Res. Lett. Inf. Math. Sci., 2005, Vol.**7**, pp171-186
Available online at http://iims.massey.ac.nz/research/letters/

Evaluation of numerical integration schemes for a partial integro-differential equation

A. Elvin & C. Laing

Institute of Information & Mathematical Sciences
Massey University at Albany, Auckland, New Zealand.

Numerical methods are important in computational neuroscience where complex nonlinear systems are studied. This report evaluates two methodologies, finite differences and Fourier series, for numerically integrating a nonlinear neural model based on a partial integro-differential equation. The stability and convergence criteria of four finite difference methods is investigated and their efficiency determined. Various ODE solvers in Matlab are used with the Fourier series method to solve the neural model, with an evaluation of the accuracy of the approximation versus the efficiency of the method. The two methodologies are then compared.

---

1 Introduction

Some mathematical systems of equations, given their complexity, must be solved using numerical computation techniques. This report explores the application of two such techniques – finite difference methods and the Fourier series method.

Finite difference methods can be used to numerically solve an equation such as a PDE by discretising both time and space into a grid with equal time steps and equal spatial steps, then replacing derivatives with finite difference approximations. These derivative approximations can be calculated in different ways, resulting in methods with different criteria for stability and different truncation errors. We are interested in evaluating the suitability of each method.

First, we introduce the heat equation as an often used example in the literature for explaining the method of grid construction which permits the use of finite difference

methods. We then outline the nonlinear neural model we are interested in.

The explicit, fully implicit and Crank-Nicolson finite difference methods, along with a hybrid method we develop, are used to numerically integrate the nonlinear model. The performance of each method is assessed in terms of stability, convergence to the true solution, and the computing effort required.

The Fourier series method is then used to numerically solve the model. Various ODE solvers within Matlab, for both stiff and non stiff problems, are applied and their efficiency assessed in terms of the desired accuracy of the Fourier series approximation.

The results obtained will be of assistance in refining the approach to the numerical integration of various nonlinear models. We now introduce the heat equation and show how a spatial/temporal grid is constructed.

2 The heat equation

Assume we have a thin bar of length L that is given an initial temperature and insulated everywhere except the ends. We wish to see how the temperature of the rod varies over time.

Let x be the spatial variable representing a point along the rod so that $0 < x < L$. Let $u(x,t)$ be the temperature of the bar at position x at time t.

The partial differential equation

$$\frac{\partial u}{\partial t} = \frac{C^2}{L^2}\frac{\partial^2 u}{\partial x^2}$$

is used to model a one dimensional temperature progression where C/L represents the thermal diffusivity of the rod. Thermal diffusivity depends upon the thermal conductivity, density and specific heat capacity of the rod. To simplify our calculations we choose $L = C$ so that the thermal diffusivity of the rod is 1.

The initial condition is

$$u(x,0) = f(x)$$

where $f(x)$ is the temperature at position x at time $t = 0$.

We assume the ends of the rod are kept at a fixed temperature of zero which gives homogeneous Dirichlet boundary conditions of

$$u(0,t) = u(L,t) = 0 \quad \text{for all } t > 0.$$

We therefore wish to solve for the function $u(x,t)$ such that

$$\begin{aligned} \frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2} \\ u(0,t) &= u(L,t) = 0 \quad \text{for all } t > 0 \\ u(x,0) &= f(x). \end{aligned} \tag{1}$$

In order to do this, we discretise over space by dividing the length L into n equal intervals of length h such that

$$\Delta x = \frac{L}{n}$$

and write $x_i = i\Delta x = ih$ for $i = 0, 1, \ldots, n$.

To find the temperature of the bar at time T we discretise time T into m steps of size k such that

$$\Delta t = \frac{T}{m}$$

and write $t_j = j\Delta t = jk$ for $j = 0, 1, \ldots, m$.

A grid of $(n+1)$ spatial points by $(m+1)$ time points can now be constructed. Letting $v_{i,j}$ be the approximate value of u at (x_i, t_j) in (1), the initial and boundary conditions mean that $v_{i,0}, v_{0,j}$ and $v_{n,j}$ are known for all i, j. This method of discretisation produces a problem containing a finite number of function points to evaluate which allows us to approximate spatial and time derivatives with finite difference quotients and solve for $v_{i,m}$.

A forward difference scheme, backward difference scheme, or average of the two can be used, each with different stability criteria and/or truncation errors, can be used to approximate the spatial and time derivatives. In the next sections, a neural model is presented and finite difference methods are used to find a solution.

3 Overview of a neural model

Consider the one dimensional nonlinear neural model for the propagation of electrical activity in neural tissue

$$\frac{\partial u(x,t)}{\partial t} = K\frac{\partial^2 u}{\partial x^2} - u(x,t) + \int_{-\infty}^{\infty} w(x-y) f(u(y,t))\, dy, \tag{2}$$

where

$$w(x) = e^{-b|x|}(b\sin(|x|) + \cos(x)), \tag{3}$$

$$f(u) = 2e^{-r/(u-th)^2} H(u-th). \tag{4}$$

In this model we have parameters $K, b, th, r > 0$, function H represents the Heaviside function, and $u(x,t) : \mathbb{R} \times \mathbb{R}^+ \to \mathbb{R}$.

A spatially homogeneous and isotropic system is assumed, where $u(x,t)$ is a neural field representing the local activity of a population of neurons at position $x \in \mathbb{R}$ at time t, $w(y)$ the strength of the connection or coupling between neurons separated by distance y, and $f(u(y,t))$ the firing rate function of the neuron at position $y \in \mathbb{R}$ at time t. It is assumed that the synaptic input current is a function of the presynaptic firing rate function, with the nonlinear term on the right in (2) representing the synaptic input.

Further discussion of this model, with $K = 0$, can be found in Laing et al (2002). For $K > 0$, a preliminary investigation of the model is presented in Elvin (2004).

We will now find stable steady state solutions of (2) through numerical integration, using finite difference methods, then a Fourier series approximation of the solution.

4 Finite difference methods

To find steady state solutions of (2) using finite difference methods, we take a finite domain of $x \in [-15\pi, 15\pi]$ and set an initial excitation profile of

$$u(x,0) = 2\cos\left(\frac{Lx}{15\pi}\right)\exp\left(-\left(\frac{Lx}{15\pi}\right)^2\right) \tag{5}$$

The parameters are set at $L = 3, b = 0.25, r = 0.095$ and $th = 1.5$ for all numerical work in this report. We also have Dirichlet boundary conditions of

$$u(-15\pi, t) = u(15\pi, t) = 0.$$

It is noted that there are many steady state solutions to this model and the size of the time steps will influence which steady state is found. The desired stable steady state solution for these methods is a three bump steady state and this acts as a constraint upon the time step.

We wish to compare the efficiency and accuracy of four different finite difference methods in order to gain insight into the most appropriate approach to solving the neural model numerically. The derivation of the methods will not be covered here as it is explained in numerous mathematical texts, with a good account being found in Kincaid and Cheney (2001), Cooper (1998), and Rade and Westergren (2004). In particular, Kincaid and Cheney provide a thorough stability and convergence

analysis.

When evaluating the stability and convergence criteria of each finite difference methods being considered for the model in (2), we will first consider the stability criteria for the heat equation, as this is more easily evaluated and provides a useful comparison.

4.1 Explicit method

To apply the explicit method to the heat equation, we replace the derivatives in (1) by forward difference approximations, therefore,

$$u_t = \frac{v_{i,j+1} - v_{i,j}}{k} + \mathbf{O}(k)$$

and

$$u_{xx} = \frac{v_{i+1,j} - 2v_{i,j} + v_{i-1,j}}{h^2} + \mathbf{O}(h^2)$$

where v_{ij} is the approximation to $u(x_i, t_j) = u(ih, jk)$.

We solve for $v_{i,j+1}$ for $i = 1, 2, \ldots, n-1$ with $v_{0,j} = v_{n,j} = 0$ and obtain the scheme

$$v_{i,j+1} = (1 - 2s)v_{i,j} + s(v_{i+1,j} + v_{i-1,j}) \tag{6}$$

where $s = k/h^2$ for time step k and spatial step h. This is unstable unless the coefficient of $v_{i,j}$ is nonnegative and provides a stability criterion of $s \leq 1/2$.

To find the stability of the method for the neural model in (2), the nonlinear term formed by the integral

$$\int_{-\infty}^{\infty} w(x-y) f(u(y,t))\, dy$$

is ignored and after replacing the derivatives by forward difference approximations, we obtain the scheme

$$v_{j+1} = (1 - k - 2sK)v_{ij} + sK(v_{i+1,j} + v_{i-1,j}).$$

This gives the matrix equation $V_{j+1} = AV_j$ where $A = (1-k)I - sKB$ and

$$B = \begin{pmatrix} 2 & -1 & 0 & \ldots & 0 \\ -1 & 2 & -1 & \ldots & 0 \\ 0 & -1 & 2 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 2 \end{pmatrix}. \tag{7}$$

The method is stable if $1-k-2sK \geq 0$, therefore the condition for stability (ignoring the effect of the nonlinear term) is

$$k \leq \frac{h^2}{h^2 + 2K}$$

with truncation error of $\mathbf{O}(k) + \mathbf{O}(h^2)$. The stability condition demonstrates the inverse relationship between the size of the time step, k, and the coefficient of the spatial derivative, K.

To solve (2) numerically, the nonlinear term is evaluated at step j (the previous time step) using the trapezoid rule, as this is very efficient and increasingly accurate as h becomes smaller.

4.2 Fully implicit method

To construct the fully implicit method, we centre the difference quotient for the second spatial derivative in (1) at (x_i, t_{j+1}), rather than at (x_i, t_j), giving

$$u_{xx} = \frac{v_{i+1,j+1} - 2v_{i,j+1} + v_{i-1,j+1}}{h^2} + \mathbf{O}(h^2). \tag{8}$$

This creates the scheme

$$(1 + 2s)v_{i,j+1} - s(v_{i+1,j+1} + v_{i-1,j+1}) = v_{i,j} \tag{9}$$

for $i = 1, 2, \ldots, n-1$ with $v_{0,j} = v_{n,j} = 0$, which is stable for all $s \geq 0$.

To find the stability condition for (2), we ignore the nonlinear term again and replace the second spatial derivative with the difference approximation of (8), giving the scheme of

$$(1 + k + 2sK)v_{i,j+1} - sK(v_{i+1,j+1} + v_{i-1,j+1}) = v_{i,j}.$$

With B defined as in (7), we write $A = (1 + k)I + sKB$ and obtain the matrix equation $AV_{j+1} = V_j$. Given the eigenvalues of B are $(1 - \cos(\theta_i))$, the eigenvalues of A are

$$\lambda_i = 1 + k + 2sK(1 - cos(\theta_i)) > 1,$$

therefore the eigenvalues of A^{-1} are less than 1 and the fully implicit method is stable for all $s \geq 0$, with truncation error of $\mathbf{O}(k) + \mathbf{O}(h^2)$.

However, once the nonlinear term

$$\int_{-\infty}^{\infty} w(x - y) f(u(y, t))\, dy$$

is included, the system is no longer one of linear equations and Newton's method must be used to solve the function g, say, such that

$$g(V_{j+1}) = A\left[V_{j+1} + \frac{X_{j+1}}{2}\right] - \left[V_j + \frac{X_j}{2}\right] = 0$$

where X_j is the nonlinear term at the j^{th} time level. The stability of the method is now constrained by the Newton criteria for convergence and is therefore not stable for all $s \geq 0$ for (2).

4.3 Crank-Nicolson method

By combining the explicit and implicit schemes, the accuracy of the approximation can be improved and larger step sizes in both spatial and time directions can be taken. We do this by replacing the second spatial derivative with the mean of its finite difference representations at the j^{th} and $(j+1)^{th}$ time levels

$$u_{xx} = \frac{v_{i+1,j+1} - 2v_{i,j+1} + v_{i-1,j+1} + vv_{i+1,j} - 2v_{i,j} + v_{i-1,j}}{2h^2} + \mathbf{O}(h^2). \tag{10}$$

This gives the Crank-Nicolson finite difference scheme for the heat equation in (1) of

$$-sv_{i-1,j+1} + (1+2s)v_{i,j+1} - sv_{i+1,j+1} = sv_{i-1,j} + (1-2s)v_{i,j} + sv_{i+1,j} \tag{11}$$

for $0 \leq i \leq n+1$ and $j \geq 0$, which is stable for all $s \geq 0$.

To apply to our neural model, ignoring the nonlinear term, the second spatial derivative in (2) is approximated by (10), resulting in a difference scheme of

$$\begin{aligned}(2+k+2sK)v_{i,j+1} - sK(v_{i+1,j+1} + v_{i-1,j+1}) \\ = (2-k-2sK)v_{i,j} + sK(v_{i+1,j} + v_{i-1,j}).\end{aligned} \tag{12}$$

Using the matrix B in (7), we write (12) in matrix form as

$$((2+k)I + sKB)V_{j+1} = ((2+k)I - sKB)V_j$$

or

$$V_{j+1} = ((2+k)I + sKB)^{-1}((2+k)I - sKB)V_j. \tag{13}$$

Defining the spectral radius of an $n \times n$ matrix A containing real or complex elements with eigenvalues $\lambda_1, \ldots, \lambda_n$ as

$$\rho(A) = \max_{1 \leq i \leq n} |\lambda_i|,$$

we then say the system (13) is stable if

$$\rho\left(\left((2+k)I + sB\right)^{-1}\left((2+k)I - sB\right)\right) < 1. \tag{14}$$

Letting w_i be the eigenvalues of B, this becomes the stability requirement

$$\mid \left((2+k)I + sw_i\right)^{-1}\left((2+k)I - sw_i\right) \mid < 1.$$

Given $w_i = 2(1-\cos\theta_i)$ (see Kincaid and Cheney), we have $0 < w_i < 4$, therefore (14) holds and the Crank-Nicolson method is stable for all $s \geq 0$ for this neural model (when the nonlinear term is not included), with truncation error of $\mathbf{O}(k^2) + \mathbf{O}(h^2)$.

Once again, inclusion of the nonlinear integral creates a nonlinear system of equations which must be solved using Newton's method, creating a constraint on s, therefore we can no longer claim stability for all $s > 0$.

4.4 Hybrid method

The need to use Newton's method in the fully implicit and Crank-Nicolson methods makes these methods much slower than the explicit method. To speed up the Crank-Nicolson method, we take the nonlinear term at the j^{th} time level only, creating the linear system of equations

$$AV_{j+1} = V_j + X_j = \hat{V}_j$$

where X_j is the nonlinear term at the j^{th} time level. Newton's method is now not required and this hybrid method can be solved effectively using LU decomposition. This has speed comparable to the explicit method with stability and truncation error similar to that of the Crank-Nicolson method. LU decomposition is used to calculate matrix inversions where possible which can make this method very fast.

4.5 Results

These four methods are used to numerically solve the neural model in (2). The restriction on the valid range of s for each method is still determined by the steady state we wish to converge to as larger spatial steps will converge to different stable steady states. Given that the fully implicit and Crank-Nicolson methods use Newton's method, there is an upper bound on s to guarantee convergence to the desired steady state.

Four computer programs were written in Matlab to numerically integrate the neural model with an identical initial condition over 40 time units for various uniformly discretised spatial grids. The maximum time step was used for each spatial step

size, that is, the maximum value of s was used that would converge to the accurate solution. Each method was evaluated for both a small and large value of K. The log of CPU time versus the log of the number of spatial points for $K = 0.05$ was graphed in Fig. 1.

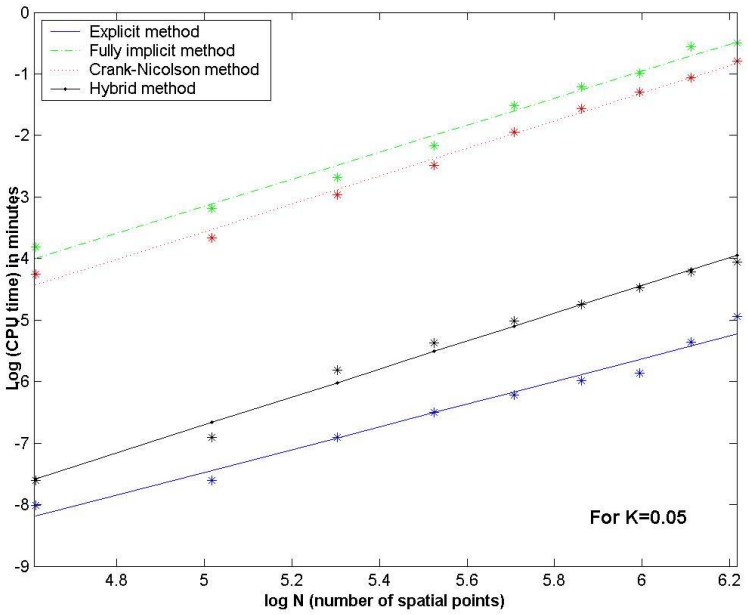

Figure 1: Results of numerically integrating the neural model for $K = 0.05$.

It can be seen that the explicit and hybrid methods are much faster than the implicit and Crank-Nicolson methods due to the use of LU decomposition rather than Newton's method. The Crank-Nicolson method is always faster than the fully implicit method by a constant factor. Even though the strict stability criterion of the explicit method means that the time step taken must be very small relative to the spatial step, the explicit method is still faster than all other three methods.

The disadvantage of the explicit method in terms of the nonlinear neural model is the severe constraint in the size of the coefficient of the second spatial derivative, K. As K increases, the time step used for the explicit method reduces rapidly in order to maintain stability. We are therefore interested in any advantage the other three methods might offer here. To investigate this we compute the model again using $K = 0.45$ in (2) as this is close to the maximum value of K where stable 3-bump steady state solutions of the model still exist, as discussed in Elvin (2004). The results can be seen in Fig. 2.

The advantage of a method which only needs carry out a single matrix inversion is

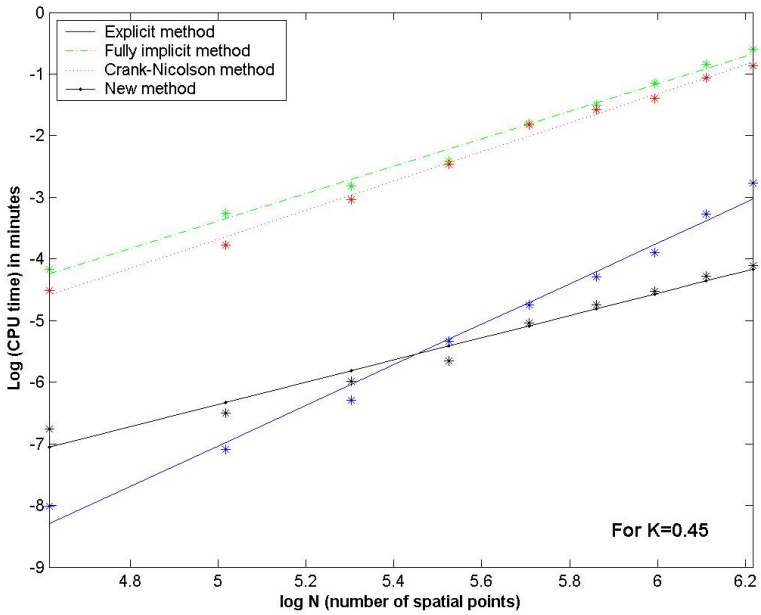

Figure 2: Results of numerically integrating the neural model for $K = 0.45$.

clear when compared to those methods using Newton's method. Surprisingly, for larger K, the time step size for the fully implicit and Crank-Nicolson methods stay relatively unchanged and the hybrid method becomes slightly more efficient. The explicit method is fastest for small N, however, as N increases past a threshold, the time step for the explicit method becomes rapidly smaller and the hybrid method becomes the better performer.

5 Fourier series method

We now investigate the trigonometric approximation of the solution of the neural model using the Fourier series which consists of sine and cosine functions. The theory and techniques of Fourier series is examined in detail in Gowar and Baker (1974). We are looking for a solution that is even about some point in space. As the system is translationally invariant, we can choose that point to be $x = 0$ and thus represent the solution by a cosine Fourier series.

5.1 Theory of Fourier series

Consider an even function f which is periodic over the domain $[-\pi, \pi]$. We wish to find the finite Fourier series for f and therefore write the approximation to f over

$x \in [-\pi, \pi]$ as

$$\hat{f}(x) = a_0 + \sum_{i=1}^{M} a_i \cos(ix)$$

where the a_i for $i = 0, 1, \ldots, M$ are the Fourier coefficients. If the necessary integrations can be carried out, these M Fourier coefficients are found by evaluating

$$\begin{aligned} a_0 &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x)dx \quad \text{and} \\ a_i &= \frac{2}{\pi} \int_{-\pi}^{\pi} f(x) \cos(ix)dx \quad \text{for} \quad i = 1, 2, \ldots, M. \end{aligned} \tag{15}$$

Gowar and Baker define a successful approximation as that where the distance between the given function f and its Fourier series $\hat{f}$ is small, that is,

$$\| f - \hat{f} \| < \epsilon, \qquad \epsilon > 0. \tag{16}$$

The number of Fourier coefficients, M, is chosen to satisfy (16) for small ϵ.

5.2 Application to the neural model

Consider the neural model in (2) where $x \in (-\pi, \pi)$. Let

$$\hat{u}(x) = a_0 + \sum_{i=1}^{M} a_i \cos(ix)$$

be an approximation to u.

After scaling to the smaller domain, the coupling function in (3) becomes

$$w(x) = e^{-15b|x|}[b \sin(15|x|) + \cos(15x)]$$

and its Fourier series is found by using the formulae in (15) such that

$$\hat{w}(x) = b_0 + \sum_{i=1}^{M} b_i \cos(ix),$$

where

$$\begin{aligned} b_0 &= \frac{1}{\pi} \int_{-\pi}^{\pi} w(x)dx \quad \text{and} \\ b_i &= \frac{2}{\pi} \int_{-\pi}^{\pi} w(x) \cos(ix)dx \quad \text{for} \quad i = 1, 2, \ldots, M. \end{aligned} \tag{17}$$

Therefore

$$\hat{w}(x-y) = b_0 + \sum_{i=1}^{M} b_i \cos(i(x-y))$$

and using a trigonometric identity this becomes

$$\hat{w}(x-y) = b_0 + \sum_{i=1}^{M} b_i[\cos(ix)\cos(iy) + \sin(ix)\sin(iy)].$$

Calculating the required partial derivatives and making the substitutions for the functions u and w, the partial integro-differential equation (2), becomes

$$\begin{aligned}\dot{a}_0 + \sum_{i=1}^{M} \dot{a}_i \cos(ix) &= -a_0 - \sum_{i=1}^{M} a_i \cos(ix) - K \sum_{i=1}^{M} i^2 a_i \cos(ix) \\ &+ \int_{-\pi}^{\pi} \left(b_0 + \sum_{i=1}^{M} b_i[\cos(ix)\cos(iy) + \sin(ix)\sin(iy)] \right) f\left(a_0 + \sum_{i=1}^{M} a_i \cos(iy) \right) dy\end{aligned}$$

which is rearranged to become

$$\begin{aligned}\dot{a}_0 + \sum_{i=1}^{M} \dot{a}_i \cos(ix) = -a_0 &- \sum_{i=1}^{M} a_i \cos(ix) - K \sum_{i=1}^{M} i^2 a_i \cos(ix) \\ &+ \int_{-\pi}^{\pi} b_0 f\left(a_0 + \sum_{i=1}^{M} a_i \cos(iy) \right) dy \\ &+ \sum_{i=1}^{M} b_i \cos(ix) \int_{-\pi}^{\pi} \cos(iy) f\left(a_0 + \sum_{i=1}^{M} a_i \cos(iy) \right) dy \\ &+ \sum_{i=1}^{M} b_i \sin(ix) \int_{-\pi}^{\pi} \sin(iy) f\left(a_0 + \sum_{i=1}^{M} a_i \cos(iy) \right) dy.\end{aligned}$$

Multiplying by $\cos(jx)$ and integrating over $[-\pi, \pi]$ for $j = 0, 1, \ldots, M$, we obtain the following nonlinear system of ODEs

$$\dot{a}_j = -a_j(1 + Kj^2) + \int_{-\pi}^{\pi} \cos(jy) f\left(a_0 + \sum_{i=1}^{M} a_i \cos(iy) \right) dy. \tag{18}$$

Here we can write a as a column vector of the $(M+1)$ Fourier coefficients, $a = [a_0, a_1, \ldots, a_M]^T$.

Let $J = [-1, -(1+K), -(1+4K), \ldots, -(1+M^2K)]^T$ and c be a column vector such that its j^{th} term is

$$\int_{-\pi}^{\pi} \cos(jy) f\left(a_0 + \sum_{i=1}^{M} a_i \cos(iy) \right) dy.$$

The system in (18) can now be expressed in vector form as

$$\dot{a} = J \cdot a + c \tag{19}$$

where $J \cdot a$ represents element by element multiplication.

5.3 Results

A program was written in Matlab to apply an ODE solver to the system of ODEs in (19). In order to evaluate the integral term, the standard Matlab routine for numerical integration, QUADL, which uses adaptive Lobatto quadrature, and the trapezoidal approximation method were evaluated. Under both methods, convergence was obtained, however QUADL was found to be very slow therefore the faster method of trapezoidal approximation method was chosen.

Six different ODE solvers were used for two different values of K, as before, with the number of Fourier coefficients being varied. The larger M, the number of Fourier coefficients in the approximation of u, the more accurate the approximation as given by (16).

For small K (see Fig. 3), it was found that nonstiff ODE solvers are the most effective. ODE45, a one-step solver based upon the Runge-Kutta method, performs best until a threshold value of M is reached where ODE113, a multi-step solver based on the variable Adams-Bashforth-Moulton method, becomes the most effective. As M increases further, ODE23, another Runge-Kutta based solver that may be more effective in the presence of mild stiffness, improves on ODE45 also however ODE113 remains the most efficient. As M increases there is less differentiation between the efficiency of the solvers, except for ODE23s which shows poor performance for all M.

For larger K, (see Fig. 4), there is increasing stiffness as M increases. For small M (a less accurate approximation), ODE113, ODE23 and ODE45 are most effective. Once M exceeds a threshold, ODE15s and ODE23t (ODE solvers suitable for stiff problems) become much more efficient than the nonstiff solvers. Again, ODE23s has very low performance.

6 Discussion

We investigated the stability and convergence conditions of the explicit, fully implicit and Crank-Nicolson finite difference methods. A new hybrid method was developed and all four methods applied to a nonlinear neural model.

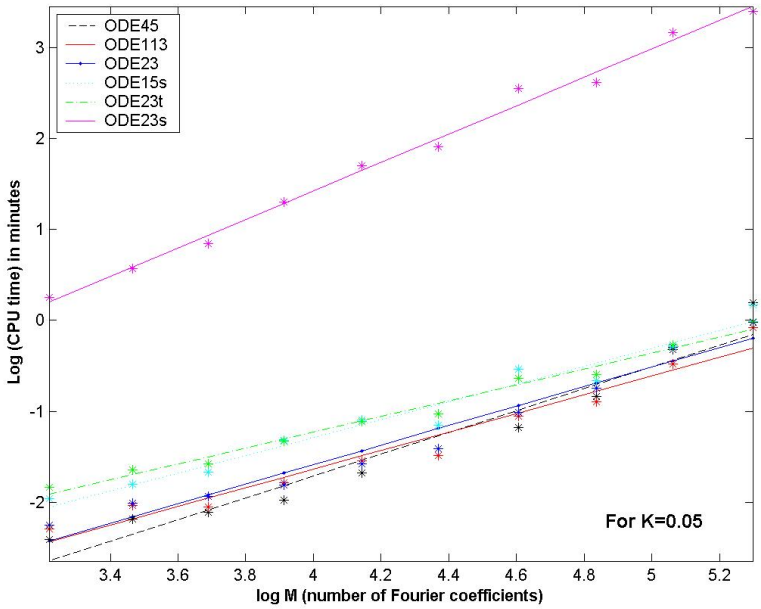

Figure 3: Using Fourier series method to solve the neural model for $K = 0.05$ using various ODE solvers in Matlab.

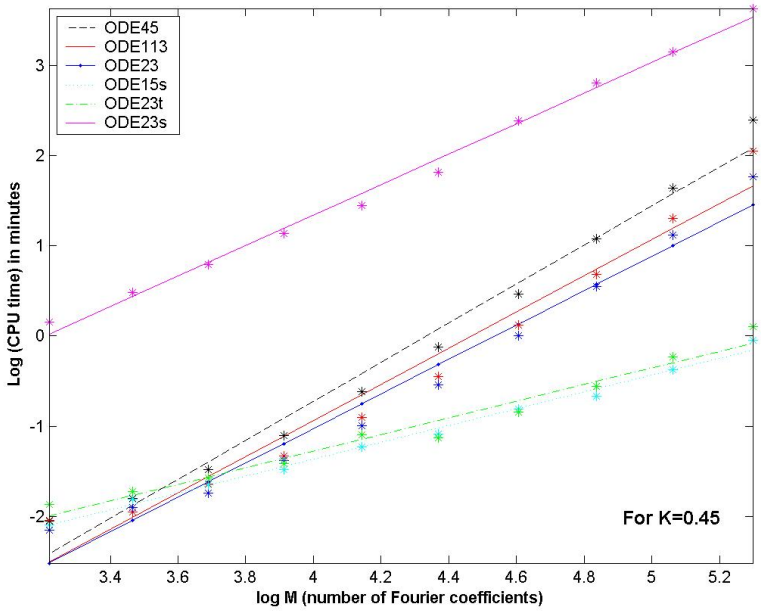

Figure 4: Using Fourier series method to solve the neural model for $K = 0.45$ using various ODE solvers in Matlab.

The explicit and hybrid methods were found to be more efficient than the fully implicit and Crank-Nicolson methods for all N, the number of spatial steps, over both a small and large second spatial derivative coefficient, K, due to the nonlinearity in the neural model. The explicit finite difference method has the advantage of simplicity of programming and for small K is very efficient for all N. For larger K, the hybrid method is the most efficient finite difference method once N passes a threshold. Both the fully implicit and the Crank-Nicolson methods are much less efficient due to the evaluation of the nonlinear term using Newton's method.

We then investigated using a Fourier series approximation to the neural model. It was found that the nonstiff ODE solvers are the most effective in solving the system for small K. As K increases and the stiffness of the system grows, ODE solvers for stiff systems become the most effective. For both small and large K, if a highly accurate approximation is required (a large number of Fourier coefficients), multistep solvers are the most efficient.

There are other considerations, however, when choosing between finite difference or Fourier series methods. The finite difference methods evaluated all require storage of large matrices in order to solve the system which can be a disadvantage if the system is large. The Fourier series method requires less storage but may be perceived as a conceptually more difficult method to implement given the need to evaluate the nonlinear term. Constraints on computer memory may determine the choice of method.

In terms of further work, the finite element method could be evaluated to see if it offers any advantages over the finite difference and Fourier series methods.

This work was part of a Summer Scholar research project in 2004/2005 for Dr Carlo Laing of the Institute of Information and Mathematical Sciences(IIMS), Massey University, Albany campus, Auckland, New Zealand, and was funded by the Massey University Research Fund and IIMS.

References

[1] C.R. Laing, W.C. Troy, B. Gutkin, and G.B. Ermentrout, Multiple Bumps in a Neuronal Model of Working Memory, *SIAM J. Appl. Math*, Vol. 63 (2002), No. 1, pp. 62-97

[2] D. Kincaid and W. Cheney (2001), *Numerical Analysis: Mathematics of Scientific Computing*, 3rd edition, Brooks/Cole

[3] J.M. Cooper (1998), *Introduction to Partial Differential Equations with MATLAB*, Birkhauser

[4] L. Rade and B. Westergren (2004), *Mathematics Handbook for Science and Engineering*, 5th edition, Springer

[5] A. Elvin (2004), PGDipSci Research Project: Pattern Formation in Neural Models

[6] N.W. Gowar and J.E. Baker (1974), *Fourier Series*, Chatto & Windus Ltd with William Collins Sons and Co. Ltd