Answer Key

CS 6220: Homework 2

Professor Mike Kirby
Problem 1

Page 131 Exercise 2 25 pts

1. Suppose that the points \( x_{j-1}, x_j = x_{j-1} + h_{j-1} \) and \( x_{j+1} = x_j + h_j \) are used to derive a second order formula for \( u_x(x_j) \) which holds even when \( h_{j-1} \neq h_j \). A careless approach, which has actually seen use in the research literature, would suggest the formula

\[
\frac{u_{j+1} - u_{j-1}}{2h_j} = u_x(x_j)
\]

which quickly generalizes the familiar centered three-point formula (3.5b).

Instead consider the formula

\[
\frac{h_j - h_{j-1}}{h_j} u_j + \frac{1}{h_{j-1} + h_j} \left( \frac{h_{j-1}}{h_j} u_{j+1} - \frac{h_j}{h_{j-1}} u_{j-1} \right)
\]

a. Apply the two formulas (3.38) for the function \( u(x) = e^x \) at \( x_j = 0 \) with \( h_{j-1} = .5h, h_j = h \). Record the errors using both methods for \( h = 10^{-l}, l = 1, \ldots, 5 \). 5 pts

b. Derive an error expression for the formula (3.38b), assuming that \( u \) is smooth (i.e., that it has all the derivatives you encounter, bounded). 10 pts

c. It should be apparent from your numerical results that the formula (3.38a) is only first order accurate. But why is this necessarily so? 10 pts

Solution:

Formulating the question more generally, let \( u(x) \) be smooth with \( u_{xx}(x_j) \neq 0 \). Show that the truncation error in the formula (3.38a) with \( h_j = h \) and \( h_{j-1} = h/2 \) must decrease linearly, and not faster, as \( h \to 0 \)

a.

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
\text{error with (3.38a)} & 2.2\text{e-2} & 2.5\text{e-3} & 2.5\text{e-4} & 2.5\text{e-5} & 2.5\text{e-6} \\
\text{error with (3.38b)} & 8.4\text{e-4} & 8.3\text{e-6} & 8.3\text{e-8} & 8.3\text{e-10} & 1.53\text{e-11} \\
\hline
\end{array}
\]

b. In equation (3.38b)

\[
\frac{h_j - h_{j-1}}{h_{j-1} h_j} u_j + \frac{1}{h_{j-1} + h_j} \left( \frac{h_{j-1}}{h_j} u_{j+1} - \frac{h_j}{h_{j-1}} u_{j-1} \right)
\]

Use Taylor series to expand \( u_{j+1} = u_j + h_j u'_j + \frac{h_j^2 u''_j}{2!} + \frac{h_j^3 u'''_j}{3!} + O(h_j^4) \)

\( u_{j-1} = u_j - h_{j-1} u'_j + \frac{h_{j-1}^2 u''_j}{2!} - \frac{h_{j-1}^3 u'''_j}{3!} + O(h_{j-1}^4) \) and simplifying the terms we get the respective coefficients.
If we assume $h_{j-1} = \alpha h_j$, we have $\frac{\alpha h_j^2}{3!} \Rightarrow O(h^2)$

c. In equation 3.38a

$$\frac{u_{j+1} - u_{j-1}}{h_{j-1} + h_j}$$

Use Taylor series to expand $u_{j+1} = u_j + h_ju'_j + h_j^2u''_j + h_j^3u'''_j + O(h_j^4)$

$$u_{j-1} = u_j + h_{j-1}u'_j + \frac{h_{j-1}^2u''_j}{2!} + \frac{h_{j-1}^3u'''_j}{3!} + O(h_{j-1}^4)$$

and simplifying the terms we get the respective coefficients. $u_j : 0$

$u'_j : 1$

$u''_j : \frac{h_j - h_{j-1}}{2!}$

If we assume $h_{j-1} = \alpha h_j$, we have $\frac{(1 - \alpha)h_j}{2} \Rightarrow O(h)$. Thus we can say the error decreases linearly and not faster.
Problem 2

Page 131 Exercise 3 25 pts

Consider the following two methods for deriving an approximate formula for the second derivative $u_x x(x_j)$ of a smooth function $u(x)$ using three points $x_{j-1}$, $x_j = x_{j-1} + h_{j-1}$, and $x_{j+1} = x_j + h_j$, where $h_{j-1} \parallel h_j$.

- Define $w(x) = u_x(x)$ and seek a staggered mesh, centered approximation
  
  
  \[
  w_{j+1/2} = \frac{u_{j+1} - u_j}{h_j}; \quad w_{j-1/2} = \frac{u_j - u_{j-1}}{h_{j-1}}; \quad u_{xx}(x_j) \approx \frac{2(w_{j+1/2} - w_{j-1/2})}{h_{j-1} + h_j}
  \]

  The idea is that all the differences are short (i.e., no long differences for first derivatives) and centered.

- Using the second degree interpolating polynomial in Newton form, differentiated twice, define
  
  \[
  u_{xx}(x_j) \approx 2u[x_{j-1}, x_j, x_{j+1}].
  \]

a. Show that the above two methods are equivalent. 5 pts

b. Show that this method is general is only first order accurate. 10 pts

c. Run the method for the example of Exercise 2 (but for the second derivative of $u(x) = e^x$). What are your findings? Discuss. 10 pts

Solution:

a. Considering the staggered grid approach, we can clearly see that $w_{j+1/2} = u[x_j, x_{j+1}]$ and $w_{j-1/2} = u[x_{j-1}, x_j]$.

\[
\frac{w_{j+1/2} - w_{j-1/2}}{h_{j-1/2} + h_j} = \frac{w_{j+1/2} - w_{j-1/2}}{h_{j-1/2} + h_j} = \frac{u[x_j, x_{j+1}] - u[x_{j-1}, x_j]}{x_{j+1/2} - x_{j-1/2}} = u_{xx}(x_j) \approx 2u[x_{j-1}, x_j, x_{j+1}]
\]

Thus we can say staggered mesh centered approximation is equivalent to an interpolating polynomial in Newton form, differentiated twice.

b. In general case of polynomial interpolation, the interpolation error at any point is

\[
g(x) - \phi(x) = g[x_0, ..., x_q, x] \Pi_{i=0}^q (x - x_i)
\]

and $g[x_0, ..., x_q, x] \approx g_{q+1}(\xi)/(q+1)!$

Refer page 76 of Numerical Methods for Evolutionary Differential Equations by Uri M. Ascher

Differentiating $g(x) - \phi(x)$ twice, we have $g''(x) - \phi''(x) \approx \frac{g_{q+1}(\xi)}{(q+1)!}(4h)$

Therefore the error is $O(h)$. So, the method is first order accurate.
c.

\[ h = 10^{-1} \text{ where } l = \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ \text{Error at } u_{xx}(0) & 1.7e-2 & 1.7e-3 & 1.7e-4 & 1.7e-4 & 8.3e-8 \end{array} \]

We can clearly see the value of \( u_{xx}(0) \) has first order accuracy till \( l = 4 \). For \( l = 5 \) the value is influenced by floating point error.
Problem 3

50 pts

2. For the problem below, solve the problem specified from the text in the “assignment report” format described in the syllabus. Please keep your answer to 10 single-column pages.

Consider the following linear advection-diffusion equation:

\[ u_t + au_x = u_{xx} \]

where \( a \) is a real-valued wave speed and \( \mu \geq 0 \) denotes the diffusion coefficient. For this problem, we will take \( a = 1 \) and \( \mu = 1 \). Assume a domain of \([0, 1)\) with periodic boundary conditions. Take as your initial condition:

\[ u(x, 0) = \sin(2\pi x) \]

Consider the difference approximation given by:

\[ u_j^{n+1} = u_j^n + k(\mu \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{h^2} - a \frac{u_{j+1}^n - u_{j-1}^n}{2h}) \]

for \( N = 10, 20, 40, 80 \) and \( 160 \). Choose the time step \( k \) such that if \( h \) denotes your grid spacing and if \( \mu_k \) and \( \lambda = \frac{\alpha k}{h} \), then you satisfy either:

- \( \lambda^2 \leq \alpha \leq 1 \)
- \( \alpha^2 k \leq 2\nu \leq h^2/k \).

Compare the solutions and explain the differences in their behavior. What is the impact of which relation you choose to satisfy? Is it possible to satisfy both simultaneously?

Solution:
Written Report

University of Utah
School of Computing
CS 6220 TA Assignment Prob 3 Report Spring 2013

Purpose/Statement of the problem

10 pts
The problems consists of advection-diffusion equation given by

\[ u_t + au_x = u_{xx} \]

We will be using explicit difference equation:

\[ u_j^{n+1} = u_j^n + k(\mu \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{h^2} - a \frac{u_{j+1}^n - u_{j-1}^n}{2h}) \]

a. 1st conditions given in the problem.

\[ \alpha \leq 1 \]

\[ \Rightarrow a^2 k \leq 2 \mu \] Substituting \( a = 1 \) we have \( 2k/h^2 \leq 1 \)

Since the method is bounded by periodicity \([0,1)\), \( h \) can take maximum value 1, then we have \( k \leq 1/2 \)

b. 2nd conditions given in the problem.

\[ \lambda^2 \leq \alpha \]

\[ \Rightarrow 2a^2 k/h^2 \leq 1 \] Substituting \( a = 1, \mu = 1 \) we have \( k \leq 2 \)

For stability, we need to match both the CFL and diffusion numbers. \( \frac{k}{h} \leq 1 \) and \( \frac{k}{h^2} \leq \frac{1}{2} \). For both conditions we need \( k = \min(h, 1/2h^2) \).

Thus both the conditions can be satisfied at the same time.

Description of the Algorithms

10 pts
Describe the computational algorithms used to implement the mathematical techniques previously described. The algorithm needed for explicit equation is

a. Initialize grid and time-step
b. Initialize the grid with Initial values
c. For each step in time(t).
   For each step in space(j).
      . update using forward euler

\[ u_j^{n+1} = u_j^n + k(\mu \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{h^2} - a \frac{u_{j+1}^n - u_{j-1}^n}{2h}) \]

      . Increment j
   Increment t
d. end of time.

**Demonstration of the Correctness of Implementation**

*10 pts*

Provide discussion and tests demonstrating that your implementation is correct. Results/Analysis of Results

*a.* when $\mu = 0$ and $a = 1$. The sin wave should have move in x-direction.

![Graph 1](image1)

*b.* when $\mu = 1$ and $a = 0$. The sin wave should have decay as time increases.

![Graph 2](image2)

*c.* when $\mu = 1$ and $a = 1$. The sin wave should have decay as time increases, and have a shift in x-direction.

*Provide results of your computations and analysis of your results*

*10 pts*

Are your results consistent with the mathematical theory for the method that you are using? If not, what assumptions have been broken?
a. Varying N and keeping $\alpha$ constant clearly acts like a attenuation function.

b. Varying N and keeping $\frac{\lambda}{2}$ constant clearly acts like a attenuation function with small shift in x-direction.

c. Keep N constant and changing $\alpha$ clearly acts like shifting function. I.e, different alpha implies different wave speeds.

d. In all the figures $\lambda^2 \leq \alpha$ and $\alpha \leq 1$ are satisfied.

**Summary and Conclusions**

10 pts

*Provide a summary of your findings and give concluding statement*

a. As proved above (part 3a), we can satisfy both the conditions at the same time.

b. $\alpha \leq 1$ brings stability for the simulation.

c. Increasing N and keeping $\alpha$ constant makes the graph converge to a solution.

d. Any change in alpha makes the graph shift in x-direction.
%initialze the grid
% assuming the space variable is s [0,2) with .1 grid spacing
% 
% clear;
function u_solve = Hw2(lambda,N)
FinalTime = .5;
FinalSpace = 1;
spaceStep = 1/N;
% a = 0.75

% a = 0.75
% spaceStep = timeStep/lambda;
% a = 2*timeStep/(spaceStep^2)
numofTsteps = ceil(FinalTime/timeStep) ;
numofSsteps = ceil(FinalSpace/spaceStep);

Eval_at = [0.1];
Eval_size = size(Eval_at);
Eval_at_s = [0.0;0.1; 0.2;0.3;0.4;0.5;0.6;0.7;0.8; 0.9;1];
Eval_size_s = size(Eval_at_s);

t_g = (0:timeStep:numofTsteps*timeStep);
s_g = (0:spaceStep:numofSsteps*spaceStep);
fun_sim = double(zeros(numofTsteps+1,numofSsteps+1));

% Forward Euler
[t,size_t] = size(t_g);
[t,size_s] = size(s_g);
test = zeros(size(s_g));

% initial Time Step
for s = 1:size_s  % time
    fun_sim(1,s) = sin(2*pi*s_g(s));
end

for t = 2:size_t  % time
    for s = 1:size_s-1  % space
        % evaluate fun_sim (t_g(t),s_g(s))
        if(s == 1)
            fun_sim(t,s) = fun_sim(t-1,s) + timeStep*u_func(fun_sim(t-1,s+1),fun_sim(t-1,s),spaceStep,mu,alpha);
        elseif (s ~= 1)
            fun_sim(t,s) = fun_sim(t-1,s) + timeStep*u_func(fun_sim(t-1,s+1),fun_sim(t-1,s-1),spaceStep,mu,alpha);
        end
    end
    fun_sim(t,size_s) = fun_sim(t,1);  % since periodic
end

% need to take for three final times. t = 0,0.2,0.5,1.5

% Reducing Data Projection for understanding better.
u_solve = zeros(Eval_size,Eval_size_s);

for i = 1:Eval_size
    for j = 1:Eval_size_s
        u_solve(i,j) = fun_sim(ceil(Eval_at(i)/timeStep)+1 , ceil(Eval_at_s(j)/spaceStep)+1 );
    end
end

%mesh(Eval_at_s,Eval_at,u_solve);

mesh(fun_sim)

%plot()

%loop to calculate till the final time
%function to calculate value of the grid

Listing 3.2: lambda_var.m

%for changing alpha
clear;
s = [0.0;0.1; 0.2;0.3;0.4;0.5;0.6;0.7;0.8; 0.9;1];

u1 = hw2(5*10^(-3),10);
u2 = hw2(5*10^(-3),20);
u3 = hw2(5*10^(-3),40);
u4 = hw2(5*10^(-3),80);
u5 = hw2(5*10^(-3),160);

figure(1);
plot(s,u1,'b');
xlabel('x[0,1]');
ylabel('u');
title('constant lambda = 0.005, varing spaceStep, Final time 0.5');
hold on;
plot(s,u2,'g');
plot(s,u3,'r');
plot(s,u4,'c');
plot(s,u5,'c');
legend('N =10','N =20','N =40','N=80');
hold off;

Listing 3.3: alpha_var.m

%for changing alpha
clear;
s = [0.0;0.1; 0.2;0.3;0.4;0.5;0.6;0.7;0.8; 0.9;1];

u1 = hw2(1,.1);
u2 = hw2(.75,.1);
u3 = hw2(.5,.1);
u4 = hw2(.25,.1);

figure(1);
plot(s,u1,'b');
xlabel('x[0,1]');
ylabel('u');
title('varing alpha,N= 10, Final time 0.05')
hold on;
plot(s,u2,'g');
plot(s,u3,'r');
legend('alpha = 1','alpha = .75','N=.5')
hold off;

u5 = hw2(1,.1);
u6 = hw2(1,.05);
u7 = hw2(1,.025);
u8 = hw2(1,.0125);

figure(2);
plot(s,u5,'b');
xlabel('x[0,1]');
ylabel('u')
title('varing N, Alpha=1, Final time 0.05')
hold on;
plot(s,u6,'g');
plot(s,u7,'r');
legend('N = 10','N =20','N = 40')
hold off;