Math 563 Lecture Notes
Numerical methods for boundary value problems

Jeffrey Wong

April 22, 2020

Related reading: Leveque, Chapter 9.

1 PDEs: an introduction

Now we consider solving a **parabolic PDE** (a time dependent diffusion problem) in a finite interval. For this discussion, we consider as an example the heat equation

$$\begin{aligned} u_t = u_{xx}, \quad & x \in [0, L],\, t > 0 \\ u(0,t) = u(L,t) = 0, \quad & t > 0 \\ u(x,0) = f(x) & \end{aligned} \tag{HD}$$

to be solved for $x \in [0, L]$ and $t > 0$. Due to the initial condition at $t = 0$, we can think of the heat equation as taking an initial function $u(x, 0) = f(x)$ and evolving it over time, i.e. the function $u(\cdot, t)$ defined in the interval $[0, L]$ is changing as t increases.

1.1 The method of lines

The fact that the heat equation looks like an IVP of sorts for the function $u(\cdot, t)$ suggests that IVP can be used. To do so, we can use the **method of lines**:

1) Discretize the PDE in **space** on a grid $\{x_j\}$

2) Write down a system of ODEs for the functions $u(x_j, t)$ from the discretization, BCs

3) Solve the system of IVPs for $u(x_j, t)$ (by any reasonable method)

We have already seen step (1) in discussing finite differences. The process is similar here. Take a uniform mesh

$$x_j = jh, \quad h = L/(N + 1)$$

and let $u_j(t)$ denote the approximation to $u(x_j, t)$. Using centered differences, we get the (space) discretized system

$$\frac{du_j}{dt} = \frac{u_{j+1}(t) - 2u_j(t) + u_{j-1}(t)}{h^2}, \quad j = 1, \cdots N$$

along with $u_0(t) = u_N(t) = 0$ from the BCs. Letting $\mathbf{u} = (u_1, \cdots, u_N)$, we have a system

$$\frac{d\mathbf{u}}{dt} = A\mathbf{u}, \qquad A = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & -2 & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{bmatrix}$$

with initial conditions $u_j(0) = f(x_j)$. This is just a system of ODEs that can be solved by standard methods. The catch is that because the system is large and A has a particular structure, some methods may work better than others. Two possibilities are

- **Forward time, centered space** (FTCS), obtained using Euler's method. Letting u_j^k denote $u(x_j, t_k)$, we get

$$\frac{u_j^{k+1} - u_j^k}{\Delta t} = \frac{u_{j+1}^k - 2u_j^k + u_{j-1}^k}{\Delta x^2}$$

- **Backward time, centered space** (BTCS), obtained using Backward Euler. Letting u_j^k denote $u(x_j, t_k)$, we get

$$\frac{u_j^k - u_j^{k-1}}{\Delta t} = \frac{u_{j+1}^k - 2u_j^k + u_{j-1}^k}{\Delta x^2}$$

Finite difference formulas can be represented by a useful diagram called a **stencil**. Stencils for FTCS and BTCS are shown below; they depict which step is the 'current time' (indicating which methods are explicit/implicit) and which grid points are involved with the PDE approximation at each (x, t).

1.2 stability: the hard way

To better understand the method, we need to understand its stability. Unfortunately, applying stability theory for IVPs requires knowing the structure of the eigenvalues of A. Let's see how this would go for the FTCS scheme.

For the FTCS scheme applied to the Dirichlet problem (HD), we have

$$\mathbf{u}^k = \mathbf{u}^{k-1} + \frac{\Delta t}{\Delta x^2} A\mathbf{u}^{k-1}.$$

Letting $\beta = \Delta t/\Delta x^2$, we get

$$\mathbf{u}^k = V\mathbf{u}^{k-1}, \quad V = I + \beta A = \begin{bmatrix} 1-2\beta & \beta & 0 & \cdots & 0 \\ \beta & 1-2\beta & \beta & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \beta & 1-2\beta & \beta \\ 0 & \cdots & 0 & \beta & 1-2\beta \end{bmatrix}$$

If β is small enough then V is diagonally dominant. However, we really want to show that

$$\|V\| \leq 1 \text{ or } \max|\lambda_j| \leq 1$$

to demonstrate that the method is stable. Note that if $\beta < 1/2$ then $\|V\|_\infty = 1$ (consider the maximum absolute row sum), so

$$\|\mathbf{u}^k\|_\infty \leq \|\mathbf{u}^{k-1}\|_\infty.$$

However, this is unsatisfactory, as it is just a sufficient condition. To be more precise, we would need to bound the eigenvalues, which requires a better understanding of the eigenvectors/values. While such an analysis is possible, there is a less rigorous but simpler way to show stability.

1.3 Convergence

First, we should identify what is the right notion of stability. The definition of convergence is the same as in ODEs. Suppose $U_n^k \approx u(x,t)$ approximates the solution to a PDE in an interval $[0,T]$, using time/space steps Δt and Δx. Then if

$$\max_k |U_n^k - u(x_n, t_k)| = O(\Delta x^p + \Delta t^q) \text{ for } k \text{ such that } k\Delta t \leq T \tag{1.1}$$

we say the method converges with **order** p in space (x) and **order** q in time (t).

Convergence matters in the limit when both Δx and Δt go to zero. Technically, the bound (1.1) only needs to hold for Δ's taken to zero at certain rates. In practice, there is some link between the two, e.g.

$$\Delta t = C\Delta x^2.$$

which allows us to think of convergence in the limit of $\Delta t \to 0$ (rather than as $\Delta t, \Delta x \to 0$), with the other Δ tied to the first.

Recall that for ODEs, convergence requires two conditions - consistency and zero-stability. A similar result holds for linear PDEs like the heat equation as well, but we need to identify what is meant by 'stability'.

First, the consistency part is easy - it is the same as for ODEs. The local truncation error is the error obtained by plugging the exact solution into the difference equation (with the same 'divide by Δt' convention). For instance, for

$$u_t = u_{xx}$$

the FTCS method

$$\frac{U_n^{k+1} - U_n^k}{\Delta t} = \frac{U_{n+1}^k - 2U_n^k + U_{n-1}^k}{\Delta x^2}$$

has a truncation error defined by

$$\frac{u_n^{k+1} - u_n^k}{\Delta t} = \frac{u_{n+1}^k - 2u_n^k + u_{n-1}^k}{\Delta x^2} + \tau_{nk}$$

where $u_n^k = u(x_n, t_k)$ denotes the exact solution. Using the error calculations from finite differences for derivatives, we obtain

$$\begin{aligned}\tau &= \frac{\Delta t}{2} u_{tt}(x_n, t_k) - \frac{\Delta x^2}{12} u_{xxxx}(x_n, t_k) + \cdots \\ &= (\frac{\Delta t}{2} - \frac{\Delta x^2}{12}) u_{xxxx} + O(\Delta t^2 + \Delta x^4).\end{aligned}$$

Note that the PDE $u_t = u_{xx}$ was used to simplify $u_{tt} = u_{xxt} = (u_t)_{xx} = u_{xxxx}$. In particular,

$$\tau = O(\Delta t + \Delta x^2)$$

if the solution u is smooth enough (which is the case because according to the properties of the heat equation). Thus, the method is consistent, **first order in time** and **second order in space**.

Stability is more subtle, and departs from the ODE case a bit. Suppose a method for a linear PDE takes the form

$$\vec{U}^{k+1} = V\vec{U}^k + b \tag{1.2}$$

where V, b may depend on Δx and Δt. Further assume that Δx is tied to Δt in the limit $\Delta t \to 0$ (e.g. $\Delta x = \Delta t^2$). Then the method is called (Lax-Richtmeyer) **stable** in a time interval $[0, T]$ if there is a constant C_T such that

$$\|V^k\| \le C_T, \qquad \text{for all integers } k \text{ such that } k\Delta t \le T$$

for sufficiently small values of Δt. The constant cannot depend on $k, \Delta t$. Ideally, we would like to have the bound

$$\|V\| \le 1 + C\Delta t$$

from which it would follow (same argument as for ODEs) that

$$\|V^k\| \le \|V\|^k \le (1 + C\Delta t)^k \le e^{Ck\Delta t} \le e^{CT}.$$

This definition leads to an important theorem:

Lax Equivalence Theorem: A difference method for a linear PDE of the form (1.2) is convergent as $\Delta x, \Delta t \to 0$ if it is consistent and stable in that limit.[1]

Note that the theory applies only for **linear** PDEs, for which the associated numerical method will be a linear iteration like (1.2). For non-linear PDEs, the principle here is still useful, but the theory is much more challenging since non-linear effects can change stability.

1.4 Connection to ODEs

Recall that for initial value problems, we had

$$\text{consistency} + \text{zero-stability} \implies \text{convergence}.$$

Bounds on eigenvalues and 'growth rates' were the subject of **absolute stability**, which could be separated from the convergence theorem. The reason is that convergence requires the error to go to zero **in the limit** as $\Delta t \to 0$. The 'eigenvalues' for the stability condition are **independent of** Δt (e.g. $\partial f/\partial y$ for $y' = f(y)$), so the absolute stability condition is automatically satisfied for small enough Δt.

On the other hand, the Lax 'stability' bound is really an absolute stability condition and is required for convergence. For instance, a sufficient condition is

$$|\lambda_j| \leq 1 \text{ for all eigenvalues of } V$$

to have $\|V\|_2 \leq 1$. This may not be intuitive, but the method of lines provides some insight. Recall that if we use the method of lines to solve

$$u_t = u_{xx}$$

with grid points x_n, $n = 0, \cdots N$, the ODEs for $U_n(t) = u(x_n, t)$ are

$$\vec{U}'(t) = A\vec{U}(t), \quad A = \frac{1}{\Delta x^2}\begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & -2 & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{bmatrix}$$

for $\vec{U} = (U_1, \cdots, U_N)$. The eigenvalues of A depend on Δx - and in particular, the largest in magnitude grows as $\Delta x \to 0$. Thus, the relevant eigenvalues for absolute stability change as the grid is refined ($\Delta x \to 0$), as do the 'stiffness' requirements.

This is why the FTCS method requires $\Delta t \leq C\Delta x^2$: as $\Delta x \to 0$, the largest eigenvalue in the 'method of lines' ODE system grows as $\Delta x \to 0$; the Forward Euler time-step must be made small to compensate.

2 Von Neumann stability analysis

In general, obtaining a bound

$$\|V^k\| \leq C$$

for a family of matrices V depending on Δx etc. is difficult. The idea of **Von Neumann analysis** is to forgo rigor in favor of deriving a simple, practical result.

On an infinite domain, a function $u(x,t)$ can be written in terms of its Fourier transform $\hat{u}(\omega, t)$ by

$$u(x,t) = \int_{-\infty}^{\infty} \widehat{u}(\omega, t) e^{i\omega x}\, d\omega$$

and derivatives in x are converted to multiplication by $i\omega$:

$$\frac{\partial u}{\partial x} = i\omega \widehat{u}.$$

The Fourier transform converts a linear, constant coefficient PDE on an infinite interval into an ODE for its transform, e.g.

$$u_t = u_{xx} \to \widehat{u}_t = -\omega^2 \widehat{u}.$$

This suggests that we may gain insight into stability of a numerical method by considering a single 'Fourier mode'

$$U_n^k = \alpha_k e^{i\omega n \Delta x} \tag{2.1}$$

and determining the growth of the amplitude α_k over time and how it depends on ω. FIrst, we apply the idea to an example, the FTCS scheme

$$\frac{U_n^{k+1} - U_n^k}{\Delta t} = \frac{U_{n+1}^k - 2U_n^k + U_{n-1}^k}{\Delta x^2}.$$

First plug the ansatz (2.1) into the formula to get

$$(\alpha_{k+1} - \alpha_k)e^{i\omega nh} = \beta\alpha_k(e^{i\omega(n+1)h} - 2e^{i\omega nh} + e^{i\omega(n-1)h}), \qquad \beta := \frac{\Delta t}{\Delta x^2}.$$

Then simplify, yielding

$$\begin{aligned}\alpha_{k+1} &= (1 + \beta(e^{i\omega h} - 2 + e^{-i\omega h}))\alpha_k \\ &= +\beta(2\cos\omega h - 2)\alpha_k\end{aligned}$$

It follows that

$$\alpha_{k+1} = A(\omega)\alpha_k$$

for the 'amplification factor'

$$A(\omega) = 1 - 4\beta \sin^2 \frac{\omega h}{2}.$$

To ensure stability, we need

$$|A(\omega)| < 1 \text{ for all } \omega$$

or else there will be an unstable mode that may grow. This condition holds if and only if

$$\frac{2\Delta t}{\Delta x^2} \leq 1$$

which is the **stability condition** for this scheme and PDE. The required time step scales with the **square** of the grid spacing, which puts a severe restriction on Δt that can be prohibitive even for moderate accuracy (consider, e.g. $\Delta x = 10^{-3}$, integrated up to $t = 1$).

In contrast, for the backward scheme, we obtain

$$A(\omega) = \frac{1}{1 + 4\beta \sin^2 \frac{\omega h}{2}}$$

which suggests that the backward scheme is stable regardless of grid size and time step. There is an obvious advantage here! Even though one has to solve a linear system at each step (since the method is implicit), one can typically use a much larger Δt.

Here is a summary of the approach:

- Plug the 'Fourier mode' $\alpha_k e^{i\omega n x}$ into the difference equation[2]
- Solve for the growth rate of the amplitude α_k as a function of ω
- Find the condition on $\Delta t, \Delta x$ such that there is no growth for any ω (i.e. $|A(\omega| < 1$)

There are a few points worth highlighting here, on why the method is not rigorous:

- (Assumption) The PDE is linear (needed for the Fourier transform)
- (Assumption) The modes for each ω are 'decoupled', so the equation for each ω can be solved for $A(\omega)$ on its own. (This is a consequence of the Fourier transform derivative property, plus the PDE needs to be constant coefficient).
- (Hand-waving) The boundary conditions are **not** considered. This omission can be critical, as the BCs **can effect stability**.

As we saw for the test equation $y' = \lambda y$, linearization can be used to extend the result to more general equations. Von Neumann stability can be done on more complicated equations by 'freezing' other variables that are slowly varying, thus obtaining a local stability result. More caution is required for PDEs, where such approximations can omit key behavior.

In practice, Von-Neumann stability does a good job of predicting stability **when it applies**. It tends to be true that if Δt is chosen too large (so some $|A(\omega)|$ is > 1) then the numerical solution will blow up as this unstable mode grows. On the other hand, if the stability condition is satisfied, the numerical method does tend to be stable. However, to reiterate, **Von Neumann stability does not imply actual stability**; it's just a good approximation to it.

2.1 What about other equations?

Numerical methods for PDEs can be subtle because the fundamental numerical properties depend considerably on the equation (compared to ODEs). For ODEs, the test equation $y' = \lambda y$ was enough to get a general sense of how IVP solvers perform.

To illustrate the point, consider the FTCS scheme applied to the **advection** equation

$$u_t + cu_x = 0$$

with some boundary conditions (not relevant for the point). The scheme reads

$$\frac{u_j^{k+1} - u_j^k}{\Delta t} = -c\frac{u_{j+1}^k - u_{j-1}^k}{2\Delta x}.$$

Setting $\beta = \Delta t/\Delta x$ and plugging in the Von Neumann analysis ansatz, we get

$$(A-1)e^{in\omega h} = -\frac{c\beta}{2}(e^{i(n+1)\omega h} - e^{i(n-1)\omega h})$$

[2]Since the difference equation for α_k will be linear, we can also just plug in $A^k e^{i\omega n x}$ and solve for A from the start, which is the typical shortcut.

$$\implies A(\omega) = 1 - ic\beta \sin(\omega h)$$

But $|A(\omega)| > 1$ for all ω regardless of Δt or Δx, so the method is always unstable. The FTCS method relies on the diffusion in the heat equation to be stable - without it, the method fails.

Surprisingly, a simple fix can provide stability. We can replace the u_j^k on the left hand side by an average to get the 'Lax-Friedrichs' scheme

$$\frac{u_j^{k+1} - \frac{1}{2}(u_{j-1}^k + u_{j+1}^k)}{\Delta t} = -c\frac{u_{j+1}^k - u_{j-1}^k}{2\Delta x}.$$

After a calculation, one obtains an amplification factor $A(\omega)$ with the property that

$$|A(\omega)| \leq 1 \text{ for all } \omega \text{ if } c\frac{\Delta t}{\Delta x} < 1.$$

This is the **CFL condition** (short for Courant-Friedrichs-Lewy). There is a nice interpretation of the condition as follows: We know that information propagates in the advection equation at a speed exactly c. In order for a numerical scheme to work properly, it should be able to propagate information (on the grid) at least at this speed.

Since each grid point at time $t + \Delta t$ depends only on adjacent grid points at time t, information travels at a speed $\Delta x/\Delta t$ on the grid (i.e. it moves by one grid point per time step). Thus, we must have

$$\text{'grid speed'} \geq \text{advection speed} \implies \frac{\Delta x}{\Delta t} \geq c.$$

Note that the argument here relies fundamentally on the structure of the PDE - it would not apply to the diffusion equation (or even a convection-diffusion equation $u_t + cu_x = u_{xx}$), where the solution does not evolve in the same way.

The idea behind the trick in 'Lax-Friedrichs' scheme is revealed by rewriting it in the form

$$\frac{u_j^{k+1} - u_j^k}{\Delta t} = -c\frac{u_{j+1}^k - u_{j-1}^k}{2\Delta x} + \frac{\Delta x^2}{2\Delta t}\left(\frac{u_{j+1}^k - 2u_j^k + u_{j-1}^k}{\Delta x^2}\right).$$

which is the FTCS scheme applied to that 'modified PDE'

$$u_t + cu_x = \frac{\Delta x^2}{2\Delta t} u_{xx}.$$

If Δ is chosen to satisfy the CFL condition, then the coefficient on u_{xx} is small, and as $\Delta t \to 0$, it goes to zero. Thus, Lax-Friedrichs can be seen as stabilizing the transport equation by adding **artificial diffusion**, which smooths out the solution.

This insight also identifies the key disadvantage of Lax-Friedrichs. We have introduced an error in the form of diffusion, and this smoothing will tend to smear out the solution, reducing (correct) sharp features over time. Such a method is called **dissipative**.

2.2 More on modified equations

We can be a bit more precise about the 'modified equation'. Consider, for example, the advection equation

$$u_t + cu_x = 0$$

and the **Lax-Friedrichs** scheme

$$U_n^{k+1} - \frac{1}{2}(U_{n-1}^k + U_{n+1}^k) = -c\beta \frac{U_{n+1}^k - U_{n-1}^k}{2}.$$

with $\beta = \Delta t/\Delta x$. To obtain the modified equation, we 'derive in reverse', plugging in $u(x,t)$ for U and Taylor expanding. The goal is to **find the relevant PDE terms**, e.g.

$$u_t, u_x, u_{xx}.$$

Other terms might matter, depending on the analysis. For convenience, let u, u_t etc. denote values at (x_n, t_k). Using the expansions

$$U_n^{k+1} = u + \Delta t u_t + \frac{1}{2}\Delta t^2 u_{tt} + \cdots$$

$$U_{n\pm1}^k = u \pm \Delta x u_x + \frac{1}{2}\Delta x^2 u_{xx} + \cdots$$

we get

$$\Delta t u_t + \frac{1}{2}\Delta t^2 u_{tt} - \frac{\Delta x^2}{2} u_{xx} = -c\beta \Delta x u_x - c\beta \frac{\Delta x^3}{6} u_{xxx} + \cdots \tag{2.2}$$

Now replace u_{tt} with $c^2 u_{xx}$ (from the PDE) and note that $\beta \Delta x = \Delta t$; then

$$u_t = -cc u_x + (\frac{\Delta x^2}{2\Delta t} - \frac{c^2}{2}\Delta t) u_{xx} + \cdots$$

which gives us the **modified equation**

$$u_t + cu_x = \frac{\Delta x^2}{2\Delta t}\left(1 - c^2\beta^2\right) u_{xx}.$$

The equation can tell us a necessary condition for stability - if the method is stable, its modified equation should also be stable (non-negative, diffusion). Here, the diffusion term

$$(1 - c^2\beta^2) u_{xx}$$

must have a positive coefficient to have stability (so $c\Delta t/\Delta x \leq 1$). Note that we have omitted 'higher order' terms, which includes the u_{xxx} term in (2.2).

For the upwind scheme, we get a similar result, but the diffusion coefficient is different ($1 - c\beta$ instead).

Notably, the analysis lets us compare 'how diffusive' the errors are. For Lax-Friedrichs and upwind, the diffusion coefficients are

$$D_{LF} = 1 - c^2\beta^2, \qquad D_U = 1 - c\beta.$$

If, say, $c\beta = 1/2$ then $D_{LF} = 3/4$ but $D_U = 1/4$, so upwind is **much less diffusive** than Lax-Friedrichs, despite having a lower order of accuracy.

2.3 Leapfrog and more

To obtain methods that do not degrade in this way, other approaches are necessary. A full discussion of the ways to improve solvers for wave-like equations is well beyond the scope of these notes. However, one more example may be useful. Not all errors are diffusive in nature; take for instance the **leapfrog scheme** for

$$u_t + cu_x = 0$$

given by

$$\frac{U_n^{k+1} - U_n^{k-1}}{2\Delta t} = -\frac{c}{2\Delta x}(U_{n+1}^k - U_{n-1}^k)$$

or more simply, with $\beta = \Delta t/\Delta x$,

$$U_n^{k+1} - U_n^{k-1} = -c\beta(U_{n+1}^k - U_{n-1}^k)$$

That is, centered differences are used in both time and space, yielding a 'multi-step method' that is consistent with second order accuracy. Plugging in the ansatz

$$U_n^k = \alpha_k e^{i\omega nh}$$

for Von Neumann analysis, we get

$$\alpha_{k+1} - \alpha_{k-1} = -c\beta\alpha_k(e^{i\omega h} - e^{-i\omega h})$$

which gives

$$\alpha_{k+1} - \alpha_{k-1} = -2i\theta\alpha_k, \quad \theta = -c\beta \sin \omega h.$$

This difference equation has solutions $\alpha_k = A^k$ for two roots A; both need to be ≤ 1 in magnitude for stability. We get

$$A^2 + 2i\theta A - 1 = 0 \implies A = -i\theta \pm \sqrt{1 - \theta^2}.$$

It is straightforward to check that $|A| \leq 1$ if and only if $\theta \leq 1$ or, more directly,

$$\frac{\Delta t}{\Delta x} \leq c$$

which is the familiar CFL condition. Notice that in this case

$$|A|^2 = \theta^2 + 1 - \theta^2 = 1 \text{ for both roots}$$

which means that the amplification factor is **neither dissipative nor unstable**.

You may think that this is a good thing for the advection equation, and it is in the sense that the method will not diffuse like Lax-Friedrichs. However, the pure leapfrog method is dangerous to use more generally for two reasons. First, it is just on the border of unstable, so other PDE features (of any more complicated PDE) could cause trouble.

Second, the amplification factor has a magnitude of 1, but does have an imaginary part. A factor $e^{i(\cdots)}$ produces a phase shift, leading to **dispersion**: a frequency-dependent change in the wave speed. You can think of it this way: At some point in the Taylor series

$$e\frac{1}{6}\Delta^2 u_{tt} + \frac{1}{6}\Delta^3 u_{ttt} + \cdots$$

we need to truncate, and that remaining term will lead to some error, the nature of which depends on the term. For Lax-Friedrichs, we get something like $\Delta t u_{xx}$ (diffusion); for leapfrog, we get $\Delta^2 u_{xxx}$ (dispersion, no diffusion).

(The hand-waving here is made rigorous by use of a **modified equation analysis**, as we did above for Lax-Friedrichs).

3 Elliptic problems

We now consider solving **Laplace's** (or **Poisson's**) equation

$$\Delta u = f, \quad (x, y) \in \Omega$$

in a domain $\Omega \subset \mathbb{R}^2$ where $\Delta u = u_{xx} + u_{yy}$ is the Laplacian. Our 'simplest' example problem will be the homogeneous Dirichlet problem in a rectangle,

$$\begin{aligned} u_{xx} + u_{yy} &= f, \quad (x, y) \in (0, W) \times (0, H) \\ u(x, 0) = u(x, H) &= 0, \quad x \in (0, W) \\ u(0, y) = y(W, y) &= 0, \quad y \in (0, H) \end{aligned}$$

This is a PDE **boundary value problem**, in contrast to the initial value problem presented by the heat equation. The approach, therefore, is like the one-d boundary value problems we solved before:

1) Discretize the domain (in both x and y)

2) Discretize the PDE/BCs to get a system to solve for u

3) Solve the system

The extra dimension makes step (3) a bit more complicated than in 1d. Let's see how the straightforward finite difference approach extends to this problem.

First, we define a grid - uniform, for simplicity, with the same number of grid points in each direction. We set

$$x_j = j\Delta x, \; y_k = k\Delta y, \qquad \Delta x = W/(N+1), \quad \Delta y = H/(N+1).$$

Now the PDE can be discretized using centered differences on each term:

$$\frac{u_{j+1,k} - 2u_{j,k} + u_{j-1.k}}{\Delta x^2} + \frac{u_{j,k+1} - 2u_{j,k} + u_{j.k-1}}{\Delta y^2} = f_{jk}. \tag{3.1}$$

The left hand side is the **five point stencil** for the Laplacian, because it approximates Δu using five points (the least possible). The BCs here are

$$u_{0,k} = u_{N+1,k} = u_{j,0} = u_{j,N+1} = 0. \tag{3.2}$$

There are $M = N^2$ unknowns. To construct the system to solve, we must write it in the form $Ax = b$, which requires **flattening** the two-dimensional array $u_{k,j}$ into a one dimensional vector.

For simplicity, let us suppose that $\Delta x = \Delta y$, so the system is

$$u_{j+1,k} + u_{j-1,k} + u_{j,k+1} + u_{j,k-1} - 4u_{j,k} = \Delta x^2 f_{jk}, \quad 1 \le j, k \le N. \tag{3.3}$$

Let

$$\mathbf{v} = (u_{1,1}, u_{1,2}, \cdots, u_{1,N},\ u_{2,1}, u_{2,2}, \cdots u_{2,N}, \cdots, u_{N,1}, u_{N,2}, \cdots, u_{N,N})$$

which stacks the data row-by-row (from the first to the last row). Then (3.3) has the form

$$A\mathbf{v} = \Delta x^2 \mathbf{f}$$

where $\mathbf{f}$ contains the values $f(x_j, y_k)$ in the same ordering as v and A has the following structure (in terms of $N \times N$ blocks):

$$A = \begin{bmatrix} D & I_N & 0 & \cdots & 0 \\ I_N & D & I_N & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & I_N & D & I_N \\ 0 & \cdots & 0 & I_N & D \end{bmatrix}$$

Here I_N is an $N \times N$ identity block, 0 is a zero block and D is the $N \times N$ matrix

$$D = \begin{bmatrix} -4 & 1 & 0 & \cdots & 0 \\ 1 & -4 & 1 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & -4 & 1 \\ 0 & \cdots & 0 & 1 & -4 \end{bmatrix}$$

Note that the 'boundary' terms vanish because of the homogeneous Dirichlet conditions. Otherwise, there would be some terms added to the right hand side (or adjusted coefficients in the matrix for Neumann problems).

For non-homogeneous boundary conditions, the boundary terms (3.2) contribute terms to the RHS. For instance, if

$$u(x, 0) = f_1(x), \quad u(x, H) = f_2(x), \quad u(0, y) = g_1(y), \quad u(W, y) = g_2(y)$$

then the system is

$$A\mathbf{v} = \Delta x^2 \mathbf{f} + \mathbf{b} + \mathbf{c}$$

where $\mathbf{b}$ contains boundary terms from the $x = 0$ and $x = W$ boundaries and $\mathbf{c}$ contains terms from the $y = 0$ and $y = H$ boundaries:

$$\mathbf{b} = \begin{bmatrix} \vec{b}_1 \\ \vec{b}_2 \\ \vdots \\ \vec{b}_N \end{bmatrix}, \quad \vec{b}_k = \begin{bmatrix} -g_1(y_k) \\ 0 \\ \vdots \\ -g_2(y_k) \end{bmatrix} \in \mathbb{R}^N$$

$$\mathbf{c} = \begin{bmatrix} \vec{c}_1 \\ 0 \\ \vdots \\ 0 \\ \vec{c}_N \end{bmatrix}, \quad \vec{c_1} = \begin{bmatrix} -f_1(x_1) \\ -f_1(x_2) \\ \vdots \\ -f_1(x_N) \end{bmatrix}, \quad \vec{c_N} = \begin{bmatrix} -f_2(x_1) \\ -f_2(x_2) \\ \vdots \\ -f_2(x_N) \end{bmatrix}$$

Implementation note: A simple trick to avoid working out the extra boundary terms is to include explicit ghost points in the data. That is, we store the $(N+1) \times (N+1)$ grid U_n^k, including the boundary terms. Then, the FD formula (3.1) can be computed for all interior j, k.

That is, we **store** the array using all the grid points, then **compute** for interior indices. At the end of each step, we just need to ensure that the boundary values are updated, something like:

1) Populate U_n^k with initial values

2) At the m-th step of the algorithm (e.g. an iterative method), update the interior of U_n^k in the interior using (3.1) and some scheme

3) Update boundary values in U_n^k (if Dirichlet, these values don't need to change ('extend' to the boundary)

The benefit is that implementation can be greatly simplified. The disadvantages are that (i) some efficiency is lost and (ii) the properties of this 'expanded' system may be slightly different or unnatural (e.g. Neumann BCs), depending on the boundary conditions

For Neumann boundary conditions, we would need to create the ghost points **outside** the domain and store those; then the trick still works but is less natural.

3.1 Short note on linear algebra

This matrix has $O(N^2)$ elements and a bandwidth of $2N$. Because the bandwidth scales with the size (unlike the tridiagonal matrix from 1d BVPs), the LU factorization method loses some efficiency. Instead of remaining $O(M)$ (where $M = N^2$ is the number of matrix entries), it takes $O(M^2)$ operations to solve. With some work, one can reduce this (called 're-ordering strategies').

Instead, a simpler (and popular) approach is to use an **iterative method** that requires only a map

$$\mathbf{x} \to A\mathbf{x}$$

as an input, and **does not care about the structure of** A and outputs a sequence of approximations $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \cdots$ that converge to the solution. This means that only $O(M)$ operations are required per step (to do $A\mathbf{x}$ calculations), so if the iteration converges rapidly, the method is efficient.

The **SOR** method (Successive over-relaxation) can be used for Poisson's equation, and is a standard 'simple' choice; others include **MINRES** and **GMRES**. Some amount of work is required to make these methods truly efficient by pre-processing the system to be easier to solve ('preconditioning').

3.2 Relaxation (briefly)

An interesting approach is to observe that, in a region Ω,

$$0 = \nabla^2 u, \quad \text{...BCs on the boundary of } \Omega\text{...} \tag{3.4}$$

is the steady state solution of the diffusion equation

$$\frac{\partial u}{\partial t} = \nabla^2 u \tag{3.5}$$

with the same boundary conditions. It follows that we can solve the boundary value problem (3.4) by instead solving the parabolic PDE (3.5) for a long enough time that the solution nears convergence. Because the convergence rate is exponential (like $e^{-\lambda t}$ where λ is the smallest eigenvalue of $\nabla^2 u = \lambda u$ with the given BCs), and we know that the heat equation is well behaved numerically, the approach should be reasonable.

General principle: The approach here is called **relaxation**, because we are constructing a process that allows some initial guess to 'relax' to the the desired solution. Relaxation can be used in a variety of places in numerical methods - if you can't solve a problem directly, build a path to get there.

Let's, for example, consider the one dimensional BVP

$$u'' = f, \quad u(0) = u(1) = 0.$$

We instead consider the heat equation

$$u_t = u_{xx} - f, \quad u(0) = u(1) = 0.$$

Solving using the FTCS method for $U_n^K \approx u(x_n, t_k)$ gives

$$U_n^{k+1} = U_n^k + \frac{\Delta t}{\Delta x^2}(U_{n+1}^k - 2U_n^k + U_{n-1}^k) - f_n$$

with the appropriate boundary conditions. Since we do not care about the solution except after a long time, we should take the largest timestep possible, so $\Delta t = \Delta x^2/2$, which yields the iteration

$$U_n^{k+1} = U_n^k + \frac{1}{2}(U_{n+1}^k - 2U_n^k + U_{n-1}^k) - f_n.$$

We can view this as an iteration for approximations $U^{(0)}, U^{(1)}, \cdots$ that converge to a solution U^∞ of the BVP. This, in fact, is the **Jacobi iterative method** applied to the BVP directly.

From here, we can use the 'relaxation' intuition to develop better methods. Essentially, we are using numerical intuition from parabolic equations to design an iterative scheme for solving a boundary value problem. (For details: see the **SOR (successive over-relaxation)** method).

It's worth noting that in a sense, the **parabolic problem** (time-dependent) is easier to solve than the steady-state one (the pure boundary value problem) - we are permitted to start with an initial condition and run it forward in time.

3.3 Alternating direction (ADI) method

An appealing approach is to try to decompose the two-dimensional operator into simpler one-dimensional parts by a splitting scheme (see ODE notes on splitting for a review), Let's see how this might work for the heat equation

$$u_t = u_{xx} + u_{yy}, \quad 0 \le x, y \le 1, \quad u = 0 \text{ on the boundary}$$

since the Laplacian $u_{xx} + u_{yy}$ is a sum of an x-part and a y-part. As per usual, define a grid $x_j = j\Delta x$ and $y_k = k\Delta y$ with $\Delta x = \Delta y = 1/(N+1)$. Note that

$$u_t = u_{xx}, \quad 0 \le x, y \le 1$$

can be easily solved by solving the 1d problems

$$(u(x, y_k))_t = (u(x, y_k))_{xx}, \quad 0 \le x \le 1, \quad u(0, y_k) = u(1, y_k) = 0$$

for each index k (that is, we can update each horizontal 'slice' of the data separately).

To separate the directions, we use a trick similar to the operator factoring in operator splitting. Let $h = \Delta x = \Delta y$ and let

$$U^n(x, y) \approx u(x, y, t_n).$$

The BTCS method is good for the heat equation, so it makes sense to discretize time using a backward difference. Then

$$U^n = U^{n-1} + \Delta t(\frac{\partial^2 U^n}{\partial x^2} + \frac{\partial^2 U^n}{\partial y^2})$$

which, in operator notation, yields

$$\mathcal{L}U^n = U^{n-1}, \qquad \mathcal{L} = I - \Delta t(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})$$

Now 'factor' $\mathcal{L}$, observing that it looks like a product $(1+a)(1+b) = 1+(a+b)+\cdots$ to get

$$\mathcal{L} = (I - \Delta t\frac{\partial^2}{\partial x^2})(I - \Delta t\frac{\partial^2}{\partial y^2}) + O(\Delta t^2).$$

Then at each step, we can solve

$$(I - \Delta t\frac{\partial^2}{\partial x^2})V = U^{n-1}$$
$$(I - \Delta t\frac{\partial^2}{\partial y^2})U^n = V.$$

The appropriate boundary conditions must also be imposed. This strategy is the **alternating direction implicit** (ADI) method for linear equations. The various techniques we have for improving accuracy in finite differences can be employed here, although there are limitations to the success of the splitting approach.

4 Spectral methods (a first example)

Before getting into details, let's see how the Fourier transform can be used to solve PDEs numerically. To start, consider the (almost trivial) example

$$u'' = f, \quad x \in [0, 2\pi], \quad u \ 2\pi\text{-periodic.}$$

Recall the **discrete Fourier transform** and its inverse defined as

$$\hat{f}_k = \frac{1}{N}\sum_{j=0}^{N-1} f_j\omega^{jk}, \quad f_j = \sum_{k=0}^{N-1}\hat{f}_k\omega^{-jk}, \qquad \omega = e^{-2\pi i/N}$$

Let us denote by $\widehat{U}_k$ the DFT of the data U_n. From the difference equation

$$U_{n+1} - 2U_n + U_{n-1} = \Delta x^2 f_n$$

plug in the expression

$$U_n = \sum_{k=0}^{N-1}\widehat{U}_k\omega^{-nk}$$

and the expression for f_n to get

$$\sum_{k=0}^{N-1}\hat{U}_k(\omega^{-k} - 2 + \omega^k)\omega^{-nk} = \Delta x^2\sum_{k=0}^{N-1}\hat{f}_k\omega^{-nk}$$

which gives

$$\widehat{U}_k = \frac{\Delta x^2 \widehat{f}_k}{\omega^{-k} + \omega^k - 2} = \frac{\Delta x^2 \widehat{f}_k}{2\cos(2\pi k/N) - 2}. \tag{4.1}$$

Note that one can also view this calculation as transforming the difference equation using the discrete shift rule

$$(\widehat{U_{n+r}})_k = \omega^{-rk}\widehat{U}_k.$$

Thus, we may compute the approximate solution U_n as follows:

1) Compute the DFT of f

2) Compute the DFT of the solution using (4.1)

3) Get the solution by inverse transforming (2)

The process extends directly to two-dimensions. Suppose we want to solve Laplace's equation

$$u_{xx} + u_{yy} = f$$

using equally spaced points (with spacing $\Delta x = \Delta y = \Delta$) in a square and N_x and N_y grid points in each direction. Then

$$U_{mn} = \sum_{k=0}^{N_x-1} \sum_{\ell=0}^{N_y-1} \hat{U}_{k\ell}\omega_x^{-mk}\omega_y^{-n\ell}, \qquad \omega_x = e^{-2\pi i/N_x}, \quad \omega_y = e^{-2\pi i/N_y}$$

where $U_{mn} \approx u(x_m, y_n)$ (this is the 2d discrete Fourier transform). Note that the 2d transform is just the 1d transform applied in succesion in each direction:

$$\tilde{U}_{kn} = \sum_{\ell=0}^{N_y-1} \hat{U}_{k\ell}\omega_y^{-n\ell}$$

$$U_{mn} = \sum_{k=0}^{N_x-1} \tilde{U}_{kn}\,\omega_x^{-mk}$$

where $\tilde{U}$ is the IDFT in y. Plugging this into the centered difference formula for Laplace's equation, we get

$$\hat{U}_{k\ell}(\omega^{-k} + \omega^k - 4 + \omega^{-\ell} + \omega^{\ell}) = \Delta^2 \hat{f}$$

so

$$\hat{U}_{k\ell} = \frac{\Delta^2 \hat{f}}{2\cos(2\pi k/N_x) + 2\cos(2\pi m/N_y) - 4)}.$$

The process of solving is the same as before; compute $\hat{U}_{k\ell}$ above, then inverse transform to get the solution.

4.1 A brief reminder of spectral accuracy, DFT

(Fourier coefficient decay) Suppose $f(x)$ has period 2π and a Fourier series

$$f = \sum_{k=-\infty}^{\infty} c_k e^{ikx}. \tag{4.2}$$

Then the decay rate of $|c_k|$ depends on the smoothness of f:

- (slow) If f is only piecewise continuous, then $|c_k| \leq C/|k|$.
- If f has p continuous derivatives, there is a constant C such that

$$|c_k| \leq \frac{C}{|k|^{p+2}}.$$

- (fast) If f is smooth[3] then $|c_k| \leq Ce^{-a|k|}$.

We will refer to **spectral accuracy** as the property that the error in approximation decays in this way. (For instance, by definition, the Fourier series partial sum $S_N = \sum_{k=-N}^{N} c_k e^{ikx} \approx f(x)$ has spectral accuracy).

Review of the DFT and interpolation: The discrete Fourier transform is not quite the actual Fourier series. Let $x_n = nh$ with $h = 2\pi/N$, i.e. equally spaced points in $[0, 2\pi]$ with $x_0 = 0$ and $x_N = 2\pi$. Given data $f_n = f(x_n)$ for a periodic function f, we have the inverse transform

$$f_n = \sum_{k=0}^{N-1} \hat{f}_k e^{ikx_n}, \qquad n = 0, \cdots N-1.$$

The right expression is a 'complex' trigonometric interpolant (with x_n replaced by x).

For real functions, we should symmetrize, pairing $+k$ frequencies with $-k$ ones. Assuming N is even, we get

$$f(x) \approx \sum_{k=-N/2+1}^{N/2} \hat{f}_k e^{ikx} \tag{4.3}$$

which can then be written in real form as

$$f(x) \approx \frac{a_0}{2} + \frac{a_m}{2} \cos mx + \sum_{k=1}^{m} (a_k \cos kx + b_k \sin kx)$$

where $m = N/2$. Where available, it's most convenient to leave it in complex form, making both theory and implementation much easier.

The interpolant is **not** quite the same as the Fourier series partial sum; recall that from the DFT definition,

$$c_k = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{ikx}\,dx \approx \sum_{n=0}^{N-1} f_n e^{ikx_n} = f_k$$

approximated using the trapezoidal rule. As it turns out, despite this error, methods that make use of Fourier-like representations can recover spectral accuracy.

4.2 (Pseudo)-spectral methods: an example

While the method above is fast, it does not give us improved accuracy because it solves the difference equation (which already has an $O(\Delta x^2)$ error). Ideally, we would like to transform the derivative directly without approximating it.

Let's contrast the methods by considering the following example:

$$u'' - \beta u = f(x), \quad u(0) = u(\pi) = 0 \tag{D}$$

with a grid $x_n = nh$ for $n = 0, \cdots N$ and $h = \pi/N$.

Method 1: Finite differences: We have solved the problem before; we obtain

$$U_{n+1} - (2 + \beta h^2)U_n + U_{n-1} = h^2 f_n, \qquad n = 1, \cdots N-1 \tag{4.4}$$

and $U_0 = U_N = 0$. Note that $\beta > 0$ ensures that there is a unique solution since then the matrix is diagonally dominant ($2 + \beta h^2 > 1 + 1$).

Assuming, at least, that the solution $u(x)$ is smooth enough, the error should scale like $O(1/N^2)$ as $N \to \infty$ (second order), regardless of the properties of f.

Method 2 - Finite differences, with Fourier: Using the method from the previous section, we take the discrete Fourier transform of the difference equation (4.4) to obtain a simpler system (effectively diagonalizing it). However, the boundary conditions are not correct!

The solution is to instead take the **sine transform**, which is valid for Dirichlet boundary conditions (we saw the cosine version of this in a homework). Rather than derive it here, we can use an equivalent trick. Let

$$F(x) = \begin{cases} f(x) & 0 < x < \pi \\ -f(-x) & -\pi < x < 0 \end{cases}, \qquad F(x) = F(x + 2\pi)$$

i.e. the odd periodic extension of F with period 2π. Then we can solve

$$u'' - \beta u = F(x), \quad x \in [0, 2\pi], \quad u \text{ } 2\pi\text{-periodic} \tag{P}$$

and restrict back to $[0, \pi]$ to obtain the the solution to the Dirichlet problem (D). For a fair comparison, we use $h = \pi/N$ as before, so the points are now $x_n = nh$ for $n = 0, \cdots, K$ for $K = 2N$.

Taking the transform of the difference equation (4.4) as before, we get

$$\hat{U}_k(\omega^{-k} - 2 - \beta h^2 + \omega^k) = \hat{F}_k, \quad k = 0, \cdots, K-1$$

where $\omega = e^{-2\pi i/K}$. This simplifies to

$$\hat{U}_k = \frac{\hat{F}_k}{2\cos 2\pi k/K - 2 - \beta h^2}$$

which can then be inverse transformed to obtain u_n (and then restricted to $[0, \pi]$). Note that $\beta > 0$ guarantees that the division is always well-defined.

This is a solution to the **difference equation** with spectral accuracy, which in turn has an $O(h^2)$ error. Thus, the total error has a 'spectral' part from the DFT plus $O(h^2)$:

$$\text{error} = O(1/N^{p+2}) + O(1/N^2)$$

where p is the number of continuous derivatives of $F(x)$ (the periodic extension). If continuous, then the spectral error is at most the same magnitude as the FD error. But if f is very smooth, there is no gain from the small spectral error (the FD error dominates).

Method 3 - The (pseudo)-spectral method: We can instead try to avoid a finite difference approximation of the derivatives. Consider the same setup as above, solving (P). Recall that from the discrete Fourier transform we can construct the interpolant

$$u(x) \approx \sum_{k=-K/2+1}^{K/2} \hat{U}_k e^{ikx}. \tag{4.5}$$

with the 'mod K' convention on the indices (so $-1 \iff K-1$ is the last index of the transform vector, etc.). This is a starting point for a numerical method: we consider an approximation of the form (4.5), plug into the DE and solve for $\hat{U}_k$.

Plugging this in, and doing the same expansion for F, we get

$$\sum_{k \in S} \hat{U}_k((ik)^2 - \beta h^2) e^{ikx} = \sum_{k \in S} \hat{F}_k e^{ikx}$$

where $S = (-K/2+1, \cdots, 0, \cdots, K/2)$ is the vector of frequencies in the sum. By orthogonality, the coefficients of each term should match, which gives

$$\hat{U}_k = -\frac{1}{k^2 + \beta h^2} \hat{F}_k, \quad k \in S$$

Again, we have effectively diagonalized the derivative operator, but in a different way from the finite difference/Fourier version.

This spectral method then uses the following steps to solve (P):

- Compute the DFT of F
- Plug the Fourier interpolant into the DE and solve for the coefficients
- Compute the transform of the solution $\hat{U}$
- Compute the IDFT of $\hat{U}$ to get the solution U

The second step will provide a trivial system for a constant coefficient differential equation with periodic boundary conditions, as the derivatives are then transformed exactly into 'multiplication by ik'.

Important note: There is one subtletly in implementation to note. After the first step, we have, with K total points,

$$\text{DFT}(F) = (\hat{F}_0, \cdots, \hat{F}_{K-1}).$$

From the expression for the interpolant,

$$u(x) \approx \sum_{k=-K/2+1}^{K/2} \hat{U}_k e^{ikx}.$$

correspond to the frequencies

$$\vec{s} = (0, 1, \cdots, K/2, \quad -K/2+1, -K/2+2, \cdots, -1).$$

Thus, the actual computation should be done as

$$\hat{U}_k = \frac{1}{\vec{s}_k^2 + \beta h^2} \hat{F}_k, \quad k = 0, \cdots K-1$$

or, in vectorized form,

$$U = \text{IDFT}\Big[q(\vec{s}) * \text{DFT}(F)\Big]$$

where $*$ denotes pointwise multiplication of the vectors and $q(s) = 1/(s^2 + \beta h^2)$.

That is, it is important to be careful to **align the frequencies correctly** so they are properly associated with the coefficients as in the interpolant (4.5).

4.3 Comparison

To summarize, suppose we solve the BVP using these three methods and the right hand side $f(x)$ in the ODE has p continuous derivatives.

1) Finite differences yield an approximation with an error $O(1/N^2)$, independent of p. This can be improved to $O(1/N^{2m})$ by using higher-order difference formulas. The linear system is quick to solve ($O(N)$).

   However, for a 2d problem like $u_{xx} + u_{yy} - u = f$, the linear system is much more complicated, (still roughly $O(N)$, but the 'constant' is larger).

2) Finite differences (Fourier) yield a solution similar to (1) (solving the same difference equation). However, the error has a spectral part ($O(1/N^{p+2})$) plus the $O(1/N^2)$ error.

   This is only a problem if f is piecewise continuous; then the error is $O(1/N)$.

   Note that for the 2d problem, we saw that the Fourier transform drastically simplifies the system, so there is a significant gain in efficiency (compared to solving the linear system by method (1)).

3) Spectral method: The solution has spectral accuracy; the error is $O(1/N^{p+2})$ or even exponential in N if f is smooth (a compelling reason to use it!). For smooth functions f, this method can be far more accurate at smaller values of N than the other two.

4.4 Other boundary conditions

The catch here is that the Fourier spectral method requires periodic boundary conditions (or equivalent). At least, for a **constant coefficient**, **linear** DE, the Fourier transform can work.

- For Dirichlet boundary conditions, use the 'sine transform' (odd extension)
- For Neumann boundary conditions (e.g. $u'(0) = u'(\pi) = 0$), use the 'cosine transform'

Note that there is no need to actually do the extension as shown in the examples here; one can use a fast direct sine/cosine transform instead.

However, it should be clear that the use of the Fourier transform itself is powerful but quite limited. By extending to other sets of orthogonal bases, we can arrive at **pseudo-spectral methods** with the same nice properties that can handle more general problems.

As it turns out, the Chebyshev polynomials play an important role in extending spectral methods to non-periodic problems.

5 Pseudo-spectral methods (briefly)

(Adapted from Numerical Recipes, Section 20.7). First, let us consider a connection between interpolation and the approximation scheme. Suppose we have a function $u(x)$ and want to approximate it, obtaining values at a set of points $x_0, \cdots x_N$ in $[0, b]$. Moreover, suppose we want to have the values agree at those points,

$$P_N(x_j) = u(x_j), \quad j = 0, \cdots N.$$

(This is called **collocation** in the context of BVPs).

The strategy is to approximate a function $u(x)$ by the 'pseudo-spectral' representation

$$u(x) \approx P_N(x) = \sum_{k=0}^{N} a_k \phi_k(x) \tag{P}$$

where $\{\phi_k\}$ is an orthogonal basis in the interval $[0, b]$ with weight function $w(x)$:

$$\langle \phi_k, \phi_m \rangle = \int_0^b \phi_k \phi_m w(x)\, dx = 0. \quad k \neq m.$$

Let us further suppose this is a polynomial basis, so ϕ_k has degree k. The function u also has a 'spectral' expansion in terms of this basis,

$$u(x) = \sum_{k=0}^{\infty} c_k \phi_k(x), \quad c_k = \frac{\langle \phi_k, u \rangle}{\langle \phi_k, \phi_k \rangle}. \tag{S}$$

Now recall **Gaussian quadrature**, where we let $(x_0, \cdots, x_N)$ be the roots of ϕ_{N+1} and then approximated integrals by

$$\int_0^b g(x) w(x)\, dx \approx \sum_{k=0}^{N} v_k g(x_k)$$

where the v_k's were weights (formula not needed here). These Gaussian quadrature points will be our grid points. The Gaussian quadrature formula provides a discrete inner product

$$\langle \phi_k, \phi_m \rangle_G := \sum_{k=0}^{N} v_k \phi_k(x) \phi_m(x)$$

i.e. it approximates the continuous inner product with the GQ formula.

As we saw with the Fourier transform, the 'collocation' and 'spectral' representation are closely related. In fact, we claim that

$$a_k = \frac{\langle u, \phi_k \rangle_G}{\langle \phi_k, \phi_k \rangle_G}, \quad k = 0, \cdots N.$$

Proof. Starting with the representation (S), take the discrete inner product with ϕ_k:

$$\langle P_N, \phi_k \rangle_G = \sum_{m=0}^{N} a_m \langle \phi_m, \phi_k \rangle_G.$$

But recall that Gaussian quadrature has a degree of exactness $2N+1$. Each of the cross terms has the form

$$\text{Gaussian quadrature applied to } 0 = \int_0^b \phi_k(x)\phi_m(x)\,dx, \quad m+k \leq 2N+1$$

so it vanishes (the approximation to the integral is exact). It follows that

$$\langle P_N, \phi_k \rangle_G = a_N \langle \phi_k, \phi_k \rangle_G.$$

Finally, since P_N interpolates u at the grid points, and the LHS only involves its values at those points,

$$\langle u, \phi_k \rangle_G = \langle P_N, \phi_k \rangle_G.$$

So what does this result give us? The prize here is spectral accuracy. Let us see how the pseudo-spectral coefficients are related to the true spectral ones:

$$a_k = \langle u, \phi_k \rangle_G = \sum_{m=0}^{N} c_m \langle \phi_m, \phi_k \rangle_G + \sum_{m>N} c_m \langle \phi_m, \phi_k \rangle_G.$$

Using the orthogonality relation from before on the first term (all vanishes except the $m = k$ term, we get

$$a_k = c_k + \sum_{m>N} \langle \phi_m, \phi_k \rangle_G \cdot c_m.$$

Thus the error between the pseudo- and true spectral coefficients looks like

$$\sum_{m>N} (\text{stuff}) \cdot c_m$$

i.e. if the the true coefficients have nice decay properties, this error will also have those nice properties.

This tells us that interpolation at Gaussian points can actually provide spectral accuracy, due to this remarkable connection to the spectral representation.

5.1 For BVPs

Now suppose we need to solve a linear, constant coefficient BVP like

$$Lu := u'' + u' = f, \quad u(0) = u(1) = 0.$$

The **pseudospectral collocation method** proceeds by taking the Gaussian points and an orthogonal basis of polynomials $\{\phi_k\}$. Typically, these are the Chebyshev polynomials, which have nice computational properties (we saw this from noting that the Chebyshev series is related to the cosine series by a transform $x \to \cos\theta$).

We then form the representation

$$u \approx P_N := \sum_{k=0}^{N} a_k \phi_k(x)$$

and require that the **collocation equations** hold,

$$\sum_{k=0}^{N} a_k (L\phi_k)(x_j) = f(x_j).$$

That is, 'the ODE is satisfied by P_N at each of the grid points'. The theory outlined above ensures that we can achieve spectral accuracy by this method.

The rest of the details are a bit messy. Unlike the Fourier basis, $L\phi_k$ will not be a multiple of ϕ_k (polynomials are not eigenfunctions of the derivative operator, unlike e^{ikx}!). One then needs to figure out the derivative values in terms of the basis,[4]

$$\phi_k'(x_j) = \sum_{m=0}^{N} \alpha_{jm} \phi_k(x_m)$$

and so on. The collection of quantities demands a **differentiation matrix**, i.e. a matrix $\hat{D}$ such that

$$\begin{bmatrix} \phi_k'(x_0) \\ \vdots \\ \phi_k'(x_N) \end{bmatrix} = \hat{D} \begin{bmatrix} \phi_k(x_0) \\ \vdots \\ \phi_k(x_N) \end{bmatrix}$$

In the Fourier case, $\hat{D}$ was just a diagonal matrix. As it turns out, such matrices can be computed analytically by way of some clever recurrences for Chebyshev polynomials. □

[4] One can also do this in 'physical space', which is traditional when dealing with small values of N; here we write $u = \sum u(x_j)\ell(x)$ - the Lagrange interpolant form - and then figure out $\ell'(x_j)$ directly.