The 2D Heat Equation

— Analysis and Numerical Approaches

Outline

An Evolutional Problem

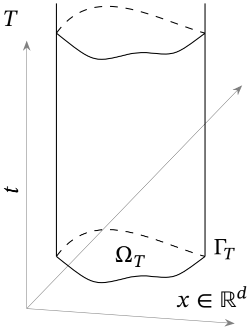


Figure: Region Ω_T

- Let $\Omega \subset \mathbb{R}^d$ be an open set with boundry $\Gamma := \partial\Omega$
- Set $\Omega_T = \Omega\times]0,T[$, $\Gamma_T := \Gamma\times]0,T[$, Γ_T is called the *lateral* boundary of the cylinder Ω_T.
- Consider the heat equation with L-periodic boundary condition:

$$\begin{cases} \partial_t u - \partial_{xx} u = f & \text{on } \Omega_T \\ u(x_i,t) = u(x_i+L,t) & \text{on } \Gamma_T \\ u(x,0) = u_0(x) & \text{on } \Omega \times \{t=0\} \end{cases}$$

Definition 1.1 (fundamental solution)

The function

$$\Phi(x,t) = \begin{cases} \frac{1}{(4\pi t)^{d/2}} e^{-\frac{|x|^2}{4t}} & x \in \mathbb{R}^d, t > 0 \\ 0 & x \in \mathbb{R}^d, t < 0 \end{cases} \tag{1.1}$$

is called the fundamental solution of the heat equation.

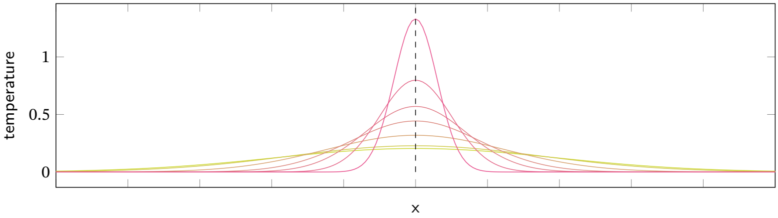


Figure: evolution of temperature on a rod along the time

The Riesz-Fredholm Theory

Theorem 1.2

Suppose that H is a separable Hilbert space, T is a compact self-adjoint operator, then there exists a Hilbert basis composed of eigenvectors of T.

Theorem 1.3 (the spectrum of Laplacian operator)

Suppose $T : L^2(\Omega) \to L^2(\Omega); f \mapsto u$, where u is the weak solution (i.e. solution in $H_0^1(\Omega)$) of

$$\begin{cases} -\Delta u_f &= f & \text{in } \Omega \\ u_f &= 0 & \text{on } \partial\Omega \end{cases}$$

Then T is compact and self-adjoint. Moreover T is positive defined.

u_f is characterized by $u_f \in H_0^1(\Omega), \forall \varphi \in H_0^1(\Omega), \int_\Omega \nabla u_f \nabla \varphi = \int_\Omega f\varphi$ and $S : f \mapsto u_f, L^2(\Omega) \to H_0^1(\Omega)$ is linear continuous.

$$\begin{tikzcd} f \arrow[r, "T"] \arrow[dr, "S"'] & u \\ & u_f \arrow[u, "i"'] \end{tikzcd}$$

$T = i \circ S$, where S is linear continuous and $i : H_0^1(\Omega) \to L^2(\Omega)$ is a canonical injection which is compact (it's a result of Reillich-Kondrachov's Theorem, we don't discuss Sobolev's injection here)
To prove that T self-adjoint is direct consequnce of properties of $L^2(\Omega)$.

Corollary 1.4

Let $\{e_n\}$ be the eigenfunctions of Laplacian, $\left\{\sqrt{\lambda_n} e_n\right\}_n$ is a Hilbert basis for $H_0^1(\Omega)$ equipped with the inner product

$$\langle v, w\rangle_{H_0^1(\Omega)} = \int_\Omega \nabla v \nabla w$$

However, sprectral analysis in $H_0^m(\Omega)$ doesn't fully meet our requirement, we must prescribe Ω and associated boundary conditions. Generally, we could sketch out the proper working space V as

$$V = \{f \in H^m(\Omega) : f \text{ satisfies BCs}\} \tag{1.2}$$

The maximum principle

Theorem 1.5

Assume that $u_0 \in L^2(\Omega)$, u is the solution of the heat equation we mentioned antecedently, then we have, for all $(x,t) \in \Omega_T$

$$\min\left\{0, \inf_{\Omega} u_0\right\} \leq u(x,t) \leq \max\left\{0, \sup_{\Omega} u_0\right\}$$

Outline

Discrete Fourier Transform (DFT)

- Separation of Variables and Superpostition Principle: A History
- DFT:

$$\hat{f}(k) = \int_0^{\tau} f(x) e^{-2i\pi kx} \,\mathrm{d}x \xrightarrow{\text{discretization}} U_k = \frac{1}{N} \sum_{j=0}^{N-1} f\left(\frac{j}{N}\right) e^{-2i\pi k \frac{j}{N}} \tag{2.1}$$

- FFT

Discrete Fourier Transform (DFT)

- Separation of Variables and Superpostition Principle: A History
- DFT:

$$\widehat{f}(k) = \int_0^\tau f(x)e^{-2i\pi kx}\,\mathrm{d}x \xrightarrow{\text{discretization}} U_k = \frac{1}{N}\sum_{j=0}^{N-1} f(\frac{j}{N})e^{-2i\pi k\frac{j}{N}} \tag{2.1}$$

- FFT

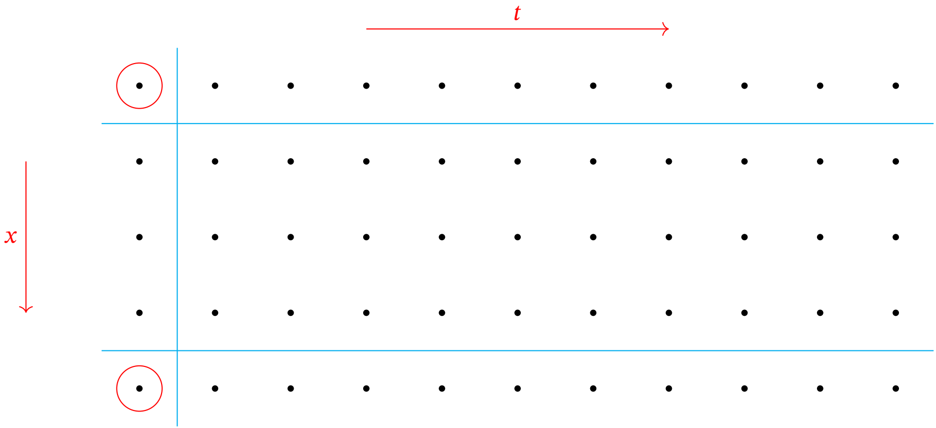
Discrete Fourier Transform (DFT)

- Separation of Variables and Superpostition Principle: A History
- DFT:

$$\widehat{f}(k) = \int_0^{\tau} f(x)e^{-2i\pi kx}\,\mathrm{d}x \xrightarrow{\text{discretization}} U_k = \frac{1}{N}\sum_{j=0}^{N-1} f(\frac{j}{N})e^{-2i\pi k\frac{j}{N}} \tag{2.1}$$

- FFT

Discretization



Schemes based on FDM

- The Explicit Euler Three Point Finite Difference Scheme
- The Implicit Euler and Leapfrog Schemes
- The Crank-Nicolson Scheme

Schemes based on FDM

- The Explicit Euler Three Point Finite Difference Scheme
- The Implicit Euler and Leapfrog Schemes
- The Crank-Nicolson Scheme

Schemes based on FDM

- The Explicit Euler Three Point Finite Difference Scheme
- The Implicit Euler and Leapfrog Schemes
- The Crank-Nicolson Scheme

```python
def compability_check(rod, t_interval):
# IC BC compability check
    try:
        #BC:
        bc = func_bc(t_interval)
        #IC:
        ic = initial_condition(rod)
        assert(ic[0] != bc[0] or ic[-1] !=bc[0])
    except AssertionError:
        print('IC and BC are not compatible')
    else:
        u = np.zeros(( rod_knots,time_knots))
        u[0, 1: ]=u[-1,1:] = bc[1:]
        u[: ,0]= ic
        return u, ic, bc

```

Compatility check for IC and BCs

In constructing numerical approximations to solutions of the heat equation in a bounded domain Ω of the plane, approximately, with a lattice and replaces the second partial derivatives with **centered differences**:

$$u_{xx} \approx \frac{u_W - 2u_O + u_E}{h_x^2}, \qquad u_{yy} \approx \frac{u_N - 2u_O + u_S}{h_y^2}$$

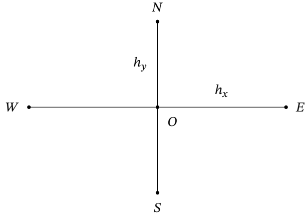


Figure: 2D centered difference approximation

and h_x (h_y) is the horizonal (with respect to vertical) mesh spacing.

For simplicity, we will use the same mesh spacing $h_x = h_y = h$ for both two dimensions. Let us consider the following scheme : given IC $u_0(x, y)$, we define iteratively the sequence $u_j(x, y)$ by

$$\frac{u_{j+1}(x,y) - u_j(x,y)}{k} - \Delta u(x,y) = f(x,y,j)$$

$$\implies \frac{u_{j+1}(x,y) - u_j(x,y)}{k} - \frac{u_j(x+h,y) + u_j(x-h,y) + u_j(x,y+h) + u_j(x,y-h) - 4u_n(x,y)}{h^2}$$

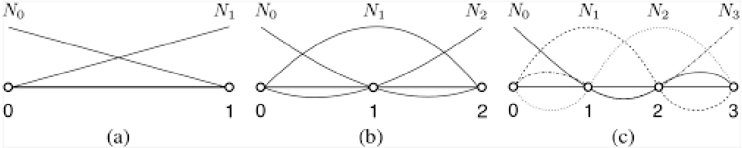
$$= f(x,y,j)$$

$$\implies u_{j+1} - u_j = \frac{k}{h^2}\left[u_j(x+h,y) + u_j(x-h,y) + u_j(x,y+h) + u_j(x,y-h) - 4u_n(x,y)\right] + kf(x,y,j)$$

We shall write it

$$u_{j+1} = u_j + A * u_j + kf_j, \quad \text{with} \quad A = \begin{bmatrix} 0 & s & 0 \\ s & -4s & s \\ 0 & s & 0 \end{bmatrix} \text{ and } s = \frac{k}{h^2}$$

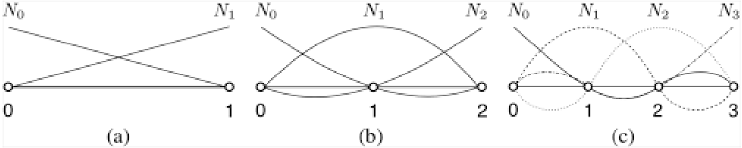
- 1D case: space-time partitioning of the rod



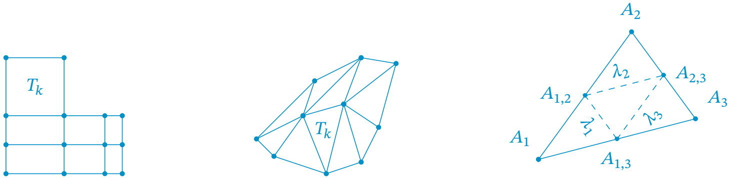
- Meshes in higher dimensions:

(a) Rectangular Q_1 Finite Elements (b) Triangular P_1 Lagrange Elements (c) Triangular P_2 Lagrange Elements

- 1D case: space-time partitioning of the rod



- Meshes in higher dimensions:



(a) Rectangular Q_1 Finite Elements (b) Triangular P_1 Lagrange Elements (c) Triangular P_2 Lagrange Elements

Outline

1D case:

$$\begin{cases} \frac{\partial u(x,t)}{\partial t} + \frac{\partial^2 u(x,t)}{\partial x^2} & = f(x,t) & \text{in } \Omega \\ u(x,0) & = u_0(x) & \text{in } \Omega \times \{t=0\} \\ u(0,t) = u(L,t) & = g(t) & \text{on } \partial\Omega \times]0,T[\end{cases}$$

2D case:

$$\begin{cases} \frac{\partial u(x,y,t)}{\partial t} + \frac{\partial^2 u(x,y,t)}{\partial x^2} + \frac{\partial^2 u(x,y,t)}{\partial y^2} & = f(x,y,t) & \text{in } \Omega \\ u(x,y,0) & = u_0(x,y,0) & \text{in } \Omega \times \{t=0\} \\ u(0,y,t) = g_1(y,t), \quad u(L,y,t) & = g_2(y,t) & \\ u(x,0,t) = g_3(x,t), \quad u(x,L,t) & = g_4(x,t) & \text{on } \partial\Omega \times]0,T[\end{cases}$$

Simulation

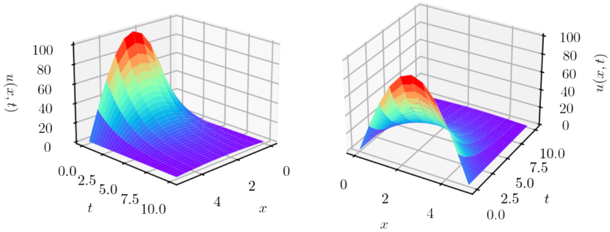


Figure: The numerical simulation for the 1D diffusion equation

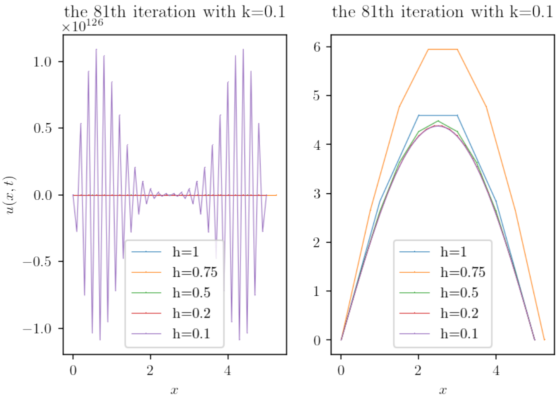


Figure: Approximation by explicit method (left) and implicit method (right)

Restriction and Possibility of Improvement

- FDM: simple to construct, but not flexible enough to treat complex boundary, restriction under CFL condition is not sufficient.
- FEM: more powerful for complex boundary problems, but hard to design meshs with well computational properties.
- Other methods? Finite Volume Method (FVM), Bhatnagar-Gross-Krook (BGK) for Boltzmann equation, Boundary Element Method(BEM), etc.
- Assemble with sparse matrix: `import scipy.sparse`
- Python - CPython memory overhead problem

Restriction and Possibility of Improvement

- FDM: simple to construct, but not flexible enough to treat complex boundary, restriction under CFL condition is not sufficient.
- FEM: more powerful for complex boundary problems, but hard to design meshs with well computational properties.
- Other methods? Finite Volume Method (FVM), Bhatnagar-Gross-Krook (BGK) for Boltzmann equation, Boundary Element Method(BEM), etc.
- Assemble with sparse matrix: `import scipy.sparse`
- Python - CPython memory overhead problem

Restriction and Possibility of Improvement

- FDM: simple to construct, but not flexible enough to treat complex boundary, restriction under CFL condition is not sufficient.
- FEM: more powerful for complex boundary problems, but hard to design meshs with well computational properties.
- Other methods? Finite Volume Method (FVM), Bhatnagar-Gross-Krook (BGK) for Boltzmann equation, Boundary Element Method(BEM), etc.
- Assemble with sparse matrix: `import scipy.sparse`
- Python - CPython memory overhead problem

Restriction and Possibility of Improvement

- FDM: simple to construct, but not flexible enough to treat complex boundary, restriction under CFL condition is not sufficient.
- FEM: more powerful for complex boundary problems, but hard to design meshs with well computational properties.
- Other methods? Finite Volume Method (FVM), Bhatnagar-Gross-Krook (BGK) for Boltzmann equation, Boundary Element Method(BEM), etc.
- Assemble with sparse matrix: `import scipy.sparse`
- Python - CPython memory overhead problem

Outline

- The Black-Scholes PDE for the Pricing of Financial Derivatives

$$-rF + \partial_t F + \partial_t F \partial_t S + \frac{\sigma_t^2}{2} \partial_{ss} F \partial_t^2 S = 0$$
$$S_t \in [0, +\infty[, \quad t \in [0, T]$$
$$F(T) = (S_T - K)_+$$

- Optimal Consumption and Investment

Thank you for listening!

Questions?

All resources including this diapositive, the final report, code and user's manual are submitted to:

Project Repository: https://github.com/derrring/2d_heat_equation

Your feedbacks are valuable to us :)

Outline