Numerical Linear Algebra Workshop

Duo Cao

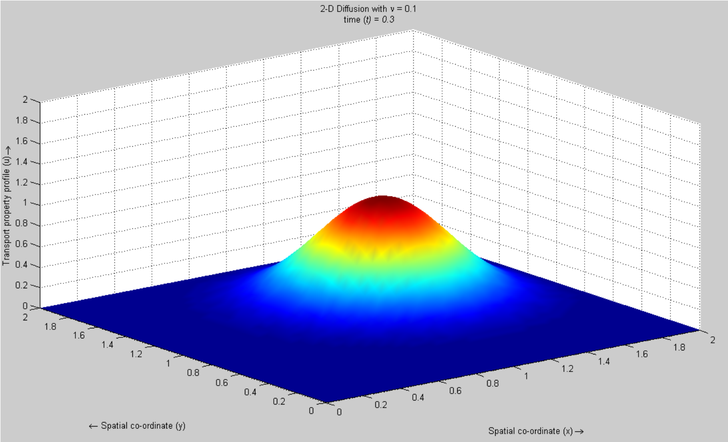
Department of Mathematics, Purdue University

cao157@purdue.edu

March 27, 2016

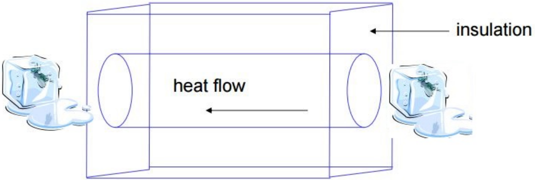
Overview

Diffusive Transport: Dye in water, pollution, heat, perfume...



Diffusion Equations Cont'd

- Consider temperature in a long thin tube of constant cross section.
- The tube is perfectly insulated laterally. Heat only flow along the tube.
- Its ends maintain at zero temperature.[1]

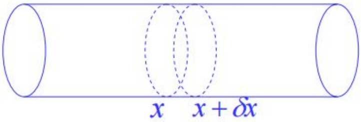


2

[1] Kreysig, 8th Edn, Sections 11.4b

[2] Professor Michael Brady's lecture note

Diffusion Equations Cont'd

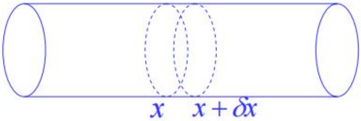


Suppose

- thermal conductivity in the wire is K.
- Cross sectional area is A.
- Material density ρ.
- Heat capacity is σ.
- Temperature at point x at time t is $u(x,t)$.

Then the heat flow into bar across face at x : $-KA\frac{\partial u}{\partial x}|_x$.
At the face $x+\delta x$: $-KA\frac{\partial u}{\partial x}|_{x+\delta x}$

Diffusion Equations Cont'd



- The net flow out is: $KA\frac{\partial^2 u}{\partial x^2}\delta x$
- $Q = \sigma m \Delta T$
- So, the conservation of heat gives: $KA\frac{\partial^2 u}{\partial x^2}\delta x = \sigma\rho A\frac{\partial u}{\partial t}\delta x$

$$\boxed{\frac{\partial u}{\partial t} = c^2\frac{\partial^2 u}{\partial x^2}}$$

Boundary Conditions

- Homogeneous Dirichlet Boundary Condition: $u(0,t) = u(L,t) = 0$
- Homogeneous Neumann Boundary Condition: $\frac{\partial u}{\partial x}(0,t) = \frac{\partial u}{\partial x}(L,t) = 0$

Miscellaneous Equations

- Navier-Stokes Equation: $\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \gamma \nabla^2 \mathbf{u} + \frac{1}{\rho}\mathbf{F}$
- Fisher's Equation: $\frac{\partial u}{\partial t} = u(1-u) + \frac{\partial^2 u}{\partial x^2}$
- Nonlinear Schrodinger Equation: $i\partial_t \phi = -\frac{1}{2}\partial_x^2 \phi + \kappa |\phi|^2 \phi$
- Black-Scholes Equation: $\frac{\partial V}{\partial t} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$
- etc

Finite Difference Method

We consider the two-point boundary value problem:

$$Au: \quad = -au'' + bu' + cu = f \ \ in \ \Omega = (0,1), \tag{1}$$

$$u(0) = u_0, \ \ u(1) = u_1 \tag{2}$$

where the coefficients $a = a(x), b = b(x)$,and $c = c(x)$ are smooth functions satisfying $a(x) > 0$ and $c(x) \geq 0$ in $\overline{\Omega}$. And f, u_0, u_1 are given.

To find numerical solution of (2) we introduce $M+1$ grid points $0 = x_0 < x_1 < ... < x_M = 1$ by setting $x_j = jh, j = 0, ..., M$, where $h = 1/M$. We denote the approximation of $u(x_j)$ by U_j and use the following finite difference approximation for derivatives.

$$
\begin{aligned}
\partial U_j &= \frac{U_{j+1} - U_j}{h}, (forward\ \ difference) \\
\partial \overline{U_j} &= \frac{U_j - U_{j-1}}{h}, (backward\ \ difference) \\
\widehat{\partial} U_j &= \frac{U_{j+1} - U_{j-1}}{2h}, (central\ \ difference) \\
\partial \overline{\partial} U_j &= \frac{U_{j+1} - 2U_j + U_{j-1}}{h^2}
\end{aligned}
$$

Setting also $a_j = a(x_j), b_j = b(x_j), c_j = c(x_j), f_j = f(x_j)$, we now define a finite difference approximation of (2) by

$$
\begin{aligned}
A_h U_j &:= -a_j \partial\bar{\partial} U_j + b_j \widehat{\partial} U_j + c_j U_j = f_j, \textit{for } j = 1, \ldots, M-1, &(3)\\
&U_0 = u_0,\ U_M = u_1. &(4)
\end{aligned}
$$

Then, after simplification, the equation at the interior point x_j may be written as

$$
(2a_j + h^2 c_j)U_j - (a_j - \frac{1}{2}hb_j)U_{j+1} - (a_+ \frac{1}{2}hb_j)U_{j-1} = h^2 f_j \qquad (5)
$$

for all j.

Put (5)into a matrix form:

$$\boxed{AU = g}$$

We finally comes to LINEAR ALGEBRA!! OH YEAH!!

In our system $AU = g$:
$U = (U_1, \ldots, U_{M-1})^T$ and the first and last components of the vector $g = (g_1, \ldots, g_{M-1})^T$ contain contributions from the boundary values u_0, u_1 as well as f_1 and f_{M-1}, respectively. The $(M-1) \times (M-1)$ matrix A is tridiagonal and diagonally dominant for h sufficiently small.

Finite Element Method for BVP

We consider the special case $b = 0$ of the two-point boundary value problem of (2),

$$Au := -(au')' + cu = f \ \ in \ \Omega := (0,1), \ \ with \ u(0) = u(1) = 0,$$

where $a = a(x), c = c(x)$ are smooth functions with $a(x) \geq q_0 > 0$, $c(x) \geq 0$ in $\overline{\Omega}$ and $f \in L_2 = L_2(\Omega)$.

Recall the variational formulation of this problem is to find $u \in H_0^1$ such that

$$a(u, \phi) = (f, \phi), \ \forall \phi \in H_0^1,$$

where

$$a(v, w) = \int_\Omega (av'w' + cvw)dx \ and \ (f, v) = \int_\Omega fvdx,$$

and that this problem has a unique solution $u \in H^2$.

For the purpose of finding an approximate solution of (15) we introduce a partition of Ω,

$$0 = x_0 < x_1 < \ldots < x_M = 1,$$

and set

$$h_j = x_j - x_{j-1}, K_j = [x_{j-1}, x_j], \; for \; j = 1, \ldots, M, \; and \; h = \max_j h_j.$$

The discrete solution will be sought in the finite-dimensional space of functions

$$S_h = v \in C = C(\overline{\Omega}) : v \ linear \ on \ each \ K_j, v(0) = v(1) = 0.$$

The set $\{\Phi_i\}_{i=1}^{M-1} \subset S_h$ is defined by

$$\Phi_i(x_j) = \begin{cases} 1 & if \ i = j \\ 0 & if \ i \neq j \end{cases}$$

and any $v \in S_h$ may be written as

$$v(x) = \sum_{i=1}^{M-1} v_i \Phi_i(x), \ with \ v_i = v(x_i).$$

Now we pose the finite-dimensional problem to find $u_h \in S_h$ such that

$$a(u_h, \chi) = (f, \chi), \ \forall \chi \in S_h. \tag{6}$$

In terms of the basis $\{\Phi_i\}_{i=1}^{M-1}$ we write $u_h(x) = \sum_{j-1}^{M-1} U_j \Phi_j(x)$ and insert this into (6) to find that this equation is equivalent to

$$\sum_{j=1}^{M-1} U_j a(\Phi_j, \Phi_i) = (f, \Phi_i), \ \mathit{for} \ i = 1, \ldots, M-1.$$

This linear system of equations could be expressed in matrix form as

$$\boxed{AU = b}$$

Finished!!!

In our system $U = (U_j), A = (a_{ij})$ is the stiffness matrix with elements $a_{ij} = a(\Phi_j, \Phi_i)$, and $b = (b_i)$ with elements $b_i = (f, \Phi_i)$. The matrix A is symmetric and positive definite, because for $V = (V_i)$ and $v(x) = \sum_{i=1}^{M-1} V_i \Phi_i(x)$ we have

$$V^T A V = \sum_{i,j=1}^{M-1} V_i a_{ij} V_j = a\Big(\sum_{j=1}^{M-1} V_j \Phi_j, \sum_{i=1}^{M-1} V_i \Phi_i \Big) = a(v, v) \geq a_0 ||v'||^2,$$

and hence $V^T A V = 0$ implies $v' = 0$, so that v is 0. Matrix A is tridiagonal since$a_{ij} = 0$ when x_i and x_j are not neighbors, i.e., when $|i - j| \geq 2$.

Miscellaneous Numerical Methods

- Finite Volume Method
- Spectral Method
- Meshfree Methods
- Multigrid
- etc.

$$\boxed{AU = b}$$

Direct Solver–LU Decomposition

Let $A \in \mathbb{R}^{m\times m}$ be a square matrix. (Algorithm can also be applied to rectangular matrices, but as this is rarely done in practice, we shall confine our attention to the square case.) The idea is to transform A into an $m \times m$ upper-triangular matrix U by introducing zeros below the diagonal, first in column 1, then in column 2. This is done by subtracting multiples of each row from subsequent rows. This "elimination" process is equivalent to multiplying A by a sequence of lower-triangular matrices L_k on the left:

$$L_{m-1} \dots L_2 L_1 A = U$$

We obtain a factorization of matrix A,

$$A = LU,$$

where U is upper-triangular and L is lower-triangular.

$$\underset{A}{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}} \xrightarrow{L_1} \underset{L_1A}{\begin{bmatrix} \times & \times & \times & \times \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}} \xrightarrow{L_2} \underset{L_2L_1A}{\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}} \xrightarrow{L_3} \underset{L_3L_2L_1A}{\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & \mathbf{0} & \boldsymbol{\times} \end{bmatrix}}$$

In Matlab, just input $[L, U, P] = lu(A)$.

Why LU?

- Cost of solving $AU = b$ is $\sim \frac{2}{3}m^3$ which is twice faster than QR factorization(Sorry no time to cover this).
- Easy to implement
- Applicable for any matrix

Why LU?

- Cost of solving $AU = b$ is $\sim \frac{2}{3}m^3$ which is twice faster than QR factorization(Sorry no time to cover this).
- Easy to implement
- Applicable for any matrix

Why not LU?

- In most cases unstable, you may improve it by 'pivoting'
- Solution is always exact – cannot find approximate solutions

Direct Solver–Cholesky Decomposition

If our A is symmetric(Hermitian) and positive-definite, applying similar Gauss-Elimination, we could decompose A into:

$$A = U^T U$$

where U is upper-triangular.

Note

Cholesky decomposition is stable.
In Matlab, input $chol(A)$.

Iterative Method

Why iterate?

- Direct Method: $O(m^3)$
- Sometimes A cannot be explicitly worked out (See 'real example 2')

Iterative Method : Gauss–Seidel & Jacobi Method

Jacobi

Write $A = D + E$, where D is the diagonal of A and E is the remaining part. Then the iteration is defined as

$$x^{(k+1)} = D^{-1}(b - Ex^{(k)})$$

Gauss-Seidel

Similar with Jacobi Method, but solve x_i^{k+1} using updated x_j^{k+1}, *for* $j < i$.

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j>i} a_{ij}x_j^{(k)}\right)$$

Advantages

- Stable
- Fast

Disadvantages

- Only converge for p.d. symmetric matrices or diagonally dominant matrices,
- Only gives one solution, which means unavailable for singular matrices.

Iterative Method : Conjugate Gradients

Where is it come from?

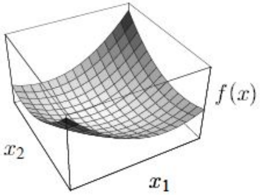
Let

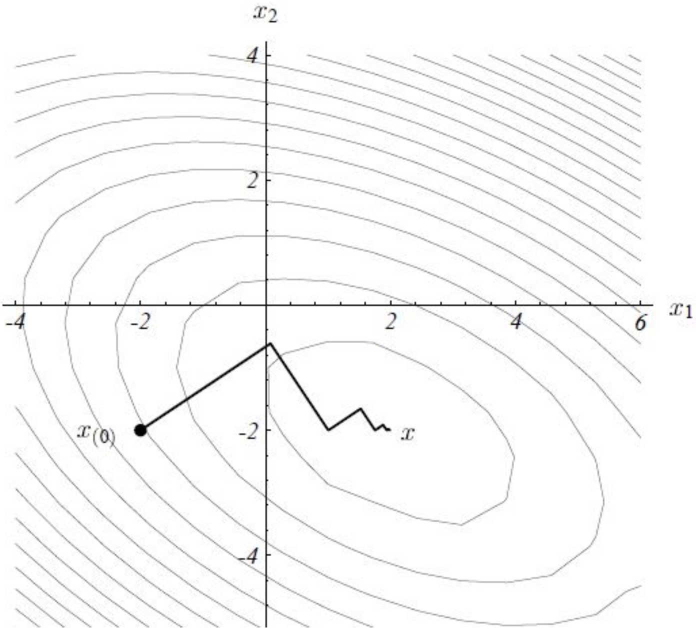
$$f(x) = \frac{1}{2}x^T Ax - b^T x,$$

then

$$\nabla f(x) = Ax - b$$

which means solution $x = A^{-1}b$ minimizes $f(x)$.
Here we require $f(x)$ to be a convex function, that's why matrix A needs to be symmetric and positive definite.




x_2
4
2
-4
-2
2
4
6
x_1
$x_{(0)}$
-2
x
-4

Boring but useful definitions

- Residual : $r_i = b - Ax_i$ indicates how far we are from the correct value of b
- Error : $e_i = x - x_i$ indicates how far we are from the solution
- Search direction : p_i
- Step size : α_i

It's easy to see

$$
\begin{aligned}
r_{i+1} &= -Ae_{i+1} \\
&= -A(e_i + \alpha_i p_i) \\
&= r_i - \alpha_i A p_i
\end{aligned}
$$

Definition (A-conjugacy)

A set of nonzero vectors $\{p_0, p_1, \ldots, p_{n-1}\}$ are called A-conjugacy if $p_i^T A p_j \; \forall i \neq j$.

This definition is important because we could write the exact solution x as:

$$x - x_0 = \alpha_0 p_0 + \alpha_1 p_1 + \ldots + \alpha_{n-1} p_{n-1}$$

To find direction p_k, we choose each new direction as a linear combination of negative residual $-r_k$ and the previous search vector p_{k-1}. So $p_k = -r_k + \beta_k p_{k-1}$ and α_k, β_k are found using conjugacy condition.

$r_{i+1} = r_i - \alpha_i A p_i$ tells us each new residual r_i is just a linear combination of the previous residual and Ap_{i-1}. It's natural to introduce a new subspace based on this fact:

Definition (Krylov subspace)

$$\mathcal{K} = span\{b, Ab, \ldots A^{k-1} b\}$$

And these methods involving Krylov subspace are also called 'Krylov subspace method'.

Algorithm

Compute $r_0 = Ax_0 - b, p_0 = -r_0$

For $k = 0, 1, 2, \ldots$ until converge

$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}$

$x_{k+1} = x_k + \alpha_k p_k$

$r_{k+1} = r_k + \alpha_k A p_k$

$\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$

$p_{k+1} = -r_{k+1} + \beta_k p_k$

Comments

CG method is not stable with respect to even small perturbations and it only works for symmetric and p.d. matrix. To improve it, you may consider biconjugate gradient method which is applicable for general matrix and biconjugate gradient stablized method(BICGSTAB).

Iterative Method : GMRES

Ideas

Approximate $Ax = b$ by a linear combination of Krylov vectors, i.e. $x^{m+1} = x^0 + \alpha_0 r^0 + \ldots + \alpha_m A^m r^0$ ($r_0 = b - Ax^0$ is initial vector). We need to find $\alpha_0, \ldots, \alpha_m$ such that r_n is minimized which is actually a least square problem.

Remarks

GMRES is suitable for all invertible square matrices.

Preconditioning

Definition

Condition Number Consider all small changes δA and δb in A and b and the resulting change, δx, in the solution x. Define

$$K(A) = ||A|| \cdot ||A^{-1}||$$

.

If the condition number of a matrix is very large, our solver would be unstable. However, our 'demand' is:

Preconditioning

Definition

Condition Number Consider all small changes δA and δb in A and b and the resulting change, δx, in the solution x. Define

$$K(A) = ||A|| \cdot ||A^{-1}||$$

.

If the condition number of a matrix is very large, our solver would be unstable. However, our 'demand' is:

- Accurate: we want fine mesh in our PDE solvers

Preconditioning

Definition

Condition Number Consider all small changes δA and δb in A and b and the resulting change, δx, in the solution x. Define

$$K(A) = ||A|| \cdot ||A^{-1}||$$

.

If the condition number of a matrix is very large, our solver would be unstable. However, our 'demand' is:

- Accurate: we want fine mesh in our PDE solvers
- Stable: we want system have small condition number

Preconditioning

Definition

Condition Number Consider all small changes δA and δb in A and b and the resulting change, δx, in the solution x. Define

$$K(A) = ||A|| \cdot ||A^{-1}||$$

.

If the condition number of a matrix is very large, our solver would be unstable. However, our 'demand' is:

- Accurate: we want fine mesh in our PDE solvers
- Stable: we want system have small condition number
- **mesh size decrease $\Rightarrow$ conditioning number increases**

Preconditioning

Definition

Condition Number Consider all small changes δA and δb in A and b and the resulting change, δx, in the solution x. Define

$$K(A) = ||A|| \cdot ||A^{-1}||$$

.

If the condition number of a matrix is very large, our solver would be unstable. However, our 'demand' is:

- Accurate: we want fine mesh in our PDE solvers
- Stable: we want system have small condition number
- **mesh size decrease $\Rightarrow$ conditioning number increases**

$\Rightarrow$ Let's use preconditioning matrix.

Suppose we wish to solve an $m \times m$ system $Ax = b$. For any nonsingular $m \times m$ matrix M, the system

$$M^{-1}Ax = M^{-1}b$$

has the same solution. If the *preconditioner* M is well chosen, we could solve the system more rapidly.

Example 1

Using $M = diag(A)$, very simple, cheap but usually insufficient.

Example 2

If A is symmetric and p.d., one trick is to use M also a symmetric p.d. matrix, with $M = CC^T$ for some C. Then system becomes

$$[C^{-1}ACC^{-T}]C^T x = C^{-1}b$$

The matrix in bracket is symmetric and p.d., so the equation can be solved easily by CG or other iteration method.

Example3

For some numerical PDE method, we may use a Finite Difference matrix as a preconditioner.e.g.

$$u_t + u_{xx} + \mathcal{N}\big(u(x,t)\big) = f(x,t)$$

References

Erwin Kreyszig, *Advanced Engineering Mathematics*, 8th Edn, Sections 11.4b

Trefethen, L.N. and Bau, D. 1997 *Numerical Linear Algebra*

Jonathan Richard Shewchuk *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*

Larsson, Stig, Thomee, Vidar 2003 *Partial Differential Equations with Numerical Methods*

The End