Mech 510

Programming Mini-assignment

Due date: None

Abstract

The purpose of this assignment is to get you accustomed to handling arrays — setting them up, passing them back and forth to subroutines, etc — and to data output and plotting. This assignment is not mandatory. If you wish to turn it in, I'll be happy to make comments on the results. If you are confident that you've done things properly, you needn't bother. Suppose that we have some sort of problem defined on the unit square $[0,1] \times [0,1]$, and that we are discretizing the problem using a uniform 20×20 mesh with ghost cells (for a total mesh size of 22×22). In practice, we'll often use oversized arrays, because we'll be interested in using more than one mesh size without having to change in the array sizes all over the program. So for this assignment, we'll be using arrays of size 42×42, and keeping track of the size we actually want to use separately.

1. Declare an array that will hold a scalar solution on that mesh, including the ghost cells. I recommend that you number cells from 1 to 40, with the ghost cells as indices 0 and 41. In C/C++, this would be `array[42][42]`; in Fortran, `array(0:41,0:41)`.

2. Initialize that array so that the value in each cell, including the ghost cells, is $x^2 + 2y$, where x and y are evaluated at the center of the cell. For reference, the value in the $(13,17)$ cell should be 2.040625.

3. Copy all of the data into a second array of the same size as the first.

4. Pass the second array to a subroutine (=function returning `void` in C/C++). In that subroutine, add 1 to the value of all cells where $i = 14$. Then add 1 to the value of all cells where $j = 13$. Return from this subroutine. In C/C++, you'll need to tell the compiler the size of the *second* array dimensions (in general, the size of all but the first dimension). In Fortran, you'll need to tell the compiler the size of all but the last dimension. This difference is related to the way in which the two languages store arrays in memory.

5. Pass to another subroutine a slice of the second array with $i = 14$. In this subroutine, subtract 1 from every value in the slice. Return. (In C/C++, you can do this directly with `array[14]`. In Fortran 77, you'll have to copy data to a vector and copy the result back.) In Fortran 9X, you can just pass an array slice explicitly.

6. Pass to the same subroutine as in #5 a slice of the second array with $j = 13$. (C/C++

and Fortran 77 language-specific stuff is opposite to 5.)

7. Now the copied and original array should be the same again (except just possibly for a bit of roundoff error). Verify this by summing the absolute values of the differences between comparable entries in the original and scratch arrays.

8. Write the original array to a file in a format that your favorite plotting software can read. Read the data in, and create a surface plot with contours (either line contours or color shading). Matlab, gnuplot, and Excel are all commonly used; TecPlot, VU, and other visualization packages can also be used if you like.