Notes on Diffusion

These summarize methods for solving the diffusion equation.

1 Parabolic equations

The diffusion equation is

$$\frac{\partial \phi}{\partial t} = \frac{\partial}{\partial x} \left( k \frac{\partial \phi}{\partial x} \right)$$  \hspace{1cm} (1)

This can describe thermal diffusion (for example, as part of the energy equation in compressible flow), species/mass diffusion for multispecies flows, or the viscous terms in incompressible flows. In this form, the diffusion coefficient (or conductivity), $k$, can be a function of $x$, or even $\phi$. We will consider a constant diffusion coefficient:

$$\frac{\partial \phi}{\partial t} = k \frac{\partial^2 \phi}{\partial x^2}$$  \hspace{1cm} (2)

The diffusion equation is the prototypical parabolic PDE. The basic behavior of the diffusion equation is to take strongly peaked concentrations of $\phi$ and smooth them out with time.

2 Explicit differencing

The simplest way to difference this equation is explicit in time (i.e. the righthand side depends only on the old state):

$$\frac{\phi^n_i - \phi^{n-1}_i}{\Delta t} = k \frac{\phi^{n+1}_{i+1} - 2\phi^n_i + \phi^{n-1}_{i-1}}{\Delta x^2}$$  \hspace{1cm} (3)

This is second-order accurate in space, but only first order accurate in time (since the righthand side is not centered in time).

As with the advection equation, when differenced explicitly, there is a constraint on the timestep required for stability. Looking at the growth of a single Fourier mode, $\phi = A^n e^{ij \theta}$ with $j = \sqrt{-1}$, we find:

$$\frac{A^{n+1}}{A^n} = 1 + 2 \frac{k \Delta t}{\Delta x^2} (\cos \theta - 1)$$  \hspace{1cm} (4)

Stability requires that $|A^{n+1}/A^n| \leq 1$, which can only be true if $2k \Delta t / \Delta x^2 \leq 1$. Therefore, our timestep constraint in this case is

$$\Delta t < \frac{\Delta x^2}{2k}$$  \hspace{1cm} (5)

Note the $\Delta x^2$ dependence—this constraint can become really restrictive.

To complete the solution, we need boundary conditions at the left ($x_l$) and right ($x_r$) boundaries. These are typically either Dirichlet:

$$\phi|_{x=x_l} = \phi_l$$  \hspace{1cm} (6)
$$\phi|_{x=x_r} = \phi_r$$  \hspace{1cm} (7)

or Neumann:

$$\phi_x|_{x=x_l} = \phi_x|_l$$  \hspace{1cm} (8)
$$\phi_x|_{x=x_r} = \phi_x|_r$$  \hspace{1cm} (9)
3 Implicit with direct solve

A backward-Euler implicit discretization would be:

\[
\phi_i^{n+1} - \phi_i^n = k \frac{\Delta t}{\Delta x^2} \frac{\phi_i^{n+1} - 2\phi_i^{n+1} + \phi_i^{n+1}}{\Delta x^2}
\]  

This is still first-order in time, but is not restricted by the timestep constraint (although the timestep will still determine the accuracy). Defining:

\[
\alpha \equiv k \frac{\Delta t}{\Delta x^2}
\]

we can write this as:

\[-\alpha \phi_{i+1}^{n+1} + (1 + 2\alpha) \phi_i^{n+1} - \alpha \phi_{i-1}^{n+1} = \phi_i^n
\]

This is a set of coupled algebraic equations. We can write this in matrix form. Using a cell-centered grid, spanning \([lo, hi]\), and Neumann BCs on the left:

\[
\phi_{lo-1} = \phi_{lo}
\]

the update for the leftmost cell is:

\[(1 + \alpha) \phi_{lo}^{n+1} - \alpha \phi_{lo+1}^{n+1} = \phi_{lo}^n
\]

If we choose Dirichlet BCs on the right \((\phi|_{x=x_l} = \Lambda)\), then:

\[\phi_{hi+1} = 2\Lambda - \phi_{hi}
\]

and the update for the rightmost cell is:

\[-\alpha \phi_{hi-1}^{n+1} + (1 + 3\alpha) \phi_{hi}^{n+1} = \phi_{hi}^n + \alpha 2\Lambda
\]

For all other interior cells, the stencil is unchanged. The resulting system can be written in matrix form and appears as a tridiagonal matrix.

\[
\begin{pmatrix}
1 + \alpha & -\alpha \\
-\alpha & 1 + 2\alpha & -\alpha \\
& -\alpha & 1 + 2\alpha & -\alpha \\
& & \ddots & \ddots & \ddots \\
& & & -\alpha & 1 + 2\alpha & -\alpha \\
& & & & -\alpha & 1 + 3\alpha
\end{pmatrix}
\begin{pmatrix}
\phi_{lo}^{n+1} \\
\phi_{lo+1}^{n+1} \\
\phi_{lo+2}^{n+1} \\
\vdots \\
\phi_{hi-1}^{n+1} \\
\phi_{hi}^{n+1}
\end{pmatrix} =
\begin{pmatrix}
\phi_{lo}^n \\
\phi_{lo+1}^n \\
\phi_{lo+2}^n \\
\vdots \\
\phi_{hi-1}^n \\
\phi_{hi}^n + \alpha 2\Lambda
\end{pmatrix}
\]  

Exercise 1: Write a one-dimensional implicit diffusion solver for the domain \([0, 1]\) with Neumann boundary conditions at each end and \(k = 1\). Your solver should use a tridiagonal solver and initialize a matrix like that above. Use a timestep close to the explicit step, a grid with \(N = 128\) zones.

If we begin with a Gaussian, the resulting solution is also a Gaussian. Initialize using the following with \(t = 0\):

\[
\phi(x, t) = (\phi_2 - \phi_1) \sqrt{\frac{t_0}{t + t_0}} e^{-\frac{1}{2}(x-x_c)^2/k(t+t_0)} + \phi_1
\]

with \(t_0 = 0.001\), \(\phi_1 = 1\), and \(\phi_2 = 2\), and \(x_c\) is the coordinate of the center of the domain. Run until \(t = 0.01\) and compare to the analytic solution above.

(Note: the solution for two-dimensions differs slightly)
4  Implicit multi-dimensional diffusion via multigrid

Consider a second-order accurate time discretization (this means that the RHS is centered in time), for the multi-dimensional diffusion equation:

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} = \frac{1}{2} \left( k \nabla^2 \phi_i^n + k \nabla^2 \phi_i^{n+1} \right)$$  \hspace{1cm} (19)

This time-discretization is sometimes called Crank-Nicolson. Grouping all the $n+1$ terms on the left, we find:

$$\phi_i^{n+1} - \frac{\Delta t}{2} k \nabla^2 \phi_i^{n+1} = \phi_i^n + \frac{\Delta t}{2} k \nabla^2 \phi_i^n$$  \hspace{1cm} (20)

This is in the form of a constant-coefficient Helmholtz equation,

$$(\alpha - \beta \nabla^2) \phi = f$$  \hspace{1cm} (21)

with

$$\alpha = 1$$  \hspace{1cm} (22)
$$\beta = \frac{\Delta t}{2} k$$  \hspace{1cm} (23)
$$f = \phi_i^n + \frac{\Delta t}{2} k \nabla^2 \phi_i^n$$  \hspace{1cm} (24)

This can be solved using multigrid techniques with a Helmholtz operator. The same boundary conditions described above apply here.

Note: when using multigrid, you do not need to actually construct the matrix. This is usually the most efficient way to implement diffusion in a multi-dimensional simulation code, especially when distributing the grid across parallel processors.

5  Going further

- **Non-constant conductivity**: for the case where $k = k(x)$, we discretize as:

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} = \frac{1}{2} \left( k \nabla \phi \right)_{i+1/2} - \frac{1}{2} \left( k \nabla \phi \right)_{i-1/2}$$  \hspace{1cm} (25)

Here we need the values of $k$ at the interfaces, $k_{i-1/2}$ and $k_{i+1/2}$. We can get these from the cell-centered values in a variety of ways including straight-averaging:

$$k_{i+1/2} = \frac{1}{2} (k_i + k_{i+1})$$  \hspace{1cm} (26)

or averaging the inverses:

$$\frac{1}{k_{i+1/2}} = \frac{1}{2} \left( \frac{1}{k_i} + \frac{1}{k_{i+1}} \right)$$  \hspace{1cm} (27)

The latter may be the right method for conduction, since it is behaves like resistance.

- **State-dependent transport coefficients**: many times the transport coefficients themselves depend on the quantity being diffused:

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} = \frac{1}{2} \left\{ \nabla \cdot [k(\phi^n) \nabla \phi^n] + \nabla \cdot [k(\phi^{n+1}) \nabla \phi^{n+1}] \right\}$$  \hspace{1cm} (28)
(for example, with thermal diffusion, the conductivity can be temperature dependent). In this case, we can achieve second-order accuracy by doing a predictor-corrector. First we diffuse with the transport coefficients evaluated at the old time, giving a provisional state, \( \phi^* \):

\[
\frac{\phi_i^n - \phi_i^n}{\Delta t} = \frac{1}{2} \left\{ \nabla \cdot [k(\phi^n) \nabla \phi^n] + \nabla \cdot [k(\phi^n) \nabla \phi^n] \right\}
\]

Then we redo the diffusion, evaluating \( k \) with \( \phi^* \) to center the righthand side in time, giving the new state, \( \phi^{n+1} \):

\[
\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} = \frac{1}{2} \left\{ \nabla \cdot [k(\phi^n) \nabla \phi^n] + \nabla \cdot [k(\phi^*) \nabla \phi^{n+1}] \right\}
\]

This is the approach used, for example, in [1].

• **Temperature diffusion in energy equation**: Often we find diffusion represented as one of many physical processes in a single equation. For example, consider the internal energy equation with both reactions and diffusion:

\[
\rho \frac{\partial e}{\partial t} + \rho U \cdot \nabla e + p \nabla \cdot U = \nabla \cdot k \nabla T + \rho S
\]

This can be solved via an explicit-implicit discretization. First the advection terms are computed as:

\[
A = \rho U \cdot \nabla e + p \nabla \cdot U
\]

Then the advective-diffusive part is solved implicitly. Expressing \( e = e(\rho, T) \), and rewriting

\[
\nabla T = (\nabla e - e_p \nabla \rho) / e_T
\]

and then

\[
\rho \frac{\partial e}{\partial t} = \nabla \cdot (k/e_T) \nabla e - \nabla \cdot (k e_p/e_T) \nabla \rho - A + \rho S
\]

This is now a diffusion equation for \( e \), which can be solved by the techniques described above. This is discussed, for example, in [1, 2].

**References**
