Chapter 6

Diffusion: Diffusive initial value problems and how to solve them

Selected Reading

Numerical Recipes, 2nd edition: Chapter 19

This section will consider the physics and solution of the simplest partial differential equations for diffusive initial value problems in the absence of advection. The archetypal equation of this sort is the "Heat flow equation" which for constant density material can be written

$$\frac{\partial T}{\partial t} = \boldsymbol{\nabla} \cdot \kappa \boldsymbol{\nabla} T \tag{6.0.1}$$

where T is the temperature and $\kappa = k/(\rho c_P)$ is the thermal diffusivity (which has units of length^2/time). I hope most people have some physical feeling for diffusion and heat flow as a smoothing and smearing process, however, we will start by emphasizing some of the more quantitative relationships between diffusive timescales and length scales (as well as developing a simple rule for how not to scorch a turkey).

6.1 Basic physics of diffusion

As a representative problem we will consider one-dimensional diffusion with constant diffusivity.

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2} \tag{6.1.1}$$

If we consider the evolution of a single sin wave of wavelength λ in an infinite medium and seek solutions of the form

$$T = T_0 e^{\sigma t} \sin\left(\frac{2\pi x}{\lambda}\right) \tag{6.1.2}$$

Then substitution of (6.1.2) into (6.1.1) shows that it is a valid solution if

$$\sigma = -\frac{4\pi^2\kappa}{\lambda^2} \tag{6.1.3}$$

i.e. the full solution is

$$T = T_0 \exp\left[-\frac{4\pi^2\kappa t}{\lambda^2}\right] \sin\left(\frac{2\pi x}{\lambda}\right) \tag{6.1.4}$$

and the time it takes for the amplitude of this sin wave to decay by a factor of $e^{-1} = .37$ is the thermal relaxation time

$$t = \frac{\lambda^2}{4\pi^2\kappa} \tag{6.1.5}$$

The important points to notice are that the amplitude decays *exponentially* with a "decay rate" that only depends on wavelength, not on the initial temperature, and that short wavelength variations decay much more quickly than long wavelength variations. Physically, this makes sense as heat flows down temperature *gradients* and for the same amplitude, short wavelength variations have much sharper gradients than longer wavelength ones. Because we can construct any arbitrary initial condition in an infinite (or periodic) medium out of sines and cosines[1], the overall behaviour of these equations is that the high-frequency information (corners, wiggles and edges) will rapidly smooth out but the larger scale features will remain for a while. The point is that for any scale feature and diffusivity there will be a characteristic time scale that we need to resolve if we want to solve for these things accurately. This time scale will motivate much of the discussion of stability and efficiency of numerical schemes for solving diffusive equations.

As for the turkey, business. The important point of heat conduction is that the time it takes for heat to move a fixed distance is independent of the temperature. Thus if a 20 lb turkey takes 5 hours to cook at 375, you can't cook it for 2.5 hours at 750. All you'll get is a Turkey tootsie pop - hard shell outside, chewy-gooey inside.

6.2 The numerics of diffusion

We'll begin considering how to solve diffusion numerically by deriving some finite difference approximations to the diffusion term. A control volume approach is particularly useful for this purpose. Given a 1-D grid with n control volumes we can consider the divergence of the heat flux out of cell j as

$$\boldsymbol{\nabla} \cdot \mathbf{F} = \frac{1}{\Delta x}\left(F_{j+1/2} - F_{j-1/2}\right) \tag{6.2.1}$$

but for diffusion the heat flux at point $j + 1/2$ is

$$F_{j+1/2} = \kappa_{j+1/2} \frac{\partial T}{\partial x}_{j+1/2} \tag{6.2.2}$$

[1] Well actually we can construct any piecewise continuous function.

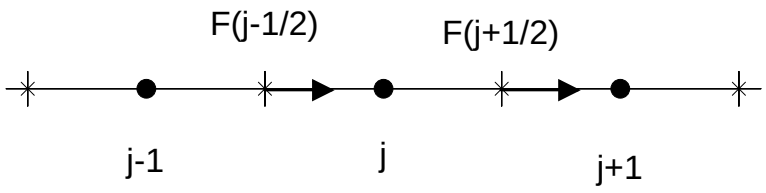


Figure 6.1: our handy-dandy control volume grid again.

If we do a centered difference for the temperature gradient at the half-point we get

$$\frac{\partial T}{\partial x}_{j+1/2} \approx \frac{1}{\Delta x}\left(T_{j+1} - T_j\right) \tag{6.2.3}$$

and therefore we can approximate the entire diffusion term as

$$\frac{\partial}{\partial x}\left(\kappa\frac{\partial T}{\partial x}\right) \approx \frac{1}{\Delta x^2}\left[\kappa_{j-1/2}T_{j-1} - (\kappa_{j-1/2} + \kappa_{j+1/2})T_j + \kappa_{j+1/2}T_{j+1}\right] \tag{6.2.4}$$

For convenience and future notation, it is useful to write (6.2.4) as a *stencil*

$$\frac{\partial}{\partial x}\left(\kappa\frac{\partial T}{\partial x}\right) \approx \frac{1}{\Delta x^2}\Big[\begin{array}{ccc} \kappa_{j-1/2} & -(\kappa_{j-1/2} + \kappa_{j+1/2}) & \kappa_{j+1/2} \end{array}\Big]T_j \tag{6.2.5}$$

which is an operator that acts on a point T_j and its two nearest neighbors. Unless it is specifically noted, the central point in the stencil is the coefficient of point T_j. In this notation, the approximate diffusion term for a constant diffusivity looks like

$$\frac{\kappa}{\Delta x^2}\Big[\begin{array}{ccc} 1 & -2 & 1 \end{array}\Big]T_j \tag{6.2.6}$$

Using a Taylor's series approach it is straightforward to show that the truncation error on this discretization is order $\Delta x^2 T_{xxxx}$ for a *regularly spaced* grid. Thus, in this case, while the error only changes as Δx^2 this approximation is actually accurate up to third order as any function that can be fit exactly by a cubic polynomial will have no truncation error (i.e. $T_{xxxx} = 0$). Thus diffusion tends to be quite accurate. Higher order schemes, are readily derived but cause the stencils to be larger.

Adding time Given a discretization of the diffusion term we still need to add the time derivative for the left hand side. While the FTCS scheme was always unstable for advection problems, it turns out that it can be stable for diffusion problems (why? As an interesting point the centered time, centered space staggered leapfrog approach is always *unstable* for diffusion problems…c'est la guerre.) Using a forward time step for the LHS and rearranging, the FTCS approximation to the diffusion equation with constant diffusivity can be written

$$\begin{aligned} T_j^{n+1} &= T_j^n + \beta\Big[\begin{array}{ccc} 1 & -2 & 1 \end{array}\Big]T_j^n \\ &= \Big[\begin{array}{ccc} \beta & (1-2\beta) & \beta \end{array}\Big]T_j^n \end{aligned} \tag{6.2.7}$$

where

$$\beta = \frac{\kappa \Delta t}{\Delta x^2} \tag{6.2.8}$$

is similar to the courant number and physically corresponds to the number of grid points that heat can diffuse in a time step (or it is the inverse of the number of time steps required for heat to diffuse a grid space). Another way to look at β is to note that it is the number of decay times for the highest frequency components on the grid (compare to Eq. (6.1.5).

Inspection of (6.2.7) shows that as long as $\beta < 1/2$, the FTCS scheme acts as a simple 3 point smoother where every point is replaced by some fraction $(1 - 2\beta)$ of its previous value and some mixture of the values of it's nearest neighbor. For example, if $\beta = 1/4$ the stencil looks like $\begin{bmatrix} 1/4 & 1/2 & 1/4 \end{bmatrix}$ and therefore, after one pass of the algorithm, a unit spike on the grid gets reduced to half its height and smeared out over its two nearest neighbors (i.e $\begin{bmatrix} 0 & 1 & 0 \end{bmatrix} T_j \rightarrow \begin{bmatrix} 1/4 & 1/2 & 1/4 \end{bmatrix}$). If $\beta > 1/2$, however, the scheme is clearly unstable as a single step will make the temperature go negative (which is not physical and will eventually explode), thus the stability criteria for the FTCS step is $\beta < 1/2$. Expanding (6.2.8) shows that this stability criterion requires that

$$\Delta t < \frac{\Delta x^2}{2\kappa} \tag{6.2.9}$$

which is the time constant for the highest frequency components to decay. This is the curse of the FTCS explicit method. You need to track the decay of the fastest decaying wavelength to remain stable even when you are not interested in features that are spanned by 1 or 2 grid points (if you are then your grid is too coarse and those features will be badly underresolved). Unfortunately, as the grid is refined so that the features of interest are spanned by at least 3 to 4 grid points (or more), the time step needs to get smaller as Δx^2. This means that to reduce the grid spacing by a factor of 2 requires 8 times longer to calculate to the same t_{max} (it's 8, not 4 because there are twice as many points and 4 times as many steps). Even in 1-D, this sort of scaling can make it costly to have well resolved grids. Fortunately, FTCS is not the only scheme available and the following section will develop *implicit* schemes which are unconditionally stable and you can (but shouldn't) take as large a step as you want. Before we develop these schemes, however, we need to consider how to include boundary conditions.

6.2.1 Boundary conditions

Any PDE with 2nd derivatives (and above) must have boundary conditions specified at the edges of the domain. Boundary conditions are extremely important and often control the behaviour of the solution. Unfortunately, they are also often the most difficult part of a problem to get to behave (many times the boundary conditions can be a source of instability even if the general scheme is stable). In discrete systems, boundary conditions are also required to make sure there are enough equations for the unknowns, i.e. within the domain, all the points satisfy

the stencil equation (6.2.7), however, at the edges T_1 and T_N we have to specify additional information to make up for the lack of a T_0 or T_{N+1} point. For second order differential equations, there are basically 4 types of boundary conditions we will be concerned with

Dirichlet Boundary conditions The simplest boundary conditions are known as Dirichlet conditions and simply state that the value at the boundary (T_1 or T_N) is a known function of time. In a 1-D problem, if both boundaries are of dirichlet type, then a unique solution exists (I think) and the equations are easy to solve as the only points that are unknown are the interior points $T_j, j = 2, \ldots, N-1$.

Neumann Boundary conditions In addition to specifying the temperature at the boundary, one could also specify the heat flux (or temperature gradient) at the boundary. These conditions are known as Neumann conditions and produce a great deal more freedom in the behaviour of the governing equations (although the equations may become ill-posed). If a boundary has Neumann BC's then the temperature on that boundary is variable with time and requires an additional equation. To derive the boundary conditions for a specified flux boundary it is convenient to use the control volume formalism. If we specify that the heat flux at point 1 is F_1, then the diffusion term for the half-sized control volume between x_1 and $x_{1+1/2}$ is

$$\frac{\partial F}{\partial x} \approx \frac{1}{\Delta x/2}\left(\kappa_{j+1/2}\frac{T_{j+1}-T_j}{\Delta x} - F_1\right) \tag{6.2.10}$$

or in the case of an insulating boundary $F_1 = 0$

$$\approx \beta\Big[\begin{array}{ccc} 0 & -2 & 2 \end{array}\Big]T_1 \tag{6.2.11}$$

Note that (6.2.11) would have the identical effect to having T_1 be an internal point but $T_0 = T_2$ i.e. a zero flux boundary must also be a reflection boundary. The extension to the right hand boundary at T_N is straightforward.

Mixed Boundary conditions It is also possible to have mixed Dirichlet/Neumann boundary conditions of the form

$$T_1 + a\frac{\partial T}{\partial x}_1 = f(t) \tag{6.2.12}$$

This is just a more general version of the Neumann condition and will produce an additional auxilary equation (stencil) for the boundary points.

Periodic or wraparound Boundary conditions These boundary conditions are extremely convenient numerically because they approximate an infinite periodic system by simply specifying that $T_N = T_1$ and therefore $T_{N+1} = T_2$ and $T_0 = T_{N-1}$. With just a simple patch at the end of each iteration, every point behaves as an interior point. These boundary conditions can be very useful ones when developing a code as only the stability of the internal solution is important.

6.3 Implicit Schemes and stability

The stability of the FTCS scheme is contingent on tracking the fastest decaying wavelengths (which is the wavelength of the grid noise) even when those wavelengths are not of interest. The problem arises because the rate of diffusion calculated at the present time is always greater than at later times and thus too large a time step will overshoot the solution. The fix for this is to use an *implicit method* which uses information about diffusion rates at *future times*. To see how it works, it is useful to develop implicit approaches for the analogous problem of radioactive decay.

6.3.1 An analogy with radioactive decay

Radioactive decay is actually an almost identical problem to diffusion if we note that individual frequencies decay exponentially with a "decay rate" given by the thermal time-constant. Now the simplest ODE for the decay of a radioactive species is

$$\frac{dc}{dt} = -\lambda c \tag{6.3.1}$$

which we could solve by taking an explicit Euler step

$$c^{n+1} = c^n - \lambda \Delta t c^n = (1 - \lambda \Delta t) c^n \tag{6.3.2}$$

which says to use the decay rate at time n ($-\lambda c^n$) and use this rate to extrapolate to a future time by taking a step of Δt. Unfortunately, the rate at time n is always an overestimate and we have to take very small steps in order to not undershoot the solution (remember how inaccurate the euler method is). Moreover, if we take too big a step ($\lambda \Delta t > 1$) we will drive the concentration to negative values which are unphysical and unstable.

The fix to this problem is to take an implicit step that uses the decay rate at a future time to predict the rate of change, i.e. the implicit version of the euler step (also called a backwards euler step) is

$$c^{n+1} = c^n - \lambda \Delta t c^{n+1} \tag{6.3.3}$$

While we don't know what the concentration c^{n+1} is *a priori*, we can rearrange (6.3.3) to solve for it by

$$c^{n+1} = \frac{c^n}{1 + \lambda \Delta t} \tag{6.3.4}$$

This scheme is stable for any sized time step Δt as in the limit of an enormous time step $c^{n+1} \to 0$ which is the steady-state solution. Just because this scheme is stable, however, does not mean that it is accurate! If you want to track the decay of an element accurately with time you still need to respect the intrinsic time scale of decay and take a time step that can resolve the changes (or use a higher order scheme that incorporates more information about the decay rate). Nevertheless, the implicit schemes allow you to choose a time scale of interest rather than being forced to follow the stability criterion. The same is true for implicit methods for solving diffusion equations.

6.3.2 Fully implicit schemes

The generalization of an implicit scheme to diffusion is straightforward. All we do is to rewrite Eq. (6.2.7) and replace the diffusion rate at timestep n with that at timestep $n+1$. For a constant diffusivity we can write

$$T_j^{n+1} = T_j^n + \beta \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} T_j^{n+1} \tag{6.3.5}$$

with the obvious extension to non-constant κ. This is an implicit method because we don't know in advance what the RHS will evaluate to, however, we can again rearrange (6.3.5) to solve for T^{n+1} as

$$\begin{bmatrix} -\beta & (1+2\beta) & -\beta \end{bmatrix} T_j^{n+1} = T_j^n \tag{6.3.6}$$

Inspection of (6.3.6) shows that it is actually a system of linear equations of the form

$$\mathsf{A}\mathbf{x} = \mathbf{b} \tag{6.3.7}$$

where A is a *tridiagonal matrix* that is primarily zero except for the diagonal (which has value $(1+2\beta)$) and one super and sub diagonal of value $-\beta$. For (6.3.7) the vector $\mathbf{x} = \mathbf{T}^{n+1}$ corresponds to the array of temperature values at timestep $n+1$ and the vector $\mathbf{b}$ is the known temperatures at time n. Thus if we can invert A we can solve for $\mathbf{T}^{n+1}$ as

$$\mathbf{T}^{n+1} = \mathsf{A}^{-1}\mathbf{T}^n \tag{6.3.8}$$

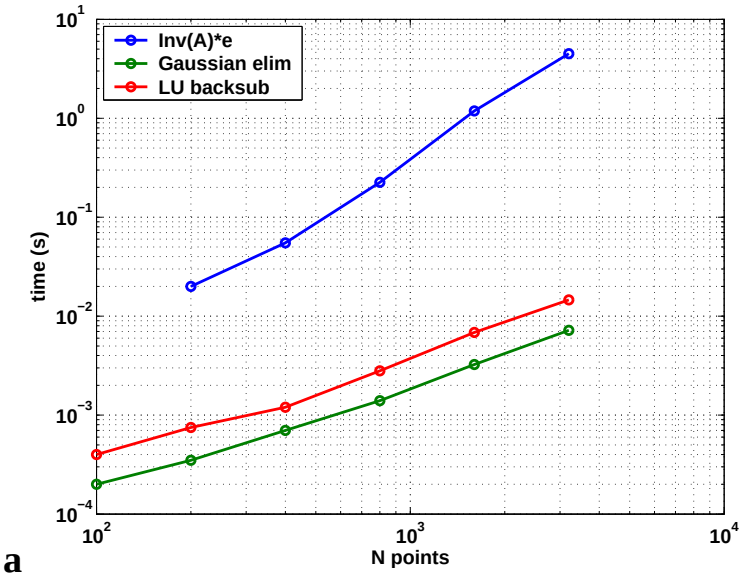
Fortunately, A is symmetric and positive definite (depending on the boundary conditions) and is readily inverted. Better yet, tridiagonal matrices can be inverted in order N operations where N is the total number of grid points. To understand what a blessing this is, normal dense matrix inversion requires order N^3 operations and rapidly becomes unfeasible for even moderately large grids. Numerical recipes (and netlib) provide efficient implementations of tridiagonal solvers and Matlab now handles them simply using Gaussian elimination via (`x=A\b`) (previous versions did this poorly and it was necessary to use an explicit LU solver (see Figure 6.2b). Whatever you do, however, never use `x=inv(A)*b` as this is completely unecessary in both time and storage.

Given an efficient solver, , this scheme is unconditionally stable so there is no critical time-step at which the scheme explodes. This can be seen by noting that Eq. (6.3.8 is an iterative scheme that can be rewritten as

$$T^n = B^n T^1 \tag{6.3.9}$$

where $B = A^{-1} = (I + \beta L)^{-1}$ and L is the standard 1-D Laplacian (negative second derivative operator). As B is diagonalizable, the general solution to Eq. (6.3.9) is just $T^n = S\Lambda^n S^{-1} T^1$ where S is a matrix of eigenvectors of B and Λ is a diagonal matrix of corresponding eigenvalues. Clearly, this scheme will be stable as long as the eigenvalues of B are positive and ≤ 1. For this particular matrix, this will always be the case for all values of β because L is positive definite (see [1]). The largest eigenvalue of B is

$$\lambda_{\max} = \frac{1}{1 + \beta\lambda_{\min}^L} \leq 1 \tag{6.3.10}$$



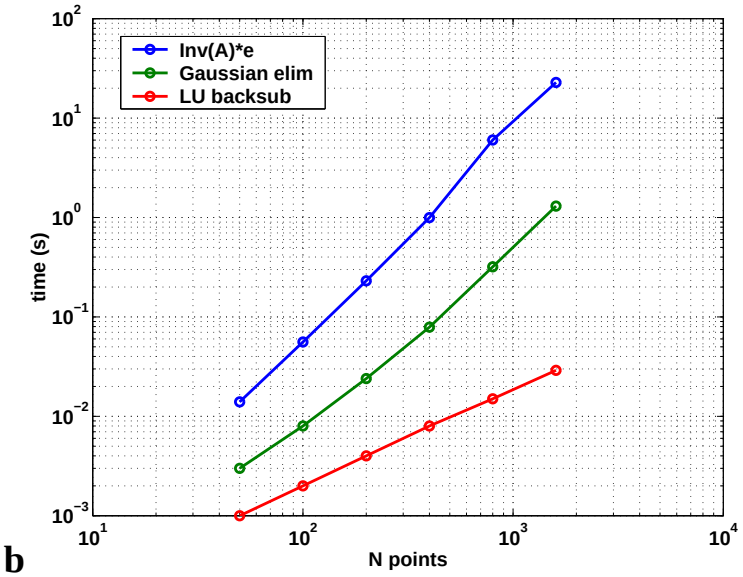


Figure 6.2: Comparison of solution times for three different solutions of $\mathsf{A}\mathbf{x} = \mathbf{b}$ where A is tridiagonal. The blue curve calculates full inverse of A which scales as N^2, the green curve uses Gaussian elimination, and the red curve uses LU decomposition and back substitution. For good solvers these last two should scale as order N. Figure 6.2a shows results from a recent version of matlab (6.5.1, 2004) which shows the proper scaling while 6.2b shows performance from an older version of matlab (in 2002) where Gaussian Elimination was actually slower than the LU scheme.

where $\lambda^L_{\min}$ is the smallest eigenvalue of L.

If you want the scheme to be accurate, however, requires taking time steps

$$\Delta t < \frac{\lambda^2}{4\pi^2\kappa} \tag{6.3.11}$$

where λ is the wavelength of interest which for a well resolved feature should be at least 3–4 Δx. This scheme, however is still only first order in Δt and therefore halving Δt will only improve the accuracy by a factor of two. A better second order scheme is the *Crank-Nicholson scheme*.

6.3.3 Crank-Nicholson schemes

The Crank-Nicholson scheme is a second order scheme because it uses the diffusion at a time step that is centered in time between t^n and t^{n+1} (remember that centered differences are second order while forward and backward differences are first order). Symbolically we can write the CN scheme as

$$T_j^{n+1} = T_j^n + \beta\Big[\begin{array}{ccc} 1 & -2 & 1 \end{array}\Big]T_j^{n+1/2} \tag{6.3.12}$$

Unfortunately, we don't have a solution for the half time step, however, we can estimate it as the average of the values at time T^n and T^{n+1} i.e.

$$T_j^{n+1/2} = \frac{1}{2}\left(T_j^n + T_j^{n+1}\right) \tag{6.3.13}$$

Substituting into (6.3.12) and rearranging gives us a new matrix equation

$$\Big[\begin{array}{ccc} -\frac{\beta}{2} & (1+\beta) & -\frac{\beta}{2} \end{array}\Big]T_j^{n+1} = \Big[\begin{array}{ccc} \frac{\beta}{2} & (1-\beta) & \frac{\beta}{2} \end{array}\Big]T_j^n \tag{6.3.14}$$

or multiplying by 2 as

$$\Big[\begin{array}{ccc} -\beta & 2(1+\beta) & -\beta \end{array}\Big]T_j^{n+1} = \Big[\begin{array}{ccc} \beta & 2(1-\beta) & \beta \end{array}\Big]T_j^n \tag{6.3.15}$$

which can be solved by tridiagonal inversion. Note: although the RHS of (6.3.14) is more complicated it is still known. This scheme is also unconditionally stable (with all the previous caveats in place). For simple diffusional problems, this is probably your best bet.

6.3.4 Boundary conditions for implicit schemes

Implicit schemes also require boundary conditions at possibly several time steps. Fortunately, they are easy to implement by modifying the form of the matrix equation. In the simplest form, Dirichlet conditions modify the RHS side vector (i.e. **b**) while Neumann and periodic boundary conditions modify the matrix A. For dirichlet conditions, consider the fully implicit stencil for the first unknown T_2. Expanding (6.3.6) we for $j = 2$ gives

$$-\beta T_1^{n+1} + (1+2\beta)T_2^{n+1} - \beta T_3^{n+1} = T_2^n \tag{6.3.16}$$

Since we already know the value of T_1^{n+1}, then we can move it over to the RHS and rewrite the equation as

$$(1+2\beta)T_2^{n+1} - \beta T_3^{n+1} = T_2^n + \beta T_1^{n+1} \tag{6.3.17}$$

Thus in matrix form we simply add βT_1^{n+1} to b_1 and βT_N^{n+1} to b_N. Crank nicholson schemes produce similar modifications. An alternative approach for dirichlet conditions that is often easier to implement is to change both the stencil and the RHS for the *Dirichlet points*, i.e. if T_1 is dirichlet and equals T_b then an appropriate stencil for this point is simply

$$\begin{bmatrix} 0 & 1 & 0 \end{bmatrix} T_1 = T_b \tag{6.3.18}$$

and then Eq. (6.3.16) can just be solved as an interior point.

For Neumann conditions, the first unknown is the boundary temperature T_1 and we need to use the auxilary equation (6.2.10). Using this equation for the RHS of the diffusion equation in a fully implicit scheme (with constant κ) gives

$$T_1^{n+1} = T_1^n + \beta\left(-2T_1^{n+1} + 2T_2^{n+1} - 2\Delta x F_1\right) \tag{6.3.19}$$

or in stencil notation as

$$\begin{bmatrix} 0 & 1+2\beta & -2\beta \end{bmatrix} T_1^{n+1} = T_1^n - 2\beta\Delta x F_1 \tag{6.3.20}$$

where F_1 is the heat flux at the $j = 1$ boundary. Thus a full Neumann boundary modifies the coefficient of T_2 in the matrix A and changes the RHS vector $\mathbf{b}$ by an amount proportional to the flux. If the boundary is a no-flux boundary, only the matrix is modified. At the $j = N$ boundary a similar thing occurs, however it is the coefficient of the first interior point T_{N-1}^{n+1} that is modified. Mixed boundary conditions and Crank Nicholson schemes are readily generalized (although in CN schemes, boundary conditions are needed at both times n and $n+1$).

Wraparound boundaries, are a bit more difficult, because they add off-diagonal terms to the matrix A and make it ever-so-slightly non-tridiagonal (by 2 points). This may seem to be trivial but it is enough to destroy the utility of the standard tridiagonal inversion. This change is transparent in matlab but requires a different routine than `tridag` in numerical recipes. Fortunately, this feature occurs so frequently that people more clever than myself have a fix for it. Methods for solving these *cyclic* tridiagonal problems using a Sherman-Morrison correction are discussed in detail in Numerical recipes (2nd edition) where a subroutine `cyclic` is given. Due to the nature of the correction, cyclic tridiagonal systems take approximately twice as much work to invert as simple tridiagonal systems. Nevertheless, for the gain in stability in time stepping, a cyclic crank-nicholson scheme is still more efficient and accurate than an explicit scheme.

6.4 Non-constant diffusivity

Much of the discussion here has considered constant diffusivities although the generalizations to non-constant diffusivities are straightforward using the control volume approach. This scheme is always flux conservative, however sometimes, if

$\kappa(x,t)$ is known analytically[2], then it can be more accurate to expand the diffusion term by the chain-rule to get

$$\frac{\partial \kappa}{\partial x}\frac{\partial T}{\partial x} + \kappa \frac{\partial^2 T}{\partial x^2} \tag{6.4.1}$$

which introduces an effective advection term. Whatever you do, never make the mistake of simply writing the diffusion term as

$$\kappa(x)\frac{\partial^2 T}{\partial x^2} \tag{6.4.2}$$

as this is simply incorrect. When in doubt, start deriving the governing equations from the integral form of the conservation equations and you'll never go wrong. Non-constant diffusivities can cause apparent source terms and strange behaviour (as well as some artifacts). Some care should be taken to understand the effects. Another difficulty may arise if the diffusivities are temperature dependent as this will lead to non-linear equations. The non-linearites are not difficult to deal with using explicit schemes but cause difficulties for implicit schemes. Several options are to linearize the equations and iterate, or a more powerful option is to use a non-linear relaxation scheme such as FAS multi-grid (which we will deal with in future lectures).

6.5 Summary

For simple 1-D diffusion use a Crank-Nicholson scheme.

Bibliography

[1] G. Strang. Introduction to Linear Algebra, Wellesley-Cambridge Press, Wellesley, MA, 2nd edn., 1998.

[2] M. Abramowitz and I. Stegun. Handbook of Mathematical Functions, vol. 55 of Applied Maths series, National Bureau of Standards, Washington, 1964.

[2] Actually, if the diffusivity is only a function of space $\kappa(x)$ and it is sufficiently simple analytically, then there may be an analytic solution in terms of the eigenfunctions of the separated equations. When in doubt check the bible [2].