Matrix-based numerical modelling of financial differential equations

Robert Piché* and Juho Kanniainen

Tampere University of Technology
PO Box 553, FI-33101 Tampere, Finland
E-mail: robert.piche@tut.fi
E-mail: juho.kanniainen@tut.fi
*Corresponding author

Abstract: Differentiation matrices provide a compact and unified formulation for a variety of differential equation discretisation and time-stepping algorithms. This paper illustrates their use for solving three differential equations of finance: the classic Black-Scholes equation (linear initial-boundary value problem), an American option pricing problem (linear complementarity problem), and an optimal maintenance and shutdown model (nonlinear boundary value problem with free boundary). We present numerical results that show the advantage of an L-stable time-stepping method over the Crank-Nicolson method, and results that show how spectral collocation methods are superior for boundary value problems with smooth solutions, while finite difference methods are superior for option-pricing problems.

Keywords: differentiation matrix, finite difference, Chebyshev spectral collocation, method of lines, Runge-Kutta, option pricing, American options, optimal shutdown.

Biographical Notes: Robert Piché (PhD 1986, University of Waterloo) is professor of mathematics at Tampere University of Technology. His research interests include mathematical modelling and numerical algorithms in applications including finance, navigation, image processing, and mechanics.

Juho Kanniainen (MSc 2002 and PhD 2005, Tampere University of Technology) is adjunct professor of mathematical finance and senior lecturer of finance at the Tampere University of Technology. His research interests are in the areas of derivative securities, computational finance, real options, risk management and corporate finance.

1 Introduction

Matrix notation is widely used in applied mathematics because of its conciseness and flexibility. By reducing complicated formulas teeming with summation symbols and indices to a few symbols, the analyst gains better insight into the essential properties of mathematical models. The concise representation also facilitates the algebraic manipulations in the development of solution algorithms, such as the derivation of the jacobian matrix for the Newton method.

A good example of the power of matrix notation is the discretisation of differential equation problems using *differentiation matrices*, a technique originally developed in the context of the spectral collocation (also known as pseudospectral) method. 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 $y(x)$ are approximated by

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

Differential equations can be discretised by simply replacing derivative operators with differentiation matrices. Boundary conditions can be modelled by adding equations to the model.

The following simple example illustrates how the discretised version of a differential equation problem is written in a high-level matrix notation that closely mimics the mathematical notation of the problem. Consider the simple boundary value problem

$$y'' + y = 0, \quad y(0) = 0, \quad y'(1) = 2. \tag{2}$$

Let $\mathbf{y}$ denote the vector that is the approximation of the solution at the nodes $\mathbf{x}$, with $0 = \mathbf{x}_1 < \mathbf{x}_2 < \ldots < \mathbf{x}_N = 1$. The left boundary condition can be modelled by $\mathbf{I}_{1,:}\,\mathbf{y} = 0$, where $\mathbf{I}_{1,:}$ denotes the first row of the $N \times N$ identity matrix. Similarly, the right boundary condition can by modelled by $\mathbf{D}^{(2)}_{N,:}\mathbf{y} = 2$ and the differential equation by $(\mathbf{D}^{(2)}_{2:N-1,:} + \mathbf{I}_{2:N-1,:})\mathbf{y} = \mathbf{0}$. Putting these together, the differentiation matrix model of the boundary value problem (2) is the linear matrix equation

$$\begin{bmatrix} \mathbf{D}^{(2)}_{2:N-1,:} + \mathbf{I}_{2:N-1,:} \\ \mathbf{I}_{1,:} \\ \mathbf{D}_{N,:} \end{bmatrix} \mathbf{y} = \begin{bmatrix} \mathbf{0} \\ 0 \\ 2 \end{bmatrix}.$$

This model is in a form suitable for solution with standard linear algebra software.

Modern scientific computing programming languages such as MATLAB and Fortran 90 allow coding of matrix operations using syntax that is very close to standard mathematical notation. This has several advantages. First, the translation of mathematical formulas into code is faster and less prone to errors. Secondly, the matrix computations can be carried out using existing high-performance linear algebra libraries that implement architecture-dependent (e.g. parallel) algorithms. Third, because the details of the discretisation are hidden in the matrix notation, it is easy to compare different discretisation schemes for a given problem using the same high-level code.

The merging of matrix-based coding and matrix-based modelling of differential equation problems was pioneered by Weideman and Reddy (2000) and Trefethen

(2000). They demonstrate how complete numerical solution of linear two-point boundary value problems, eigenvalue problems, and linear initial-boundary value problems of mathematical physics can be computed with 4–10 lines of MATLAB code. Weideman and Reddy (2000) also provide support functions for spectral collocation differentiation matrices corresponding to Chebyshev, Legendre, Hermite, Laguerre, Fourier, and sinc interpolants. These matrices are full and become ill-conditioned for $N > 100$ or so, but these drawbacks are often compensated by the exponential convergence rate for problems with smooth solutions. Differentiation matrices corresponding to second-order finite difference formulas are given by Trefethen (2000); these matrices are sparse banded and the solution convergence rates are typically at most $O(N^{-2})$.

The objective of this paper is to introduce differentiation-matrix based modelling to numerical finance practicioners by presenting step-by-step solutions of some typical financial differential equations problems. These problems present features distinct from those of the mathematical physics problems previously solved using differentiation matrices: linear complementarity constraints (American options), nonlinear coefficients (optimal shutdown model), and free boundaries (American options, optimal shutdown model). We introduce techniques that extend differentiation matrix modelling to deal with these features.

To illustrate the flexibility of the modelling approach, we present and compare solutions with different state-variable discretisations and time-stepping schemes. These or similar methods have been presented elsewhere in the literature, but results are widely scattered; the present paper serves mainly to collect these methods into a common methodological framework, and the example's results serve to confirm the methods' known properties.

- Bunnin et al. (1999) and Caporale and Cerrato (2008) have presented finance examples computed using Chebyshev spectral collocation, but they do not make comparisons with the finite difference method, which is widely used in finance (Duffy, 2005; Tavella and Randall, 2000). In the differentiation matrix framework, the modification of the spectral collocation code into finite difference code is trivial, which greatly facilitates comparison of the two methods. Here we present results showing that finite difference methods are more effective than spectral collocation in solving basic option-pricing problems.

- Ikonen and Toivanen (2007) have presented a matrix-based approach to solving American options. Here this algorithm is adapted to a put option, and generalised to include spectral collocation discretisations.

- The Crank-Nicolson time stepping scheme, which is widely used in computational finance (Duffy, 2005; Tavella and Randall, 2000), is known to produce spurious oscillations near the expiry time for fine state variable discretisation and coarse time discretisation. The backward Euler scheme is often presented as an alternative time stepping scheme that does not produce these oscillations because it is L-stable. However, the backward Euler scheme only has first-order accuracy, compared to the second-order accuracy of the Crank-Nicolson scheme. Second-order accurate L-stable time-stepping schemes for option pricing have been investigated by Ikonen and Toivanen (2007), and fourth-order schemes by Voss et al. (2003). We present computational results

that confirm the good stability and accuracy properties of a second-order L-stable scheme.

- Dangl and Wirl (2004) used Chebyshev spectral collocation with front-fixing to solve their optimal shutdown problem, but the finite difference method they used for comparison was rather ineffective. We present a finite difference implementation that is much faster, but find that spectral collocation remains the most effective method.

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.). Codes for all the computations are available at `http://alpha.cc.tut.fi/~piche/finance/2009/`

2 BLACK-SCHOLES MODEL

The model of Black and Scholes (1973) for the value $y(x,t)$ of a European put option on an asset of value x with volatility $\sqrt{v}$ and interest rate r is

$$\left.\begin{aligned} \frac{\partial y}{\partial t} &= \overbrace{\left(r - rx\frac{\partial}{\partial x} - \tfrac{1}{2}vx^2\frac{\partial^2}{\partial x^2}\right)}^{\mathcal{H}} y \\ y(0,t) = Ee^{-r(T-t)}, &\quad \lim_{x\to\infty} y(x,t) = 0 \\ y(x,T) &= \max(E-x,0) \end{aligned}\right\} \tag{3}$$

Let $\mathbf{x}$ denote the nodes (with $0 = \mathbf{x}_1 < \mathbf{x}_2 < \ldots < \mathbf{x}_N = L$). We also introduce the notation $\mathbf{y} \approx y(\mathbf{x},t)$, $\mathbf{X} = \operatorname{diag}(\mathbf{x})$, and $\mathbf{i} = [2 : N-1]$. Applying the differential matrix approximation formula (1), the discretisation of the partial differential equation in (3) is

$$\dot{\mathbf{y}}_\mathbf{i} = \mathbf{I}_{\mathbf{i},:}\underbrace{(r\mathbf{I} - r\mathbf{X}\mathbf{D} - \tfrac{1}{2}v\mathbf{X}^2\mathbf{D}^{(2)})}_{\mathbf{H}}\mathbf{y}. \tag{4}$$

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

Substituting the boundary conditions equations

$$\mathbf{y} = \mathbf{I}_{:,\mathbf{i}}\mathbf{y}_\mathbf{i} + \mathbf{I}_{:,1}\cdot Ee^{-r(T-t)}$$

into (4) yields the linear constant-coefficient ordinary differential equation

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

where

$$\begin{aligned} \mathbf{A} &= \mathbf{I}_{:,\mathbf{i}}\mathbf{H}\mathbf{I}_{:,\mathbf{i}} = \mathbf{H}_{\mathbf{i},\mathbf{i}} \\ \mathbf{b} &= \mathbf{H}_{\mathbf{i},1}\cdot Ee^{-r(T-t)}. \end{aligned}$$

The Crank-Nicolson scheme for solving this ODE requires solving the linear matrix equation

$$\mathbf{M}_{\mathrm{cn}}\mathbf{y_i}(t_k + h) = \mathbf{y_i}(t_k) + \tfrac{1}{2}h(\mathbf{A}\mathbf{y_i}(t_k) + \mathbf{b}(t_k) + \mathbf{b}(t_k + h))$$

at each time step, where h is the (negative) time increment and $\mathbf{M}_{\mathrm{cn}} = \mathbf{I} - \frac{1}{2}h\mathbf{A}$. The coefficient matrix $\mathbf{M}_{\mathrm{cn}}$ is constant, so its LU factorization only need be calculated once and the factors can be re-used at each time step.

The European put option pricing problem can be completely formulated and solved with just a dozen lines of Matlab code:

```
E=1; v=(0.3)^2; r=0.1; T=1; L=4*E; nt=100; N=500;
[x,D1,D2]=get_diffmatrix('fd',N,0,L);
i=2:N-1; I=speye(N); X=spdiags(x,0,N,N); X2=X.^2;
H=r*I-r*X*D1-0.5*v*X2*D2; A=H(i,i);
xi=x(i); yi=max(E-xi,0);
h=-T/nt; t=T:h:0;
b=H(i,N)*E;
[l,u]=lu(I(i,i)-0.5*h*A); % LU decomposition
for it=2:length(t)
    b1=H(i,1)*E*exp(-r*(T-t(it)));
    yi=u\(l\(yi+0.5*h*(A*yi+b+b1)));
    b=b1;
end
```

In this code, the sparse banded differentiation matrices are generated by the call to the function `get_diffmatrix`, which implements the formulas of Appendix A. The matrix algebra operators are overloaded in MATLAB to implement sparse matrix algorithms transparently.

A spectral collocation solution can be obtained simply by changing `'fd'` to `'sc'` in the call to `get_diffmatrix`; this changes the formulas used to generate the differentiation matrices. The spectral collocation differentiation matrices are full, and their order N is typically limited to 100 to avoid problems due to ill-conditioning. For this problem, the spectral collocation discretisation is not as efficient as finite differences (Figure 1). This is because the lack of smoothness of the solution (derivative jump at $x = E$ and $t = T$) prevents the method from achieving the exponential convergence rates that it typically exhibits for problems with smooth solutions.

The Crank-Nicolson scheme is known to produce spurious numerical oscillation when the terminal condition is not smooth. The oscillation due to the derivative jump at $x = E$ and $t = T$ is seen in Figure 2, which shows the error of the numerical solution relative to the exact solution of (3). This oscillation becomes more serious as the state variable discretisation is made finer, which makes the ordinary differential equation (5) stiffer.

The implicit two-stage Runge-Kutta scheme of Alexander (1977) has the same numerical complexity and the same order of time-discretisation error as the Crank-Nicolson scheme, but because it is L-stable, it is not so prone to producing spurious numerical oscillation (Figure 2) . The scheme requires solving the two linear matrix equations

$$\mathbf{M}_{\mathrm{a}}\mathbf{z} = \mathbf{A}\mathbf{y_i}(t_k) + \mathbf{b}(t_k + \gamma h)$$

$$\mathbf{M}_{\mathrm{a}}\mathbf{y_i}(t_k + h) = \mathbf{y_i}(t_k) + (1 - \gamma)h\mathbf{z} + \gamma h\mathbf{b}(t_k + h)$$

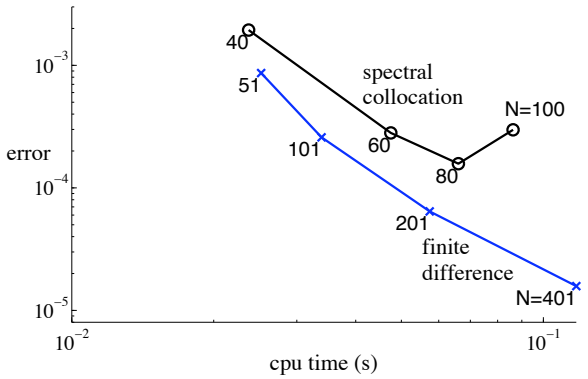

Figure 1 Error and computing time of numerical solutions of European put option pricing problem ($v = (0.3)^2$, $r = 0.1$, $E = 1$, $L = 4$, $T = 1$, 100 Crank-Nicolson steps) and two discretisations. The error is that of the option price at $x = E$ and $t = 0$. Computing times are for Matlab 7.1 on an Apple PowerBook with 1.5 GHz G4 processor.

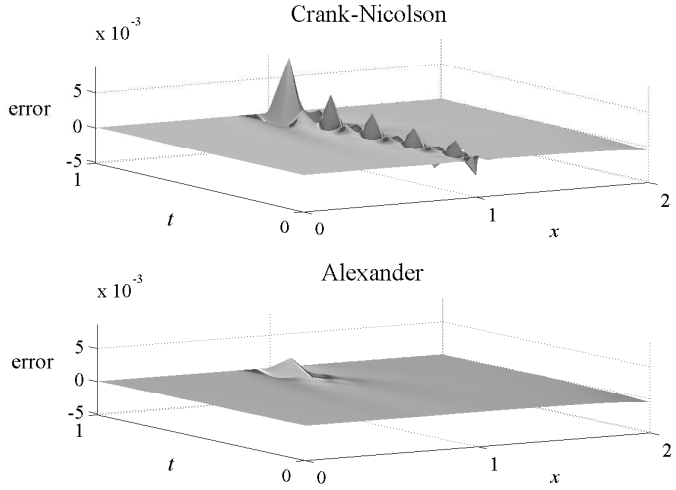

Figure 2 Error in numerical solution of European put option pricing problem ($v = (0.3)^2$, $r = 0.1$, $E = 1$) with finite difference discretisation and two time-stepping formulas ($N = 500$, $L = 2$, and 10 time steps).

where $\mathbf{M}_\mathrm{a} = \mathbf{I} - \gamma h \mathbf{A}$ with $\gamma = 1 - \frac{1}{\sqrt{2}}$. The coefficient matrix $\mathbf{M}_\mathrm{a}$ is constant and is the same for both stage equations, so the LU factorization only need be performed once.

3 AMERICAN OPTION

The value of an American put option is governed by the linear complementarity problem

$$\left(y - g(x)\right) \cdot \left(\mathcal{H}y - \frac{\partial y}{\partial t}\right) = 0$$
$$y - g(x) \geq 0, \quad \mathcal{H}y - \frac{\partial y}{\partial t} \geq 0$$

where $g(x) = \max(E - x, 0)$ is the payoff value. Approximating x-derivatives similarly as for the Black-Scholes equation, we obtain the discretised problem

$$\left(\mathbf{y_i} - g(\mathbf{x_i})\right) \odot (\mathbf{A}\mathbf{y_i} + \mathbf{b} - \dot{\mathbf{y}}_\mathbf{i}) = 0 \tag{6}$$

$$\mathbf{y_i} - g(\mathbf{x_i}) \geq 0, \quad \mathbf{A}\mathbf{y_i} + \mathbf{b} - \dot{\mathbf{y}}_\mathbf{i} \geq 0 \tag{7}$$

where $\odot$ denotes elementwise (Hadamard) product.

Generally, a linear complementarity problem with implicit time-stepping scheme requires iterative solution methods. However, in the case of an American option, the fact that the early-exercise constraint $y \geq g$ is active in a single contiguous domain allows the solution can be computed directly using a simple modification of the European option calculation, known as the Brenner-Schwarz algorithm. A matrix-based presentation of the algorithm for American get options and finite differences is presented by Ikonen and Toivanen (2007); the modification for put options and extension to general differentiation matrices is straightforward and is as follows.

Recall that the time stepping scheme for the European option requires solving a linear system of the form $\mathbf{M}\mathbf{z} = \ldots$ for $\mathbf{z} := \mathbf{y_i}(t_k + h)$. Let $\mathbf{U}\mathbf{L} = \mathbf{M}$ be a triangular factorisation of the time stepping matrix with $\mathbf{U}$ upper triangular and $\mathbf{L}$ lower triangular. This UL factorisation can be computed with standard LU factorisation code and reordering of the indices. In a European option calculation, the linear system is solved by a back substitution $\mathbf{z} \leftarrow \mathbf{U}^{-1}\mathbf{z}$ followed by a forward substitution $\mathbf{z} \leftarrow \mathbf{L}^{-1}\mathbf{z}$. To calculate the value of an American option, simply apply the early-exercise constraint during the forward substitution:

$$\mathbf{z}_j \leftarrow \max\left(g(\mathbf{x}_j), \frac{\mathbf{z}_j - \mathbf{L}_{j,1:j-1}\mathbf{z}_{1:j-1})}{\mathbf{L}j, j}\right).$$

A complete formulation and solution of the American put option pricing problem is thus obtained by replacing the last 6 lines of the European put option code by:

```
[u,l]=lu(I(i,i)-0.5*h*A(end:-1:1,end:-1:1));
u=u(end:-1:1,end:-1:1); l=l(end:-1:1,end:-1:1);
for it=2:length(t)
  b1=H(i,1)*E*exp(-r*(T-t(it)));
```

```
    yi=u\(yi+0.5*h*(A*yi+b+b1));
    for j=1:length(y)
      yi(j)=yi(j)-sum(l(j,1:j-1)'.*yi(1:j-1));
      yi(j)=max(yi(j)/l(j,j),E-xi(j));
    end
    b=b1;
  end
```

4 OPTIMAL SHUTDOWN

The decision whether and when to shut down a plant to end an expected negative profit flow is an optimal stopping problem. In the model for optimal maintenance and shutdown given by Dangl and Wirl (2004), 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 $l \le \pi$, with boundary conditions $F(l) = 0$, $F'(l) = 0$, and $F(\pi) \approx \frac{a}{r^2} + \frac{1}{2cr^3} + \frac{\pi}{r}$ as $\pi \to \infty$. 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 l (the "critical level") is also to be determined.

A standard approach to one-dimensional free-boundary problems is to map the problem to a fixed domain by a change of variables. Fixing the right boundary at $\pi = L$ and making the change of variables

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

transforms the ODE into

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

on $-1 \le x \le 1$, with boundary conditions

$$y(-1) = \frac{2a}{L-l}y'(-1) = y(1) - \frac{a}{r^2} - \frac{1}{2cr^3} - \frac{L}{r} = 0.$$

Denoting $\mathbf{y}_i = y(\mathbf{x}_i)$ for $i = 1, \ldots, N$ and $\mathbf{i} = 2 : N-1$, the differentiation matrix approximation of the ODE is

$$\begin{aligned} 0 = {} & (\mathbf{x_i}+1)\frac{L}{2} + \frac{1}{2}(1-\mathbf{x_i})l + \frac{2a}{L-l}\mathbf{D}_{\mathbf{i},:}\mathbf{y} \\ & + \frac{2}{(L-l)^2 c}(\mathbf{D}_{\mathbf{i},:}\mathbf{y}) \odot (\mathbf{D}_{\mathbf{i},:}\mathbf{y}) \\ & + \frac{2\sigma^2}{(L-l)^2}\mathbf{D}^{(2)}_{\mathbf{i},:}\mathbf{y} - r\mathbf{I}_{\mathbf{i},:}\mathbf{y}. \end{aligned}$$

This, together with the boundary conditions

$$\mathbf{y}_1 = 0, \quad \frac{2a}{L-l}\mathbf{D}_{1,:}\mathbf{y} = 0, \quad \mathbf{y}_N - \frac{a}{r^2} - \frac{1}{2cr^3} - \frac{L}{r} = 0,$$

gives $N+1$ equations for the $N+1$ unknowns $\mathbf{z} = [\mathbf{y}_1, \ldots, \mathbf{y}_N, l]^T$.

The system of equations $\mathbf{f}(\mathbf{z}) = \mathbf{0}$ can be readily solved by Newton's iteration

$$\mathbf{z} \leftarrow \mathbf{z} - \mathbf{J}^{-1}\mathbf{f}(\mathbf{z}),$$

where $\mathbf{J} = \partial\mathbf{f}/\partial\mathbf{z}$. The matrix-based formulation of the equations facilitates the derivation of the formula for the jacobian, and allows it to be written (and coded) compactly as

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{11} & \mathbf{J}_{12} \\ \mathbf{I}_{1,:} & \mathbf{0} \\ \frac{2a}{L-l}\mathbf{D}_{1,:} & \frac{2a}{(L-l)^2}\mathbf{D}_{\mathbf{i},:}\mathbf{y} \\ \mathbf{I}_{N,:} & \mathbf{0} \end{bmatrix}$$

with

$$\begin{aligned} \mathbf{J}_{11} &= \frac{2a}{L-l}\mathbf{D}_{\mathbf{i},:} + \frac{4}{c(L-l)^2}\mathrm{diag}(\mathbf{D}_{\mathbf{i},:}\mathbf{y})\mathbf{D}_{\mathbf{i},:} \\ &\quad + \frac{2\sigma^2}{(L-l)^2}\mathbf{D}^{(2)}_{\mathbf{i},:} - r\mathbf{I}_{\mathbf{i},:} \\ \mathbf{J}_{12} &= \tfrac{1}{2}(1-\mathbf{x}_\mathbf{i}) + \frac{2a}{(L-l)^2}\mathbf{D}_{\mathbf{i},:}\mathbf{y} \\ &\quad + \frac{4}{(L-l)^3 c}(\mathbf{D}_{\mathbf{i},:}\mathbf{y}) \odot (\mathbf{D}_{\mathbf{i},:}\mathbf{y}) + \frac{4\sigma^2}{(L-l)^3}\mathbf{D}^{(2)}_{\mathbf{i},:}\mathbf{y}. \end{aligned}$$

Using initial values

$$l = 0, \quad \mathbf{y} = \frac{1}{2}(\mathbf{x}+1)\left(\frac{a}{r^2} + \frac{1}{2cr^3} + \frac{L}{r}\right)$$

to define initial values of $\mathbf{z}$ for the iteration, the Newton iterations typically converge to machine precision in 5 iterations.

Because the solution is smooth, the spectral collocation method is much more efficient than the finite difference method (Figure 3). However, the performance of the finite difference method used here is acceptable for moderate precision levels, and is orders of magnitude faster than the finite difference method used by Dangl and Wirl (2004).

5 CONCLUDING REMARKS

In this paper we presented three case studies to illustrate the differentiation matrix formalism as a tool for developing discretised models for financial differential equations in one state variable. The use of a high level notation that closely resembles the original differential equation was exploited to rapidly produce effective solution code a modern matrix-oriented scientific programming language. The

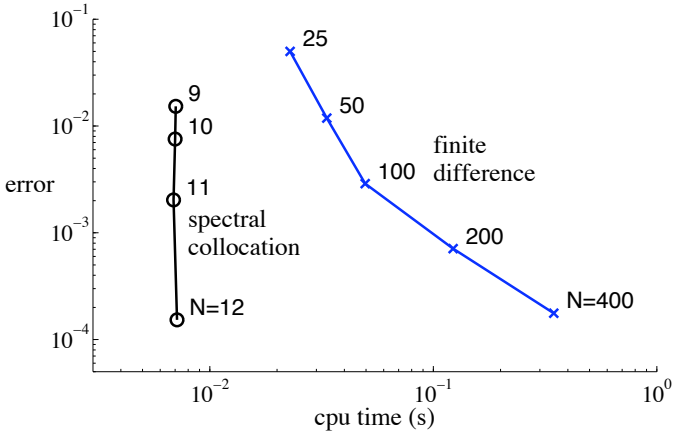

Figure 3 Error vs. computing time for the optimal maintenance and shutdown problem with $a = -0.1$, $\sigma = 0.2$, $r = 0.1$, $c = 200$, $L = 10$. The error is that of the critical level l, using as reference the value $l = -0.1794460361577$ computed using spectral collocation with $N = 50$.

unified formulation facilitated numerical comparison of different state-variable discretisations (finite difference vs. spectral collocation) and time-stepping schemes (Alexander vs. Crank-Nicolson).

The Alexander scheme deserves to be better known in computational finance, as it provides better stability and accuracy with essentially the same numerical complexity as the widely-used Crank-Nicolson method. The spectral collocation discretisation also deserves wider attention, at least for problems with smooth solutions such as optimal stopping models or perpetual options. For problems with non-smooth terminal conditions (e.g. time-dependent option pricing models) or inconsistent boundary conditions (e.g. interest rate models), however, finite difference methods are generally more effective.

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 using other basis functions (radial, sinc, Fourier, Laguerre, etc.), higher order finite differences, finite elements, or finite volumes.

Differentiation matrix modelling can also be applied to problems with two state variables, such as the stochastic volatility model of Heston (1993). The numerical solution is represented as an array $\mathbf{y} = y(\mathbf{x}\cdot\mathbf{1}^T, \mathbf{1}\cdot\mathbf{w}^T)$ of values on a tensor product grid, and the partial derivatives are approximated by

$$\frac{\partial^{d_1+d_2} y}{\partial x^{d_1} \partial y^{d_2}} \approx \mathbf{D}^{(d_1)} \mathbf{y} \mathbf{D}^{(d_2)T}.$$

With dimension higher than two one could use multilinear algebra, but this is not widely supported in current scientific computing software.

References

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

F. Black & M. Scholes, The pricing of options and corporate liabilities. *J. Political Economy*, **81**, 1973, 637–654.

F. O. Bunnin et al., Pseudospectral symbolic computation for financial models, International Mathematica Symposium, 1999.

G. M. Caporale & M. Cerrato, Chebyshev polynomial approximation to approximate partial differential equations, University of Glasgow Department of Economics Working Paper, 2008.

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.

D. J. Duffy, *Finite Difference Methods in Financial Engineering: A Partial Differential Equation Approach*, Wiley, 2005

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.

S. Ikonen & J. Toivanen, Pricing American options using LU decomposition, *Applied Mathematical Sciences*, **1** (51), 2007, pp. 2529–2551.

D. Tavella & C. Randall, *Pricing financial instruments, the finite difference method*, John Wiley & Sons, New York, 2000.

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

D. A. Voss et al., A fourth order-stable method for Black-Scholes model with barrier options, *Computational Science and Its Applications – ICCSA 2003*, Springer Lecture Notes in Computer Science, **2669**, 2003, 199–207.

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

A.1 Finite differences

The central difference formula for approximating the first derivative of $y : [a, b] \to \mathbb{R}$ is

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

The one-sided difference formulas

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

have the same order of accuracy as the central difference formula. Let $\mathbf{x}_i = a + (i-1)(b-a)/N$ for $i = 1, 2, \ldots N$ be a grid of equally spaced points in $[a, b]$, 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

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

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 $N-1$ in domain $x \in [-1, 1]$ 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, as well as codes for interpolation of the solution, are given by Weideman and Reddy (2000). Differentiation matrices corresponding to functions on domain $\xi \in [a, b]$ are obtained by scaling by the factor $(\frac{2}{b-a})^d$.