2.29 Numerical Fluid Mechanics
Spring 2015 – **Lecture 8**

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 (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

p super-diagonals
q sub-diagonals
$w = p + q + 1$ bandwidth

- Iterative Methods
  - Concepts and Definitions $\boxed{\mathbf{x}^{k+1} = \mathbf{B}\,\mathbf{x}^k + \mathbf{c} \qquad k = 0,1,2,\ldots}$
  - Convergence: Necessary and Sufficient Condition

$$\boxed{\rho(\mathbf{B}) = \max_{i=1\ldots n} |\lambda_i| < 1, \quad \text{where } \lambda_i = \text{eigenvalue}(\mathbf{B}_{n\times n})}$$ (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 $\mathbf{Ax=b}$

Reading Assignment

- Chapters 11 of Chapra and Canale, Numerical Methods for Engineers, 2006/2010/2014."
  - Any chapter on "Solving linear systems of equations" in references on CFD references provided. For example: chapter 5 of "J. H. Ferziger and M. Peric, Computational Methods for Fluid Dynamics. Springer, NY, 3rd edition, 2002"
- Chapter 14.2 on "Gradient Methods" of Chapra and Canale, Numerical Methods for Engineers, 2006/2010/2014."
  - Any chapter on iterative and gradient methods for solving linear systems, e.g. chapter 7 of Ascher and Greif, SIAM, 2011.

Error Estimation and Stop Criterion

Express error as a function of latest increment:

$$\begin{aligned}\overline{\mathbf{x}}^{(k)} - \overline{\mathbf{x}} &= \overline{\overline{\mathbf{B}}}\left(\overline{\mathbf{x}}^{(k-1)} - \overline{\mathbf{x}}\right) \qquad\qquad \pm\, \overline{\overline{\mathbf{B}}}\,\overline{\mathbf{x}}^{(k)} \\ &= -\overline{\overline{\mathbf{B}}}\left(\overline{\mathbf{x}}^{(k)} - \overline{\mathbf{x}}^{(k-1)}\right) + \overline{\overline{\mathbf{B}}}\left(\overline{\mathbf{x}}^{(k)} - \overline{\mathbf{x}}\right)\end{aligned}$$

$$\Rightarrow \quad \left\|\overline{\mathbf{x}}^{(k)} - \overline{\mathbf{x}}\right\| \;\le\; \left\|\overline{\overline{\mathbf{B}}}\right\|\left\|\overline{\mathbf{x}}^{(k)} - \overline{\mathbf{x}}^{(k-1)}\right\| + \left\|\overline{\overline{\mathbf{B}}}\right\|\left\|\overline{\mathbf{x}}^{(k)} - \overline{\mathbf{x}}\right\|$$

$$\left\|\overline{\mathbf{x}}^{(k)} - \overline{\mathbf{x}}\right\| \le \frac{\left\|\overline{\overline{\mathbf{B}}}\right\|}{1 - \left\|\overline{\overline{\mathbf{B}}}\right\|}\left\|\overline{\mathbf{x}}^{(k)} - \overline{\mathbf{x}}^{(k-1)}\right\| \qquad (\text{ if } \|\mathbf{B}\| < 1 \,)$$

$$\boxed{\left\|\overline{\overline{\mathbf{B}}}\right\| < 1/2 \Rightarrow \left\|\overline{\mathbf{x}}^{(k)} - \overline{\mathbf{x}}\right\| \le \left\|\overline{\mathbf{x}}^{(k)} - \overline{\mathbf{x}}^{(k-1)}\right\|}$$

If we define $\beta = \|\mathbf{B}\| < 1$, it is only if $\beta <= 0.5$ 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)

General Case and Stop Criteria

- General Formula

$$Ax_e = b$$

$$x_{i+1} = B_i x_i + C_i b \qquad\qquad i = 1, 2, \ldots..$$

- Numerical convergence stops:

$$i \le n_{\max}$$

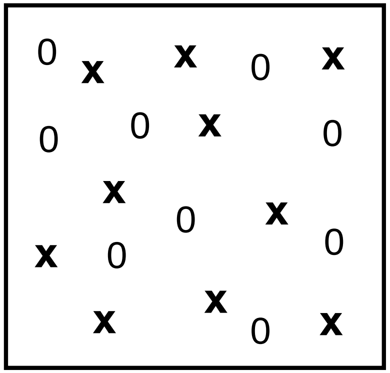
$$\|x_i - x_{i-1}\| \le \varepsilon$$

$$\|r_i - r_{i-1}\| \le \varepsilon, \ \ where \ \ r_i = Ax_i - b$$

$$\|r_i\| \qquad \le \varepsilon$$

(if x_i not normalized, use relative versions of the above)

Linear Systems of Equations: **Iterative Methods**
Element-by-Element Form of the Equations

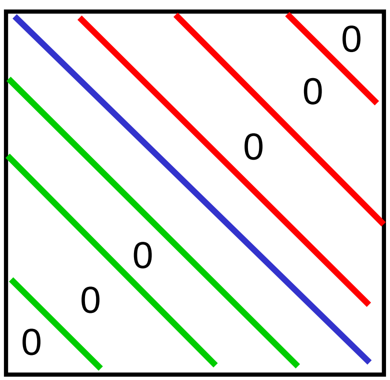


Sparse (large) Full-bandwidth Systems (frequent in practice)

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



$$\overline{\overline{\mathbf{A}}}\,\overline{\mathbf{x}} = \overline{\mathbf{b}} \quad \Leftrightarrow \quad \sum_{j=1}^{n} a_{ij} x_j = b_i$$

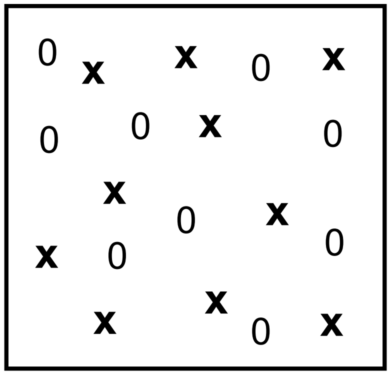
$$\boxed{a_{ii} \neq 0 \Rightarrow x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j - \sum_{j=i+1}^{n} a_{ij} x_j}{a_{ii}}\ ,\ i = 1, \ldots n}$$

Note: each x_i is a scalar here, the i^{th} element of $\overline{\mathbf{x}}$

Iterative Methods: Jacobi and Gauss Seidel

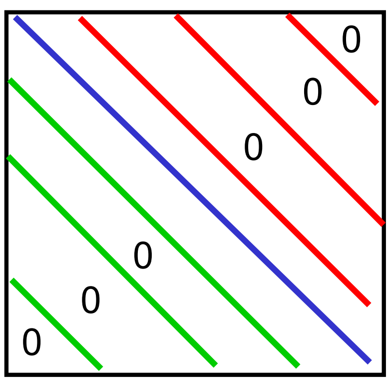
Rewrite Equations: $\bar{\bar{\mathbf{A}}}\bar{\mathbf{x}} = \bar{\mathbf{b}} \quad \Leftrightarrow \quad \sum_{j=1}^{n} a_{ij} x_j = b_i$

Sparse, Full-bandwidth Systems



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

=> Iterative, Recursive Methods:



Jacobi's Method

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

Computes a full new **x** based on full old **x**, i.e.
Each new x_i is computed based on all old x_i's

Gauss-Seidel's Method

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

New **x** based most recent **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

$$\overline{\mathbf{x}}^{(k+1)} = \overline{\overline{\mathbf{B}}}\,\overline{\mathbf{x}}^{(k)} + \overline{\mathbf{c}}\ ,\ k = 0, \ldots$$

Decompose Coefficient Matrix

$$\overline{\overline{\mathrm{A}}} = \overline{\overline{\mathrm{D}}} + \overline{\overline{\mathrm{L}}} + \overline{\overline{\mathrm{U}}}$$

with

$$\overline{\overline{\mathbf{D}}} = \operatorname{diag} a_{ii}$$

$$\overline{\overline{\mathbf{L}}} = \begin{cases} a_{ij} & ,\quad i > j \\ 0, & \quad i \le j \end{cases}$$

Note: this is NOT LU-factorization

$$\overline{\overline{\mathbf{U}}} = \begin{cases} a_{ij} & ,\quad i < j \\ 0, & \quad i \ge j \end{cases}$$

Jacobi's Method

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

$$\boxed{\overline{\mathbf{x}}^{(k+1)} = -\overline{\overline{\mathbf{D}}}^{-1}\left(\overline{\overline{\mathbf{L}}} + \overline{\overline{\mathbf{U}}}\right)\overline{\mathbf{x}}^{(k)} + \overline{\overline{\mathbf{D}}}^{-1}\overline{\mathbf{b}}}$$

Iteration Matrix form

$$\begin{cases} \overline{\overline{\mathbf{B}}} = -\overline{\overline{\mathbf{D}}}^{-1}\left(\overline{\overline{\mathbf{L}}} + \overline{\overline{\mathbf{U}}}\right) \\ \overline{\mathbf{c}} = \overline{\overline{\mathbf{D}}}^{-1}\overline{\mathbf{b}} \end{cases}$$

Convergence of Jacobi and Gauss-Seidel

- Jacobi: $\mathbf{A}\,\mathbf{x} = \mathbf{b} \quad \Rightarrow \quad \mathbf{D}\,\mathbf{x} + (\mathbf{L}+\mathbf{U})\,\mathbf{x} = \mathbf{b}$

$$\boxed{\mathbf{x}^{k+1} = -\mathbf{D}^{-1}(\mathbf{L}+\mathbf{U})\,\mathbf{x}^{k} + \mathbf{D}^{-1}\mathbf{b}}$$

- Gauss-Seidel: $x_i^{(k+1)} = \dfrac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}}{a_{ii}} \;, \; i = 1, \ldots n$

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

$$\mathbf{x}^{k+1} = -\mathbf{D}^{-1}\mathbf{L}\,\mathbf{x}^{k+1} - \mathbf{D}^{-1}\mathbf{U}\,\mathbf{x}^{k} + \mathbf{D}^{-1}\mathbf{b} \qquad \text{or}$$

$$\boxed{\mathbf{x}^{k+1} = -(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}\,\mathbf{x}^{k} + (\mathbf{D}+\mathbf{L})^{-1}\mathbf{b}}$$

- 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

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

$$\boxed{\mathbf{x}^{k+1} = \underbrace{-\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})}\,\mathbf{x}^{k} + \mathbf{D}^{-1}\mathbf{b}}$$

Sufficient Convergence Condition $\left\|\overline{\overline{\mathbf{B}}}\right\| < 1$

Jacobi's Method $b_{ij} = -\frac{a_{ij}}{a_{ii}} \;,\; i \neq j$

Using the ∞-Norm
(Maximum Row Sum) $\left\|\overline{\overline{\mathbf{B}}}\right\|_{\infty} = \max_{i} \sum_{j=1, j\neq i}^{n} \frac{|a_{ij}|}{|a_{ii}|}$

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

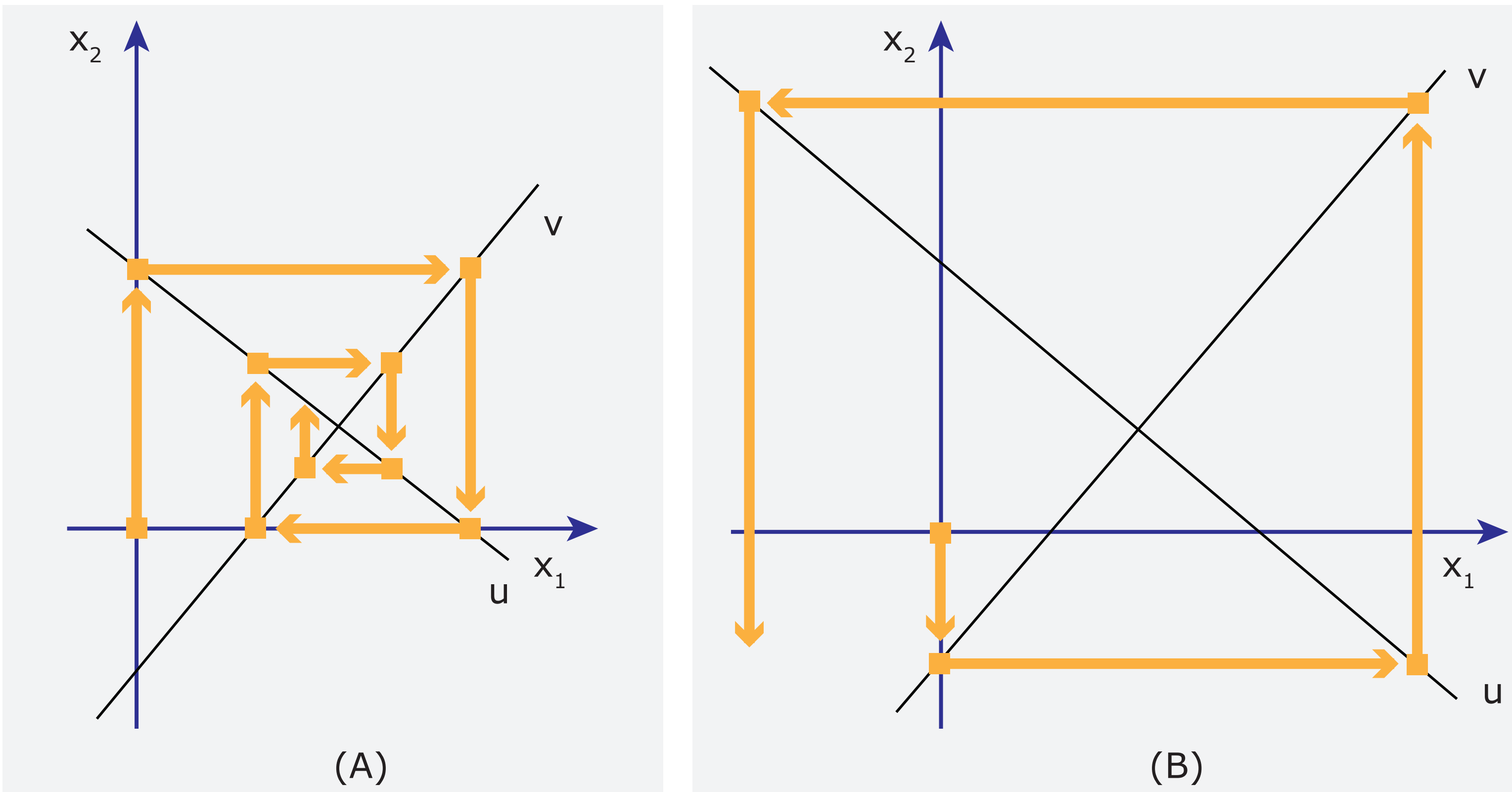


Illustration of (A) convergence and (B) divergence of the Gauss-Seidel method.
Notice that the same functions are plotted in both cases (u:11x1+13x2=286; v:11x1-9x2=99).

Image by MIT OpenCourseWare.

Special Matrices: **Tri-diagonal Systems**

Example "Forced Vibration of a String"

Finite Difference

$$\left.\frac{d^2y}{dx^2}\right|_{x_i} \simeq \frac{y_{i-1} - 2y_i + y_{i+1}}{h^2}$$

Discrete Difference Equations

$$y_{i-1} + \left((kh)^2 - 2\right) y_i + \ y_{i+1} = f(x_i)h^2$$

Differential Equation: $\frac{d^2y}{dx^2} + k^2 y = f(x)$

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

Matrix Form

$$\begin{bmatrix} (kh)^2-2 & 1 & . & . & . & . & 0 \\ 1 & (kh)^2-2 & 1 & & & & . \\ . & & . & . & & & . \\ . & & 1 & (kh)^2-2 & 1 & & . \\ . & & & . & . & & . \\ 0 & . & . & . & . & 1 & (kh)^2-2 \end{bmatrix} \overline{y} = \begin{Bmatrix} f(x_1)h^2 \\ . \\ . \\ f(x_i)h^2 \\ . \\ . \\ f(x_n)h^2 \end{Bmatrix}$$

Tridiagonal Matrix

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)**: If $|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}| \quad \Rightarrow \quad \|\mathbf{B}\|_\infty = \max_{i=1\ldots n} \left(\sum_{j=1}^{n} |b_{ij}|\right) = \max_{i=1\ldots n} \left(\sum_{j=1, j\neq i}^{n} \frac{|a_{ij}|}{|a_{ii}|}\right) < 1$

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);
subplot(2,1,1); p=plot(x,f,'r');set(p,'LineWidth',2);
p=title('Force Distribution');
set(p,'FontSize',14)
```

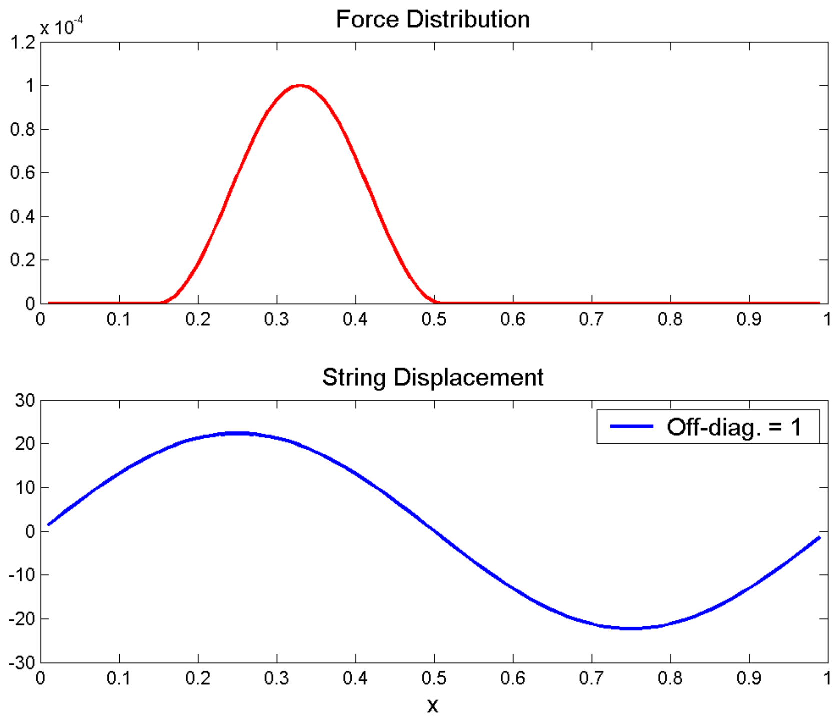
```
% Exact solution
y=inv(a)*f;
subplot(2,1,2); p=plot(x,y,'b');set(p,'LineWidth',2);
p=legend(['Off-diag. = ' num2str(o)]);
set(p,'FontSize',14);
p=title('String Displacement (Exact)');
set(p,'FontSize',14);
p=xlabel('x');
set(p,'FontSize',14);
```

vib_string.m (Part II)

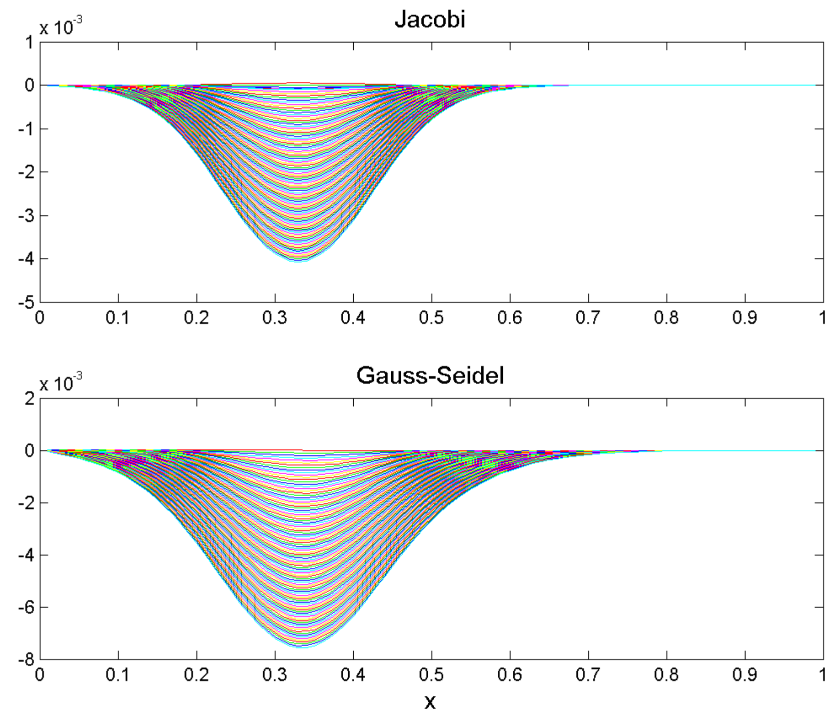
```
%Iterative solution using Jacobi's and Gauss-Seidel's methods
b=-a;
c=zeros(n,1);
for i=1:n
    b(i,i)=0;
    for j=1:n
        b(i,j)=b(i,j)/a(i,i);
        c(i)=f(i)/a(i,i);
    end
end

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

figure(2)
nc=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



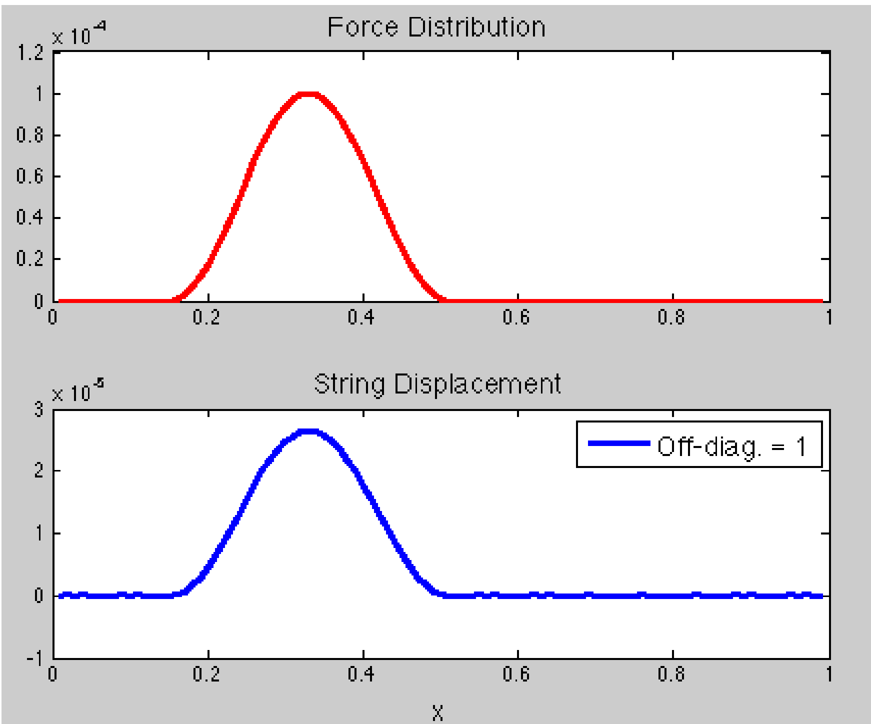
Iterative Solutions



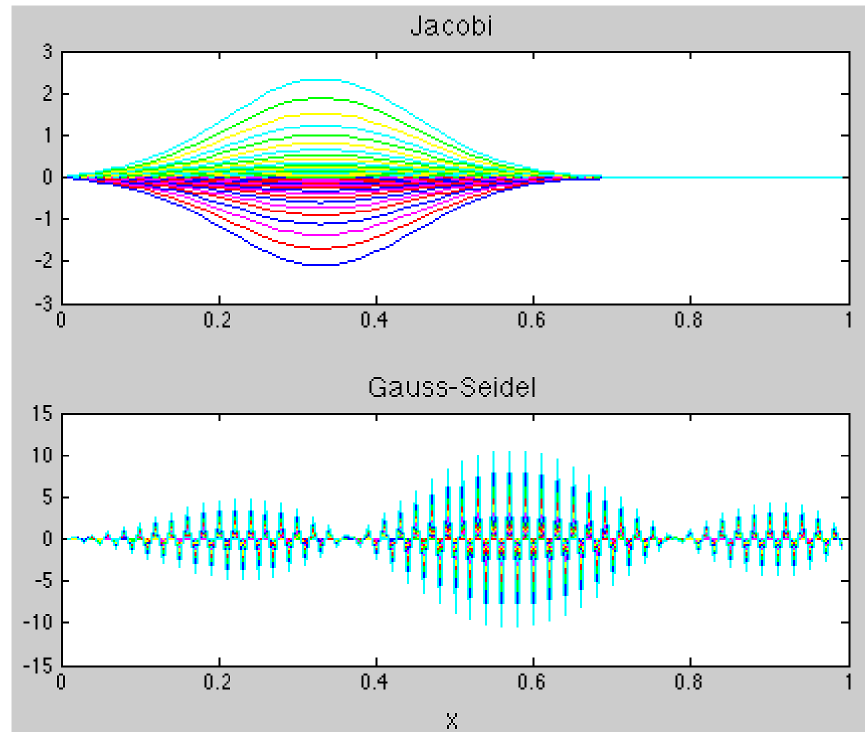
Coefficient Matrix Not Strictly Diagonally Dominant

vib_string.m
o=1.0, , k=2*pi*31, h=.01, kh<2

Exact Solution



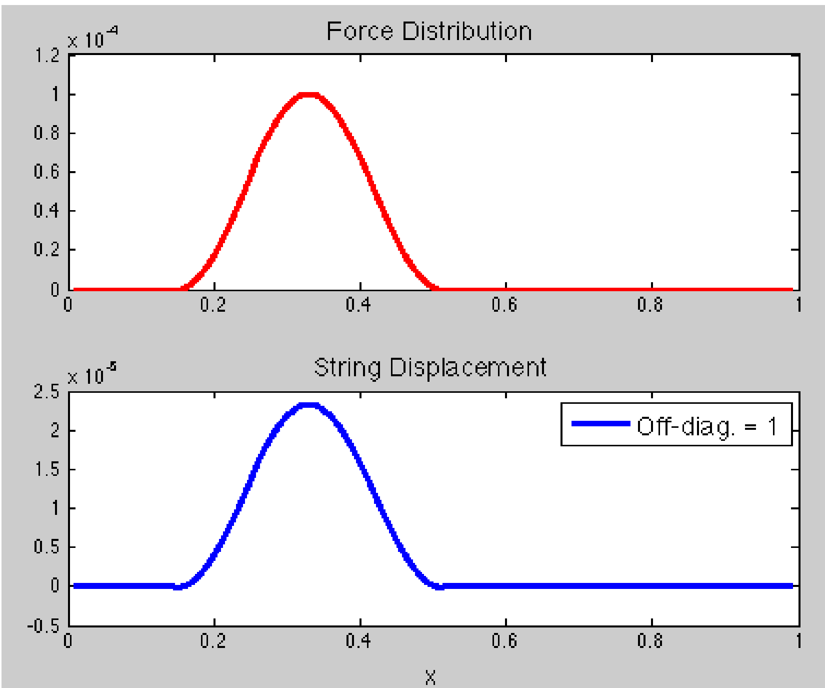
Iterative Solutions



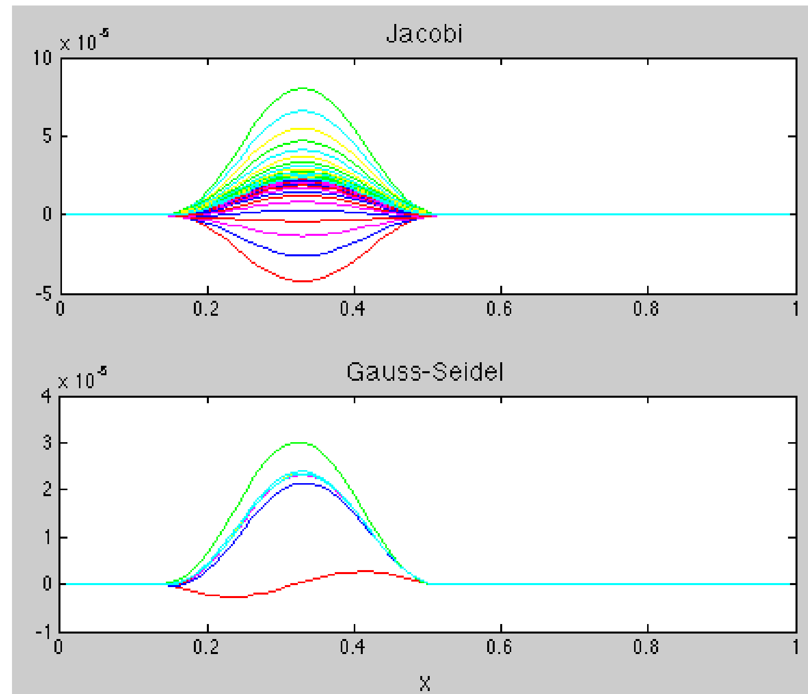
Coefficient Matrix Not Strictly Diagonally Dominant

vib_string.m
o=1.0, k=2*pi*33, h=.01, kh>2

Exact Solution



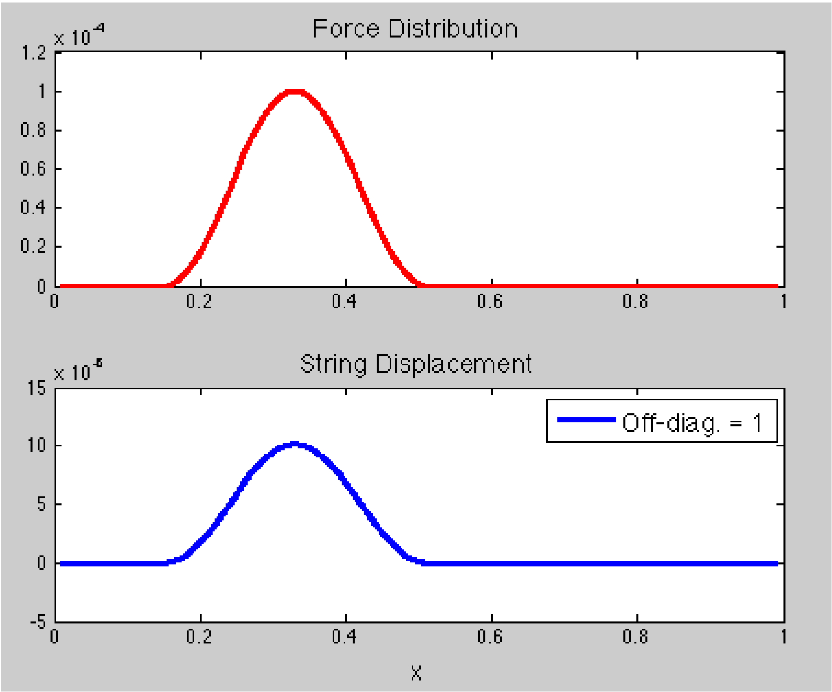
Iterative Solutions



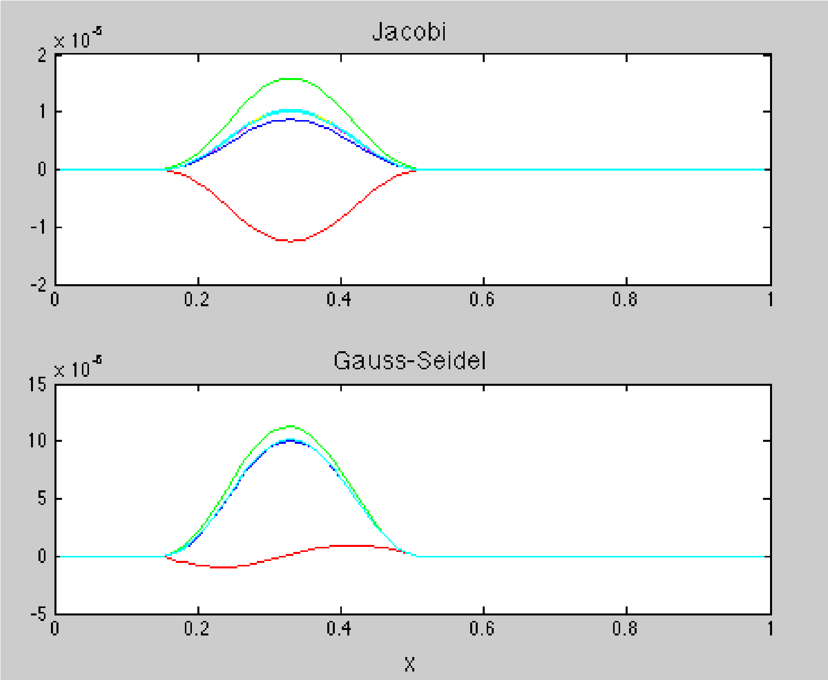
Coefficient Matrix Strictly Diagonally Dominant

vib_string.m
o=1.0, k=2*pi*50, h=.01, kh>2

Exact Solution



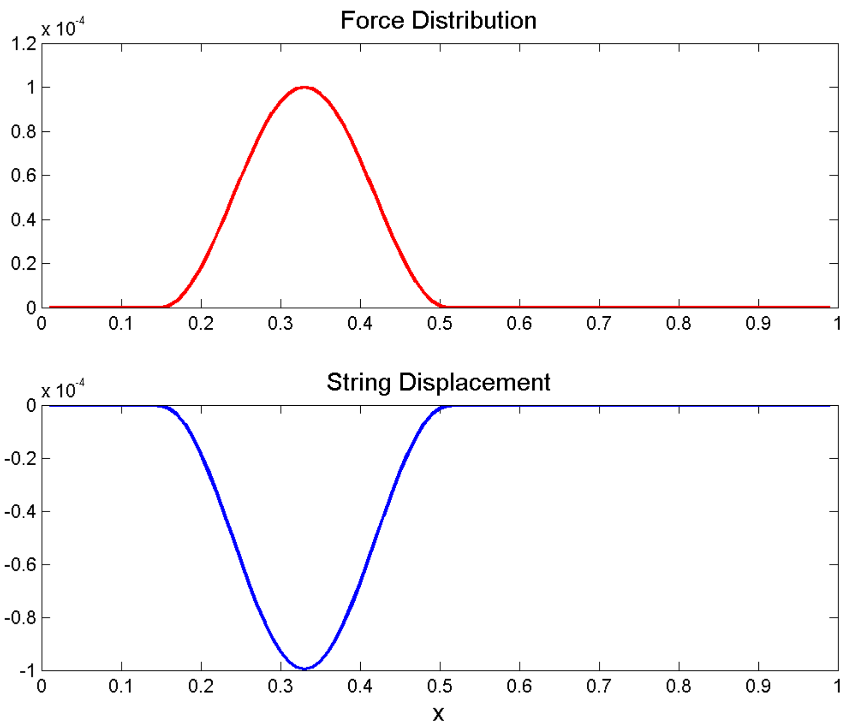
Iterative Solutions



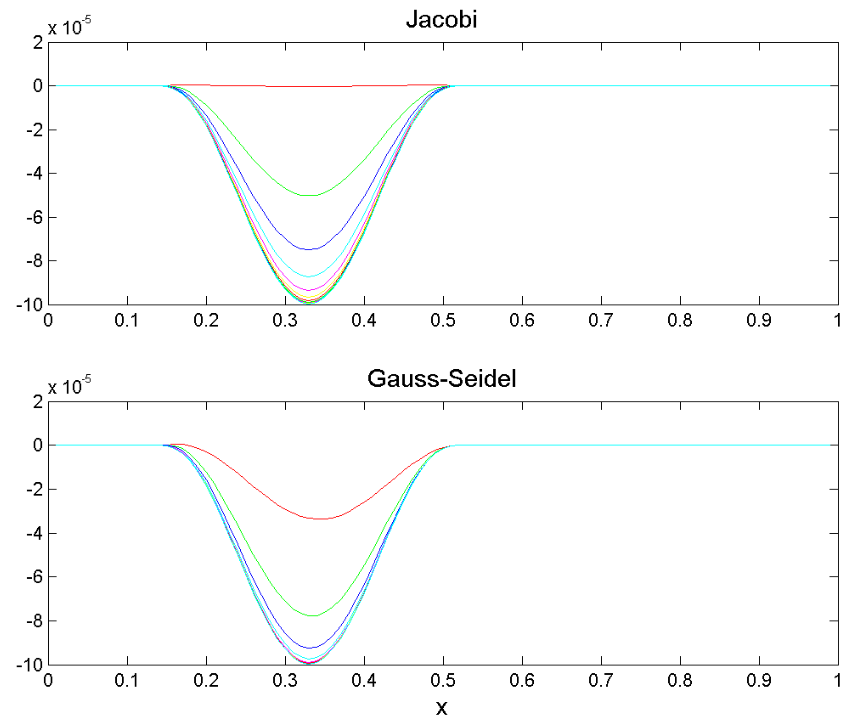
Coefficient Matrix Strictly Diagonally Dominant

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

$$\mathbf{x}_i^{k+1} = \omega \bar{\mathbf{x}}_i^{k+1} + (1-\omega)\mathbf{x}_i^k \text{, } where \quad \bar{\mathbf{x}}_i^{k+1} \ computed\ by\ Gauss-Seidel$$

$$\omega = 1 \Rightarrow \ SOR \equiv \text{Gauss-Seidel}$$

$$1 < \omega < 2 \Rightarrow \text{ Over-relaxation (weight new values more)}$$

$$0 < \omega < 1 \Rightarrow \text{ Under-relaxation}$$

- If "**A**" symmetric and positive definite $\Rightarrow \ converges\ for\ 0 < \omega < 2$
- Matrix format:

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

- Hard to find optimal value of over-relaxation parameter for fast convergence (aim to minimize spectral radius of $\mathbf{B}$) 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

$$Q(x) = \frac{1}{2} x^T A x - x^T b$$

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

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

- Propose step rule

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

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

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

$$r : residual, \;\; r_i = b - Ax_i$$

- Step rule (obtained in lecture)

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

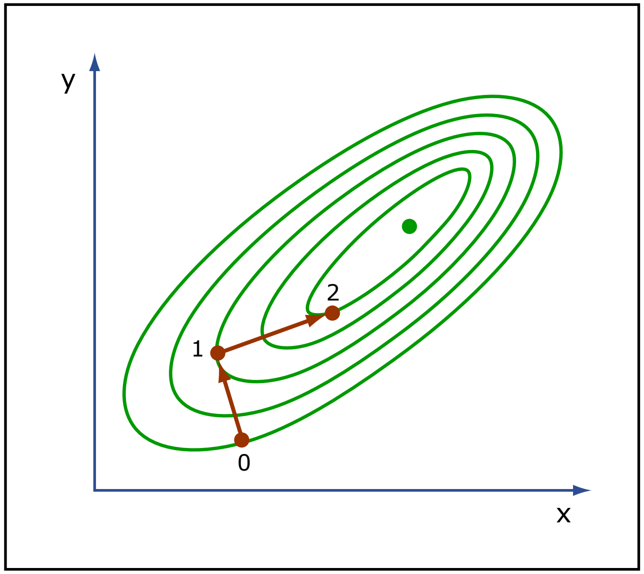


Image by MIT OpenCourseWare.

Graph showing the steepest descent method.

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

Conjugate Gradient Method

- Derivation provided in lecture
- Check CGM_new.m

- Definition: "**A**-conjugate vectors" or "Orthogonality with respect to a matrix (metric)":

  if $\mathbf{A}$ is symmetric & positive definite,

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

- 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:

$\boldsymbol{v}_0 = \boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$

do

$\alpha_i = (\boldsymbol{v}_i^\top \boldsymbol{r}_i)/(\boldsymbol{v}_i^\top \boldsymbol{A}\boldsymbol{v}_i)$	Step length
$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i \boldsymbol{v}_i$	Approximate solution
$\boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \alpha_i \boldsymbol{A}\boldsymbol{v}_i$	New Residual
$\beta_i = -(\boldsymbol{v}_i^\top \boldsymbol{A}\boldsymbol{r}_{i+1})/(\boldsymbol{v}_i^\top \boldsymbol{A}\boldsymbol{v}_i)$	Step length &
$\boldsymbol{v}_{i+1} = \boldsymbol{r}_{i+1} + \beta_i \boldsymbol{v}_i$	new search direction

until a stop criterion holds

Note: $\boldsymbol{A}\, \boldsymbol{v}_i$ = one matrix vector multiply at each iteration

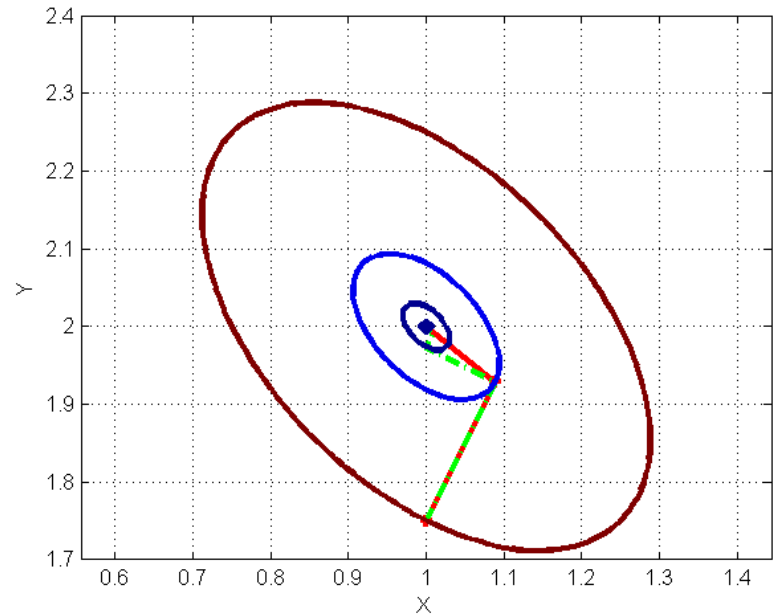


Figure indicates solution obtained using Conjugate gradient method (red) and steepest descent method (green).