The Heat Equation

Problem Setup & Spatial Discretization

CS 410 / 510 — Scientific Computing

Lecture: Numerical Methods for PDEs

Where we are going

Over the next two lectures we will build, from scratch, a numerical method that solves the one-dimensional heat equation.

- **Today:** state the problem, set up the grid, and discretize in *space*.
- **Next time:** discretize in *time* and solve the resulting system on a computer.

By the end of the week, you will be able to implement the whole method yourself (this is what HW 3 is about).

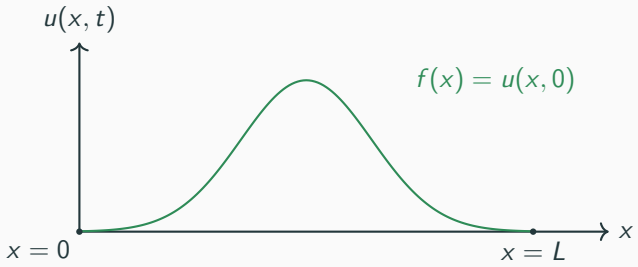
The physical problem

A rod that is cooling off

Imagine a thin rod of length L. We know:

- the temperature at every point at time $t = 0$ (call it $f(x)$),
- the temperature is held at 0 at both endpoints for all time,
- heat may be added or removed along the rod by a source $F(x, t)$.

Question: what is the temperature $u(x, t)$ at every point, for every time?



The initial-boundary value problem (IBVP)

Putting all the physics together, we get a partial differential equation plus two kinds of side conditions:

PDE: $u_t = K\,u_{xx} + F(x,t), \quad 0 \le x \le L,\ 0 \le t \le T$

Initial condition: $u(x,0) = f(x)$

Boundary conditions: $u(0,t) = u(L,t) = 0$

- K is the thermal diffusivity (a positive constant).
- $F(x,t)$ is a known heat source term.
- $f(x)$ is the known initial temperature profile.

Why we need a numerical method

For very simple f, F, and boundary conditions, you can solve this IBVP by hand (Fourier series, separation of variables, ...).

In real life:

- $f(x)$ comes from measurements, not a formula,
- the domain may be complicated,
- the heat source $F(x, t)$ may be messy,
- closed-form solutions usually do not exist.

Our plan
Approximate $u(x, t)$ only at a finite grid of points in x and t, and use the PDE to connect those values to each other.

The numerical grid

The spatial grid

Cut the interval $[0, L]$ into M equal pieces. That gives $M+1$ grid points:

$$x_j = j\,\Delta x, \qquad j = 0, 1, \ldots, M, \qquad \Delta x = \frac{L}{M}.$$

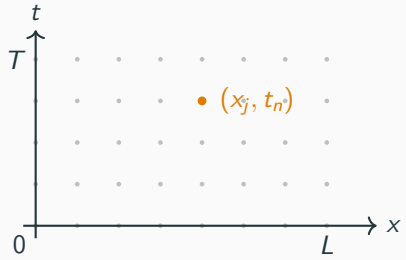
- Endpoints x_0 and x_M are the boundary nodes.
- The rest, $x_1, \ldots, x_{M-1}$, are the interior nodes.
- Larger M = finer grid = better accuracy (and more work).

The temporal grid

Do the same thing in time: cut $[0, T]$ into N equal pieces, giving $N+1$ time levels

$$t_n = n\,\Delta t, \qquad n = 0, 1, \ldots, N, \qquad \Delta t = \frac{T}{N}.$$

So we end up with a 2D lattice of points (x_j, t_n):



Our numerical unknowns

At each grid point we want *one number* that approximates the true temperature there:

$$\boxed{u_j^n \approx u(x_j, t_n)} \qquad \begin{matrix} j = 0, 1, \ldots, M \\ n = 0, 1, \ldots, N \end{matrix}$$

The initial condition gives us the entire $n = 0$ row for free:

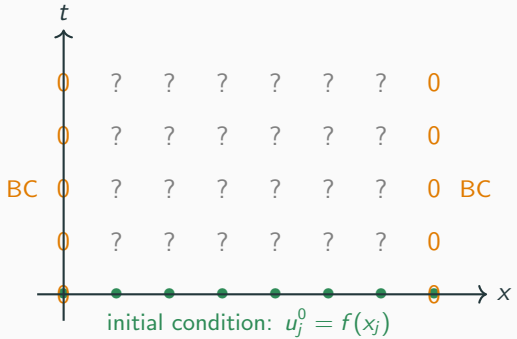
$$u_j^0 = f(x_j), \qquad j = 0, 1, \ldots, M.$$

The boundary conditions give us the two edge columns for free:

$$u_0^n = 0, \qquad u_M^n = 0, \qquad n = 0, 1, \ldots, N.$$

Everything else — the interior, for $n \geq 1$ — must be computed. That is our job.

Picture of the knowns and the unknowns



Our task: fill in the "?" s using the PDE.

Spatial discretization

Strategy: discretize space first

The PDE

$$u_t(x,t) = K\, u_{xx}(x,t) + F(x,t)$$

has derivatives in both space and time. We will replace the *spatial* derivative u_{xx} with something that only uses values on our x-grid. This leaves the *time* derivative alone — for now.

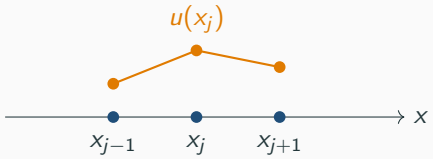
What we need

A formula that approximates $u_{xx}\big(x_j, t\big)$ using only the values $u(x_{j-1}, t)$, $u(x_j, t)$, $u(x_{j+1}, t)$.

A reminder: the central difference for u_{xx}

From Taylor's theorem (you have seen this earlier in the course):

$$u_{xx}\big|_{x=x_j} \approx \frac{u(x_{j+1}) - 2\,u(x_j) + u(x_{j-1})}{\Delta x^2}.$$



Using our grid notation, evaluated at time $t = t_n$:

$$u_{xx}\big|_{(x_j,t_n)} \approx \frac{u^n_{j+1} - 2\,u^n_j + u^n_{j-1}}{\Delta x^2}.$$

This formula is $O(\Delta x^2)$ accurate — halving Δx cuts the error by a factor of four.

Plugging it back into the PDE

Take the original PDE at the grid point (x_j, t_n)

$$u_t|_{(x_j,t_n)} = K \; u_{xx}|_{(x_j,t_n)} + F(x_j, t_n)$$

and replace the spatial derivative by our approximation:

$$\left.\frac{du_j}{dt}\right|_{t=t_n} = K\left(\frac{u^n_{j+1} - 2\,u^n_j + u^n_{j-1}}{\Delta x^2}\right) + F(x_j, t_n)$$

Notice what changed:

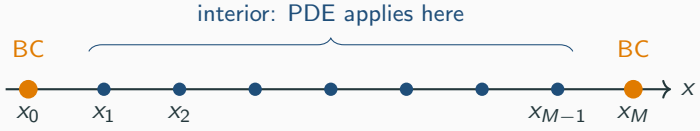
- u_{xx} is gone — replaced by an algebraic combination of three neighbors.
- But u_t is still a true time derivative. We have NOT discretized time yet.

Which nodes does this equation apply to?

The central-difference formula uses *both* a left and a right neighbor:

$$u_{j-1}, \quad u_j, \quad u_{j+1}.$$

So it only makes sense at nodes that have both neighbors:



So the equation holds only for $j = 1, 2, \ldots, M-1$.

What about the boundaries? The values u_0^n and u_M^n are not unknowns — they are given by the boundary conditions:

$$u_0^n = 0, \qquad u_M^n = 0, \qquad \forall n.$$

Re-indexing for implementation (Julia-friendly)

A quick practical note before we stop.

In math we like to start counting from 0. In Julia (and in HW 3), arrays start at index 1. It is easier to just use 1-based indexing everywhere:

$$x_1 = 0, \quad x_2, \quad \ldots, \quad x_{M+1} = L.$$

Then the roles shift by one:

- Boundary nodes: x_1 and x_{M+1}, with $u_1(t) = u_{M+1}(t) = 0$.
- Interior nodes: $x_2, x_3, \ldots, x_M$.
- The semi-discrete equation holds for $j = 2, \ldots, M$.

We will use this convention from next lecture onward.

Summary

What we did today

1. **Stated the problem:** heat equation IBVP on $[0, L]$ with Dirichlet BCs and initial data $f(x)$.
2. **Set up a grid:** $M+1$ spatial points and $N+1$ time levels, unknowns $u_j^n \approx u(x_j, t_n)$.
3. **Discretized in space:** replaced u_{xx} by a central difference.
4. Got a semi-discrete equation

$$\frac{du_j}{dt} = K \cdot \frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta x^2} + F(x_j, t), \quad j = 2, \ldots, M.$$

Looking ahead

We have one equation like this at every interior node. Stacking them gives a system of ODEs in time:

$$\vec{Y}'(t) = A\,\vec{Y}(t) + \vec{b}(t), \qquad \vec{Y}(0) = \vec{Y}_0.$$

Next lecture:

- identify the matrix A and the vector $\vec{b}(t)$ explicitly,
- apply time-stepping schemes (Forward Euler, Backward Euler) to march this system forward in time,
- end up with a complete numerical method — the method of lines.

Questions?