Homework 5

May 7, 2004

This homework is due Monday, May 17.

1. Consider the PDE:
$$\begin{cases} u_t + \frac{x}{1-t} u_x = 0 & x \in [-\infty, \infty], t \in [0,1]. \\ u(x,0) = u_0(x) & x \in [-\infty, \infty]. \end{cases}$$
   (a) Show that $u(x,t) = u_0(x - xt)$ is a solution to this PDE.
   (b) What does this tell you about $u(x,1)$?
   (c) This equation is a form of the transport (or 'advection') equation, with variable speed $a(x,t) = x/(1-t)$. What happens to the speed as $t \to 1$? This should explain your answer to the previous part, as we expect the transport equation to model motion of something, and not its dissipation.
   (d) When dealing with difference schemes we are interested in the magnitude of $\nu = a\Delta t/\Delta x$. Bound $|a|$ for $(x,t) \in [-1,1] \times \left[0, \frac{1}{2}\right]$. If you want to ensure that $|\nu| \leq 1$, what relationship do you have to enforce on $\Delta t, \Delta x$?
   (e) We analyzed characteristic curves in the setting of a *vector* PDE; here we consider a scalar PDE. Here is the scalar analogue: the curve $(x,t) = (\xi(t), t)$ for $t \in [t_0, t_f]$, is a characteristic curve of $u_t + a(x,t)u_x = 0$ if and only if
      i. $a(\xi(t), t) = \xi'(t)$ for $t \in [t_0, t_f]$.
      ii. $u(\xi(t), t)$ is constant for $t \in [t_0, t_f]$.

      Show that $\xi(t) = k/(1-t)$ defines a characteristic curve for any k, and for $t_0 = 0, t_f < 1$.
2. Implement the upwind scheme in your favorite programming language to solve the PDE
$$\begin{cases} u_t + \frac{x}{1-t} u_x = 0 & x \in [-\infty, \infty], t \in [0,1]. \\ u(x,0) = \sin(\pi x) & x \in [-\infty, \infty]. \end{cases}$$
   On the interval $[-1,1]$ for $t = \frac{1}{2}$.
   - Let $M = 40$, and let $\Delta x = 2/M$. Let $x_i = i\Delta x$ for $x = -20, -19, \ldots, 0, \ldots, 19, 20$.
   - Your program should accept parameter $r = \Delta t/\Delta x$. Thus let $\Delta t = r\Delta x = 2r/M$.
   - Let N be the largest integer less than $1/2\Delta t$. That is,
   $$N = \left\lfloor \frac{M}{4r} \right\rfloor.$$
   - Run your program with $r = 0.1, 0.25, 0.4, 0.48, 0.5, 0.52, 0.75, 1, 2$.
   - Compare your approximate solution to the actual answer, obtained above, at time $t^* =_{\text{df}} N\Delta t$. Note t^* will not always be $\frac{1}{2}$, but should be close, especially when r is smaller.
   - Remember that although the update is explicit, the PDE is time dependant, so if you are going to do an explicit update as a matrix multiply
   $$\boldsymbol{U^{n+1}} = \mathsf{A}\boldsymbol{U^{n}},$$
   you must make A a function of time.

If you are going to implement the code in Matlab, here's one way you could try it. Call the file "upwind.m":

```
function err = upwind(r)
%
% upwind scheme for u_t + x/(1-t) u_x = 0
% accepts parameter r = del t / del x
% outputs maximum absolute error

% initialization
M     = 20;
X     = ((-M:M)/M)';    %make U a column
delt  = 2.0 * r / M;
N     = floor(1.0 / (2.0 * delt));
U     = sin(pi * X);

%figure out how to make A
...

t     = 0;
for i = 1:N
  t   = t + delt;
  U   = (A * U) ./ (1.0 - t);
end

%figure out the actual solution.
...

err   = max(abs(U - Uac));
```

You will have to be careful to figure out the matrix A. Remember that this is an upwind scheme, but that the sign of $a(x, t)$ is rather predictable– it has the same sign as x. Use this in constructing A. If need be, write out what you think the matrix should look like for M small, like $M = 2$. Note that A will be a $(2M + 1) \times (2M + 1)$ matrix, and likely will not be Toeplitz.

This is only one way of doing it. Another way would be to actually do the upwind procedure "long-hand" at each step. This may be more straightforward, but will certainly be slower in Matlab. If you work in C or Fortran, likely you will program the code in this way.