

Algorithms of Scientific Computing

Fast Poisson Solvers

Michael Bader

Summer Term 2013



Part I

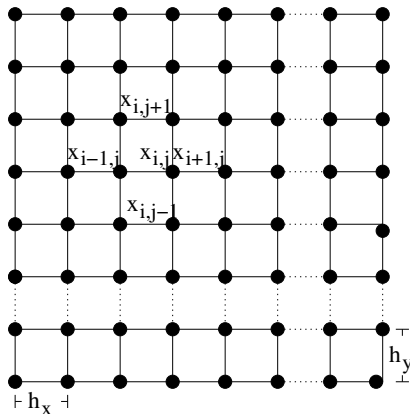
Excursion: Discrete Models for Heat Transfer and the Poisson Equation

Modelling of Heat Transfer

- objective: compute the temperature distribution of some object
- under certain prerequisites:
 - temperature T at object boundaries given
 - heat sources
 - material parameters k, \dots
- observation from physical experiments: $q \approx k \cdot \delta T$
heat flow proportional to temperature differences

A Wiremesh Model

- consider rectangular plate as fine mesh of wires
- compute temperature $T_{ij} \approx T(x_{ij})$ at nodes x_{ij} of the mesh



Wiremesh Model (2)

- model assumption:
temperatures in equilibrium at every mesh node
- for all temperatures T_{ij} :

$$T_{ij} = \frac{1}{4} (T_{i-1,j} + T_{i+1,j} + T_{i,j-1} + T_{i,j+1})$$

- temperature known at (part of) the boundary; for example:

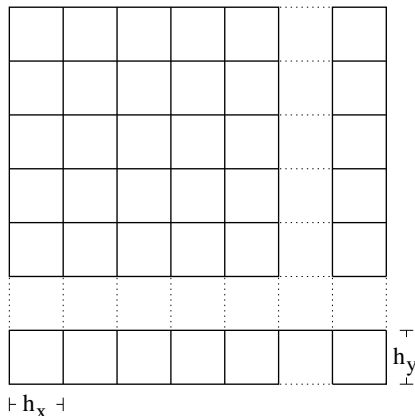
$$T_{0,j} = 0 \quad T_{n,j} = 1$$

- task: solve system of linear equations
- in standard notation:

$$-T_{i-1,j} - T_{i+1,j} + 4T_{ij} - T_{i,j-1} - T_{i,j+1} = 0$$

A Finite Volume Model

- object: a rectangular metal plate (again)
- model as a collection of small connected rectangular cells



- examine the heat flow across the cell edges

Heat Flow Across the Cell Boundaries

- Heat flow across a given edge is proportional to
 - temperature difference ($T_1 - T_0$) between the adjacent cells
 - length h of the edge
- e.g.: heat flow across the left edge:

$$q_{ij}^{(\text{left})} = k_x (T_{ij} - T_{i-1,j}) h_y$$

k_x depends on material

- heat flow across all edges determines change of heat energy:

$$\begin{aligned} q_{ij} &= k_x (T_{ij} - T_{i-1,j}) h_y + k_x (T_{ij} - T_{i+1,j}) h_y \\ &+ k_y (T_{ij} - T_{i,j-1}) h_x + k_y (T_{ij} - T_{i,j+1}) h_x \end{aligned}$$

Temperature change due to heat flow

- model assumption: conservation of energy, i.e.,
in equilibrium, total heat flow equal to 0 for each cell
- or: consider additional source term F_{ij} due to
 - external heating
 - radiation
- $F_{ij} = f_{ij} h_x h_y$ (f_{ij} heat flow per area)
- equilibrium with source term requires $q_{ij} + F_{ij} = 0$:

$$\begin{aligned} f_{ij} h_x h_y &= -k_x h_y (2T_{ij} - T_{i-1,j} - T_{i+1,j}) \\ &\quad -k_y h_x (2T_{ij} - T_{i,j-1} - T_{i,j+1}) \end{aligned}$$

Finite Volume Model

- divide by $h_x h_y$:

$$f_{ij} = -\frac{k_x}{h_x} (2T_{ij} - T_{i-1,j} - T_{i+1,j}) - \frac{k_y}{h_y} (2T_{ij} - T_{i,j-1} - T_{i,j+1})$$

- again, system of linear equations
- how to treat boundaries?
 - prescribe temperature in a cell (e.g. boundary layer of cells)
 - prescribe heat flow across an edge; for example insulation at left edge:

$$q_{ij}^{(\text{left})} = 0$$

From Discrete to Continuous

- system of equations derived from the discrete model:

$$f_{ij} = -\frac{k_x}{h_x} (2T_{ij} - T_{i-1,j} - T_{i+1,j}) \\ -\frac{k_y}{h_y} (2T_{ij} - T_{i,j-1} - T_{i,j+1})$$

- assumption: heat flow across edges is proportional to temperature *difference*

$$q_{ij}^{(\text{left})} = k_x (T_{ij} - T_{i-1,j}) h_y$$

- in reality: heat flow proportional to temperature *gradient*

$$q_{ij}^{(\text{left})} \approx kh_y \frac{T_{ij} - T_{i-1,j}}{h_x}$$

From Discrete to Continuous (2)

- replace k_x by k/h_x , k_y by k/h_y , and get:

$$f_{ij} = -\frac{k}{h_x^2} (2T_{ij} - T_{i-1,j} - T_{i+1,j}) \\ -\frac{k}{h_y^2} (2T_{ij} - T_{i,j-1} - T_{i,j+1})$$

- consider arbitrary small cells: $h_x, h_y \rightarrow 0$:

$$f_{ij} = -k \left(\frac{\partial^2 T}{\partial x^2} \right)_{ij} - k \left(\frac{\partial^2 T}{\partial y^2} \right)_{ij}$$

- leads to *partial differential equation* (PDE):

$$-k \left(\frac{\partial^2 T(x, y)}{\partial x^2} + \frac{\partial^2 T(x, y)}{\partial y^2} \right) = f(x, y)$$

Part II

Fast Poisson Solvers and the Sine Transform

- situation: solve a system of linear equations

$$-u_{i-1,j} - u_{i+1,j} + 4u_{ij} - u_{i,j-1} - u_{i,j+1} = f_{ij} \quad \forall i, j$$

- or, simpler, for a 1D problem:

$$-u_{n-1} + 2u_n - u_{n+1} = f_n \quad \text{for } n = 1, \dots, N-1$$

with $u_0 = u_N = 0$

- consider very fine meshes, for example with 1000×1000 unknowns (in 2D)
- actually simple to solve in 1D (tri-diagonal system), but hard to solve in 2D (and even harder in 3D)

Applying the Sine Transform

Idea: apply discrete sine transform on u_n and f_n

$$u_n = 2 \sum_{k=1}^{N-1} U_k \sin \frac{\pi nk}{N}, \quad f_n = 2 \sum_{k=1}^{N-1} F_k \sin \frac{\pi nk}{N} \quad (1)$$

into the system of equations

$$-u_{n-1} + 2u_n - u_{n+1} = f_n \quad \text{for } n = 1, \dots, N-1$$

Applying the Sine Transform

Idea: apply discrete sine transform on u_n and f_n

$$u_n = 2 \sum_{k=1}^{N-1} U_k \sin \frac{\pi nk}{N}, \quad f_n = 2 \sum_{k=1}^{N-1} F_k \sin \frac{\pi nk}{N} \quad (1)$$

into the system of equations

$$-u_{n-1} + 2u_n - u_{n+1} = f_n \quad \text{for } n = 1, \dots, N-1$$

Why should that help?

- corresponding continuous problem is $-u''(x) = f(x)$
- is solved by $u(x) = \sin(x)$, if $f(x) = \sin(x)$
(with $u(x) = 0$ at both boundaries)
- sine modes are **eigenvectors** of the system matrix, and **eigenmodes** of the continuous solution

Applying the Sine Transform (2)

We insert the transformations

$$u_n = 2 \sum_{k=1}^{N-1} U_k \sin \frac{\pi nk}{N} \quad \text{and} \quad f_n = 2 \sum_{k=1}^{N-1} F_k \sin \frac{\pi nk}{N}$$

into the system of linear equations

$$\begin{aligned} -u_{n+1} + 2u_n - u_{n-1} &= f_n \quad \text{for } n = 1, \dots, N-1 \\ u_0 &= u_N = 0, \end{aligned}$$

and get, for $n = 1, \dots, N-1$:

$$\begin{aligned} -2 \sum_{k=1}^{N-1} U_k \sin \frac{\pi(n+1)k}{N} + 4 \sum_{k=1}^{N-1} U_k \sin \frac{\pi nk}{N} - 2 \sum_{k=1}^{N-1} U_k \sin \frac{\pi(n-1)k}{N} \\ = 2 \sum_{k=1}^{N-1} F_k \sin \frac{\pi nk}{N} \end{aligned}$$

Applying the Sine Transform (3)

Use theorems of addition

$$\sin(A + B) = \sin(A) \cos(B) + \cos(A) \sin(B) \quad \text{and}$$

$$\sin(A - B) = \sin(A) \cos(B) - \cos(A) \sin(B)$$

applied to:

$$\sin\left(\frac{\pi(n+1)k}{N}\right) = \sin\left(\frac{\pi nk}{N} + \frac{\pi k}{N}\right) \quad \text{and}$$

$$\sin\left(\frac{\pi(n-1)k}{N}\right) = \sin\left(\frac{\pi nk}{N} - \frac{\pi k}{N}\right)$$

We have the situation

$$\sin(A + B) + \sin(A - B) = 2 \sin(A) \cos(B)$$

or, particularly:

$$\sin\left(\frac{\pi(n+1)k}{N}\right) + \sin\left(\frac{\pi(n-1)k}{N}\right) = 2 \sin\left(\frac{\pi nk}{N}\right) \cos\left(\frac{\pi k}{N}\right)$$

Applying the Sine Transform (4)

Use theorems of addition in the left-hand side:

$$\begin{aligned}
 -2 \sum_{k=1}^{N-1} \left(U_k \sin \frac{\pi(n+1)k}{N} + U_k \sin \frac{\pi(n-1)k}{N} \right) + 4 \sum_{k=1}^{N-1} U_k \sin \frac{\pi nk}{N} \\
 -4 \sum_{k=1}^{N-1} U_k \sin \frac{\pi nk}{N} \cos \frac{\pi k}{N} + 4 \sum_{k=1}^{N-1} U_k \sin \frac{\pi nk}{N}
 \end{aligned}$$

and obtain simplified system of equations:

$$\begin{aligned}
 -4 \sum_{k=1}^{N-1} U_k \sin \frac{\pi nk}{N} \cos \frac{\pi k}{N} + 4 \sum_{k=1}^{N-1} U_k \sin \frac{\pi nk}{N} &= 2 \sum_{k=1}^{N-1} F_k \sin \frac{\pi nk}{N} \\
 \Leftrightarrow 2 \sum_{k=1}^{N-1} U_k \sin \frac{\pi nk}{N} \left(1 - \cos \frac{\pi k}{N} \right) &= \sum_{k=1}^{N-1} F_k \sin \frac{\pi nk}{N}
 \end{aligned}$$

Solution of the System of Equations

We transformed the system of equations into

$$2 \sum_{k=1}^{N-1} U_k \sin \frac{\pi nk}{N} \left(1 - \cos \frac{\pi k}{N} \right) = \sum_{k=1}^{N-1} F_k \sin \frac{\pi nk}{N}$$

All equations are satisfied, if

$$2U_k \left(1 - \cos \frac{\pi k}{N} \right) = F_k \quad \text{for all } k = 1, \dots, N-1$$

This is true, if we set

$$U_k = \frac{F_k}{2 - 2 \cos \frac{\pi k}{N}} \quad \text{for all } k = 1, \dots, N-1.$$

Fast Poisson Solver – Algorithm

1. Compute the coefficients F_k by a **Fast Sine Transform**:

$$F_k = \frac{1}{N} \sum_{n=1}^{N-1} f_n \sin \frac{\pi nk}{N}$$

2. Compute all coefficients U_k from the F_k as

$$U_k = \frac{F_k}{2 - 2 \cos \frac{\pi k}{N}} \quad \text{for all } k = 1, \dots, N-1.$$

3. Compute the u_n from the U_k by means of an Inverse **Fast Sine Transform**:

$$u_n = 2 \sum_{k=1}^{N-1} U_k \sin \frac{\pi nk}{N},$$

Fast Poisson Solver – Algorithm (2)

Computational Costs:

- the two Fast Sine Transforms require $\mathcal{O}(N \log N)$ operations
- step 2 needs only $\mathcal{O}(N)$ operations
 - ⇒ **total computational effort is $\mathcal{O}(N \log N)$**
- thus: slower than solving the tridiagonal system of equations directly, which has effort $\mathcal{O}(N)$
- however: pays off in 2D and higher-dimensional settings!

Fast Poisson Solver – Algorithm (2)

Computational Costs:

- the two Fast Sine Transforms require $\mathcal{O}(N \log N)$ operations
- step 2 needs only $\mathcal{O}(N)$ operations
 - ⇒ **total computational effort is $\mathcal{O}(N \log N)$**
- thus: slower than solving the tridiagonal system of equations directly, which has effort $\mathcal{O}(N)$
- however: pays off in 2D and higher-dimensional settings!

When can the Algorithm be applied:

- boundary conditions need to be $u_0 = u_N = 0$
(otherwise: different transform required)
- requires rectangular/cuboid domain and Cartesian mesh
- requires uniform material parameters k