REVIEW Lecture 7:

• Direct Methods for solving linear algebraic equations
  – Gauss Elimination, LU decomposition/factorization
  – Error Analysis for Linear Systems and Condition Numbers
  – Special Matrices: LU Decompositions
    • Tri-diagonal systems: Thomas Algorithm \( (\text{Nb Ops: } 8 O(n)) \)
    • General Banded Matrices
      – Algorithm, Pivoting and Modes of storage
      – Sparse and Banded Matrices
    • Symmetric, positive-definite Matrices
      – Definitions and Properties, Choleski Decomposition

• Iterative Methods
  – Concepts and Definitions
    \[ x^{k+1} = B x^k + c \quad k = 0, 1, 2, \ldots \]
  – Convergence: Necessary and Sufficient Condition
    \[ \rho(B) = \max_{i=1 \ldots n} |\lambda_i| < 1, \quad \text{where } \lambda_i = \text{eigenvalue}(B_{n \times n}) \] (ensures \( ||B|| < 1 \))
TODAY (Lecture 8): Systems of Linear Equations IV

• Direct Methods
  – Gauss Elimination
  – LU decomposition/factorization
  – Error Analysis for Linear Systems
  – Special Matrices: LU Decompositions

• Iterative Methods
  – Concepts, Definitions, Convergence and Error Estimation
  – Jacobi’s method
  – Gauss-Seidel iteration
  – Stop Criteria
  – Example
  – Successive Over-Relaxation Methods
  – Gradient Methods and Krylov Subspace Methods
  – Preconditioning of $Ax=b$
Reading Assignment


  – Any chapter on iterative and gradient methods for solving linear systems, e.g. chapter 7 of Ascher and Greif, *SIAM*, 2011.
Express error as a function of latest increment:

\[
\bar{x}^{(k)} - \bar{x} = \Bar{B} (\bar{x}^{(k-1)} - \bar{x}) \pm \Bar{B} \bar{x}^{(k)} \\
= -\Bar{B} (\bar{x}^{(k)} - \bar{x}^{(k-1)}) + \Bar{B} (\bar{x}^{(k)} - \bar{x})
\]

\[
\Rightarrow \|\bar{x}^{(k)} - \bar{x}\| \leq \|\Bar{B}\| \|\bar{x}^{(k)} - \bar{x}^{(k-1)}\| + \|\Bar{B}\| \|\bar{x}^{(k)} - \bar{x}\|
\]

\[
\|\bar{x}^{(k)} - \bar{x}\| \leq \frac{\|\Bar{B}\|}{1 - \|\Bar{B}\|} \|\bar{x}^{(k)} - \bar{x}^{(k-1)}\| \quad \text{(if } \|\Bar{B}\| < 1)\]

If we define \(\beta = \|\Bar{B}\| < 1\), it is only if \(\beta \leq 0.5\) that it is adequate to stop the iteration when the last relative error is smaller than the tolerance (if not, actual errors can be larger)
Linear Systems of Equations: Iterative Methods

General Case and Stop Criteria

• General Formula

\[ Ax_e = b \]

\[ x_{i+1} = B_i x_i + C_i b \quad i = 1, 2, \ldots \]

• Numerical convergence stops:

\[ i \leq n_{\text{max}} \]

\[ \| x_i - x_{i-1} \| \leq \varepsilon \]

\[ \| r_i - r_{i-1} \| \leq \varepsilon, \ \text{where} \ r_i = Ax_i - b \]

\[ \| r_i \| \leq \varepsilon \]

(if \( x_i \) not normalized, use relative versions of the above)
Linear Systems of Equations: **Iterative Methods**

Element-by-Element Form of the Equations

Sparse (large) Full-bandwidth Systems (frequent in practice)

**Iterative Methods are then efficient**

Analogous to iterative methods obtained for roots of equations, i.e. Open Methods: Fixed-point, Newton-Raphson, Secant

Rewrite Equations

\[
\begin{align*}
\overline{A}\overline{x} &= \overline{b} \\
\Leftrightarrow \sum_{j=1}^{n} a_{ij}x_j &= b_i
\end{align*}
\]

\[
a_{ii} \neq 0 \Rightarrow x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}}, \quad i = 1, \ldots, n
\]

Note: each \( x_i \) is a scalar here, the \( i^{th} \) element of \( \overline{x} \)
Iterative Methods: Jacobi and Gauss Seidel

Rewrite Equations:

\[ \overline{A} \mathbf{x} = \mathbf{b} \iff \sum_{j=1}^{n} a_{ij} x_j = b_i \]

\[ a_{ii} \neq 0 \Rightarrow x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j - \sum_{j=i+1}^{n} a_{ij} x_j}{a_{ii}} , \; i = 1, \ldots n \]

=> Iterative, Recursive Methods:

**Jacobi’s Method**

\[ x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}}{a_{ii}} , \; i = 1, \ldots n \]

Computes a full new \( \mathbf{x} \) based on full old \( \mathbf{x} \), i.e.
Each new \( x_i \) is computed based on all old \( x_i \)’s

**Gauss-Seidel’s Method**

\[ x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}}{a_{ii}} , \; i = 1, \ldots n \]

New \( \mathbf{x} \) based most recent \( \mathbf{x} \) elements, i.e.
The new \( x_1^{k+1} \cdots x_{i-1}^{k+1} \) directly used to compute next element \( x_i^{k+1} \)
Iterative Methods: Jacobi’s Matrix form

Iteration – Matrix form

\[ \mathbf{x}^{(k+1)} = \mathbf{B}\mathbf{x}^{(k)} + \mathbf{c}, \quad k = 0, \ldots \]

Decompose Coefficient Matrix

\[ \overline{\mathbf{A}} = \overline{\mathbf{D}} + \overline{\mathbf{L}} + \overline{\mathbf{U}} \]

with

\[ \overline{\mathbf{D}} = \text{diag} \ a_{ii} \]

\[ \overline{\mathbf{L}} = \begin{cases} a_{ij}, & i > j \\ 0, & i \leq j \end{cases} \]

\[ \overline{\mathbf{U}} = \begin{cases} a_{ij}, & i < j \\ 0, & i \geq j \end{cases} \]

Note: this is NOT LU-factorization

Jacobi’s Method

\[ x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}}{a_{ii}}, \quad i = 1, \ldots n \]

Iteration Matrix form

\[ \mathbf{x}^{(k+1)} = -\overline{\mathbf{D}}^{-1}(\overline{\mathbf{L}} + \overline{\mathbf{U}}) \mathbf{x}^{(k)} + \overline{\mathbf{D}}^{-1} \mathbf{b} \]

\[ \mathbf{B} = -\overline{\mathbf{D}}^{-1}(\overline{\mathbf{L}} + \overline{\mathbf{U}}) \]

\[ \mathbf{c} = \overline{\mathbf{D}}^{-1} \mathbf{b} \]
Convergence of Jacobi and Gauss-Seidel

- **Jacobi:**
  \[ \mathbf{A} \mathbf{x} = \mathbf{b} \quad \Rightarrow \quad \mathbf{D} \mathbf{x} + (\mathbf{L} + \mathbf{U}) \mathbf{x} = \mathbf{b} \]

  \[ \mathbf{x}^{k+1} = -\mathbf{D}^{-1} (\mathbf{L} + \mathbf{U}) \mathbf{x}^{k} + \mathbf{D}^{-1} \mathbf{b} \]

- **Gauss-Seidel:**
  \[
  x_{i}^{(k+1)} = \frac{b_{i} - \sum_{j=1}^{i-1} a_{ij} x_{j}^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_{j}^{(k)}}{a_{ii}}, \quad i = 1, \ldots, n
  \]

  \[ \mathbf{A} \mathbf{x} = \mathbf{b} \quad \Rightarrow \quad (\mathbf{D} + \mathbf{L}) \mathbf{x} + \mathbf{U} \mathbf{x} = \mathbf{b} \]

  \[ \mathbf{x}^{k+1} = -\mathbf{D}^{-1} \mathbf{L} \mathbf{x}^{k+1} - \mathbf{D}^{-1} \mathbf{U} \mathbf{x}^{k} + \mathbf{D}^{-1} \mathbf{b} \quad \text{or} \]

  \[ \mathbf{x}^{k+1} = -(\mathbf{D} + \mathbf{L})^{-1} \mathbf{U} \mathbf{x}^{k} + (\mathbf{D} + \mathbf{L})^{-1} \mathbf{b} \]

- Both converge if \( \mathbf{A} \) strictly diagonal dominant
- Gauss-Seidel also convergent if \( \mathbf{A} \) symmetric positive definite matrix
- Also Jacobi convergent for \( \mathbf{A} \) if
  - \( \mathbf{A} \) symmetric and \( \{\mathbf{D}, \mathbf{D} + \mathbf{L} + \mathbf{U}, \mathbf{D} - \mathbf{L} - \mathbf{U}\} \) are all positive definite
Sufficient Condition for Convergence
Proof for Jacobi

\[
A \mathbf{x} = \mathbf{b} \quad \Rightarrow \quad D \mathbf{x} + (L + U) \mathbf{x} = \mathbf{b}
\]

\[
\mathbf{x}^{k+1} = -D^{-1} (L + U) \mathbf{x}^k + D^{-1} \mathbf{b}
\]

**Sufficient Convergence Condition**

\[
\left\| \mathbf{B} \right\| < 1
\]

**Jacobi’s Method**

\[
b_{ij} = -\frac{a_{ij}}{a_{ii}}, \quad i \neq j
\]

**Using the \( \infty \)-Norm (Maximum Row Sum)**

\[
\left\| \mathbf{B} \right\|_{\infty} = \max_{i} \sum_{j=1, j \neq i}^{n} |a_{ij}|
\]

Hence, Sufficient Convergence Condition is:

\[
\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|
\]

**Strict Diagonal Dominance**
Illustration of Convergence (left) and Divergence (right) of the Gauss-Seidel Method

Illustration of (A) convergence and (B) divergence of the Gauss-Seidel method. Notice that the same functions are plotted in both cases (u: 11x1 + 13x2 = 286; v: 11x1 - 9x2 = 99).

Image by MIT OpenCourseWare.
Special Matrices: **Tri-diagonal Systems**

Example “Forced Vibration of a String”

Finite Difference

\[
\frac{d^2 y}{dx^2} \bigg|_{x_i} \approx \frac{y_{i-1} - 2y_i + y_{i+1}}{h^2}
\]

Discrete Difference Equations

\[
y_{i-1} + ((kh)^2 - 2)y_i + y_{i+1} = f(x_i)h^2
\]

Matrix Form

\[
\begin{bmatrix}
(kh)^2 - 2 & 1 & \cdot & \cdot & \cdot & 0 \\
1 & (kh)^2 - 2 & 1 & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & 1 & (kh)^2 - 2 & 1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
0 & \cdot & \cdot & \cdot & 1 & (kh)^2 - 2
\end{bmatrix}
\]

Differential Equation:

\[
\frac{d^2 y}{dx^2} + k^2 y = f(x)
\]

Boundary Conditions: \(y(0) = 0, \ y(L) = 0\)

Strict Diagonal Dominance?

\(kh > 2 \Rightarrow h > \frac{2}{k}\)

For Jacobi, recall that a sufficient condition for convergence is:

With \(B = -D^{-1}(L+U)\):

\[
|a_{ii}| > \sum_{j=1, j \neq i}^{n} |a_{ij}| \quad \Rightarrow \quad \|B\|_\infty = \max_{i=1 \ldots n} \left( \sum_{j=1}^{n} |b_{ij}| \right) = \max_{i=1 \ldots n} \left( \sum_{j=1, j \neq i}^{n} \frac{|a_{ij}|}{|a_{ii}|} \right) < 1
\]
vib_string.m  (Part I)

n=99;
L=1.0;
h=L/(n+1);
k=2*pi;
kh=k*h

% Tri-Diagonal Linear System: Ax=b
x=[h:h:L-h]';
a=zeros(n,n);
f=zeros(n,1);

% Offdiagonal values
o=1.0

a(1,1) =kh^2 - 2;
a(1,2)=o;

for i=2:n-1
    a(i,i)=a(1,1);
    a(i,i-1) = o;
    a(i,i+1) = o;
end
a(n,n)=a(1,1);
a(n,n-1)=o;

% Hanning window load
nf=round((n+1)/3);
nw=round((n+1)/6);
nw=min(min(nw,nf-1),n-nf);
figure(1)
hold off

nw1=nf-nw;
nw2=nf+nw;
f(nw1:nw2) = h^2*hanning(nw2-nw1+1);

% Exact solution
y=inv(a)*f;
subplot(2,1,2); p=plot(x,y,'b');set(p,'LineWidth',2);
p=legend(['Off-diag. = ' num2str(o)]);
set(p,'FontSize',14);
p=title('String Displacement (Exact)');
set(p,'FontSize',14);
p=xlabel('x');
set(p,'FontSize',14);
%Iterative solution using Jacobi's and Gauss-Seidel's methods
b=-a;
c=zeros(n,1);
for i=1:n
    b(i,i)=0;
    for j=1:n
        b(i,j)=b(i,j)/a(i,i);
        c(i)=f(i)/a(i,i);
    end
end

nj=100;
xj=f;
xgs=f;

figure(2)
c=6
col=['r' 'g' 'b' 'c' 'm' 'y']
hold off
for j=1:nj
    % jacobi
    xj=b*xj+c;
    % gauss-seidel
    xgs(1)=b(1,2:n)*xgs(2:n) + c(1);
    for i=2:n-1
        xgs(i)=b(i,1:i-1)*xgs(1:i-1) + b(i,i+1:n)*xgs(i+1:n) +c(i);
    end
    xgs(n)= b(n,1:n-1)*xgs(1:n-1) +c(n);
    cc=col(mod(j-1,nc)+1);
    subplot(2,1,1); plot(x,xj,cc); hold on;
    p=title('Jacobi');
    set(p,'FontSize',14);
    subplot(2,1,2); plot(x,xgs,cc); hold on;
    p=title('Gauss-Seidel');
    set(p,'FontSize',14);
    p=xlabel('x');
    set(p,'FontSize',14);
end
vib_string.m
\( o=1.0, \, k=2\pi, \, h=.01, \, kh<2 \)

Exact Solution

Iterative Solutions

Coefficient Matrix Not Strictly Diagonally Dominant
**vib_string.m**

\( o = 1.0, \quad k = 2\pi \times 31, \quad h = 0.01, \quad kh < 2 \)

**Exact Solution**

**Iterative Solutions**

**Coefficient Matrix Not Strictly Diagonally Dominant**
vib_string.m
\( o=1.0, \ k=2*\pi*33, \ h=0.01, \ kh>2 \)

**Exact Solution**

- Force Distribution
- String Displacement

**Iterative Solutions**

- Jacobi
- Gauss-Seidel

**Coefficient Matrix Strictly Diagonally Dominant**
vib_string.m
\( \omega=1.0, \ k=2\pi \times 50, \ h=0.01, \ kh>2 \)

**Exact Solution**
- Force Distribution
- String Displacement

**Iterative Solutions**
- Jacobi
- Gauss-Seidel

**Coefficient Matrix Strictly Diagonally Dominant**
vib_string.m
\( o = 0.5, \ k = 2*\pi, \ h = .01 \)

Exact Solution

Iterative Solutions

Coefficient Matrix Strictly Diagonally Dominant
Successive Over-relaxation (SOR) Method

- Aims to reduce the spectral radius of B to increase rate of convergence
- Add an extrapolation to each step of Gauss-Seidel

\[ x_i^{k+1} = \omega \overline{x}_i^{k+1} + (1 - \omega)x_i^k, \text{ where } \overline{x}_i^{k+1} \text{ computed by Gauss–Seidel} \]

- If “A” symmetric and positive definite \( \Rightarrow \) converges for \( 0 < \omega < 2 \)

- Matrix format:

\[
x^{k+1} = (D + \omega L)^{-1}[-\omega U + (1 - \omega)D]x^k + \omega(D + \omega L)^{-1}b
\]

- Hard to find optimal value of over-relaxation parameter for fast convergence (aim to minimize spectral radius of B) because it depends on BCs, etc.

\[ \omega = \omega_{opt} = ? \]
Gradient Methods

• Prior iterative schemes (Jacobi, GS, SOR) were “stationary” methods (iterative matrices B remained fixed throughout iteration)

• Gradient methods:
  – utilize gathered information throughout iterations (i.e. improve estimate of the inverse along the way)
  – Applicable to physically important matrices: “symmetric and positive definite” ones

• Construct the equivalent optimization problem

\[ Q(x) = \frac{1}{2} x^T Ax - x^T b \]

\[ \frac{dQ(x)}{dx} = Ax - b \]

\[ \left. \frac{dQ(x)}{dx} \right|_{x_{opt}} = 0 \Rightarrow x_{opt} = x_e, \text{ where } Ax_e = b \]

• Propose step rule

\[ x_{i+1} = x_i + \alpha_i v_i \]

search direction at iteration \( i + 1 \)

step size at iteration \( i + 1 \)

• Common methods
  – Steepest descent
  – Conjugate gradient

• Note: above step rule includes iterative “stationary” methods (Jacobi, GS, SOR, etc.)
Steepest Descent Method

• Move exactly in the negative direction of the Gradient

\[
\frac{dQ(x)}{dx} = Ax - b = -(b - Ax) = -r
\]

\(r: \text{residual}, \ r_i = b - Ax_i\)

• Step rule (obtained in lecture)

\[
x_{i+1} = x_i + \frac{r_i^T r_i}{r_i^T A r_i} r_i
\]

• \(Q(x)\) reduces in each step, but slow and not as effective as conjugate gradient method

Image by MIT OpenCourseWare.

Graph showing the steepest descent method.
Conjugate Gradient Method

- Definition: “$A$-conjugate vectors” or “Orthogonality with respect to a matrix (metric)”:
  
  if $A$ is symmetric & positive definite,

  For $i \neq j$ we say $v_i, v_j$ are orthogonal with respect to $A$, if $v_i^T A v_j = 0$

- Proposed in 1952 (Hestenes/Stiefel) so that directions $v_i$ are generated by the orthogonalization of residuum vectors (search directions are $A$-conjugate)
  - Choose new descent direction as different as possible from old ones, within $A$-metric

- Algorithm:

  $v_0 = r_0 = b - Ax_0$

  do

  $\alpha_i = (v_i^T r_i) / (v_i^T A v_i)$

  $x_{i+1} = x_i + \alpha_i v_i$

  $r_{i+1} = r_i - \alpha_i A v_i$

  $\beta_i = -(v_i^T A r_{i+1}) / (v_i^T A v_i)$

  $v_{i+1} = r_{i+1} + \beta_i v_i$

  until a stop criterion holds

  Note: $A v_i =$ one matrix vector multiply at each iteration

- Derivation provided in lecture
- Check CGM_new.m

Figure indicates solution obtained using Conjugate gradient method (red) and steepest descent method (green).
2.29 Numerical Fluid Mechanics
Spring 2015

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.