methods are better than finite difference methods for problems that have a smooth solution, because the exponential convergence rate of spectral methods allows the problem to be solved with a much smaller number of nodes. Many option pricing problems have a nonsmooth terminal condition (payoff function), and so are better solved by finite difference methods. However, even though spectral methods could be competitive alternatives for problems with smooth terminal conditions (e.g. interest rate models) and for equilibrium problems with no terminal conditions (e.g. perpetual options, optimal shutdown), they seem to be only rarely used in financial computations.

In this paper we present four case studies of familiar financial differential equations and their solution using differentiation matrices.

- The **Black-Scholes equation** is taken as the first case in section 2 because of its familiarity and the availability of closed-form reference solutions. The differentiation matrix model is derived and is used to compare the stability and performance of the Crank-Nicolson time stepping scheme to the L-stable Runge-Kutta scheme of Alexander [1].
- In the pricing of an **American option** with implicit time stepping, applying the linear complementarity

*Institute of Mathematics, Tampere University of Technology, Tampere, Finland, robert.piche@tut.fi
†Institute of Industrial Management, Tampere University of Technology, Tampere, Finland, juho.kanniainen@tut.fi

constraint at each time step is generally a complicated procedure. For ordinary put options, however, where the early exercise region is a halfspace, the structure of the LU decomposition of the time stepping matrix can be exploited to give a fast direct solution [4]. In section 3 we show how this technique is applied to a differentiation matrix model.

- To illustrate the solution of problems in two dimensions, in section 4 we solve **Heston's model** [3] for pricing derivative securities with stochastic volatility. We compare the efficiency of the spectral collocation and finite difference methods for this problem.
- In section 5 we present the solution of a **nonlinear boundary value problem with free boundary**, namely the optimal maintenance and shutdown problem studied in [2]. The differentiation matrix formulation gives a unified model for comparison of spectral and finite difference discretisations.

The numerical methods presented in this paper are standard ones from the numerical finance literature, so we do not go into details about their theory (convergence, stability, etc.). Source codes for all the computations are available at
`http://alpha.cc.tut.fi/~piche/finance/2007A/`

2 Black-Scholes model

The Black-Scholes equation for the value $Y(S,t)$ of a European put option on an asset of value S with volatility $\sqrt{v}$ and interest rate r is

$$\frac{\partial Y}{\partial t} + \tfrac{1}{2}vS^2\frac{\partial^2 Y}{\partial S^2} + rS\frac{\partial Y}{\partial S} - rY = 0. \qquad (2)$$

The terminal condition is $Y(S,T) = \max(S-E,0)$, where E is the exercise price. The boundary values are

$$Y(0,t) = 0, \quad Y(S,t) \approx S - Ee^{-r(T-t)} \text{ for large } S.$$

The change of variables $y(x,t) = \frac{1}{E}Y(Ee^{Lx}, 2\tau/v)$ transforms (2) into

$$\frac{\partial}{\partial \tau}y = -L^{-2}\frac{\partial^2}{\partial x^2}y - (k-1)L^{-1}\frac{\partial}{\partial x}y + ky \qquad (3)$$

on $x \in [-1,1]$, with $k = 2r/v$. The terminal condition is $y(x, vT/2) = \max(e^{Lx}-1, 0)$ and the boundary conditions are $y(-1,\tau) = 0$ and $y(1,\tau) = e^L - e^{-k(Tv/2-\tau)}$.

Denoting $\mathbf{y}(\tau) = y(\mathbf{x},\tau)$ for the values at the nodes and $\mathbf{i} = 2:N-1$ for the indices of the inner nodes, and applying the differential matrix approximation formula (1), the discretisation of (3) is

$$\dot{\mathbf{y}}_\mathbf{i} = -L^{-2}\mathbf{D}^{(2)}_{\mathbf{i},:}\mathbf{y} - (k-1)L^{-1}\mathbf{D}_{\mathbf{i},:}\mathbf{y} + k\mathbf{y}_\mathbf{i}. \qquad (4)$$

Notice how the differential matrix notation allows the discretisation (4) to be written in essentially the same form as the differential equation (3). Formulas for differentiation matrices are given in appendix A.

Writing equation (4) together with the boundary conditions $\mathbf{y}_1(\tau) = 0$ and $-\mathbf{y}_N(\tau) + e^L - e^{-k(Tv/2-\tau)} = 0$ gives the linear constant-coefficient differential algebraic equation

$$\mathbf{M}\dot{\mathbf{y}} = \mathbf{A}\mathbf{y} + (e^L - e^{-k(Tv/2-\tau)})\mathbf{I}_{:,N},$$

where

$$\mathbf{A} = \begin{bmatrix} -L^{-2}\mathbf{D}^{(2)}_{\mathbf{i},:} - (k-1)L^{-1}\mathbf{D}_{\mathbf{i},:} + k\mathbf{I}_{\mathbf{i},:} \\ \mathbf{I}_{1,:} \\ -\mathbf{I}_{N,:} \end{bmatrix},$$

$$\mathbf{M} = \begin{bmatrix} \mathbf{I}_{\mathbf{i},:} \\ 0 \\ 0 \end{bmatrix}.$$

This DAE can readily be solved by one of the time stepping formulas presented in Appendix B, or by a general-purpose solvers such as Matlab's `ode15s`. For example, the problem can be completely formulated and solved using finite differences and fully implicit (backward Euler) time stepping in just a few lines of Matlab code:

```
E=100; v=(0.3)^2; r=0.1; T=1;
L=log(3); nt=100; N=200;
[x,D,D2]=get_diffmatrix('fd',N);
I=speye(N); i=2:N-1; k=2*r/v;
A=[-L^(-2)*D2(i,:)-(k-1)/L*D(i,:)+k*I(i,:)
    I(1,:)
   -I(N,:)];
M=I(i,:);  M(N-1,N-1)=0; M(N,N)=0;
y=max(exp(L*x)-1,0);
h=-T*v/2/nt; tau=T*v/2:h:0;
[l,u]=lu(M-h*A);      % LU decomposition
for it=2:length(tau)  % timestepping
    b=(exp(L)-exp(-k*(T*v/2-tau(it))))*I(:,N);
    z=u\(l\(A*y+b));
    y=y+h*z;
end
```

Notice how the matrices of the DAE are directly translated into Matlab commands. A spectral collocation solution can be obtained simply by replacing `'fd'` in line 2 by `'sc'`.

The advantage of the L-stable Alexander method over the Crank-Nicolson method is seen in Figure 1, which shows the error in the finite difference solution with different time integrators. The spurious oscillation at the exercise price near the expiry time damps out amost entirely in a single time step in the Alexander solution, whereas in the Crank-Nicolson solution the oscillation amplitude is larger and damps out slowly over several time steps. The results for spectral collocation are similar.

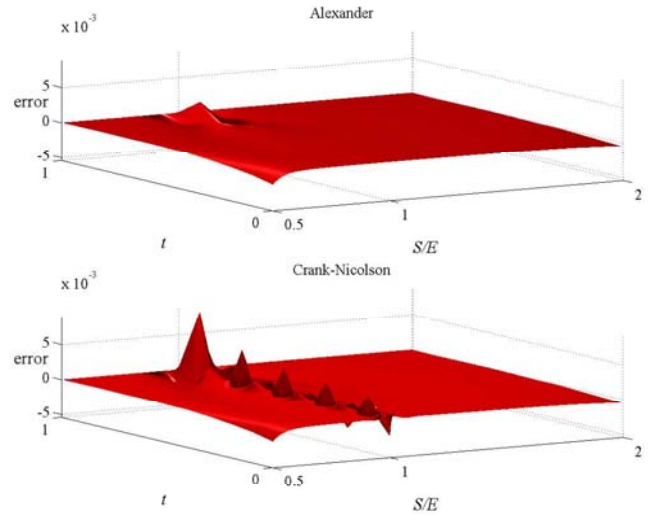

Figure 1: Errors in numerical solutions of European call pricing problem ($v = (0.3)^2$, $r = 0.1$) with finite difference discretisation and different time stepping formulas ($N = 500$, $L = \ln(2)$, 10 time steps).

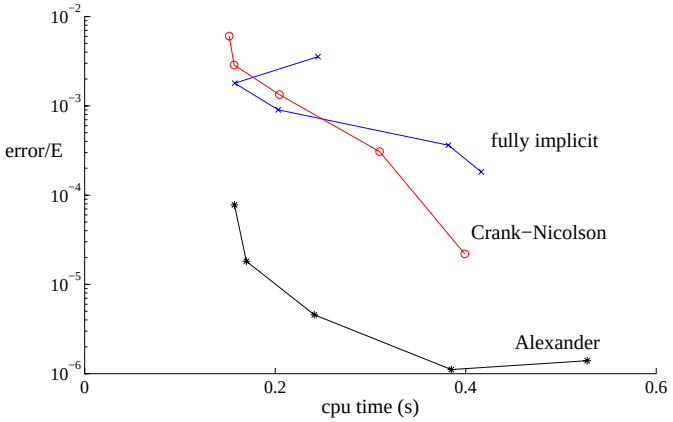

Figure 2: Error vs. computing time for a European call pricing problem. The error is the maximum absolute difference between the numerical solution and the Black-Scholes formula at $t = 0$. The finite difference grid has $N = 801$ nodes and the number of time steps varies from 5 to 100. The price range parameter is $L = \ln(4)$. Computing times are for an Apple PowerBook with 1.5 GHz G4 processor running Matlab 7.1.

The tradeoff between accuracy and computing time for different time integrators applied to this problem is shown in Figure 2. As $h \to 0$ the error for all the time stepping methods approaches the limiting error of the discretised ODE. Alexander's method is seen to provide better accuracy faster than the other time integrators.

3 American put option

The price of an American put option when S is greater than the early exercise price is governed by the nondimensionalised Black-Scholes equation (3) with boundary conditions $y(-1,\tau) = 1 - e^{-L}$, $y(1,\tau) = 0$, and terminal condition $y(x, Tv/2) = \max(1 - e^{Lx}, 0)$. Substituting the boundary conditions $\mathbf{y}_1 = 1 - e^{-L}$ and $\mathbf{y}_N = 0$ into the differentiation matrix discretisation (4) yields the ODE

$$\dot{\mathbf{y}}_\mathbf{i} = \mathbf{A}\mathbf{y}_\mathbf{i} + \mathbf{b}, \tag{5}$$

where

$$\begin{aligned} \mathbf{A} &= -L^{-2}\mathbf{D}^{(2)}_{\mathbf{i},\mathbf{i}} - (k-1)L^{-1}\mathbf{D}_{\mathbf{i},\mathbf{i}} + k\mathbf{I}_{\mathbf{i},\mathbf{i}}, \\ \mathbf{b} &= \left(-L^{-2}\mathbf{D}^{(2)}_{\mathbf{i},1} - (k-1)L^{-1}\mathbf{D}_{\mathbf{i},1} + k\mathbf{I}_{\mathbf{i},1}\right)(1 - e^{-L}). \end{aligned}$$

Denoting by $\bar{\mathbf{y}}$ the solution at the previous time step, the Crank-Nicolson formula for the solution of (5) at the next time step is

$$\mathbf{y}_\mathbf{i} = \mathbf{U}^{-1}\mathbf{L}^{-1}(\bar{\mathbf{y}}_\mathbf{i} + \tfrac{1}{2}h\mathbf{A}\bar{\mathbf{y}}_\mathbf{i} + h\mathbf{b}),$$

where $\mathbf{L}$, $\mathbf{U}$ are the LU factors of $\mathbf{I}_{\mathbf{i},\mathbf{i}} - \frac{1}{2}h\mathbf{A}$. Because these factors are triangular matrices, the operations are computed using forward and backward substitution rather than using matrix inverses.

The payoff constraint $y \geq 1 - e^{Lx}$ is applied by simply adding a thresholding operation into the Crank-Nicolson stepping formula:

$$\mathbf{y}_\mathbf{i} = \mathbf{U}^{-1} \max\left(\mathbf{L}^{-1}(\bar{\mathbf{y}}_\mathbf{i} + \tfrac{1}{2}h\mathbf{A}\bar{\mathbf{y}}_\mathbf{i} + h\mathbf{b}), \mathbf{U}(1 - e^{L\mathbf{x}_\mathbf{i}})\right).$$

Because $\mathbf{U}$ is upper triangular, the thresholding is applied in a back-substitution loop. It therefore does not affect solution components with larger indices. The following Matlab code computes an American put option price to 5 digit accuracy in less than a second:

```
E=50; v=(0.4)^2; r=0.1; T=5/12;
L=log(2); N=501; nt=500;
TT=T*v/2; h=-TT/nt; t=TT:h:0;
[x,D,D2]=get_diffmatrix('fd',N);
k=r/(v/2); I=speye(N); i=2:N-1;
S=E*exp(L*x); yc=1-S(i)/E;
y=max(1-S(i)/E,0); y1=1-S(1)/E;
A=-1/L^2*D2(i,i)-(k-1)/L*D(i,i)+k*I(i,i);
[l,u]=lu(I(i,i)-0.5*h*A); Uyc=u*yc;
b=(-1/L^2*D2(i,1)-(k-1)/L*D(i,1))*y1;
for it=2:length(t)
    y=u\max(l\(y+0.5*h*A*y+h*b),Uyc);
end
```

The code can easily be modified to use spectral collocation (replace `'fd'` in the fourth line by `'sc'`) and other implicit time stepping schemes (Appendix B). Call options with continuous dividends can be treated similarly.

4 Option with stochastic volatility

For two dimensional problems, the differentiation matrix approximation of a partial derivative at nodes on a tensor product grid is where $\mathbf{1}$ is a vector of ones.

Heston's PDE [3] for an option value $U(S,v,t)$ with asset price S, squared volatility v, volatility of volatility σ, spot price and volatility process correlation ρ, interest rate r, long term volatility θ, volatility mean-reversion rate κ, and market price of volatility risk λv absorbed into κ and σ, is

$$\begin{aligned}\frac{\partial U}{\partial t} &= rU - \tfrac{1}{2}vS^2\frac{\partial^2 U}{\partial S^2} - rS\frac{\partial U}{\partial S} - \rho\sigma vS\frac{\partial^2 U}{\partial S\partial v}\\ &\quad - \tfrac{1}{2}\sigma^2 v\frac{\partial^2 U}{\partial v^2} - \kappa(\theta - v)\frac{\partial U}{\partial v}.\end{aligned}$$

The terminal condition for a call option is

$$U(S,v,T) = \max(S-E,0).$$

Boundary conditions are

$$U(0,v,t) = 0, \quad U(S,v,t) \approx S - Ee^{-r(T-t)} \text{ for large } S.$$

Let $U(S,v,t) = Y(S,t) + C(S,v,t)$, where Y satisfies the Black-Scholes equation (2) for a call option. Subtracting off Y, Heston's equation can be written

$$\begin{aligned}\frac{\partial C}{\partial t} &= rC - \tfrac{1}{2}vS^2\frac{\partial^2 C}{\partial S^2} - rS\frac{\partial C}{\partial S} - \rho\sigma vS\frac{\partial^2 C}{\partial S\partial v}\\ &\quad - \tfrac{1}{2}\sigma^2 v\frac{\partial^2 C}{\partial v^2} - \kappa(\theta - v)\frac{\partial C}{\partial v} + F,\end{aligned}$$

where

$$F(S,v,t) = -\rho\sigma vS\frac{\partial^2 Y}{\partial v\partial S} - \tfrac{1}{2}\sigma^2 v\frac{\partial^2 Y}{\partial v^2} - \kappa(\theta - v)\frac{\partial Y}{\partial v}.$$

The terminal condition is $C(x,v,T) = 0$ and the boundary conditions are $C(0,v,t) = 0$ and $C(\infty,v,t) = 0$. F and Y can be computed using the Black Scholes formula or numerically as in section 2.

The change of variables

$$c(x,w,t) = \frac{1}{E}C(Ee^{L_1 x}, \theta e^{L_2 w}, t)$$

gives the Heston equation in the form

$$\begin{aligned}\dot{c} &= rc - \tfrac{1}{2}vL_1^{-2}\frac{\partial^2}{\partial x^2}c - \rho\sigma L_1^{-1}L_2^{-1}\frac{\partial^2}{\partial x\partial w}c\\ &\quad - \tfrac{1}{2}\sigma^2 v^{-1}L_2^{-2}\frac{\partial^2}{\partial w^2}c - (r - \tfrac{1}{2}v)L_1^{-1}\frac{\partial}{\partial x}c\\ &\quad + \left((\tfrac{1}{2}\sigma^2 - \kappa\theta)v^{-1} + \kappa\right)L_2^{-1}\frac{\partial}{\partial w}c\\ &\quad + E^{-1}F(Ee^{L_1 x}, \theta e^{L_2 w}, t) \qquad (6)\end{aligned}$$

on $(x,w) \in [-1,1]\times[-1,1]$. The terminal condition is $c(x,w,T) = 0$ and the boundary conditions are $c(-1,w,t) = 0$ and $c(1,w,t) = 0$. We assume that $\partial c/\partial w$ is zero on the boundaries $w = \pm 1$.

Introduce the notation $\mathbf{X} = \mathbf{x}\otimes\mathbf{1}_{N_2}^T$, $\mathbf{W} = \mathbf{1}_{N_1}\otimes\mathbf{w}^T$, $\mathbf{i} = 2 : N_1 - 1$, $\mathbf{j} = 2 : N_2 - 1$, $\mathbf{C} = c(\mathbf{X},\mathbf{W},t)$, $\mathbf{V} = \mathrm{diag}(\theta e^{L_2\mathbf{W}})$, and $\mathbf{D}^{(1)} = \mathbf{D}$, where $\mathbf{1}$ is a vector of ones. The differentiation matrix approximation of a partial derivative at the nodes on a tensor product grid is

$$\frac{\partial^{d_1+d_2}c}{\partial x^{d_1}\partial w^{d_2}}(\mathbf{x}\otimes\mathbf{1}^T, \mathbf{1}\otimes\mathbf{w}^T) \approx \mathbf{D}^{(d_1)}c(\mathbf{x}\otimes\mathbf{1}^T, \mathbf{1}\otimes\mathbf{w}^T)\mathbf{D}^{(d_2)T}. \qquad (7)$$

Applying this formula to (6) gives the discretisation

$$\begin{aligned}\dot{\mathbf{C}}_{\mathbf{i},\mathbf{j}} &= \mathbf{I}_{\mathbf{i},:}\Big(r\mathbf{C} - \tfrac{1}{2}L_1^{-2}\mathbf{D}^{(2)}\mathbf{C}\mathbf{V} - \rho\sigma L_1^{-1}L_2^{-1}\mathbf{D}\mathbf{C}\mathbf{D}^T\\ &\quad - \tfrac{1}{2}\sigma^2 L_2^{-2}\mathbf{C}\mathbf{D}^{(2)T}\mathbf{V}^{-1} - L_1^{-1}\mathbf{D}\mathbf{C}(r\mathbf{I} - \tfrac{1}{2}\mathbf{V})\\ &\quad + L_2^{-1}\mathbf{C}\mathbf{D}^T\left((\tfrac{1}{2}\sigma^2 - \kappa\theta)\mathbf{V}^{-1} + \kappa\mathbf{I}\right)\\ &\quad + E^{-1}F(Ee^{L_1\mathbf{X}}, \theta e^{L_2\mathbf{W}}, t)\Big)\mathbf{I}_{:,\mathbf{j}}. \qquad (8)\end{aligned}$$

This formula is rather long, but so is Heston's PDE; each term of (6) has been translated directly into a term with similar notation. Combining (8) with the boundary conditions

$$\mathbf{C}_{1,:} = \mathbf{C}_{N_1,:} = \mathbf{0}^T, \quad \mathbf{I}_{\mathbf{i},:}\mathbf{C}\mathbf{D}^T\mathbf{I}_{:,1} = \mathbf{I}_{\mathbf{i},:}\mathbf{C}\mathbf{D}^T\mathbf{I}_{:,N_2} = \mathbf{0}$$

gives $N_1\cdot N_2$ equations for the $N_1\cdot N_2$ variables in $\mathbf{C}$. Applying the Kronecker product identity that converts matrix products into matrix-vector products, these equations can be written as the linear constant-coefficient DAE

$$\mathbf{M}\mathrm{vec}(\dot{\mathbf{C}}) = \mathbf{A}\mathrm{vec}(\mathbf{C}) + \mathbf{b},$$

where

$$\mathbf{A} = \begin{bmatrix} r\mathbf{I}_{\mathbf{j},:}\otimes\mathbf{I}_{\mathbf{i},:} - \frac{1}{2}L_1^{-2}\mathbf{V}_{\mathbf{j},:}\otimes\mathbf{D}^{(2)}_{\mathbf{i},:} -\\ \rho\sigma L_1^{-1}L_2^{-1}\mathbf{D}_{\mathbf{j},:}\otimes\mathbf{D}_{\mathbf{i},:} -\\ \frac{1}{2}\sigma^2 L_2^{-2}\mathbf{V}_{\mathbf{j},:}^{-1}\mathbf{D}^{(2)}\otimes\mathbf{I}_{\mathbf{i},:} -\\ L_1^{-1}(r\mathbf{I}_{\mathbf{j},:} - \frac{1}{2}\mathbf{V}_{\mathbf{j},:})\otimes\mathbf{D}_{\mathbf{i},:} +\\ L_2^{-1}\left((\frac{1}{2}\sigma^2 - \kappa\theta)\mathbf{V}_{\mathbf{j},:}^{-1} + \kappa\mathbf{I}_{\mathbf{j},:}\right)\mathbf{D}\otimes\mathbf{I}_{\mathbf{i},:}\\ \mathbf{I}\otimes\mathbf{I}_{1,:}\\ \mathbf{I}\otimes\mathbf{I}_{N_1,:}\\ \mathbf{D}_{1,:}\otimes\mathbf{I}_{\mathbf{i},:}\\ \mathbf{D}_{N_2,:}\otimes\mathbf{I}_{\mathbf{i},:}\end{bmatrix},$$

$$\mathbf{M} = \begin{bmatrix}\mathbf{I}_{\mathbf{j},:}\otimes\mathbf{I}_{\mathbf{i},:}\\ \mathbf{0}\end{bmatrix},$$

and

$$\mathbf{b} = \begin{bmatrix}\mathrm{vec}(E^{-1}F(Ee^{L_1\mathbf{X}_{\mathbf{i},\mathbf{j}}}, \theta e^{L_2\mathbf{W}_{\mathbf{i},\mathbf{j}}}, t))\\ \mathbf{0}\end{bmatrix}.$$

This DAE is readily solved using any of the implicit timestepping formulas given in Appendix B, and the numerical solution with moderately dense grid agrees well with the solution given by Heston's formula (Figure 3).

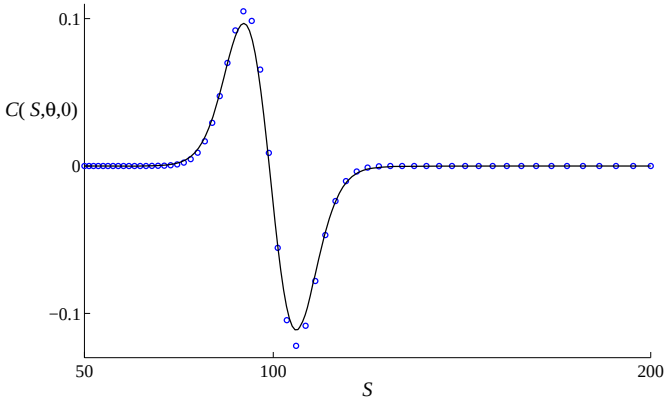

Figure 3: Finite difference/Alexander solution of the stochastic volatility model (dots) and Heston's formula (line). Model parameters are $E = 100$, $\sigma = 0.1$, $r = 0$, $T = 0.5$, $\kappa = 2$, $\theta = 0.01$, and $\rho = 0.5$. Numerical parameters are $N_1 = 60$, $N_2 = 15$, $e^{L_1} = 2$, $e^{L_2} = 4$, and $h = -0.05$.

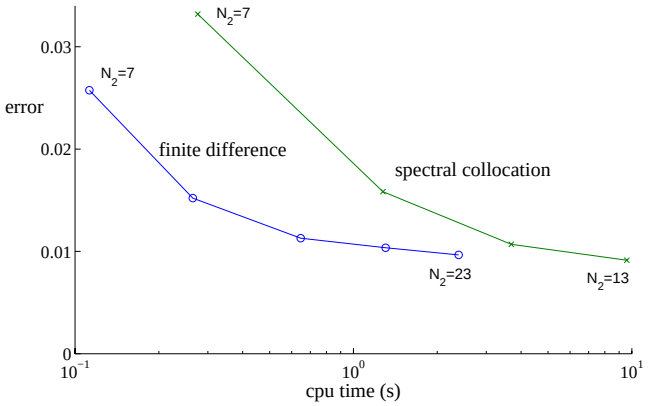

Figure 4: Efficiency of different discretisations of stochastic volatility model. Here 'error' is the maximum absolute difference between the numerical value and Heston's formula's value for $C(S, \theta, 0)$, with parameters as in Figure 3 except for N_1 and N_2, which vary with $N_1 = 4N_2$. Alexander's formula is used for time stepping.

The tradeoff between computing time and accuracy as the discretisation mesh is refined is shown in Figure 4. For this problem the finite difference method is faster than the spectral collocation method because the lack of smoothness in the terminal condition prevents the spectral method from achieving exponential convergence rates.

5 Optimal maintenance and shutdown

Sometimes machines generate (or are expected to generate) poor profit flows. The decision whether and when to shut down a machine to end an expected negative profit flow is an optimal stopping problem. In the model for optimal maintenance and shutdown given in [2], the marginal cost of maintenance equals the expected present value of marginal profits, ensuring the optimal decision. The model leads to the nonlinear ODE

$$0 = \pi + aF' + \frac{1}{2c}(F')^2 + \frac{\sigma^2}{2}F'' - rF$$

on $\pi_1 \le \pi \le \pi_N$, with boundary conditions

$$F(\pi_1) = 0, \quad F'(\pi_1) = 0, \quad F(\pi_N) - \frac{a}{r^2} - \frac{1}{2cr^3} - \frac{\pi_N}{r} = 0.$$

Here a is the depreciation rate of the profit flow, F is the rate of the maintenance, σ is the profit flow volatility, c is the maintenance cost, and π the value of the profit flow. There are three boundary conditions for the second order ODE because the value of π_1 (the "critical level") is also to be determined.

The change of variables

$$y(x) = F\left((x+1)\frac{\pi_N}{2} + (1-x)\frac{\pi_1}{2}\right)$$

transforms the ODE into

$$\begin{aligned} 0 &= (x+1)\frac{\pi_N}{2} + (1-x)\frac{\pi_1}{2} + \frac{2a}{\pi_N - \pi_1}y' \\ &\quad + \frac{2}{(\pi_N - \pi_1)^2 c}(y')^2 + \frac{2\sigma^2}{(\pi_N - \pi_1)^2}y'' - ry \end{aligned}$$

on $-1 \le x \le 1$.

Denoting $\mathbf{y}_i = y(\mathbf{x}_i)$ for $i = 1, \ldots, N$ and $\mathbf{y}_{N+1} = \pi_1$, the differentiation matrix approximation of the ODE is

$$\begin{aligned} 0 &= (\mathbf{x}_{2:N-1} + 1)\frac{\pi_N}{2} + \frac{1}{2}(1 - \mathbf{x}_{2:N-1})\mathbf{y}_{N+1} \\ &\quad + \frac{2a}{\pi_N - \mathbf{y}_{N+1}}\mathbf{D}_{2:N-1,1:N}\mathbf{y}_{1:N} \\ &\quad + \frac{2}{(\pi_N - \mathbf{y}_{N+1})^2 c}(\mathbf{D}_{2:N-1,1:N}\mathbf{y}_{1:N})^2 \\ &\quad + \frac{2\sigma^2}{(\pi_N - \mathbf{y}_{N+1})^2}\mathbf{D}^{(2)}_{2:N-1,1:N}\mathbf{y}_{1:N} - r\mathbf{y}_{2:N-1}, \end{aligned}$$

where the square is taken componentwise. This, together with the boundary conditions

$$\mathbf{y}_1 = \frac{2}{\pi_N - \mathbf{y}_{N+1}}\mathbf{D}_{1,1:N}\mathbf{y}_{1:N} = \mathbf{y}_N - \frac{a}{r^2} - \frac{1}{2cr^3} - \frac{\pi_N}{r} = 0,$$

gives $N+1$ equations for the $N+1$ unknowns. This nonlinear equation system with Newton iteration using the matrix of partial derivatives with respect to the unknowns $\mathbf{y}$, whose derivation is straightforward. Using $\pi_1 = 0$ and $y(x) = \frac{1}{2}(x+1)\mathbf{y}_N$ to define initial values of $\mathbf{y}$ for the iteration, the Newton method typically converges to machine precision in 5 iterations.

Figure 5 shows that the spectral collocation method converges much faster than the finite difference method, as

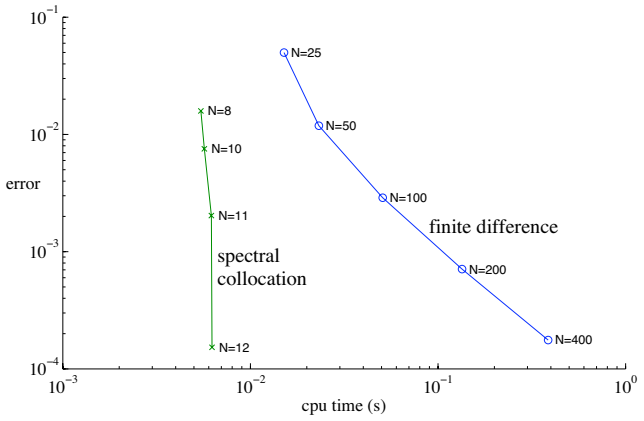

Figure 5: Error vs. computing time for the optimal maintenance and shutdown problem with $a = -0.1$, $\sigma = 0.2$, $r = 0.1$, $c = 200$, $\pi_N = 10$. The error is that of the computed value of the critical level π_1, using as reference the value $\pi_1 = -0.1794460361577$ computed using spectral collocation with $N = 100$.

expected for such a problem with a smooth solution. However, the finite difference method's performance is acceptable for moderate precision levels, and several orders of magnitude faster than the finite difference method used in [2].

6 Concluding Remarks

In this paper we presented four case studies to illustrate the differentiation matrix formalism as a tool for developing discretised models for financial differential equations in one and two dimensions. The use of a high level notation that closely resembles the original differential equation was exploited to rapidly produce effective solution code in the matrix-oriented scientific programming language Matlab. The unified formulation facilitated numerical comparison of different discretisation schemes (finite difference vs. spectral collocation) and time-stepping schemes (Alexander vs. Crank-Nicolson).

In addition to the finite difference and Chebyshev spectral collocation discretisations presented here, other discretisation methods can be implemented using the same differentiation matrix models simply by substituting the appropriate differentiation matrix formulas. These include spectral collocation schemes using other basis functions (radial, sinc , Fourier, Laguerre), and higher order finite difference schemes. Finite element methods can also be incorporated into the differentiation matrix formalism by generalising the basic approximation formula (1) to

$$f^{(d)}(\mathbf{x}) \approx \mathbf{E}^{-1}\mathbf{D}^{(d)}f(\mathbf{x}),$$

where $\mathbf{E}$ is the finite element "mass matrix".

For problems with dimension higher than two the matrix notation no longer suffices, and one would need the more general notation of multilinear algebra. For such problems we would not explicitly form the Kronecker products in the codes, and would use iterative solvers in place of gaussian elimination.

References

[1] R. Alexander, Diagonally implicit Runge-Kutta methods for stiff ODE's, *SIAM J. Numer. Anal.*, **14**, 1977, 1006–1021.

[2] T. Dangl & F. Wirl, Investment under uncertainty: calculating the value function when the Bellman equation cannot be solved analytically, *Journal of Economic Dynamics and Control*, **28**, 2004, 1437–1460.

[3] S. L. Heston, A closed-form solution for options with stochastic volatility with applications to bond and currency options, *The Review of Financial Studies*, **6** (2) 1993, 327–343.

[4] S. Ikonen & J. Toivanen, Pricing american options using LU decomposition, University of Jyväskylä Dept of Mathematical Information Technology Report, 2005.

[5] L. N. Trefethen, *Spectral Methods in Matlab*, SIAM, 2000.

[6] J. A. C. Weideman & S. C. Reddy, A Matlab differentiation matrix suite, *ACM Transactions on Mathematical Software*, **26** (4), 2000, 465–519.

A Differentiation matrix formulas

The function domain is assumed to be $x \in [-1, 1]$. Functions with domain $\xi \in [a, b]$ can be transformed using the affine change of variables

$$\xi = \frac{1-x}{2}a + \frac{1+x}{2}b.$$

A.1 Finite differences

The central difference formula for approximating the first derivative is

$$y'(x) \approx \frac{y(x+\delta) - y(x-\delta)}{2\delta}.$$

The one-sided difference formulas

$$\begin{aligned} y'(x) &\approx \frac{-3y(x) + 4y(x+\delta) - y(x+2\delta)}{2\delta}, \\ y'(x) &\approx \frac{y(x-2\delta) - 4y(x-\delta) + 3y(x)}{2\delta} \end{aligned}$$

have the same order of accuracy as the central difference formula. Let $\mathbf{x}_i = -1+(i-1)\delta$ for $i = 1, 2, \ldots N$ be a grid

of equally spaced points in $[-1,1]$, where $\delta = 2/(N-1)$, and let $y(\mathbf{x})$ be the N-vector of function values at the grid nodes. Using the above difference formulas, the first derivative values can be approximated by

$$y'(\mathbf{x}) \approx \mathbf{D}^{(1)} y(\mathbf{x}),$$

where the $N \times N$ differentiation matrix is

$$\mathbf{D}^{(1)} = \frac{1}{\delta}\begin{bmatrix} -3/2 & 2 & -1/2 & & & \\ -1/2 & 0 & 1/2 & & & \\ & -1/2 & 0 & 1/2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1/2 & 0 & 1/2 \\ & & & 1/2 & -2 & 3/2 \end{bmatrix}.$$

The differentiation matrix corresponding to the central difference formula

$$y''(x) \approx \frac{y(x+\delta) - 2y(x) + y(x-\delta)}{\delta^2}$$

for the second derivative can be derived analogously.

A.2 Spectral collocation

Let ϕ_j be the polynomials of degree $N-1$ satisfying $\phi_j(\mathbf{x}_k) = \delta_{j,k}$ for $k = 1, \ldots, N$ at the Chebyshev nodes

$$\mathbf{x}_k = \cos((N-k)\pi/(N-1)).$$

The polynomial $p(x) = \sum_{j=1}^{N} \phi_j(x) y(\mathbf{x}_j)$ interpolates the function y at the nodes, that is, $p(\mathbf{x}) = y(\mathbf{x})$. The value of the interpolating polynomial's dth derivative at the kth node is $p^{(d)}(\mathbf{x}_k) = \sum_{j=1}^{N} \phi_j^{(d)}(\mathbf{x}_k) y(\mathbf{x}_j)$. This can be written

$$p^{(d)}(\mathbf{x}) = \mathbf{D}^{(d)} y(\mathbf{x}),$$

where $[\mathbf{D}_{k,j}^{(d)}] = [\phi_j^{(d)}(\mathbf{x}_k)]$ is the dth order differentiation matrix for the spectral collocation method. Formulas and codes for generating these matrices are given in [6].

B Time integration formulas

The following formulas explain how to compute the increment $\mathbf{z}$ in the time stepping formula

$$\mathbf{y}(T+h) \approx \mathbf{y}(T) + h\mathbf{z}$$

for the solution of a differential algebraic equation of the form

$$\mathbf{M}\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t)).$$

The jacobian $\partial \mathbf{f}/\partial \mathbf{y}(T, y(T))$ is denoted $\mathbf{J}$. The solution is computed by applying the time stepping formula recursively.

Fully implicit (backward Euler): The increment $\mathbf{z}$ is the solution of the equation

$$\mathbf{r}(\mathbf{z}) = \mathbf{M}\mathbf{z} - \mathbf{f}(T+h, \mathbf{y}(T) + h\mathbf{z}) = 0$$

It can be found by starting with $\mathbf{z}^{(0)} = \mathbf{0}$ and recursively solving

$$(\mathbf{M} - h\mathbf{J})\mathbf{z}^{(k)} = \mathbf{f}(T+h, \mathbf{y}(T) + h\mathbf{z}^{(k-1)}) - h\mathbf{J}\mathbf{z}^{(k-1)}$$

until $\|\mathbf{r}(\mathbf{z}^{(k)})\|$ is small enough. In the case $\mathbf{f}(t, \mathbf{y}) = \mathbf{A}\mathbf{y} + \mathbf{b}(t)$ with constant $\mathbf{A}$, the equation for $\mathbf{z}$ is linear:

$$(\mathbf{M} - h\mathbf{A})\mathbf{z} = \mathbf{A}\mathbf{y}(T) + \mathbf{b}(T+h).$$

Crank-Nicolson (trapezoid): The increment $\mathbf{z}$ is the solution of the equation

$$\mathbf{r}(\mathbf{z}) = \mathbf{M}\mathbf{z} - \tfrac{1}{2}\mathbf{f}(T, y(T)) - \tfrac{1}{2}\mathbf{f}(T+h, \mathbf{y}(T) + h\mathbf{z}) = 0$$

It can be found by starting with $\mathbf{z}^{(0)} = \mathbf{0}$ and recursively solving

$$\begin{aligned}(\mathbf{M} - \tfrac{1}{2}h\mathbf{J})\mathbf{z}^{(k)} &= -\tfrac{1}{2}h\mathbf{J}\mathbf{z}^{(k-1)} + \tfrac{1}{2}\mathbf{f}(T, y(T)) \\ &\quad + \tfrac{1}{2}\mathbf{f}(T+h, \mathbf{y}(T) + h\mathbf{z}^{(k-1)})\end{aligned}$$

until $\|\mathbf{r}(\mathbf{z}^{(k)})\|$ is small enough. In the case $\mathbf{f}(t, \mathbf{y}) = \mathbf{A}\mathbf{y} + \mathbf{b}(t)$ with constant $\mathbf{A}$, the equation for $\mathbf{z}$ is linear:

$$(\mathbf{M} - \tfrac{1}{2}h\mathbf{A})\mathbf{z} = \mathbf{A}\mathbf{y}(T) + \tfrac{1}{2}\mathbf{b}(T) + \tfrac{1}{2}\mathbf{b}(T+h).$$

Alexander [1]: The stage equations are

$$\begin{aligned}\tilde{\mathbf{r}}(\tilde{\mathbf{z}}) &= \mathbf{M}\tilde{\mathbf{z}} - \gamma\mathbf{f}(T + \gamma h, \mathbf{y}(T) + h\tilde{\mathbf{z}}) = 0 \\ \mathbf{r}(\mathbf{z}) &= \mathbf{M}\mathbf{z} - (1-\gamma)\mathbf{f}(T + \gamma h, \mathbf{y}(T) + h\tilde{\mathbf{z}}) \\ &\quad - \gamma\mathbf{f}(T+h, \mathbf{y}(T) + h\mathbf{z}) = 0\end{aligned}$$

where $\gamma = 1 - (2)^{-1/2} \approx 0.2929$. The first stage value $\tilde{\mathbf{z}}$ can be found by starting with $\tilde{\mathbf{z}}^{(0)} = \mathbf{0}$ and recursively solving

$$(\mathbf{M} - \gamma h\mathbf{J})\tilde{\mathbf{z}}^{(k)} = \gamma\mathbf{f}(T + \gamma h, \mathbf{y}(T) + h\tilde{\mathbf{z}}^{(k-1)}) - \gamma h\mathbf{J}\tilde{\mathbf{z}}^{(k-1)}$$

until $\|\tilde{\mathbf{r}}(\tilde{\mathbf{z}}^{(k)})\|$ is small enough. Then $\mathbf{z}$ can be found by starting with $\mathbf{z}^{(0)} = \mathbf{0}$ and recursively solving

$$\begin{aligned}(\mathbf{M} - \gamma h\mathbf{J})\mathbf{z}^{(k)} &= (1-\gamma)\mathbf{f}(T + \gamma h, \mathbf{y}(T) + h\tilde{\mathbf{z}}) \\ &\quad + \gamma\mathbf{f}(T+h, \mathbf{y}(T) + h\mathbf{z}^{(k-1)}) \\ &\quad - \gamma h\mathbf{J}\mathbf{z}^{(k-1)}\end{aligned}$$

until $\|\mathbf{r}(\mathbf{z}^{(k)})\|$ is small enough. In the case $\mathbf{f}(t, \mathbf{y}) = \mathbf{A}\mathbf{y} + \mathbf{b}(t)$ with constant $\mathbf{A}$, the equations are linear:

$$\begin{aligned}(\mathbf{M} - \gamma h\mathbf{A})\tilde{\mathbf{z}} &= \gamma\mathbf{A}\mathbf{y}(T) + \gamma\mathbf{b}(T + \gamma h) \\ (\mathbf{M} - \gamma h\mathbf{A})\mathbf{z} &= (1-\gamma)\mathbf{b}(T + \gamma h) + \gamma\mathbf{b}(T+h) \\ &\quad + \mathbf{A}(\mathbf{y}(T) + h(1-\gamma)\tilde{\mathbf{z}}).\end{aligned}$$