

Solutions to Problem Set #3: PDEs and Spatial Pattern Formation

Due: Wednesday November 1, 2006

1. Consider an advection equation (already non-dimensionalised) on a finite, bounded 1D domain, $x = [0, 30]$, with no-flux boundary conditions at either $x = 0$ or $x = 30$:

$$\frac{\partial c}{\partial t} = -v \frac{\partial c}{\partial x}, \quad \left. \frac{\partial c}{\partial x} \right|_{x=0} = 0$$

and a finite distribution for initial conditions:

$$c(x, 0) = \begin{cases} 1 & \text{if } 10 \leq x \leq 20 \\ 0 & \text{otherwise} \end{cases}$$

Let $v = 0.5$. Discretize these equations three different ways, using forward, backward and centered differencing for the advection term. Implement each of these discretizations in Mathematica, Java, C or application of your choice using only loops (do not use the built in ODE or PDE solvers). As a start, use $\Delta x = 0.1$, $\Delta t = 0.01$. Let the simulation run for at least 1 second and plot your result at different time points. What happens? Why? What could you do if you didn't know the direction of the flow, or it changed sign and/or magnitude during the simulation?

Solution

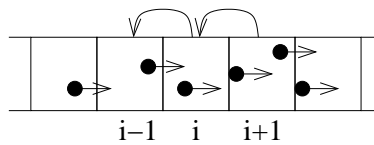
In this question, you use different discretization schemes to look at an advecting square pulse. Sample Matlab code for simulating this system is included at the end of the solutions.

For forward and centered differencing, the discretization is numerically unstable, meaning that no matter how small you take your time step, your solution will always blow up. With backward differencing, your discretization is not unstable, but the shoulders of the square pulse get rounded off as the simulation proceeds. We can understand why the scheme is numerically unstable for forward and centered differencing in this case by considering the flow of particles as shown in Figure 1 (centered differencing is the average of forward and backward differencing). In this equation, particles are always moving from left to right and only backward differencing is properly tracking the direction of flow. With forward

and centered differencing, even though they are perfectly valid mathematical expressions for a first derivative, they are attempting to take particles from in front of the flow and pull them backward. This leads to something called a “transport error”. By using backward differencing, the discretization moves the particles in the same direction as the advective flow and you avoid transport error but internal numerical roundoff by the computer during the computation leads to a smoothing of the shoulders of the square pulse.

When simulating PDEs, determining the proper discretization is often difficult and not obvious. But the proper discretization will make your life much easier and avoid numerical instability.

Forward Differencing



Backward Differencing

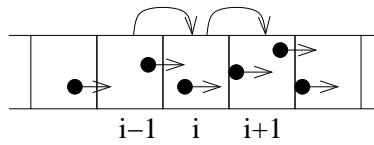
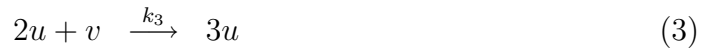


Figure 1: Schematic diagram of forward and backward differencing.

2. The Brusselator This is a model developed by mathematicians in Brussels that has both a Hopf and a Turing bifurcation. The reaction diagram is:



Write down the full model according to these reactions. Assume A , B , C and E are constant and that u and v can diffuse at the rates \overline{D}_u and \overline{D}_v . Non-dimensionalise

the equations to arrive at the following equations for u and v :

$$\frac{\partial u}{\partial t} = a - (b + 1)u + u^2v + D_u \nabla^2 u \quad (5)$$

$$\frac{\partial v}{\partial t} = bu - u^2v + D_v \nabla^2 v \quad (6)$$

Indicate the parameter groupings a , b , D_u and D_v in terms of the original rate constants.

- (a) Consider the space free model (in the absence of diffusion). Under what conditions on b does the model exhibit a Hopf bifurcation? Plot the phase portrait just before and just after this point.
- (b) Consider the full (non-dimensionalised) model. Under what conditions on b does the model exhibit a Turing bifurcation?
- (c) Using the Mathematica worksheet, show what happens as the system passes through the Turing bifurcation point. Print up some of the patterns you find. What is the progression of patterns as you increase the Turing bifurcation parameter?

Solution

- (a) See Problem Set #2, Model 3.
- (b) Testing the three criteria for a Turing bifurcation at the steady state $(a, b/a)$:
 1. $Tr(J) = b - 1 - a^2 < 0$
 2. $Det(J) = a^2 > 0$
 3. $(b - 1)D_v - a^2D_u > 2a\sqrt{D_uD_v}$

These criteria are satisfied when

$$b > \left(1 + a\sqrt{\frac{D_u}{D_v}}\right)^2.$$

Thus, a Turing bifurcation appear at the critical point

$$b_c = \left(1 + a\sqrt{\frac{D_u}{D_v}}\right)^2.$$

- (c) This model has a rich repertoire of spatial patterns that you explored in the Mathematica lab, including spots, stripes and spiral waves.

Challenge Draw the bifurcation diagram for the space free model.

Solution

See Problem Set #2, Model 3.

Sample Matlab code for advection equation that plots the spatial profile of c at every time step.

```
clear
hold off

%variable declaration
v=0.5;

L=30;
dx=0.1;
dt=0.01;
T=10;

N=L/dx;
total=T/dt;

%initial conditions
for i=1:1:N
    if (i*dx>10 && i*dx<20) c(i)=1;
    else c(i)=0;
    end
end

%main time loop
for t=1:1:total

    for i=2:1:N-1

        %forward differencing
        %    cnew(i)=c(i)+dt*(-v*(c(i+1)-c(i))/dx);

        %backward differencing
        cnew(i)=c(i)+dt*(-v*(c(i)-c(i-1))/dx);

        %centered differencing
        %    cnew(i)=c(i)+dt*(-v*(c(i+1)-c(i-1))/(2*dx));
```

```
%no flux boundary conditions
    cnew(1)=cnew(2);
    cnew(N)=cnew(N-1);

%updating the variable
    c=cnew;

    plot(c)
end
```