446-1
Numerical Solution of Partial Differential Equations
Fall 2004
Hermann Riecke

Homework 5

Due Friday December 10, 2000

1. **Burgers Equation with Diffusion**

   Burgers equation describes the dynamics of nonlinear waves for which the wave speed depends on the amplitude of the wave. With an additional diffusion term the equation is given by

   $$\partial_t u = 2\gamma u \partial_x u + D\partial_x^2 u. \tag{1}$$

   Because of the shocks that can occur for $D = 0$ it is better to write the equation in conservation form,

   $$\partial_t u = \partial_x \left(\gamma u^2 + D\partial_x u \right). \tag{2}$$

   Solve (2) numerically with Dirichlet boundary conditions, $u(0,t) = 0$ and $u(L = 2\pi, t) = 0$. Use two different schemes: a Crank-Nicholson scheme of order (2,2) and a (2,2)-MacCormack scheme[1].

   For the Crank-Nicholson treat the nonlinear term by expanding it around the solution at time step t_n, i.e. write $u^{(n+1)} = u^{(n)} + \left\{ u^{(n+1)} - u^{(n)} \right\}$ and expand in $u^{(n+1)} - u^{(n)}$. This scheme will lead to a tridiagonal matrix problem. Within Matlab this is solved without too much difficulty. Make sure you make use of the sparseness of the matrix. Below is a short list of commands and functions you may need for the Matlab code. You probably will have to read in the manual or the online help as well.

   For the MacCormack, there is no need to alternate between the FB- and the BF-version, since the coefficients in the differential equation (2) are constant.

   (a) Validate your codes.

      i. For the diffusion equation

      $$\partial_t u = \partial_x \left(\partial_x u \right) \tag{3}$$

      without the nonlinear term you can make use of the exact solution[2] for the initial condition $u(x,0) = \sin(x/2)$ on the interval $[0, 2\pi]$. Demonstrate the correct convergence of your codes by measuring the error of the numerical solution at $x = \pi$ and $t = 0.95$ for a sequence of suitable values of Δt and of Δx. (Use double-logarithmic plots).

[1] It is in general good practice to write codes in a modular fashion and provide a control menu to change the parameters. For illustration purposes there is a simple program on the class web site that solve the Kuramoto-Sivashinsky equation with forward Euler (see below).

[2] To calculate the exact solution you could use a Fourier ansatz in space and time.

ii. Study numerically also the stability limits of both schemes. How do they depend on Δx? How does this compare with your expectations?

iii. Solve now the full equation (2) using the Crank-Nicholson and the MacCormack scheme for $D = 0.001$ and $\gamma = 1$. Test the implementation of the nonlinear term by showing that the codes have the correct convergence rate: choose as initial condition

$$u(x, 0) = -2\sin(x) + \sin(x/2) \tag{4}$$

on the interval $[0, 2\pi]$ and measure $u(x = \pi, t = 0.1)$ for a suitable sequence of Δt and of Δx.

(b) Use your codes now to study the further evolution from the initial condition (4) for $D = 0.001$ and $\gamma = 1$. Attempt to resolve the solution graphically at the times $t = 0.25$ as well as for $t = 0.3$ and $t = 1$. Comment on the solution. What happens? What makes the resolution of the solution difficult? Which scheme fairs better? Why?

2. **Kuramoto-Sivashinsky Equation**

The Kuramoto-Sivashinsky equation describes among other things flame fronts and the evolution of the wavenumber of a traveling wave. It is given by

$$\partial_t u = D\partial_x^2 u - \partial_x^4 u + u\partial_x u \tag{5}$$

and can be considered to be a regularization of the Burgers equation for the case $D < 0$. As boundary conditions use $u = 0$ and $\partial_x u = 0$ at $x = 0$ and $x = L$.

(a) Solve (5) using a simple (1,2) forward Euler scheme. Also implement a (2,2) Crank-Nicholson scheme (with linearization of the nonlinearity as in the Burgers equation). Validate your codes and perform a convergence study for $D = -0.6$ and system length $L = 4\pi$ by measuring $u(x = \pi, t = 5)$ starting from the initial condition $u(x, 0) = 0.5\,\sin(2\pi\, x/L)$.

(b) Can you reasonably obtain 10%, 1%, and 0.1% accuracy with both schemes?

(c) To get an impression of the dynamics that are possible in the Kuramoto-Sivashinsky equation use the scheme of your choice to solve (5) with random initial conditions for $L = 20\pi$ and $D = -0.6$. Evolve the equation at least till $t_{max} = 100$.

Various commands and functions in Matlab

Taking transposes of matrices and vectors: $Y = y'$
shifted vector for derivatives: $yp = [y(2 : nx)\ y(1)]$
vector of ones: $e = ones(nx, 1)$
vector of zeroes: $z = zeros(nx, 1)$
tridiagonal matrix with rows ....1, −2, 1...: $tridiag = spdiags([e\ -2 * e\ e], -1 : 1, nx, nx)$
the function spdiag creates a matrix in sparse form, i.e. makes use of the sparseness. For comparison (i.e. with respect to speed) you can also try the full matrix version.
identity matrix: $Id = spdiags([e], 0, nx, nx)$
matrix with 1 in an off-diagonal; $spdiag([e], j, nx, nx)$ with $-nx + 1 < j < nx - 1$.
square each entry in the vector Y: $S = Y. * Y$
to solve the matrix equation $Ax = b$: $x = A\backslash b$
where x and b are column vectors (obtained from row vectors by transposition).