PHYS358: Homework 6 (due Nov 6, 9:30am)

PDEs: Diffusive Initial Value Problems

Note: Please read the complete homework instructions before you start.

Motivation:

In this homework, we will discuss heat diffusion, i.e. the transport of energy via "random collisions" of particles, if you will. We are interested in the time evolution of a (spatial) temperature field, $T(x,t)$. The corresponding equation in its most general form can be written (in one spatial dimension) as

$$\partial_t T = \partial_x \kappa \partial_x T, \tag{1}$$

with the conductivity κ. In the following, we'll assume that κ is a constant.

Goals: When you're finished with this homework, you should have understood

1. why (mathematically) eq. 1 removes extrema.
2. the difference between explicit and implicit PDE integration.
3. how to use the von Neumann stability analysis to arrive at a timestep criterion when solving eq. 1 in various ways.
4. what controls the timestep in explicit diffusion problems.

For more information and derivations, see Hancock (2006).

(6a) Stability of the FTCS scheme [10pts]: The simplest way to solve eq. 1 is to discretize it such that

$$T_j^{n+1} = T_j^n + \frac{\kappa \Delta t}{(\Delta x)^2}(T_{j+1}^n - 2T_j^n + T_{j-1}^n), \tag{2}$$

with the spatial index $j = 1 \dots J$ such that $x_j = j\Delta x$, and the time index $n = 0 \dots N-1$ such that $t_n = n\Delta t$. Since this is an initial value problem, we need to specify initial values at $n = 0$ for all j. We also have to specify boundary values at $j = 0$ and $j = J+1$, for all n (i.e. at every timestep). Therefore, we have $J+2$ spatial support points $j = 0 \dots J+1$, with $j = 0$ and $j = J+1$ containing the boundary values.

But before we address how to integrate eq. 2, we'll discuss it's stability. Looking at eq. 2, we can rewrite it as

$$\boldsymbol{T}^{n+1} = \boldsymbol{A}\boldsymbol{T}^n, \tag{3}$$

with the temperature vector $\boldsymbol{T}$ and a linear operator (i.e. matrix) $\boldsymbol{A}$. From eq. 3 it is clear that $\boldsymbol{T}^n$ can be reached by repeatedly applying $\boldsymbol{A}$ to $\boldsymbol{T}^0$. Clearly, the underlying physics of the diffusion problem mandates that $|\boldsymbol{T}|$ be bounded by some maximum value. Thus, the eigenvalues of $\boldsymbol{A}$ must obey $\max|\lambda| \leq 1$. A simpler, but related, notion is to view each T_j^n as an independent solution, or *eigenmode* of eq. 2, with

$$T_j^n \equiv \xi^n(k) e^{ikj\Delta x}. \tag{4}$$

Here, $\xi(k)$ is a complex function of the wavenumber k, and ξ^n is ξ to the nth power, i.e. the time dependence of each eigenmode is determined by successive integer powers of $\xi(k)$ (the *phase* information in the exponential just depends on position). Another way to see this is that eq. 4 describes the *Fourier* modes of the solution $\boldsymbol{T}$. Since $T_j^n \propto \xi^n(k)$, eq. 2 will be unstable if $|\xi(k)| > 1$ for some k. What remains to be done is to find $\xi(k)$:

1. Write down one row (away from the boundaries) of the matrix $\boldsymbol{A}$.
2. Using eq. 2 and eq. 4, show that
$$\xi(k) = 1 - \frac{4\kappa\Delta t}{(\Delta x)^2}\sin^2\frac{k\Delta x}{2}. \tag{5}$$
3. Using eq. 5, show that the time step is limited by
$$\Delta t \leq \frac{(\Delta x)^2}{2\kappa}. \tag{6}$$
4. If you double the resolution (i.e. increase the number of spatial support points J by a factor of 2 while keeping the physical extent of the domain constant), by what factor do you need to reduce Δt to keep the the method stable?

Solution: The matrix row is $(\alpha, 1-2\alpha, \alpha)$, just by comparing coefficients, with $\alpha \equiv \kappa\Delta t/(\Delta x)^2$. Replacing the T in eq. 2 with their Fourier modes yields after division by ξ^n and $e^{ijk\Delta x}$
$$\xi = 1 + \alpha\left(e^{ik\Delta x} - 2 + e^{-ik\Delta x}\right).$$
The term in parentheses is $-4\sin^2(k\Delta x/2)$, hence eq. 5. The condition $|\xi| \leq 1$ leads to $|1-4\alpha| \leq 1$, and thus to $\alpha \leq 1/2$, i.e. eq. 6. Since the $\Delta t \propto (\Delta x)^2$, doubling the resolution requires reducing the timestep by a factor of 4.

(6b) Stability of the fully implicit scheme [5pts]: Clearly, eq. 6 suggests that the timestep constraints on explicit diffusive problems are rather stringent. To make progress, remember how we got around (time) step constraints when integrating ODEs, namely by rewriting the RHS in terms of the updated values at time $n+1$.

1. Write down the fully implicit version of eq. 2. Write down the tridiagonal matrix with which the system can be solved (assume $T_0 = T_{J-1} = 0$).
2. Show that for the fully implicit update,
$$\xi(k) = \frac{1}{1 + 4\frac{\kappa\Delta t}{(\Delta x)^2}\sin^2\frac{k\Delta x}{2}}. \tag{7}$$
3. What is the limiting timestep for the fully implicit update?
4. If the boundaries are set to $T_0 = 0, T_{J-1} = 0$, to what value will the T_j converge for $\Delta t \to \infty$?

Solution: The "update" equation is

$$T_j^{n+1} = T_j^n + \alpha(T_{j+1}^{n+1} - 2T_j^{n+1} + T_{j-1}^{n+1}).$$

This can be written as a tri-diagonal linear problem $\boldsymbol{AT}^{n+1} = \boldsymbol{T}^n$, where a row of $\boldsymbol{A}$ is given by $(-\alpha, 1+2\alpha, -\alpha)$. Using the Fourier modes on the update equation yields

$$\xi = 1 + \xi\alpha\left(e^{ik\Delta x} - 2 + e^{-ik\Delta x}\right),$$

resulting in eq. 7, using the same reasoning as above. Thus $|\xi| \leq 1\,\forall k$, i.e. the scheme is unconditionally stable.

(6c) Stability of the Crank-Nicholson method [5pts]: Fully implicit updates are unconditionally stable. Yet, we loose accuracy in the small-scale time evolution for the price of getting this stability. A very popular way around this problem of loss of accuracy is a combination of the explicit and implicit update, by literally taking the average of the two RHS.

1. Write down the update rule for the *Crank-Nicholson* scheme, and show that

$$\xi(k) = \frac{1 - 2\frac{\kappa\Delta t}{(\Delta x)^2}\sin^2\frac{k\Delta x}{2}}{1 + 2\frac{\kappa\Delta t}{(\Delta x)^2}\sin^2\frac{k\Delta x}{2}}. \tag{8}$$

2. What is the limiting timestep for the Crank-Nicholson scheme?

Solution: The CN update is given by

$$T_j^{n+1} = T_j^n + \frac{\kappa\Delta t}{2(\Delta x)^2}(T_{j+1}^{n+1} - 2T_j^{n+1} + T_{j-1}^{n+1} + T_{j+1}^n - 2T_j^n + T_{j-1}^n).$$

The expression for ξ is just the combination of those in (6a) and (6b). One contains a ξ and thus needs to be divided by. $|\xi| \leq 1\,\forall k$, thus, the CN scheme is unconditionally stable.

(6d) Implementation of the three solvers [20pts]: Implement the three solvers. An example for a possible structure (with the usual function pointer mess) is given in `pde_diffusion.py`.

Note on Boundary Conditions: The tricky part is keeping track of the spatial boundary conditions. There are two ways to implement them. The most obvious way would be to place the function values T_i at the support points x_i. For example, $x_0 = -0.5$ and $x_N = 0.5$ with $\Delta x = 0.05$ requires $J = 21$ support points. Another, more elegant way, is to place the function values T_i at the cell centers. In that case, the lower cell wall for the first cell is located at $x = -0.5$, and T_0 would be placed at $x = -0.5 + \Delta x/2 = -0.475$ for $J = 20$. I strongly recommend to go with the latter approach, since it will be consistent with later discussions of flux-conservative problems.

(6e) Tests [15pts]:

1. Check out chapter 2 and the first introduction to chapter 3 ("full solution" for rod) of Hancock 2006. I recommend to use this example for testing. A rod of uniform temperature $T = 1$ is cooled at its ends to $T = 0$. Go through the analysis in Hancock (2006) and implement the analytic result for the temperature profile as a comparison.

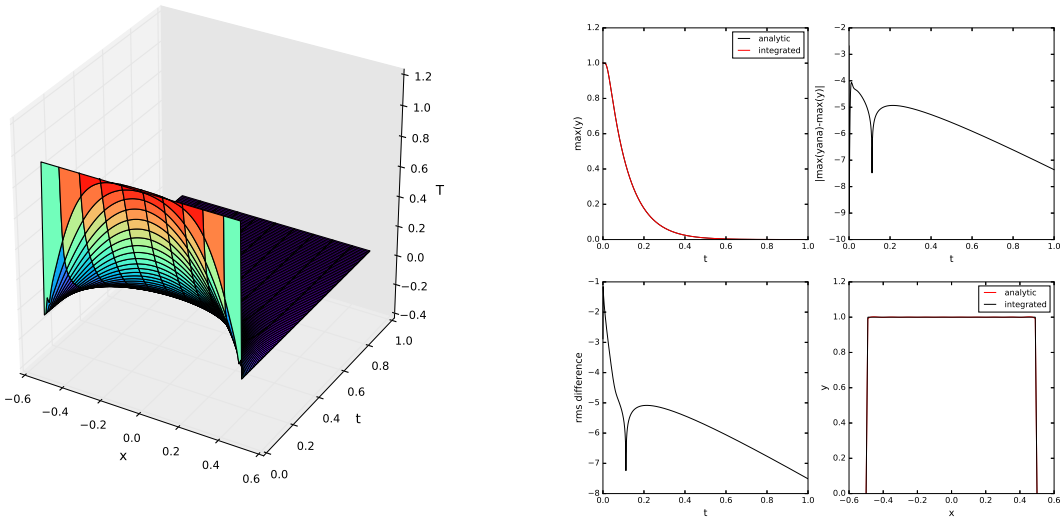

Figure 1: *Left:* Surface map of $T(x,t)$. *Right:* Various plots to test results. Here for Crank-Nicholson.

2. Your code should print out the value of $\alpha \equiv \kappa \Delta t/(\Delta x)^2$.

3. Your code should show three plots (example see below): a 3D plot of the temperature function in time, a comparison between the analytic and integrated result of the maximum temperature against time, and a root-mean-square difference between the integrated and the analytic solution.

4. Calculate Δt (argument `dt`), the timestep, for $J = 20$ and a total simulation time of $t = 1$. How many iterations should the code take when using `ftcs`? Run your code with $J = 20$, using `ftcs`. Compare the results to `implicit` and `CN`. Take note of the rms difference for each case. Next, try $J = 40$, keeping your previous Δt. Does `ftcs` still produce useful results? What about `implicit` and `CN`? Find the value of Δt to run `ftcs` stably for $J = 40$, and compare.

Solution: For $J = 20$, $\Delta t = 1.25 \times 10^{-3}$. The `ftcs` should run stably, using 800 iterations (excluding initial condition). Increasing to $J = 40$, we'll need to divide Δt by a factor of 4 to get `ftcs` to produce useful results. `implicit` and `CN` on the other hand work fine.