

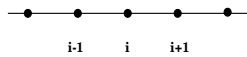
CS 689-001
Parallel Numerical Computation

Lecture 2*
Discretization and Solution Methods

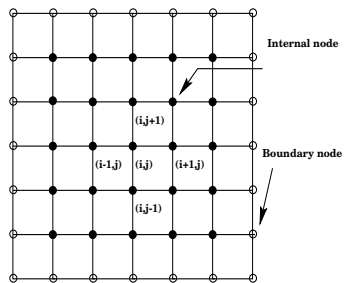
Professor Jun Zhang
 Department of Computer Science
 University of Kentucky
 Lexington, KY 40506-0046
 January 21, 1999

*Reference: *Iterative Methods for Sparse Linear Systems*, by Yousef Saad, 1996.

One Dimensional Approximation



Two Dimensional Approximation



Discretization Formulas

Approximating the first order differential operator by finite difference formulas

$$\left. \frac{du(x)}{dx} \right|_i \approx \frac{u_{i+1} - u_{i-1}}{2h} \quad \text{central difference}$$

$$\left. \frac{du(x)}{dx} \right|_i \approx \frac{u_i - u_{i-1}}{h} \quad \text{upwind difference}$$

The central difference scheme is more accurate than the upwind difference scheme.

The second order differential operators can be approximated by finite (central) difference formulas as

$$\left. \frac{d^2u(x,y)}{dx^2} \right|_{i,j} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$$

$$\left. \frac{d^2u(x,y)}{dy^2} \right|_{i,j} \approx \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2}$$

Model Problem

Suppose a two dimensional Poisson equation

$$\frac{d^2u(x,y)}{dx^2} + \frac{d^2u(x,y)}{dy^2} = f(x,y) \quad \text{in } \Omega$$

$$u(x,y) = g(x,y) \quad \text{on } \partial\Omega$$

is defined on the unit square $\Omega = [0, 1] \times [0, 1]$ with Dirichlet boundary conditions.

If we use the central difference approximation for the second differential operators, we have at an internal node (i, j)

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} = f_{i,j}$$

Difference Equations

For each internal node (i, j) the following equation holds

$$u_{i-1,j} + u_{i+1,j} - 4u_{i,j} + u_{i,j-1} + u_{i,j+1} = h^2 f_{i,j}$$

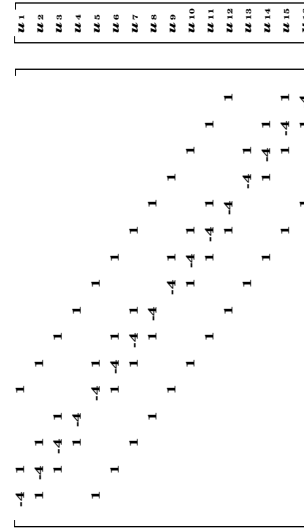
The linear equations at all internal nodes form a sparse system

$$Au = b,$$

where A is sparse (many entries are zero) and structured, consisting of 5 diagonals.

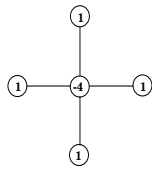
The matrix is said to be weakly diagonally dominant, if

$$|a_{i,i}| \geq \sum_{j \neq i} |a_{i,j}| \quad \text{for all } i$$

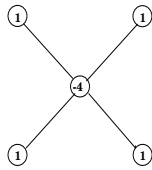


Coefficient matrix for a 4x4 Grid

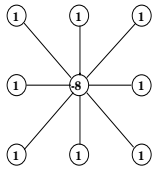
Other Discretization Schemes



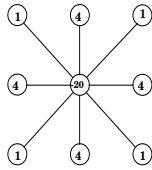
standard 5-point stencil



scheved 5-point stencil



a 9-point stencil



fourth order 9-point stencil

Direct Solution Methods

Direct solution methods based on Gaussian elimination

- robust with pivoting
- solution with high accuracy
- zero elements are filled in with nonzeros
- large memory and CPU consumption
- less parallelism due to triangular solves with forward elimination and backward substitution

Not very suitable for solving large sparse linear systems

Iterative Solution Methods

Iterative methods perform certain matrix-vector operations and generate a series of approximate vectors that may converge to the true solution (with certain accuracy)

- economic in memory consumption
- may be faster than direct methods
- easier to parallelize (than direct methods)
- viable for very large sparse linear systems
- not robustness. Iterative methods may not converge for given problems. Convergence guaranteed for restricted problems

Basic Iterative Methods (I)

Use iterative methods to solve

$$Au = b$$

Write the i th equation as

$$a_{i,i}u_i = -\sum_{j<i} a_{i,j}u_j - \sum_{j>i} a_{i,j}u_j + b_i$$

Provided $a_{i,i} \neq 0$, we have

$$u_i = \frac{-\sum_{j<i} a_{i,j}u_j - \sum_{j>i} a_{i,j}u_j + b_i}{a_{i,i}}$$

Equivalent to splitting the matrix A as

$$A = D - E - F$$

and (Jacobi iteration)

$$u^{(k+1)} = D^{-1}(E + F)u^{(k)} + D^{-1}b$$

Current Developments

- gaining industry acceptance (slowly)
- public software packages and test problems available
 - NETLIB <http://www.netlib.org>
 - National Institute of Standards and Tech.
 - <http://math.nist.gov/MatrixMarket>
- enhance robustness
- develop parallelizable iterative methods
- develop preconditioning techniques
- identify applications and categories
- efficient parallel implementations

Basic Iterative Methods (II)

Jacobi iteration updates the approximate solution vector once in a time. Each component is updated by using old values of all other components. Its convergence is slow.

In contrast, Gauss-Seidel iteration updates each components by using new approximate components as soon as they are computed, i.e.,

$$a_{i,i}u_i^{(k+1)} = -\sum_{j<i} a_{i,j}u_j^{(k+1)} - \sum_{j>i} a_{i,j}u_j^{(k)} + b_i$$

or

$$u_i^{(k+1)} = \frac{-\sum_{j<i} a_{i,j}u_j^{(k+1)} - \sum_{j>i} a_{i,j}u_j^{(k)} + b_i}{a_{i,i}}$$

Basic Iterative Methods (III)

In matrix notation, Gauss-Seidel iteration is

$$u^{(k+1)} = (D - E)^{-1}Fu^{(k)} + (D - E)^{-1}b$$

Pros and cons of Gauss-Seidel vs. Jacobi

- Gauss-Seidel is almost twice faster
- Gauss-Seidel is inherently sequential

Even faster convergence can be obtained using successive over relaxation (SOR) method following the Gauss-Seidel iteration

$$u_i^{(k+1)} = \omega u_{iGS}^{(k+1)} + (1 - \omega)u_i^{(k)}$$

ω is the SOR parameter.

Basic Iterative Methods (IV)

SOR in matrix form is

$$(D - \omega E)u^{(k+1)} = [\omega F + (1 - \omega)D]u^{(k)} + \omega b$$

Suitable ω values are between 0 and 2. Convergence can be accelerated significantly if an optimal SOR parameter can be found. This can only be done with simple model problems.

An SOR step can also be applied to a Jacobi iteration. The resulting method is called damped Jacobi method.

Symmetric SOR (SSOR) is two SOR steps performed in different directions, used mainly for providing symmetric preconditioning

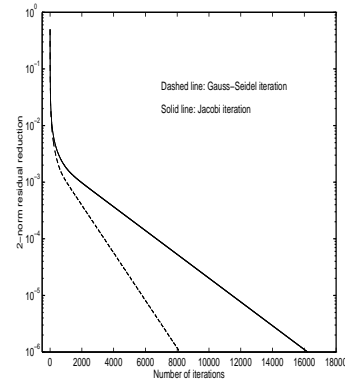


Figure 1: Jacobi and Gauss-Seidel methods for solving a one dimensional Poisson equation with $n = 100$ and 6 order of residual norm reduction.

- Gauss-Seidel is faster than Jacobi;
- Convergence slows down for both methods;
- None of them is good for real applications.

Parallelizing Gauss-Seidel

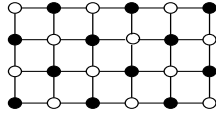
To introduce parallelism in Gauss-Seidel, a coloring scheme is needed to decouple the grid nodes.

For 5-point stencil, two colors are needed (red-black coloring)

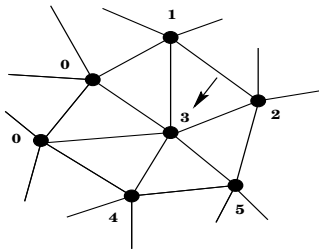
For 9-point stencil, four colors are needed

For general sparse matrices, the adjacency graph should be colored using a greedy algorithm

Ref.: L. M. Adams and J. Ortega, *A multi-color SOR method for parallel computers*, in Proceedings of the 1982 International Conference on Parallel Processing, p. 53–56, 1982.



Red-black coloring of 5-point stencil



Greedy algorithm for coloring graph

Block Relaxation Schemes (I)

A matrix may be written in block form

$$A = \begin{pmatrix} A_{11} & A_{12} & A_{13} & \dots & A_{1p} \\ A_{21} & A_{22} & A_{23} & \dots & A_{2p} \\ A_{31} & A_{32} & A_{33} & \dots & A_{3p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{p1} & A_{p2} & A_{p3} & \dots & A_{pp} \end{pmatrix}$$

Where A_{ii} are square submatrices. The vector u and the right hand side b should be partitioned conformally. The matrix A can still be splitted as $A = D - E - F$ with

$$D = \begin{pmatrix} A_{11} & & & & \\ & A_{22} & & & \\ & & A_{33} & & \\ & & & \ddots & \\ & & & & A_{pp} \end{pmatrix}$$

being a block diagonal matrix, E and F are lower and upper block triangular matrices

Block Relaxation Schemes (II)

Block Jacobi, Gauss-Seidel, and SOR can be defined similarly to the point versions, as

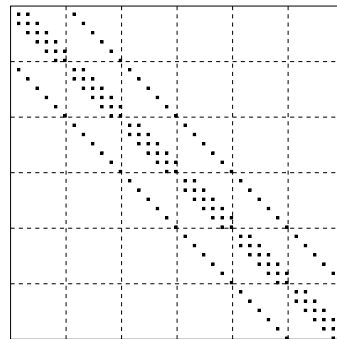
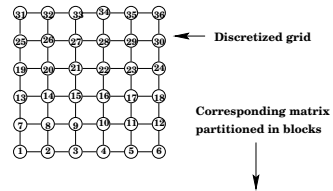
$$A_{ii}u_i^{(k+1)} = ((E + F)x^{(k)})_i + b_i$$

Provided A_{ii} is invertible, we have

$$u_i^{(k+1)} = A_{ii}^{-1}((E + F)x^{(k)})_i + A_{ii}^{-1}b_i$$

for $i = 1, 2, \dots, p$.

It is sometimes useful to partition the grid points along the vertical or horizontal lines. The corresponding schemes are called line relaxation techniques and are more robust than the analogous point schemes.



Block Relaxation Schemes (III)

Block schemes are useful in parallel and high performance computations as they involve certain domain decomposition strategies

- blocks can be overlapped. Convergence may depend on the degree of overlapping
- block size may be chosen suitable for a particular parallel architecture
- for large size blocks, inverting the block diagonal matrix may be a problem
- iterative solve for blocks and approximate inverse techniques may be employed

Iteration and Preconditioning

The iteration process

$$u^{(k+1)} = Gu^{(k)} + f$$

can be viewed as a fixed point iteration to solve

$$(I - G)u = f \quad (1)$$

With $G = I - M^{-1}A$, the system (1) can be written as

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

which is called the preconditioned system of the original system $Au = b$

Iteration Matrices

Jacobi and Gauss-Seidel iterations can be written in matrix form

$$u^{(k+1)} = Gu^{(k)} + f$$

with

$$G_{JA}(A) = D^{-1}(E + F) = I - D^{-1}A$$

$$G_{GS}(A) = (D - E)^{-1}F = I - (D - E)^{-1}A$$

In general matrix splitting, we have

$$A = M - N$$

with M^{-1} easy to compute, and iterate with

$$u^{(k+1)} = M^{-1}Nu^{(k)} + M^{-1}b$$

Convergence

If the iteration process

$$u^{(k+1)} = Gu^{(k)} + f \quad (1)$$

converges to u , then

$$u = Gu + f \quad (2)$$

Let $u^{(*)}$ be a solution to (2), subtracting (2) from (1) gives

$$\begin{aligned} u^{(k+1)} - u^{(*)} &= G(u^{(k)} - u^{(*)}) \\ &= \dots = G^{k+1}(u^{(0)} - u^{(*)}) \end{aligned}$$

Hence, in order for the iteration to converge, the spectral radius of G must satisfy

$$\rho(G) \leq \|G\| < 1$$

Example: Richardson's Iteration

$$u^{(k+1)} = u^{(k)} + \alpha(b - Au^{(k)})$$

with nonnegative α can be rewritten as

$$u^{(k+1)} = (I - \alpha A)u^{(k)} + \alpha b$$

The iteration matrix is $G_\alpha = I - \alpha A$. Let the eigenvalues of A be $\lambda_i, i = 1, \dots, n$ and real, and

$$\lambda_{\min} \leq \lambda_i \leq \lambda_{\max}$$

The eigenvalues μ_i of $G_\alpha = I - \alpha A$ are

$$1 - \alpha\lambda_{\max} \leq \mu_i \leq 1 - \alpha\lambda_{\min}$$

The method converges for any scalar α with

$$0 < \alpha < \frac{2}{\lambda_{\max}}$$

References

Many numerical computation software can be found on NETLIB at <http://www.netlib.org>. There is a collection of iterative solvers for linear systems, called *templates*, that can be found at

<http://www.netlib.org/templates/index.html>.

You can also find the book *Templates for Iterative Solution of Linear Systems* there in postscript file. This electronic book is also available on our main CS689 page. You can download and read it as a substitute for a text book, and then download software from netlib to see how an algorithm behaves.