MPO 762 – Problem Set 1 – Due Monday Sept 12 2016

Feel free to use symbolic computation software such as **mathematica** *or the symbolic toolbox in matlab to carry out the algebra. See 2 to see an example with matlab.*

1 Finite Difference Derivation

Use Taylor series to show that the 5th-order finite difference approximation to the first derivative using two different stencils are of the form

$$u_x = \frac{-2\,u_{i-3} + 15\,u_{i-2} - 60\,u_{i-1} + 20\,u_i + 30u_{i+1} - 3\,u_{i+2}}{60\Delta x} - \frac{12}{60}\frac{\Delta x^5}{6!}u_i^{(6)} + \frac{36}{60}\frac{\Delta x^6}{7!}u_i^{(7)} + \cdots (1)$$

$$u_x = \frac{3\,u_{i-4} - 20\,u_{i-3} + 60\,u_{i-2} - 120\,u_{i-1} + 65\,u_i + 12\,u_{i+1}}{60\Delta x} + \frac{24}{60}\frac{\Delta x^5}{6!}u_i^{(6)} - \frac{216}{60}\frac{\Delta x^6}{7!}u_i^{(7)} + (2)$$

Don't solve the 5x5 system by hand. Use a linear algebra tool to do it.

2 Programming discretization of a grid

Write a script that discretizes the interval $x_{\min} \le x \le x_{\max}$ into N equally spaced cells with grid spacing Δx. The function interface should have the form:

```
  [xe,dx]= FDGrid(xmin,xmax,N);
%
% usage is [xe,dx]= FDGrid(xmin,xmax,N);
% dx: uniform grid size
% xe(1:N+1): coordinate of cell edges
% xmin,xmax: Limits of interval
% N: number of cells
%
```

Use the above to discretize the interval $-1 \le x \le 1$ into 1000 cells, calculate the periodic functions on the cell edges

$$u(x) = \tanh\alpha\left(\cos\pi x + \frac{1}{2}\right) + e^{-\cos^2\pi x} \quad (3)$$

$$u_x(x) = \pi\sin\pi x\left\{\alpha\left[\tanh^2\alpha\left(\cos\pi x + \frac{1}{2}\right) - 1\right] + 2\cos\pi x\, e^{-\cos^2\pi x}\right\} \quad (4)$$

and plot the 2 curves using your favorite visualization tool. Use $\alpha = 4$. The plot should look as shown in figure 1.

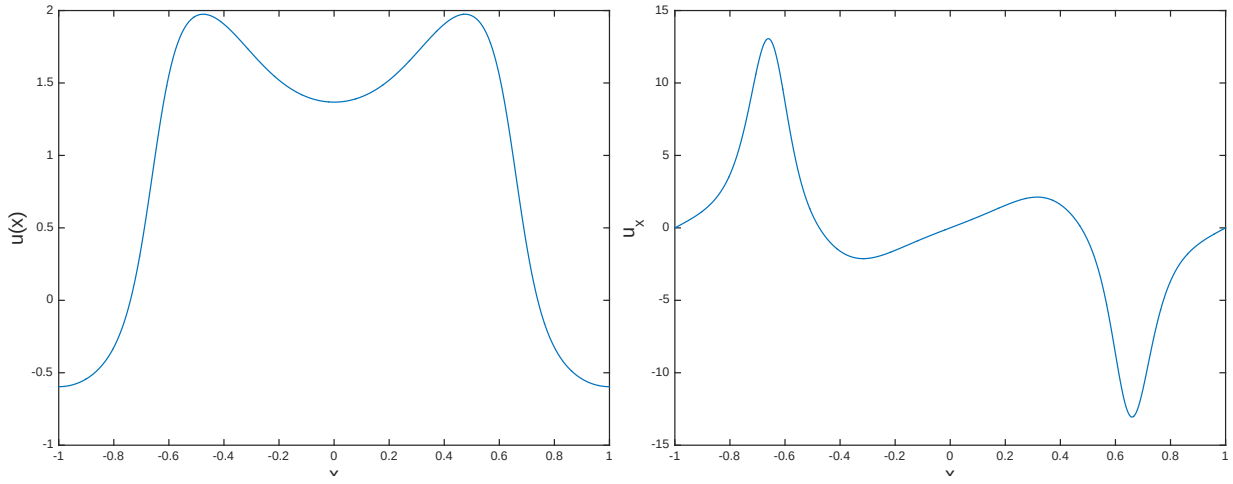

Figure 1: Plots of the function $u(x)$ defined in equation 3 (left panel) and its derivative (right panel).

3 Programming Finite Difference Calculations

Write a program to implement the numerical differentiation formula 1. Apply to the function defined in equation 3. Use the exact derivative, equation 4, to validate your work and to confirm the convergence rates predicted by the Taylor series by drawing the convergence diagram of the root mean square error and maximum error.

Bear the following in mind when writing your code:

- Dealing with the boundaries of the array can be problematic. Use the periodicity of the function to fill-in the values of the *ghost* points.
- Use the following number of *intervals*: 10 20 40 80 100 200 400 800 1600 3200.
- Draw the convergence diagrams and try to decide at what point do you start to see the convergence rate you expect.
- Explain your results, particularly if the convergence curves you see don't conform to your expectation and why.
- Use the polyfit function in matlab/octave to fit your data (the number of cells and the error norm) to a straight line and report the results. If the expected behavior of the truncation error to leading order is $E = A\Delta x^m$, then $\log(E) = m\log(\Delta x) + \log A$, and so the matlab command `polyfit(log(dx),log(E))` will spit out a vector whose first component is the slope, and the second component is $\log A$. Since $\Delta x \sim N^{-1}$, the same thing can be done with the number of points. Make sure you perform the regression on the asymptotic part of the convergence curve to get the result you expect.

- Finally, you are free to use the matlab/octave-scripts provided to perform all or some the calculations. The scripts are written in modular form, and the different steps are easily identifiable:

  1. read in input parameters
  2. initialize relevant variables
  3. perform the computations
  4. compute error metric
  5. output and plot the results.
  6. You can use your favorite tool for visualization if you don't like matlab (recommended) or octave.

4 Drifter Velocity Calculations

The GLAD experiment released a number of drifter equipped with GPS. The drifter's location is available as a time series of latitudes, $\theta(t)$ and longitudes, $\lambda(t)$. The data was pre-processed to filter high frequency noise and to report the coordinates every 15 minutes. Modify your matlab program to read-in the data and calculate the drifter's velocity using the formula:

$$u = R\cos\theta\frac{d\lambda}{dt} \tag{5}$$

$$v = R\frac{d\theta}{dt} \tag{6}$$

Use second and 4th-order centered difference formula to estimate the derivatives of the longitudes and latitudes. Plot the evolution of (u, v) in time for the second and 4th-order estimates. Produce an error measure to decide whether the 2 estimates are in agreement, e.g.

$$\|\epsilon\|^2 = \frac{1}{N-4}\sum_{i=3}^{N-2}\left[\left(u_i^{(2)} - u_i^{(4)}\right)^2 + \left(v_i^{(2)} - v_i^{(4)}\right)^2\right] \tag{7}$$

where $\left(u_i^{(2)}, v_i^{(2)}\right)$ and $\left(u_i^{(4)}, v_i^{(4)}\right)$ are the 2nd and 4-th order estimates. Use $R = 6371km$. Comment on your results and try to explain the differences in the estimates if possible. You may skip the first and last 2 data points to avoid complications. The data is available in ASCII form from blackboard.

```
syms a2 a1 a0 x real     % define symbolic variables
a=[a0 a1 a2]'            % define column vector of symbolic variables
xp=[1 x x^2]             % define row vector of powers of x
f=xp*a                   % dot product of the two vectors
Ic=int(f,'x',-1/2,1/2)   % integrate the function f for -1/2 <= x <=1/2
ue=subs(f,'x',1/2)       % subsitute 1/2 for x in expression f
ux=diff(f,'x')           % derivative of f w.r.t. x
xs=solve(ux,'x')         % solve ux=0 for x, find where derivative is zero
fext=subs(f,'x',xs)      % substitute xs for x in f (parabola extremum)
A=[1 x; x 1]           % define 2x2 symbolic matrix
Ainv=inv(A)              % invert it
Id=simple(A*Ainv)        % verify you get the Identity matrix


B=[-2  1  0  0
    1 -2  1  0
    0  1 -2  1
    0  0  1 -2]          % define a 4x4 matrix
Binv=inv(B)              % define its inverse
S=sym(B)                 % S is a symbolic (exact not numeric) version of B
Sinv=inv(S)              % Its symbolic inverse is found exactly
Sinv*[0 0 0 1]';         % exact matrix vector product
```

Figure 2: Sample symbolic manipulation commands in matlab