

Math 166 Project:
**A model of nerve impulse mechanism and the Zeeman Cusp
 Catastrophe**

Due Date: 2pm, Monday, December 6, 2004

You may either work alone or with one partner on this project. You should, however, write up your own report which should be typed and neatly organized. For reference, include a copy of your MATLAB programs. Hand in or E-mail by the above date.

The response of nerves to impulses can be modeled by a combination of Zeeman's cusp catastrophe model with the van der Pol oscillator. To see more about catastrophes, especially of the cusp kind, check out <http://www.ams.org/new-in-math/cover/cusp1.html>.

The model is given by the system of partial differential equations:

$$\frac{\partial y}{\partial t} = -\frac{1}{\epsilon}(y^3 + ay - b) + \sigma \frac{\partial^2 y}{\partial x^2} \quad (1)$$

$$\frac{\partial a}{\partial t} = b - 0.07v + \sigma \frac{\partial^2 a}{\partial x^2} \quad (2)$$

$$\frac{\partial b}{\partial t} = (1 - a^2)b - a + 0.4y - 0.035v + \sigma \frac{\partial^2 b}{\partial x^2} \quad (3)$$

where

$$u = (y - 0.7)(y - 1.3), \quad v = \frac{u}{u + 0.1}, \quad \epsilon = 0.0001, \quad \sigma = \frac{1}{144}$$

1. Divide the spatial interval (i.e., the x -direction) $[0,1]$ into $N = 10$ equal subintervals, with $x_i = 0 + i\Delta x$. Then the model equations above can be written in the form

$$y'_i = -\frac{1}{\epsilon}(y_i^3 + a_i y_i - b_i) + D(y_{i+1} - 2y_i + y_{i-1}) \quad (4)$$

$$a'_i = b_i - 0.07v_i + D(a_{i+1} - 2a_i + a_{i-1}) \quad (5)$$

$$b'_i = (1 - a_i^2)b_i - a_i + 0.4y_i - 0.035v_i + D(b_{i+1} - 2b_i + b_{i-1}) \quad (6)$$

for $i = 1, 2, \dots, N$, where $D = \sigma/\Delta x^2$. Using the periodic boundary conditions (i.e. $x_0 = x_N$), then $y_1 = y_{N+1}$, $y_0 = y_N$, $b_1 = b_{N+1}$, $b_0 = b_N$, $a_1 = a_{N+1}$, $a_0 = a_N$.

How many ordinary differential equations are you solving?

Suppose you are given the initial conditions

$$y(0) = 0 \quad (7)$$

$$a(0) = -2 \cos(2\pi x) \quad (8)$$

$$b(0) = 2 \sin(2\pi x) \quad (9)$$

Write down the algorithm (similar to Eq. (11)) that advances the system of ODEs above using a fourth-order Runge-Kutta method from t_n to t_{n+1} .

Now implement the algorithm and solve the ODEs for $t \in [0, 1.1]$. The choice of time-step is left up to you; report and justify your choice of time-step. Make sure that your answers seem reasonable. Plot your results of y_i versus t , a_i versus t , and b_i versus t , for $i = 1, \dots, N$. Group all your y_i graphs on one plot, all the a_i graphs on one plot, etc. Discuss your results.

2. Write down the algorithm that advances the system of ODEs above using the implicit Trapezoidal method.

To advance from t_n to t_{n+1} , you will need to solve a system of coupled nonlinear equations. Solve these equations using Newton's method. Using the notation in Eqs. (23) and (24), derive the $J_{i,j}$ terms.

Plot similar graphs as above. What time-step did you use? Discuss what you have learned from this exercise.

which can be written as

$$F_1(\vec{y}_n, \vec{y}_{n+1}, \vec{a}_{n+1}, \vec{b}_{n+1}) \equiv \vec{y}_{n+1} - \vec{y}_n - \Delta t \left(-\frac{1}{\epsilon}(\vec{y}_{n+1}^3 + \vec{a}_{n+1}\vec{y}_{n+1} - \vec{b}_{n+1}) + M\vec{y}_{n+1} \right) = 0 \quad (19)$$

$$F_2(\vec{a}_n, \vec{y}_{n+1}, \vec{a}_{n+1}, \vec{b}_{n+1}) \equiv \vec{a}_{n+1} - \vec{a}_n - \Delta t \left(\vec{b}_{n+1} - 0.07\vec{v}_{n+1} + M\vec{a}_{n+1} \right) = 0 \quad (20)$$

$$F_3(\vec{b}_n, \vec{y}_{n+1}, \vec{a}_{n+1}, \vec{b}_{n+1}) \equiv \vec{b}_{n+1} - \vec{b}_n - \Delta t \left((1 - \vec{a}_{n+1}^2)\vec{b}_{n+1} - \vec{a}_{n+1} + 0.04\vec{y}_{n+1} - 0.035\vec{v}_{n+1} + M\vec{b}_{n+1} \right) = 0 \quad (21)$$

The above system of nonlinear coupled equations can be solved using Newton's method. Let \vec{x} be the $3N \times 1$ vector

$$\vec{x} = [\vec{y}; \vec{a}; \vec{b}]$$

Then a Newton iteration gives

$$\vec{x}^{k+1} = \vec{x}^k - J(\vec{x}^k)^{-1}F(\vec{x}^k) \quad (22)$$

where F denotes the $3N \times 1$ vector $[F_1; F_2; F_3]$, and J denotes the $3N \times 3N$ Jacobi matrix

$$J = \begin{bmatrix} \frac{\partial F_{1,1}}{\partial y_{1,n+1}} & \frac{\partial F_{1,1}}{\partial y_{2,n+1}} & \cdots & \frac{\partial F_{1,1}}{\partial y_{N,n+1}} & \frac{\partial F_{1,1}}{\partial a_{1,n+1}} & \frac{\partial F_{1,1}}{\partial a_{2,n+1}} & \cdots & \frac{\partial F_{1,1}}{\partial a_{N,n+1}} & \frac{\partial F_{1,1}}{\partial b_{1,n+1}} & \frac{\partial F_{1,1}}{\partial b_{2,n+1}} & \cdots & \frac{\partial F_{1,1}}{\partial b_{N,n+1}} \\ \frac{\partial F_{1,2}}{\partial y_{1,n+1}} & \frac{\partial F_{1,2}}{\partial y_{2,n+1}} & \cdots & \frac{\partial F_{1,2}}{\partial y_{N,n+1}} & \frac{\partial F_{1,2}}{\partial a_{1,n+1}} & \frac{\partial F_{1,2}}{\partial a_{2,n+1}} & \cdots & \frac{\partial F_{1,2}}{\partial a_{N,n+1}} & \frac{\partial F_{1,2}}{\partial b_{1,n+1}} & \frac{\partial F_{1,2}}{\partial b_{2,n+1}} & \cdots & \frac{\partial F_{1,2}}{\partial b_{N,n+1}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_{1,N}}{\partial y_{1,n+1}} & \frac{\partial F_{1,N}}{\partial y_{2,n+1}} & \cdots & \frac{\partial F_{1,N}}{\partial y_{N,n+1}} & \frac{\partial F_{1,N}}{\partial a_{1,n+1}} & \frac{\partial F_{1,N}}{\partial a_{2,n+1}} & \cdots & \frac{\partial F_{1,N}}{\partial a_{N,n+1}} & \frac{\partial F_{1,N}}{\partial b_{1,n+1}} & \frac{\partial F_{1,N}}{\partial b_{2,n+1}} & \cdots & \frac{\partial F_{1,N}}{\partial b_{N,n+1}} \\ \frac{\partial F_{2,1}}{\partial y_{1,n+1}} & \frac{\partial F_{2,1}}{\partial y_{2,n+1}} & \cdots & \frac{\partial F_{2,1}}{\partial y_{N,n+1}} & \frac{\partial F_{2,1}}{\partial a_{1,n+1}} & \frac{\partial F_{2,1}}{\partial a_{2,n+1}} & \cdots & \frac{\partial F_{2,1}}{\partial a_{N,n+1}} & \frac{\partial F_{2,1}}{\partial b_{1,n+1}} & \frac{\partial F_{2,1}}{\partial b_{2,n+1}} & \cdots & \frac{\partial F_{2,1}}{\partial b_{N,n+1}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_{3,1}}{\partial y_{1,n+1}} & \frac{\partial F_{3,1}}{\partial y_{2,n+1}} & \cdots & \frac{\partial F_{3,1}}{\partial y_{N,n+1}} & \frac{\partial F_{3,1}}{\partial a_{1,n+1}} & \frac{\partial F_{3,1}}{\partial a_{2,n+1}} & \cdots & \frac{\partial F_{3,1}}{\partial a_{N,n+1}} & \frac{\partial F_{3,1}}{\partial b_{1,n+1}} & \frac{\partial F_{3,1}}{\partial b_{2,n+1}} & \cdots & \frac{\partial F_{3,1}}{\partial b_{N,n+1}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_{3,N}}{\partial y_{1,n+1}} & \frac{\partial F_{3,N}}{\partial y_{2,n+1}} & \cdots & \frac{\partial F_{3,N}}{\partial y_{N,n+1}} & \frac{\partial F_{3,N}}{\partial a_{1,n+1}} & \frac{\partial F_{3,N}}{\partial a_{2,n+1}} & \cdots & \frac{\partial F_{3,N}}{\partial a_{N,n+1}} & \frac{\partial F_{3,N}}{\partial b_{1,n+1}} & \frac{\partial F_{3,N}}{\partial b_{2,n+1}} & \cdots & \frac{\partial F_{3,N}}{\partial b_{N,n+1}} \end{bmatrix}$$

where $\partial F_{1,i}/\partial y_j$ denotes the partial derivative of the i -th equation in F_1 (i.e., $F_1(x_i)$) with respect to the j -th component of y (i.e., $y(x_j)$).

Perhaps the easiest way to define J is to split it up into nine $N \times N$ matrices $J_{i,j}$

$$J = \begin{bmatrix} J_{1,1} & J_{1,2} & J_{1,3} \\ J_{2,1} & J_{2,2} & J_{2,3} \\ J_{3,1} & J_{3,2} & J_{3,3} \end{bmatrix} \quad (23)$$

where $J_{i,j}$ is the Jacobi matrix corresponding to the partial derivative of F_i with respect to the j -th variable. (We let y , a , and b be the first, second, and third variables, respectively.) So

$$J_{1,1} = \begin{bmatrix} \frac{\partial F_{1,1}}{\partial y_{1,n+1}} & \frac{\partial F_{1,1}}{\partial y_{2,n+1}} & \cdots & \frac{\partial F_{1,1}}{\partial y_{N,n+1}} \\ \frac{\partial F_{1,2}}{\partial y_{1,n+1}} & \frac{\partial F_{1,2}}{\partial y_{2,n+1}} & \cdots & \frac{\partial F_{1,2}}{\partial y_{N,n+1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_{1,N}}{\partial y_{1,n+1}} & \frac{\partial F_{1,N}}{\partial y_{2,n+1}} & \cdots & \frac{\partial F_{1,N}}{\partial y_{N,n+1}} \end{bmatrix}$$

It is not difficult to see that $J_{1,1}$ (and in fact, all the J 's) is sparse. The (i,j) -th entry of $J_{1,1}$, denoted $J_{1,1}(i,j)$, is

$$J_{1,1}(i,j) = \frac{\partial F_{1,i}}{\partial y_j} = \begin{cases} 1 - \Delta t \left(-\frac{1}{\epsilon}(3y_{j,n+1}^2 + a_{j,n+1}) + \frac{-2\sigma}{\Delta x^2} \right) & j = i \\ \frac{\sigma}{\Delta x^2} & j = i \pm 1 \\ 0 & \text{otherwise} \end{cases} \quad (24)$$

So $J_{1,1}$ is given by

$$J_{1,1} = I - \Delta t \left(-\frac{1}{\epsilon}(3\text{diag}(\vec{y}^2) + \text{diag}(\vec{a})) + M \right)$$

where I denotes the identity matrix and $\text{diag}(\vec{x})$ denotes a diagonal matrix with the vector x as its diagonal. In MATLAB, you can do

```
diag(x);
```

In fact, you can form $J_{1,1}$ using the commands

```
J11 = eye(N) - dt * (-1/eps) * ( 3*diag(y.*y) + diag(a) ) +M);
```

To form the big Jacobi matrix J from its 9 components, you do

```
J = [ J11, J12, J13; J21, J22, J23; J31, J32, J33 ];
```