Finite Element Method for 2D heat equation

Tianyuan Wu, Mengying Wu

ShanghaiTech University

2020.11.26

Overview

1. 2D heat equation
2. FEM for 2D heat equation
3. Mesh generation
4. Performance evaluation & comparison

Goal

1. A FEM solver for 2D heat equation (* implemented in both Python and C++)
2. A mesh generating algorithm for given region Ω
3. Performance evaluation for our solver
4. * Data visualization (rendering) by matplotlib (Python) or OpenGL (C++)

2D heat equation

Formula:

$$\frac{\partial T}{\partial t} = \lambda(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}) + f(x, y, t)$$

where $x, y \in \Omega$, and T is the function of x, y and time t.
Also define the Neumann boundary condition:

$$\frac{\partial T}{\partial \vec{n}}\Big|_{\partial\Omega_n} = T_n$$

And the Dirichlet boundary condition:

$$T\Big|_{\partial\Omega_d} = T_d$$

Time elapse

In this project, we'll use update time variable t by finite difference method:

$$\frac{T(x,y,t_{n+1}) - T(x,y,t_n)}{dt} =$$
$$\lambda(\frac{\partial^2 T(x,y,n+1)}{\partial x^2} + \frac{\partial^2 T(x,y,n+1)}{\partial y^2}) + f(x,y,t_{n+1})$$

Denote $\frac{\partial^2 T(x,y,n)}{\partial x^2} + \frac{\partial^2 T(x,y,n)}{\partial y^2}$ as ΔT_n, and rewrite this equation as:

$$\frac{T_{n+1} - T_n}{dt} = \lambda \Delta T_{n+1} + f_{n+1}$$

Hence:

$$T_{n+1}(1 - \lambda dt \Delta T_{n+1}) = f_{n+1} dt + T_n$$

FEM method for heat equation

Denote:
Nodes: $N = \{n_1, n_2, \ldots, n_k\}$
Triangle Mesh: $M = \{m_1, m_2, \ldots, m_p\}$ Then we discrete this equation as:

$$T_{sol} = T_{sol,d} + T_{rest}$$

where $T_{sol,d}$ is to fit the dirichlet boundary condition.

FEM method for heat equation

Denote $\{\phi_1, \phi_2, \ldots, \phi_k\}$ as the basis The Galerkin discretation of this equation is:

$$\int\limits_{\Omega} T_{rest,n}\phi_j dx + (\int\limits_{\Omega} \nabla T_{rest,n} \cdot \nabla \phi_j dx) \cdot \lambda dt$$

$$= \int\limits_{\Omega} T_{rest,n-1}\phi_j dx + (\int\limits_{\partial\Omega_n} T_{rest,n}^{neum}\phi_j ds) \cdot dt + (\int\limits_{\Omega} F_n \phi_j dx) \cdot dt$$

FEM method for heat equation - Cont'd

$$T_{rest,t} = \sum_{i=1}^{k} \alpha_i \phi_t^i$$

Let Tri denote the triangle mesh, hence we can construct the mass matrix

$$M_{ij} = \sum_{A \in Tri} \int_A \phi_i \phi_j dx$$

, coefficient matrix

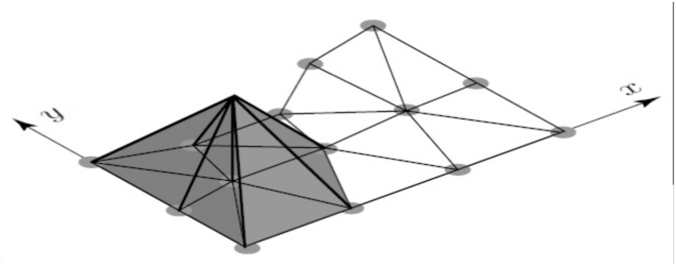
$$A_{ij} = \int_\Omega \nabla \phi_i \cdot \nabla \phi_j dx$$

Hence we can get the matrix form of this equation

$$(Adt + M)T_{rest,n} = MT_{rest,n-1}$$
$$+ \left(\int_{\partial \Omega_n} T_{rest,n}^{neum} \phi_j ds\right) \cdot dt + \left(\int_\Omega F_n \phi_j dx\right) \cdot dt$$

Triangle mesh

The triangle mesh $\phi_i(x, y)$ is shown in the following figure



Hence we can discrete the temperature function T as

$$T_{rest,t} = \sum_{i=1}^{k} \alpha_i \phi_t^i$$

Mesh generation

1. Regular region: Simple to implement
2. Irregular region: Maybe we can refer to Delaunay triangulation, or using KD Tree

Performance evaluation & comparison

1. Evaluation of different mesh size / number of triangles
2. Comparison between different implementations (Python vs. C++)
3. Evaluation of different initial conditions, boundary conditions

Thanks!