PX390 2020/21, Assignment 5: Burn Wave

January 12, 2021

In thermonuclear fusion, the ignition initiated in some part of the fuel propagates as a burn wave throughout the fuel. In this assignment you will use a numerical model to determine whether a compressed lump of fusion fuel will ignite. This is controlled by a balance between a diffusive process and energy release from the fuel. The fuel temperature $T(x, y, t)$ and stored nuclear energy $E(x, y, t)$ evolve according to the following equations:

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

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

We seek a numerical solution in 2D rectangular domain (x, y), such that $x \in [0, x_R]$, and $y \in [0, y_H]$. The spatial boundary condition is $\nabla T \cdot \hat{\mathbf{n}} = G$, where G is a constant and $\hat{\mathbf{n}}$ is outward pointing boundary normal. At $t = 0$ the initial condition is read from a file for $T(x, y, 0)$ and $E(x, y, 0)$. Your program should use a structure to store parameters and pass them to functions, rather than using global variables.

Solution strategy: A numerical solution at a given time can be found in three steps:

1. Use matrix methods to construct an implicit time update of temperature using equation (1) with only the spatial part of the right-hand side (omitting $\partial E/\partial t$ term). An implicit method allows a reasonably long time step for fine grid. Matrix inversion should use the LAPACK library and the required modules and compiler flags are the same as for assignment 4.

2. Once the spatial temperature profile is known, it can be used to calculate the time evolution of the energy $E(x, y, t)$, as in equation (2).

3. A complete solution of equation (1) is obtained by subtracting $\partial E/\partial t$ from the temperature profile found in step 1.

As before, you should test your code with simple testcases, since marking is mostly based on how well your code 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 the right/top boundaries satisfy $x_R > 0$, $y_H > 0$.
Input parameters must include:

1. t_f - length of time to simulate
2. t_d - diagnostic time step
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 - critical ignition temperature.
9. T_w - ignition temperature cut-off width.
10. G - temperature gradient boundary value.

The functions $T(x, y, 0)$ and $E(x, y, 0)$ will be given at the $I \times J$ grid points as a column of double precision numbers in a file called 'initialprofile.txt'. The order must be such that a data associated with a point (x_i, y_j) is on line $i + I \cdot j$, 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 $i + I \cdot j$. 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 you can decide whether or not you output this point.