

CSI 701 2003 Final Exam - In-class - J. Wallin

Write your name and student ID number at the top of each page of the exam. The time for the test is  $\sim 2$  hours. **To receive credit on the problems, you must show your work!**

### 1. Finite Difference Methods (50 pts total)

In homework #3, you programmed a two dimensional finite difference code to solve Poisson's equation with complicated boundary conditions. In this problem, you will need to build on those ideas. As a quick review, we will be solving the equation

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} = \rho$$

using the linear approximation

$$AU = \rho$$

Assume that we have a 3 dimensional grid with cells labeled  $U_{i,j,k}$  where  $i$  is associated with the x axis,  $j$  is associated with the y axis, and  $k$  is associated with the z axis. The maximum index for  $i, j, k$  is set to be  $L, M, N$  respectively. Assume that the grid spacing in the x, y, z directions are all constant and the same size -  $h$ .

- (a) (10 pts) Write down the three dimensional version of the finite difference approximation of Poisson's equation, using the  $i, j, k$  indexes set above.

$$\frac{U_{i+1,j,k} - 2U_{i,j,k} + U_{i-1,j,k}}{h^2} + \frac{U_{i,j+1,k} - 2U_{i,j,k} + U_{i,j-1,k}}{h^2} + \frac{U_{i,j,k+1} - 2U_{i,j,k} + U_{i,j,k-1}}{h^2} = \rho_{i,j,k}$$

- (b) (10 pts) In homework #3, you had to translate from a two dimensional grid  $(i, j)$  to a one-dimensional vector  $(\lambda)$ . Create a unique mapping between our three dimensional grid and the one-dimensional vector such that  $(i, j, k) \rightarrow \lambda$

$$\lambda = i + (j - 1) * L + (k - 1) * L * M$$

- (c) (10 pts) Rewrite the finite difference equation in part (a) in terms of  $\lambda$ .

$$\frac{U_{\lambda+1} - 2U_{\lambda} + U_{\lambda-1}}{h^2} + \frac{U_{\lambda+L} - 2U_{\lambda} + U_{\lambda-L}}{h^2} + \frac{U_{\lambda+(L*M)} - 2U_{\lambda} + U_{\lambda-(L*M)}}{h^2} = \rho_{\lambda}$$

- (d) (5 pts) How many non-zero entries will there be for each non-modified row for the matrix?  
7 nonzero entries (continued on next page)
- (e) (15 pts) In homework #3, you needed to modify the “A” matrix for boundary conditions. Specifically, you needed to move specified boundary values to the  $\rho$  side of the equation. For this problem, assume that we have Dirichlet boundary conditions on all 6 faces each with a different but constant value. In other words,  $U_{top} = \text{constant}$ ,  $U_{bottom} = \text{constant}$ , etc. In the central regions of the domain, we will have to loop over

$$\begin{aligned} i &= (2, \dots, L - 1) \\ j &= (2, \dots, M - 1) \\ k &= (2, \dots, N - 1) \end{aligned}$$

and make no modification to the finite difference equation. For each of these parts, lists the limits on  $i, j, k$  and what term(s) from part (c) will need to be moved to the  $\rho$  side of the equation.

- i. Write the limits for  $i, j, k$  and the term that is modified for the top face of the domain.

$$\begin{aligned} i &= 2, \dots, L - 1 \\ j &= 2, \dots, M - 1 \\ k &= N \end{aligned} \tag{1}$$

The  $U_{i,j,k+1}$  term is set, so this corresponds to  $\lambda + (L * M)$ . There are 6 non-zero terms left.

- ii. Write the limits for  $i, j, k$  and the term that is modified for the front, bottom edge of the domain.

$$\begin{aligned} i &= 2, \dots, L - 1 \\ j &= 1 \\ k &= 1 \end{aligned} \tag{2}$$

The  $U_{i-1,j,k}$  and  $U_{i,j,k-1}$  terms are set, so these correspond to  $U_{\lambda-1}$  and  $U_{\lambda-(L*M)}$ . There are 5 non-zero terms left.

- iii. Write the limits for  $i, j, k$  and the term that is modified for the top, front, left corner of the domain.

$$\begin{aligned} i &= 1 \\ j &= 1 \\ k &= N \end{aligned} \tag{3}$$

For this corner, we modify the  $U_{i-1,j,k}$ ,  $U_{i,j-1,k}$  and the  $U_{i,j,k+1}$  terms or the  $U_{\lambda-1}$ ,  $U_{\lambda-L}$ ,  $U_{\lambda+(L*M)}$  terms. There are 4 terms that are non-zero in these rows.

2. **ODE's and Boundary value problems** (30 pts) Consider the following boundary value problem

$$y'' - y' + 3yx = 0 \tag{4}$$

where

$$y(0) = 1 \tag{5}$$

$$y(1) = 0.2 \tag{6}$$

- (a) (5 pts) Rewrite the equation into a form suitable for solving with a 4th order Runge Kutta solver.

We define two new variables-

$$U1 = y$$

$$U2 = y'$$

So

$$U1' = U2$$

$$U2' = U2 - 3xU1$$

- (b) (5 pts) What additional steps or condition are needed to solve this boundary value problem as an initial value problem. In other words, how do you turn this into a converging method that matches the boundary conditions?

The basic steps are:

- Guess an initial derivative for the system at one of the boundary points.
- Integrate the equation to the other boundary point.
- Based on the integrated position, determine if you need to increase or decrease the initial guess for the boundary derivative.
- Repeat the last three steps until the system converges.

Step three involves using a non-linear solver to create a new estimate for the derivative. Care should be used to bracket the system so that it will converge.

- (c) (5 pts) Sketch and very briefly (about 100 words) describe how one step (iteration  $k$  to  $k+1$  in the 4th order Runge Kutta method.

One version of a 4th order Runge-Kutta solver is given by the following set of equations.

$$y_{k+1} = y_k + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \quad (7)$$

where

$$k_1 = f(x_k, y_k) \quad (8)$$

$$k_2 = f(x_k + h_k/2, y_k + k_1/2)h_k \quad (9)$$

$$k_3 = f(x_k + h_k/2, y_k + k_2/2)h_k \quad (10)$$

$$k_4 = f(x_k + h_k, y_k + k_3)h_k \quad (11)$$

Using this method, we first calculate the derivative at  $x=0$ , calling the result  $k_1$ . We then use the estimated derivative  $k_1$  to find the midpoint of the step, we estimate another derivative,  $k_2$ . Using  $k_2$ , we repeat the midpoint calculation again creating  $k_3$ . Finally, we estimate the derivative at the end of the interval using  $k_3$  to project to the estimated end point. By combining  $k_1$ ,  $k_2$ ,  $k_3$ , and  $k_4$ , we get a 4th order accurate method.

- (d) (10 pts) Take one step with  $h = 0.1$  and an initial  $y'(0) = 0.5$  using the 4th order Runge Kutta method.

are starting points are

$$U1 = 0$$

$$U2 = 0.5$$

using

$$U1' = U2$$

$$U2' = U2 - 3xU1$$

at  $x = 0$   $k_1 = (0.5, 0.5 - 3(0)(0)) = (0.5, 0.5)(h)$

using this, we find at  $x = 0.05$  using  $k_1$

$$U_1 = 0. + 0.5(0.05) = 0.025$$

$$U_2 = 0.5 + 0,5(0.05) = 0.525$$

$$k_2 = (0.525, 0.525 - 3(.025)(0.05)) = (0.525, 0.52125)(h)$$

using this, we find at  $x = 0.05$  using  $k_2$

$$U_1 = 0. + 0.525(0.05) = 0.02625$$

$$U_2 = 0.5 + 0,52125(0.05) = 0.5260625$$

$$k_3 = (0.5260625, 0.526025 - 3(0.02625)(0.05)) = (0.5260625, 0.522125)(h)$$

using this, we find at  $x=0.1$  using  $k_3$

$$U_1 = 0. + 0.5260625(0.1) = 0.05260625$$

$$U_2 = 0.5 + 0.522125(0.1) = 0.5522125$$

$$k_4 = (0.5522125, 0.5522125 - 3(0.05260625)(0.1)) = (0.5522125, 0.53643063)$$

Now we combine these

$$\begin{aligned} y(0.1) &= y(0) + 1/6(k_1 + 2k_2 + 2k_3 + k_4)h \\ &= 0 + 1/6(0.5 + 2(0.525 + 0.5260625) + 0.5522126)(0.1) \\ &= 0.052572 \end{aligned}$$

$$\begin{aligned} y'(0.1) &= y'(0) + 1/6(k_1 + 2k_2 + 2k_3 + k_4)h \\ &= 0.5 + 1/6(0.5 + 2(0.52125 + 0.5260625) + 0.53643063)(0.1) \\ &= 0.552053 \end{aligned}$$

- (e) (5 pts) Explain briefly how errors can be monitored and controlled in routines which integrate initial value problems (or boundary value problems formulated as initial value problems.) [Note: there are two fairly general methods used.]

Errors can be monitored and controlled using

- Errors can be estimated by using a small stepsize and a larger stepsize. Since methods have a rate of convergence, you can estimate the error using two small steps or one step that crosses both.
- Errors can be estimated using two methods with different orders. An example of this is the Runge-Kutta-Feldburg method using 4th and 5th order evaluations.

3. **Optimization and Parallelization** (20 pts) For these short answer questions, write one or two SHORT paragraphs for each sub-section. Assume for the first three parts, we are considering the performance of codes on modern Intel workstations with one processor.

- (a) (5 pts) Explain briefly why interchanging the order of nested loops affects the performance of scientific codes.

Changing the loop order changes the way memory is being loaded into the memory caches. If contiguous blocks of memory are loaded, the process is efficiently handled by the CPU. Large strides in memory caused by out of order loops hurts performance by causing cache misses.

- (b) (5 pts) Explain briefly what unrolling loops is and why it affects the performance of scientific codes.

Unrolling loops lowers the overhead for finishing array calculations. If you do one element at a time, you need to have a set of implicit if statements jump statements. Both of these take CPU time.

Unrolling loops also allows better pipelining of instructions. This helps keep modern processors happy and busy, instead of having them uncertain of what future instructions to execute.

- (c) (5 pts) Explain briefly why removing *if* statements from the inside of loops improves performance of scientific codes.

The primary problem of having imbedded if statements is that it can break instruction and math pipelines. By having additional non-pipelined instructions, performance suffers.

In addition, the if statement has internal overhead by its execution. This turns out to be a secondary effect compared to the breaking of vector pipelines.

- (d) (5 pts) Explain briefly what communication and load balancing are in parallel computing, and why they both must be considered in parallel code design.

Communication is the passing of data between nodes. Communication is expensive. Every message has to be considered against the bandwidth and latency between the processors.

Load balancing is the attempt to keep all processors doing useful work. If one processor is sitting idle, the efficiency of the code suffers. The only way to keep processors busy is coordinate data through message passing. Unfortunately, message passing is very inefficient and cuts efficiency on its own.