MME 9710a Assignment # 2

Date Given: October 6, 2015 Date Due: October 20, 2015

This assignment is intended to give you some experience with different types of transient discretization techniques. Modifications are mainly required in your *main.f* and *coeff.f* files, although depending on how you choose to implement the schemes, other routines might also be affected. The main objectives of this assignment are to program a general two–level transient discretization scheme and then test the difference between the first–order, fully–implicit method verses the second–order Crank–Nicolson method.

Task List for Programming:

1. Add an outer, time loop in *main*. Use `KNTOUT` as your do loop counter.
2. Add a section at the beginning of the time loop to transfer `T(I)` to `TOLD(I)` and `DE(I)` to `DEOLD(I)` in preparation for doing two–level transient calculations.
3. Add a parameter `OMEG` to control the time–discretization method.
4. Modify *coeff* to include the transient terms.
5. Modify other routines as necessary.

Problem: Transient conduction in a plane wall

The problem we consider is that of a plane wall initially at 100 $[^oC]$ with its outer surfaces exposed to an ambient temperature of 0 $[^oC]$. The wall has a thickness $2L$ and may be considered to have an infinite height and a unit depth. Initially, the wall only feels the effect of the ambient air very near the surface and thus, the temperature profile inside the solid is quite steep in the vicinity of the surface. An analytical solution for this problem involves several (at least 4) terms of a Fourier series. After some time, however, the influence of the ambient air will have reached the center of the wall and the analytical solution can be approximated by the first term of the Fourier series (see, for example, *Fundamentals of Heat and Mass Transfer*, by Incropera & Dewitt, section 5.5). To study the order of the fully–implicit and Crank–Nicolson time discretization schemes, we will consider the cooling process during a period past the initial transient where the one–term Fourier solution is valid.

The parameters for the problem are:

$$Bi = \frac{h_o L}{k} = 1.0, \quad T_i = 100\ [^oC], \quad T_\infty = 0\ [^oC]$$

The one–term Fourier solution for this problem is:

$$\frac{T - T_\infty}{T_i - T_\infty} = C_1 e^{-\zeta^2 \frac{\alpha t}{L^2}} cos(\zeta \frac{x}{L})$$

where:

$$T = T(x,t), \quad \alpha = \frac{k}{\rho C_p}, \quad C_1 = 1.1191, \quad \zeta = 0.8603$$

The solution to this problem at the two different dimensionless time levels of interest is:

$$at \ \frac{\alpha t_1}{L^2} = 0.4535, \quad T(0,t_1) = 80 \ [^oC]$$
$$at \ \frac{\alpha t_2}{L^2} = 3.2632, \quad T(0,t_2) = 10 \ [^oC]$$

Computational details:

1. Use 40 control–volumes for your spatial discretization, i.e. `IB=2, IE=IE1=41`.

2. Initialize the problem using the analytical solution at $\frac{\alpha t_1}{L^2} = 0.4535$. That is, in your main program, after the initial values have been applied by *inital*, overwrite `T(I)` by the exact solution at $\alpha t_1/L^2 = 0.4535$. (You can remove this section of code when this problem is done.)

3. Using your code, calculate the temporal temperature field over the time period described above. Solve the problem by employing 2, 4, 8, 16, and 32 time steps using both the fully–implicit scheme and the Crank–Nicolson scheme.

4. At the end of each run, calculate the absolute average error, $\overline{e}$, using:

$$\overline{e} = \frac{1}{IE + 1 - IB} \sum_{I=IB}^{IE} |e(I)|, \quad e(I) = T_{exact}(I) - T(I)$$

   Then, for each scheme, plot your results of $\overline{e}$ vs. Δt (on a log-log scale) and find the value of p in the expression:

$$\overline{e} = c(\Delta t)^p$$

   where p represents the order of the transient. Also show a separate plot of `T(0,`t_2`)` verses the number of timesteps used for each scheme employed.

Writeup:

Hand in a brief report discussing your results. Include in your report all computational details and a hard copy of your code. Finally, what would be the minimum number of timesteps required to solve this problem if a fully–explicit scheme (`OMEG=0.0`) were used? (you do not need to compute this with your codes)