Numerical Tools for Physical Scientists: Partial Differential Equations

Erik Spence

SciNet HPC Consortium

6 March 2014

Today's class

Today we will discuss the following topics:

- Basic approaches to solving PDEs.
- How to approach the temporal part of the equations.
- How to approach the spatial part of the equations.
- Assignment 8.

SciNet compute • calcul CANADA

Partial Differential Equations

Partial differential equations (PDEs) are differential equations which contain derivatives of more than one variable.

$$A\frac{\partial^2\Phi}{\partial x^2} + B\frac{\partial^2\Phi}{\partial x\partial y} + C\frac{\partial^2\Phi}{\partial y^2} = F\left(x, y, \Phi, \frac{\partial\Phi}{\partial x}, \frac{\partial\Phi}{\partial y}\right)$$

For A, B, C constant, three classes of PDEs show up repeatedly in physical systems.

- If $B^2 - 4AC < 0$, the equation is called *elliptic*.
- If $B^2 - 4AC = 0$, the equation is called *parabolic* (diffusive).
- If $B^2 - 4AC > 0$, the equation is called *hyperbolic* (wavelike).

SciNet compute • calcul CANADA

How do we solve these problems?

Let's look at parabolic equations; in particular, let us look at the heat equation.

$$\frac{\partial T}{\partial t} = D\frac{\partial^2 T}{\partial x^2},$$

where T is the temperature and k is the thermal diffusivity.

How do we solve this equation? By discretizing in both space and time, and marching an initial condition forward in time.

Dealing with time

Rewrite the right-hand side as an operator:

$$\frac{\partial T}{\partial t} = k\frac{\partial^2 T}{\partial x^2} \quad \rightarrow \quad \frac{\partial T}{\partial t} = FT$$

Basic approaches to dealing with the time part of the equation:

- Explicit methods, such as forward Euler:

$$\frac{\partial T_{i+1}}{\partial t} = FT_i \quad \rightarrow \quad T_{i+1} = (1 + hF)T_i$$

- Implicit methods, such as backward Euler:

$$\frac{\partial T_{i+1}}{\partial t} = FT_{i+1} \quad \rightarrow \quad (1 - hF)T_{i+1} = T_i$$

Where $h = \Delta t$ is our timestep.

Explicit Methods

Methods are called explicit when only T_i is on the right side of the equation. Explicit methods have some nice features:

- They are very easy to implement.
- They are usually quick to calculate (no matrix inversions).
- Easier to parallelize, since the calculation is inherently local.
- There exist more-accurate explicit methods than forward Euler.

$$\frac{\partial T_{i+1}}{\partial t} = FT_i \quad \rightarrow \quad T_{i+1} = (1 + hF)T_i$$

But there are some serious downsides as well:

- They are not very accurate at low order ($\mathcal{O}(h)$ for forward Euler).
- They can be numerically unstable (though there are exceptions).

Runge-Kutta methods

A commonly-used class of explicit methods are the Runge-Kutta methods. Assume we have a differential equation of the form

$$\frac{\partial y(t)}{\partial t} = f(t, y)$$

Then the fourth-order Runge-Kutta (RK4) method is given by

$$y_{i+1} = y_i + (k_1 + 2k_2 + 2k_3 + k_4)/6$$

where

$$\begin{aligned} k_1 &= hf(t, y_i) \\ k_2 &= hf(t + h/2, y_i + k_1/2) \\ k_3 &= hf(t + h/2, y_i + k_2/2) \\ k_4 &= hf(t + h, y_i + k_3) \end{aligned}$$

Runge-Kutta methods, continued

Notes about RK4:

- y_{i+1} is y_i plus a weighted average of four different estimates of y, at different intervals.
- The terms are derived from repeated Taylor expansions of y.
- But the terms themselves do not require calculating the derivatives of y.
- The method is globally accurate to $\mathcal{O}(h^4)$.
- The method was presented as an ODE, but can be generalized to PDEs.
- The method is not necessarily stable, but more stable than forward-Euler.
- Commonly implemented in standard C++ numeric packages (BOOST, GSL).

SciNet compute • calcul CANADA

The timestep size

Consider again the heat equation, where the spatial coordinate is given by the second index, $T_{i,j} = T(t_i, x_j)$.

$$\frac{\partial T_{i,j}}{\partial t} = k\left[\frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta x^2}\right]$$

which then becomes

$$T_{i+1,j} = T_{i,j} + \frac{k\Delta t}{\Delta x^2}\left[T_{i,j+1} - 2T_{i,j} + T_{i,j-1}\right]$$

what are the conditions for this to be stable? Look at the eigenvalues of the operator and you'll find that for $|\lambda| \leq 1$ we require $\frac{k\Delta t}{\Delta x^2} \leq 0.5$.

$$\Delta t \leq \frac{\Delta x^2}{2k}$$

If you double the spatial resolution, you need to quadruple the temporal resolution, for stability.

The CFL condition

The Courant-Friedrichs-Lewy (CFL) condition is a necessary condition for numerical stability for hyperbolic PDEs, in explicit timestepping schemes. Consider the heat equation with advection:

$$\frac{\partial T(t)}{\partial t} + (\mathbf{v} \cdot \nabla T) = f(t, T)$$

where $\mathbf{v}$ is a velocity field. Then the CFL condition, in 1D, is given by

$$C = \frac{u \Delta t}{\Delta x} \leq C_{\max} \quad \rightarrow \quad u \leq \frac{\Delta x}{\Delta t} C_{\max}$$

where u is the speed in the x direction. The value of $C_{\max}$ depends on the method, but in general for explicit methods cannot be greater than 1.

Implicit methods

Methods are called implicit when T_{i+1} is on both sides of the equation. Examples include backward Euler:

$$\frac{\partial T_{i+1}}{\partial t} = FT_{i+1} \qquad \mathcal{O}(h)$$

and Crank-Nicolson:

$$\frac{\partial T_{i+1}}{\partial t} = (FT_{i+1} + FT_i)/2 \qquad \mathcal{O}(h^2)$$

Implicit time stepping methods have some nice features:

- They are stable over a wide range of timestep sizes, sometimes unconditionally (not unconditionally accurate, though).
- Excellent for solving steady-state problems.

Downsides include being more difficult to code, and much more difficult to parallelize (inverting the operator depends upon all grid points).

Dealing with space

We've talked about time. Now, what about space? How can we deal with the spatial aspects of the problem? How should set up our spatial discretization?

Let's consider the usual approaches, when one is dealing with fields.

- Finite difference methods.
- Finite volume methods.
- Spectral methods.

SciNet compute • calcul CANADA

Finite difference methods

Finite difference methods are the simplest way to deal with your equations. You've seen finite differences before, in assignments 2 & 3.

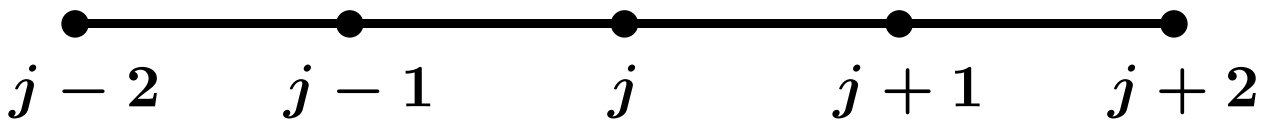
- The simulation domain is discretized.
- Derivatives are approximated by linear combinations of function values at the grid points.

We've used 'central differences' before, to calculate our second derivatives. Where did that come from?

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta x^2}$$

Calculating derivatives

We need to calculate the second spatial derivatives. How best to do that? We discretize the x domain, and examine the Taylor expansion of some function f, centered around three different points:



$$
\begin{aligned}
f(x_{j-1}) &= f(x_j) - (\Delta x)\frac{\partial f(x_j)}{\partial x} + \frac{(\Delta x)^2}{2!}\frac{\partial^2 f(x_j)}{\partial x^2} + \mathcal{O}(\Delta x^3) \\
f(x_j) &= f(x_j) \\
f(x_{j+1}) &= f(x_j) + (\Delta x)\frac{\partial f(x_j)}{\partial x} + \frac{(\Delta x)^2}{2!}\frac{\partial^2 f(x_j)}{\partial x^2} + \mathcal{O}(\Delta x^3)
\end{aligned}
$$

Where $\Delta x = x_j - x_{j-1}$.

Calculating derivatives, continued

We can write this as a matrix operation:

$$\begin{bmatrix} f(x_{j-1}) \\ f(x_j) \\ f(x_{j+1}) \end{bmatrix} = \begin{bmatrix} 1 & -\Delta x & \Delta x^2 \\ 1 & 0 & 0 \\ 1 & \Delta x & \Delta x^2 \end{bmatrix} \begin{bmatrix} f(x_j) \\ f'(x_j) \\ f''(x_j) \end{bmatrix}$$

To get the answer we invert the matrix:

$$\begin{bmatrix} f(x_j) \\ f'(x_j) \\ f''(x_j) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{-1}{2\Delta x} & 0 & \frac{1}{2\Delta x} \\ \frac{1}{\Delta x^2} & \frac{-2}{\Delta x^2} & \frac{1}{\Delta x^2} \end{bmatrix} \begin{bmatrix} f(x_{j-1}) \\ f(x_j) \\ f(x_{j+1}) \end{bmatrix}$$

$$f'(x_j) = \frac{\partial f(x_j)}{\partial x} = \frac{f(x_{j+1}) - f(x_{j-1})}{2\Delta x}$$

$$f''(x_j) = \frac{\partial^2 f(x_j)}{\partial x^2} = \frac{f(x_{j+1}) - 2f(x_j) + f(x_{j-1})}{\Delta x^2}$$

Finite volume methods

Finite volume methods are based on the discretization of the integral forms of the equations, rather than the differential form.

- Convert all divergences into fluxes through the surfaces of each volume.
- Create a grid on which we will evaluate our fields. "Finite volume" refers to the small volume of space which surrounds each grid point. These are called "control volumes" (CVs).
- Discretize the equations on the CVs.
- Solve. Make sure your fluxes balance.

SciNet compute • calcul CANADA

Finite volume methods, continued

So what have you done to the equations?

$$\frac{\partial T}{\partial t} = k\nabla^2 T = k\left(\nabla \cdot \nabla T\right)$$

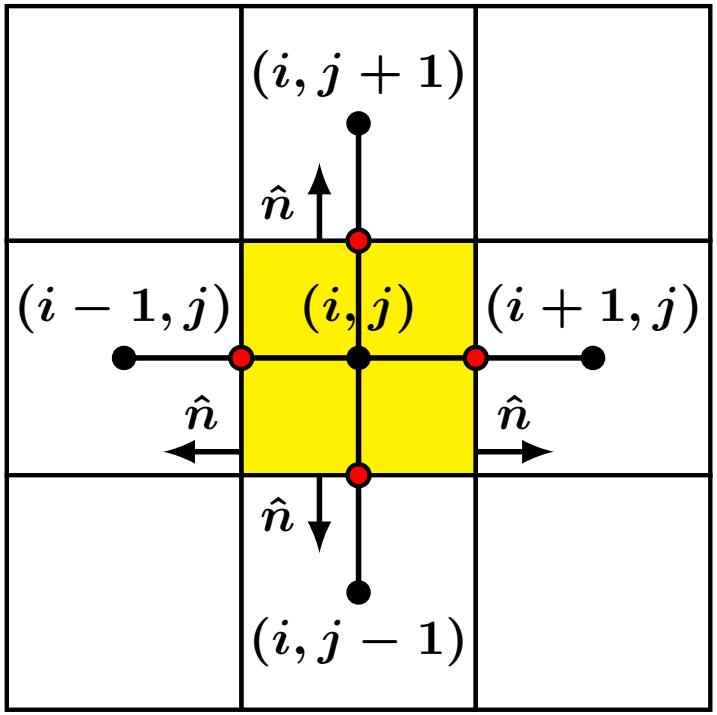
$$\begin{aligned}\frac{\partial}{\partial t}\int T dV &= k\int \left(\nabla \cdot \nabla T\right) dV \\ &= k\int \left(\nabla T \cdot \hat{n}\right) dS\end{aligned}$$

where we have used the divergence theorem, and $\hat{n}$ is the unit vector normal to the CV's surfaces.

Finite volume methods, continued

$$\frac{\partial}{\partial t}\int T dV = k\int (\nabla T\cdot\hat{n})\, dS$$

- T is calculated at the center, ∇T at the surfaces.
- Assume that T is constant throughout the control volume and ∇T constant across the surface.
- Dot ∇T with the normal vectors $\hat{n}$.



$$\begin{aligned}\Delta x^2\frac{\partial T}{\partial t} &= k\left[\left(\frac{T_{i,j}-T_{i-1,j}}{\Delta x}\right)(-1)+\left(\frac{T_{i+1,j}-T_{i,j}}{\Delta x}\right)+\right.\\ &\quad\left.\left(\frac{T_{i,j}-T_{i,j-1}}{\Delta x}\right)(-1)+\left(\frac{T_{i,j+1}-T_{i,j}}{\Delta x}\right)\right]\Delta x\\ &= k\left(T_{i-1,j}+T_{i,j-1}+T_{i+1,j}+T_{i,j+1}-4T_{i,j}\right)\end{aligned}$$

Advantages of finite volume methods

Finite volume methods have a number of advantages:

- The methods ensure that all quantities are conserved, both locally and globally, assuming that all surface fluxes balance.
- The methods are easily adapted to irregular meshes, or unstructured meshes (arbitrary polyhedra or polygons).
- The methods can be especially powerful in cases where the mesh moves or adapts to the simulation.

For the case where the grid is uniform, the finite volume method reduces to the finite difference method.

Spectral methods

Spectral methods involve expanding fields in terms of some orthogonal basis set. Sinusoids are often use in situations which are periodic (periodic boxes):

$$T(t,x) = \sum_{n=0}^{n_{\max}} a_n(t) \sin\left(\frac{2\pi n x}{L}\right) + b_n(t) \cos\left(\frac{2\pi n x}{L}\right)$$

For $0 \leq x \leq L$.

If your geometry is spherical, you might expand in spherical harmonics:

$$T(t,r,\theta,\phi) = \sum_{l,m} a_{l,m}(t,r) Y_{l,m}(\theta,\phi)$$

You might also expand your r dependence spectrally:

$$T(t,r,\theta,\phi) = \sum_{l,m,\alpha} a_{l,m,\alpha}(t) N_\alpha(r) Y_{l,m}(\theta,\phi)$$

Why would you do that?

There are a number of reasons why you might solve your equations spectrally:

- The equations may be easier to solve, especially if the operators are eigenfunctions of the basis in question.
- The expansions are global in nature. If your boundary conditions are global (such as magnetic boundary condtions), this may be the only way to easily satisfy them.
- They can be very accurate, in certain cases converging exponentially to exact solutions.
- Because they are accurate, fewer grid points are needed, resulting in less memory being needed.
- If conversion between real-space and spectral-space is needed to be regularly performed as part of the algorithm, advanced algorithms exists for some of these transforms (FFTs).

SciNet compute • calcul CANADA

Why wouldn't you do that?

Spectral methods have some downsides as well:

- The implementations can be more difficult to code.
- Because the expansions are global in nature, the domain needs to match the expansion basis. Complicated geometries lose accuracy, since the expansion must be long.
- Nonlinear terms can be very slow to be calculated.

SciNet compute • calcul CANADA

Assignment 8

The Fourier transform of the 1D diffusion equation leads to

$$\frac{\partial T}{\partial t} = D\frac{\partial^2 T}{\partial x^2} \quad \rightarrow \quad \frac{\partial \hat{T}}{\partial t} = -Dk^2\hat{T}$$

where we are now diffusing heat instead of density. The solution is

$$\hat{T}(t,k) = \hat{T}_0(k)e^{-Dk^2t}.$$

Write a program which models the following heating of a 1D 'rod', and evolves the temperature profile in Fourier space:

- Initialize the temperature profile in real space, with $T(0,x) = 0$, except with $T(0, L/2) = 1$.
- Transform the temperature profile to Fourier space.
- Step foward in time. Output the real space temperature profile every timestep.
- Every 10 timesteps add 1 to the $x = L/2$ temperature gridpoint.

Assignment 8, continued

Please use the following parameters:

- $L = 2.0$ (length of rod).
- $n = 40$ (number of grid points).
- $D = 10.0$ (coefficient of diffusivity).
- $dt = 1e-4$ (timestep size).
- $num = 50$ (total number of timesteps).

Also note that, due to the fact the we are not transforming over $(-\infty, \infty)$, but rather over $[0, L]$, the values of k must be modified to account for periodicity:

$$k[i] = \begin{cases} 2\pi i/L, & 0 \leq i < n/2 \\ 2\pi(n-i)/L, & n/2 \leq i < n \end{cases}$$

Submit your code, Makefile, 'git log' output. Remember to comment your code.