PX390, Autumn 2016, Assignment 6: burn wave

February 1, 2017

You are in charge of determining whether a compressed lump of fusion fuel will ignite (let's assume this is for peaceful purposes). This is controlled by a balance between a diffusive process and energy release from the fuel. The temperature $T(x, y, t)$ and stored nuclear energy $E(x, y, t)$ are given by

$$\frac{\partial T}{\partial t} = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) T - \frac{\partial E}{\partial t} \tag{1}$$

$$\frac{\partial E}{\partial t} = -E \frac{\gamma_B}{2} \left[1 + \tanh([T - T_C]/T_w) \right] \tag{2}$$

The spatial boundary condition is that $\nabla T.\hat{n} = 0$ where $\hat{n}$ is normal to the boundary. At $t = 0$ the initial condition is read from a file for $T(x, y, 0)$ and $E(x, y, 0)$.

The code should also output an error estimate based on simulating the same problem on a grid twice as coarse in space: output the mean squared difference in temperature $< (T_c - T)^2 >$, (between the temperature T_c on the coarse grid and temperature T on the original grid), at the final timestep to a file called 'errorest.txt'.

I would like you to use a structure to store parameters and pass them to functions rather than global variables.

I am expecting you to use matrix methods to allow at implicit solution of the equations so that the timestep can be reasonably long for fine grids. Matrix inversion should use the lapack library: the compiler flags are the same as for assignment 5. This library is installed on the lab computers and on daisy. Some instructions are given for installing it on a linux system on the course website; if you aren't used to compiling/installing libraries on your own system, it might be quicker just to log in to daisy. The code (which is called prog3.c in this example) must compile and generate no warnings when compiled with:

```
gcc -Wall -std=c99 -g -L/home/phskak/Code/BLAS/ -L/home/phskak/Code/lapack-3.5.0/  \
    -I/home/phskak/Code/lapack-3.5.0/lapacke/include/ prog3.c -llapacke         \
    -llapack -lblas -lgfortran -lm
```

Be careful copying and pasting this command line: a script containing the compiler command is also on the website.

The lab computers require different libraries (the OS and hardware are different): in the compile line, the directory 'Code' must be replaced with 'Code_lab'.

As before, you should test your code with simple testcases. I will, again, be marking it mostly based on how well it reproduces certain tests.

Specification:

The input and output x and y grid have I and J grid points, the grids will be taken to run from $x_L = 0$ to $x_R = x_{I-1}$ and $y_L = 0$ to $y_H = y_{J-1}$ respectively.

Input:

The input parameters will be read from a file called 'input.txt'.

You should assume that I and J are odd, $I \geq 3$, $J \geq 3$, $x_R > 0$, $y_H > 0$.

Inpur parameters:

1. t_f, length of time to simulate for.
2. t_d, diagnostic timestep
3. I, number of grid points in x direction.
4. J, number of grid points in y direction.
5. x_R, right boundary of x domain.
6. y_H, top boundary of y domain.
7. γ_B, burn time constant.
8. T_c, ignition temperature.
9. T_w, ignition temperature cutoff width.

The functions $T(x, y)$ and $E(x, y)$ will be given at the $I \times J$ grid points as a column of double precision numbers in a file called 'coefficients.txt', in order such that data associated with point (x_i, y_j) is on line $i + Ij$, with T as the first column and E as the second.

There are example input files and output file on the assignment page.

Output:

The code should output the solutions T and E at all the grid points as five columns in a text file called 'output.txt' with values $(t, x_i, y_j, T_{ij}, E_{ij})$. These should be output in order such that data associated with point (x_i, y_j) is output on line $i + Ij$. The time points to output are $t = nt_d$ (with $t < t_f$) where $n \geq 0$ is an integer. As before, if t_f is an 'exact' multiple of t_d I don't mind whether or not you output this point.

The file 'errorest.txt' should contain an estimate of the mean squared error in T based on a grid coarsening method: see above.