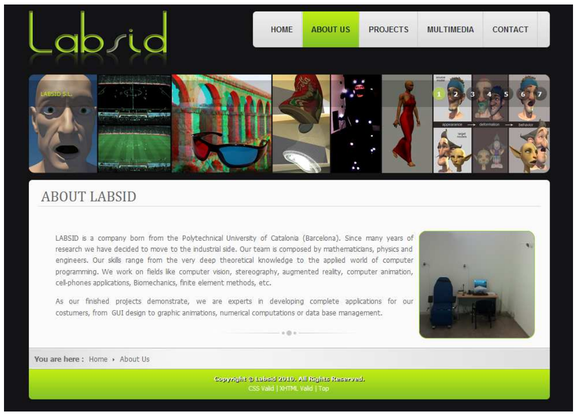
Linear Systems

Dr. Antonio Susin
Polytechnic University of Catalonia

Multimedia and Robotics Summer School 2010

University of Zürich



Some other material:

Introduction to Numerical Methods
Prof. Per-Olof Persson
MIT 18.335J/6.337J

- http://ocw.mit.edu/courses/mathematics/18-335j-introduction-to-numerical-methods-fall-2006/lecture-notes/

INDEX:

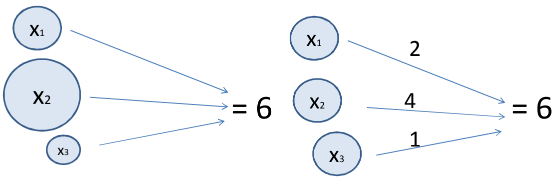
INDEX:

INDEX:

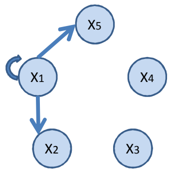
Introduction

Linear Equation

$$2x_1 + 4x_2 + x_3 = 6$$



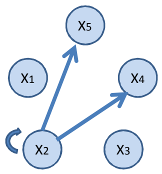
Introduction



$$x_1 + 3\,x_2 + 4\,x_5 = b_1$$

Linear System

Introduction

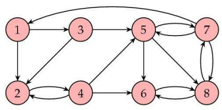


$$x_1 + 3\,x_2 + 4\,x_5 = b_1$$
$$x_2 + 2\,x_4 + 5\,x_5 = b_2$$

Linear System

Google Ranking

- Graph of citation between web pages



- Hyperlink Matrix:
  - entry if it is cited
  - weighted
  - sum by columns =1

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 0 \\ 1/2 & 0 & 1/2 & 1/3 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/3 & 1 & 1/3 & 0 \end{bmatrix}$$

Introduction

Linear System

$$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n = b_1 \\ \vdots \qquad \vdots \qquad \vdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n = b_m \end{cases}$$

a Coefficients
x Unknowns
b Independent terms

Introduction

Matrix Form

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

$$A\,x \;=\; b$$

MATRIX PROPERTIES

- Matrix Algebra:
  - Matrix-Vector product: *b=Ax*

$$b_i = \sum_{j=1}^{n} a_{ij} x_j, \qquad i = 1, \ldots, m$$

  - Linear Mapping $x \mapsto Ax$

$$A(x+y) = Ax + Ay$$
$$A(\alpha x) = \alpha A x$$

MATRIX PROPERTIES

- Columns $a_1, a_2, \ldots, a_n$ of A

$$A = \left[a_1 \middle| a_2 \middle| \cdots \middle| a_n \right]$$

- Alternative view of matrix-vector product

$$b = Ax = \sum_{j=1}^{n} x_j a_j = x_1 \left[a_1 \right] + x_2 \left[a_2 \right] + \cdots + x_n \left[a_n \right]$$

- *b* is a linear combination of the columns of *A*

MATRIX PROPERTIES

- Matrix- matrix product *B=AC*

$$b_{ij} = \sum_{k=1}^{m} a_{ik} c_{kj}$$

- Matrix-vector product for each column of *C*
- Transpose of a matrix A^T : changing rows by columns

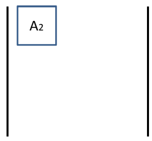
$$(A+B)^T = A^T + B^T$$
$$(AB)^T = B^T A^T$$
$$(A^T)^T = A$$

MATRIX PROPERTIES

- Determinant: $\begin{vmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{vmatrix}$
  - 2x2 dim

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11} \cdot a_{22} - a_{12} \cdot a_{21}$$

  - Principal minors:

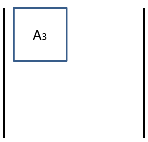


MATRIX PROPERTIES

- Determinant: $\begin{vmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{vmatrix}$
  - 2x2 dim

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11} \cdot a_{22} - a_{12} \cdot a_{21}$$

  - Principal minors:

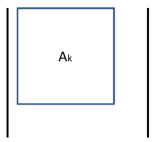


MATRIX PROPERTIES

- Determinant: $\begin{vmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{vmatrix}$
  - 2x2 dim

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11} \cdot a_{22} - a_{12} \cdot a_{21}$$

  - Principal minors:

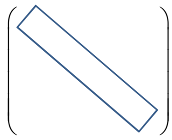


MATRIX PROPERTIES

- Det. Properties: $\det(A \cdot B) = \det(A) \cdot \det(B)$
  $\det(A^T) = \det(A)$
- A is symmetric iff $A^T = A$

MATRIX PROPERTIES

- Det. Properties: $\det(A \cdot B) = \det(A) \cdot \det(B)$
  $\det(A^T) = \det(A)$
- A is symmetric iff $A^T = A$
- Trace: Sum of the diagonal elements



$= \sum_{i=1}^{n} a_{ii}$

MATRIX PROPERTIES

- Rank: number of linear independent columns
  - Maximum order of the nonvanishing determinant extracted from *A*.
  - *A* has *full rank* if rank(*A*)= min(*m*,*n*)
  - column rank = row rank

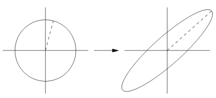
MATRIX PROPERTIES

- Rank: number of linear independent columns
  - Maximum order of the nonvanishing determinant extracted from *A*.
  - *A* has *full rank* if rank(*A*)= min(*m*,*n*)
  - column rank = row rank
- Kernel: all solutions of *Ax=0*

  rank + dim(Kernel) = m

MATRIX PROPERTIES

- Eigenvalues and Eigenvectors: $Av = \lambda v$
  - Directions preserved by *A*
  - Deformation induced by A



- Linear homogeneous system:
  $(A - \lambda Id)v = 0$
- >>eigshow

MATRIX PROPERTIES

- Inverse of a matrix: (for square matrices)
  $A \cdot A^{-1} = Id$
  - If rank(*A*)=n (or det(*A*) ≠ 0), *A* is invertible.
  - If *A* is not invertible is called *singular*

MATRIX PROPERTIES

- <u>Inverse of a matrix:</u> (for square matrices)

$$A \cdot A^{-1} = Id$$

  - If rank(A)=n (or det(A) ≠ 0), A is invertible.
  - If A is not invertible is called *singular*

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

$$(A^T)^{-1} = (A^{-1})^T$$

- A is <u>orthogonal</u> iff $A^T = A^{-1}$

MATRIX NORMS

- <u>Vector norms:</u>

$$\|x\|_1 = \sum_{i=1}^{m} |x_i|$$

$$\|x\|_2 = \left(\sum_{i=1}^{m} |x_i|^2 \right)^{1/2} = \sqrt{x^* x}$$

$$\|x\|_\infty = \max_{1 \le i \le m} |x_i|$$

$$\|x\|_W = \|Wx\|_2 = \left(\sum_{i=1}^{m} |w_i x_i|^2 \right)^{1/2}$$

MATRIX NORMS

- <u>Induced matrix norms:</u> $\| A \| = \sup \frac{\| Ax \|}{\| x \|}$

$$\|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{m} |a_{ij}|$$ "maximum column sum"

$$\|A\|_\infty = \max_{1 \le i \le m} \sum_{j=1}^{n} |a_{ij}|$$ "maximum row sum"

$$\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2}$$ The Frobenius norm

$$\|A\|_2 = \sqrt{\lambda_{\max}(A^* A)}$$ More later

MATRIX NORMS

Quantity	MATLAB Syntax
$\|x\|_1$	`sum(abs(x))` or `norm(x,1)`
$\|x\|_2$	`sqrt(x'*x)` or `norm(x)`
$\|x\|_p$	`sum(abs(x).^p).^(1/p)` or `norm(x,p)`
$\|x\|_\infty$	`max(abs(x))` or `norm(x,inf)`
$\|A\|_1$	`max(sum(abs(A),1))` or `norm(A,1)`
$\|A\|_2$	`norm(A)`
$\|A\|_\infty$	`max(sum(abs(A),2))` or `norm(A,inf)`
$\|A\|_F$	`sqrt(A(:)'*A(:))` or `norm(A,'fro')`

Special Matrices Types

- Diagonal and block diagonal:
  - Nonzero entries only at the diagonal

$$\begin{pmatrix} a_{11} & & \\ & \ddots & \\ & & a_{mn} \end{pmatrix} \qquad \begin{pmatrix} A_{11} & & \\ & \ddots & \\ & & A_{mn} \end{pmatrix}$$

Special Matrices Types

- Triangular Matrices:
  - <u>Lower triangular</u>

$$\begin{pmatrix} l_{11} & & & & \\ l_{21} & l_{22} & & & \\ l_{31} & l_{32} & l_{33} & & \\ \vdots & \vdots & \vdots & \ddots & \\ l_{m1} & \cdots & \cdots & l_{mn-1} & l_{mn} \end{pmatrix}$$

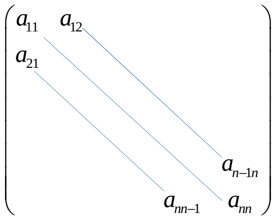
Special Matrices Types

- Triangular Matrices:
  - Upper triangular

$$\begin{pmatrix} u_{11} & u_{12} & \cdots & \cdots & u_{1n} \\ & u_{22} & u_{23} & \cdots & u_{2n} \\ & & u_{33} & \cdots & u_{3n} \\ & & & \ddots & \vdots \\ & & & & u_{nn} \end{pmatrix}$$

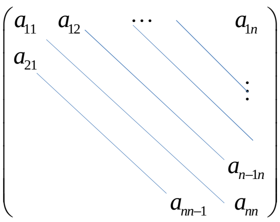
Special Matrices Types

- Tridiagonal Matrices:



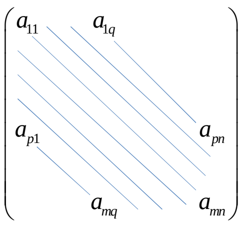
Special Matrices Types

- Hessemberg Matrices:



Special Matrices Types

- Banded Matrices:

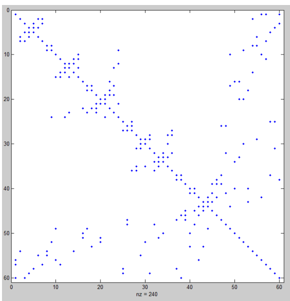


Special Matrices Types

- Sparse Matrix

Mainly zero elements

- Special data structures
  - Compression
  - Efficiency



- >>sparse1

SOLVING LINEAR SYSTEMS

- DIRECT METHODS:
  - Solution is obtained after a finite number of steps
    - Gaussian Elimination, LU, Cholesky, QR, SVD
- ITERATIVE METHODS:
  - Solution is obtained as successive approaches
    - Jacobi, Gauss-Seidel, Conjugate Gradient

DIRECT METHODS

- Naïf approach: $Ax = b \longrightarrow x = A^{-1}b$

DIRECT METHODS

- Naïf approach: $Ax = b \longrightarrow x = A^{-1}b$
- Easy solutions
  - If A is Diagonal: $x = D^{-1}b$
  - If A is Orthogonal: $x = A^{T}b$

DIRECT METHODS

- Naïf approach: $Ax = b \longrightarrow x = A^{-1}b$
- Easy solutions
  - If A is Diagonal: $x = D^{-1}b$
  - If A is Orthogonal: $x = A^{T}b$
- Already Solved
  - If A is Triangular:

$$\begin{pmatrix} u_{11} & u_{12} & \cdots & \cdots & u_{1n} \\ & u_{22} & u_{23} & \cdots & u_{2n} \\ & & u_{33} & \cdots & u_{3n} \\ & & & \ddots & \vdots \\ & & & & u_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

DIRECT METHODS

- Gaussian Elimination-LU
  - Inverse matrix
  - Determinant
- Cholesky
- QR
- SVD

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

$$Ax = b$$

Gaussian Elimination-LU

- Transform the original matrix to an easy form:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & \cdots & a_{nn} \end{pmatrix} \Longrightarrow \begin{pmatrix} u_{11} & u_{12} & \cdots & \cdots & u_{1n} \\ & u_{22} & u_{23} & \cdots & u_{2n} \\ & & u_{33} & \cdots & u_{3n} \\ & & & \ddots & \vdots \\ & & & & u_{nn} \end{pmatrix}$$

Gaussian Elimination-LU

- Using a multiplier: $l_{21} = \frac{a_{21}}{a_{11}}$

PIVOT

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ 0 & a^{1}_{22} & a^{1}_{23} & \cdots & a^{1}_{2n} \\ \vdots & \vdots & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & \cdots & a_{nn} \end{pmatrix}$$

Gaussian Elimination-LU

- Using now multiplier: $l_{31} = \frac{a_{31}}{a_{11}}$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & \vdots & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ 0 & a_{22}^1 & a_{23}^1 & \cdots & a_{2n}^1 \\ 0 & \vdots & a_{33}^1 & \cdots & a_{3n}^1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & \cdots & a_{nn} \end{pmatrix}$$

Gaussian Elimination-LU

- Using now multiplier: $l_{n1} = \frac{a_{n1}}{a_{11}}$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ 0 & a_{22}^1 & a_{23}^1 & \cdots & a_{2n}^1 \\ 0 & \vdots & a_{33}^1 & \cdots & a_{3n}^1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & a_{nn} \end{pmatrix}$$

Gaussian Elimination-LU

- Finally: $l_{nn-1} = \frac{a_{nn-1}^{n-1}}{a_{n-1n-1}^{n-1}}$

$$U = \begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ 0 & a_{22}^1 & a_{23}^1 & \cdots & a_{2n}^1 \\ 0 & 0 & a_{33}^1 & \cdots & a_{3n}^1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & a_{nn} \end{pmatrix}$$

Upper Triangular

$$L = \begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ l_{n1} & \cdots & \cdots & l_{nn-1} & 1 \end{pmatrix}$$

Lower Triangular

Gaussian Elimination-LU

- System solution from LU decomposition:

$$A = L \cdot U$$

$$Ax = b \longrightarrow LUx = b$$

1. $Ux = y$
2. $Ly = b$

- Real Live: Permutations
  – Troubles when pivot ≈ 0

Gaussian Elimination-LU

- Row permutations:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ 0 & a_{22}^1 & a_{23}^1 & \cdots & a_{2n}^1 \\ 0 & \vdots & a_{33}^1 & \cdots & a_{3n}^1 \\ \vdots & a_{k2}^1 & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & a_{nn} \end{pmatrix}$$

Gaussian Elimination-LU

- Row permutations as matrix product

$$P_{ij}^k = \begin{pmatrix} 1 & 0 & \cdots & & 0 & \cdots & 0 \\ 0 & 1 & & & & & \\ & 0 & 0 & & 1 & & \\ \vdots & & & \ddots & \vdots & 0 & \\ & \vdots & 1 & 0 & 0 & \cdots & 0 \\ & & \vdots & & \vdots & \ddots & \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix} \longrightarrow P = P_{ij}^1 P_{ij}^2 \ldots P_{ij}^{n-1}$$

Gaussian Elimination-LU

- Row permutations as matrix product

$$P_{ij}^{k} = \begin{pmatrix} 1 & 0 & \cdots & & 0 & \cdots & 0 \\ 0 & 1 & & & & & \\ & 0 & 0 & & 1 & & \\ \vdots & & & \ddots & \vdots & 0 & \\ & \vdots & 1 & 0 & 0 & \cdots & 0 \\ & & \vdots & & \vdots & \ddots & \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix} \longrightarrow P = P_{ij}^{1} P_{ij}^{2} \ldots P_{ij}^{n-1}$$

- Finally:

$$P \cdot A = LU$$
$$Ax = b \longrightarrow LUx = P^T b$$
1. $Ux = y$
2. $Ly = P^T b$

Gaussian Elimination-LU

Algorithm: Gaussian Elimination (no pivoting)

$U = A, L = I$
for $k = 1$ **to** $m - 1$
  for $j = k + 1$ **to** m
    $\ell_{jk} = u_{jk} / u_{kk}$
    $u_{j,k:m} = u_{j,k:m} - \ell_{jk} u_{k,k:m}$

- Operation count

$$\sum_{k=1}^{m} 2(m-k)(m-k) \sim 2 \sum_{k=1}^{m} k^2 \sim 2m^3/3$$

Gaussian Elimination-LU

Algorithm: Gaussian Elimination (partial pivoting)

$U = A, L = I, P = I$
for $k = 1$ **to** $m - 1$
  Select $i \geq k$ to maximize $|u_{ik}|$
  $u_{k,k:m} \leftrightarrow u_{i,k:m}$ (interchange two rows)
  $\ell_{k,1:k-1} \leftrightarrow \ell_{i,1:k-1}$
  $p_{k,:} \leftrightarrow p_{i,:}$
  for $j = k + 1$ **to** m
    $\ell_{jk} = u_{jk} / u_{kk}$
    $u_{j,k:m} = u_{j,k:m} - \ell_{jk} u_{k,k:m}$

MATLAB-LU

```
>> A=rand(6)

A =

   0.8147   0.2785   0.9572   0.7922   0.6787   0.7060
   0.9058   0.5469   0.4854   0.9595   0.7577   0.0318
   0.1270   0.9575   0.8003   0.6557   0.7431   0.2769
   0.9134   0.9649   0.1419   0.0357   0.3922   0.0462
   0.6324   0.1576   0.4218   0.8491   0.6555   0.0971
   0.0975   0.9706   0.9157   0.9340   0.1712   0.8235

>> [L,U,P]=lu(A)

L =

   1.0000        0        0        0        0        0
   0.1068   1.0000        0        0        0        0
   0.8920  -0.6711   1.0000        0        0        0
   0.9917  -0.4726   0.5368   1.0000        0        0
   0.1390   0.9491  -0.0517  -0.2586   1.0000        0
   0.6923  -0.5883   0.5947   0.8836   0.0470   1.0000

U =

   0.9134   0.9649   0.1419   0.0357   0.3922   0.0462
        0   0.8676   0.9006   0.9302   0.1293   0.8185
        0        0   1.4349   1.3846   0.4156   1.2141
        0        0        0   0.6204   0.2068  -0.2789
        0        0        0        0   0.6408  -0.5157
        0        0        0        0        0   0.0953

P =

   0   0   0   1   0   0
   0   0   0   0   0   1
   1   0   0   0   0   0
   0   1   0   0   0   0
   0   0   1   0   0   0
   0   0   0   0   1   0

>>
```


MATLAB-LU

```
b =

     1
     2
     3
     4
     5
     6
```

```
>> y=L\P*b;
x=U\y;
A*x-b

ans =

  1.0e-014 *
   -0.7105
   -0.7772
         0
         0
    0.3553
    0.7105
```


Application: Inverse of a Matrix

- Inverse as n-linear systems:

$$\underbrace{\begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & \cdots & a_{nn} \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x_{11} & x_{12} & \cdots & \cdots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2n} \\ \vdots & \vdots & x_{33} & \cdots & x_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & \cdots & \cdots & \cdots & x_{nn} \end{pmatrix}}_{A^{-1}} = \underbrace{\begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & 0 & 1 & \cdots & \vdots \\ \vdots & \vdots & 0 & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}}_{Id}$$

Application: Inverse of a Matrix

- Inverse as n-linear systems:

$$\underbrace{\begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & \cdots & a_{nn} \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x_{11} & x_{12} & \cdots & \cdots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2n} \\ \vdots & \vdots & x_{33} & \cdots & x_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & \cdots & \cdots & \cdots & x_{nn} \end{pmatrix}}_{A^{-1}} = \underbrace{\begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & 0 & 1 & \cdots & \vdots \\ \vdots & \vdots & 0 & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}}_{Id}$$

Application: Inverse of a Matrix

- The cost is NOT n times a linear system:
  - LU decomposition $A = L \cdot U$ ONLY ONCE!!
  - n times: solution of triangular systems

$$\begin{matrix} Ly = e_i \\ Ux = y \end{matrix} \quad where\ e_i = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}$$

Application: Determinant of a Matrix

- Determinant from LU decomposition:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ l_{n1} & \cdots & \cdots & l_{nn-1} & 1 \end{pmatrix} \cdot \begin{pmatrix} u_{11} & u_{12} & \cdots & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ \vdots & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & 0 & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & u_{nn} \end{pmatrix}$$

From det(AB)=det(A)·det(B)

$$\det(A) = \det(L)\cdot\det(U)/\det(P) = (-1)^p \prod_{1}^{n} u_{ii}$$

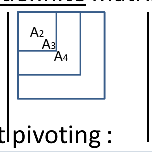
DIRECT METHODS

- Gaussian Elimination-LU
  - Inverse matrix
  - Determinant
- Cholesky
- QR
- SVD

$$Ax = b$$

LU: Cholesky Decomposition

- If A is a symmetric positive definite matrix
  - (i) $A^T = A$
  - (ii) $x^T Ax > 0$
  - (all principal minors >= 0)



- LU can be obtained without pivoting :

$$A = R^T \cdot R$$

- Cost

$$\sum_{k=1}^{m} \sum_{j=k+1}^{m} 2(m-j) \sim \sum_{k=1}^{m} k^2 \sim \frac{m^3}{3}$$

Algorithm: Cholesky Factorization

$R = A$
for $k = 1$ **to** m
  for $j = k+1$ **to** m
    $R_{j,j:m} = R_{j,j:m} - R_{k,j:m}\overline{R}_{kj}/R_{kk}$
  $R_{k,k:m} = R_{k,k:m}/\sqrt{R_{kk}}$

DIRECT METHODS

- Gaussian Elimination-LU
  - Inverse matrix
  - Determinant
- Cholesky
- QR
- SVD

$$Ax = b$$

QR Decomposition

- Matrix A as a product of an Orthogonal matrix and an upper triangular one.

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & \cdots & a_{nn} \end{pmatrix} = \underbrace{\begin{pmatrix} q_{11} & q_{12} & \cdots & \cdots & q_{1n} \\ q_{21} & q_{22} & q_{23} & \cdots & q_{2n} \\ \vdots & \vdots & q_{33} & \cdots & q_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ q_{n1} & \cdots & \cdots & \cdots & q_{nn} \end{pmatrix}}_{Q^T = Q^{-1}} \cdot \underbrace{\begin{pmatrix} r_{11} & r_{12} & \cdots & \cdots & r_{1n} \\ 0 & r_{22} & r_{23} & \cdots & r_{2n} \\ \vdots & 0 & r_{33} & \cdots & r_{3n} \\ \vdots & \vdots & 0 & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & r_{nn} \end{pmatrix}}_{R}$$

Matrix Decomposition-QR

- Solution "only" a triangular system

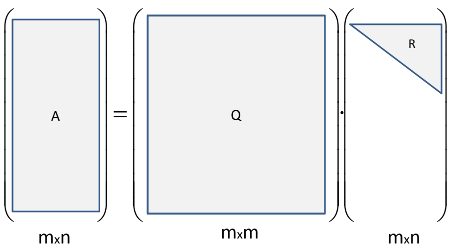
$$A = Q \cdot R$$

$$Ax = b \longrightarrow QRx = b$$

$$Rx = Q^T b$$

Matrix Decomposition-QR

- Appropriate for non-square systems (overdetermined systems)



Matrix Decomposition-QR

- Householder Matrices: $P = Id - 2ww^T$ with $\| w \| = 1$
  - Orthogonal matrix
  - *P* is symmetric
  - No storage needed $Px = x - 2(w^T x)w$
  - *y=Px* is obtained by the reflection of *x* by a plane whose normal vector is *w.*
  - Norm is preserved $\| x \| = \| y \|$

Matrix Decomposition-QR

- Householder Matrices:
  - *Idea:* Change a general vector as $Px = ke_1$

$$P_1 \cdot \begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} k_1 & a_{12}^1 & a_{13}^1 & \cdots & a_{1n}^1 \\ 0 & a_{22}^1 & a_{23}^1 & \cdots & a_{2n}^1 \\ \vdots & \vdots & \vdots & \cdots & a_{3n}^1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^1 & a_{n3}^1 & \cdots & a_{nn}^1 \end{pmatrix}$$

  - Achieve it by using the first column $x = a_1$

$$w = \frac{x - k_1 e_1}{\| x - k_1 e_1 \|} \quad \text{with } k_1 = x^T x$$

Matrix Decomposition-QR

- Householder Matrices:

$$P_2 P_1 \cdot \begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} k_1 & a_{12}^1 & a_{13}^1 & \cdots & a_{1n}^1 \\ 0 & k_2 & a_{23}^2 & \cdots & a_{2n}^2 \\ \vdots & 0 & \vdots & \cdots & a_{3n}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & a_{n3}^2 & \cdots & a_{nn}^2 \end{pmatrix}$$

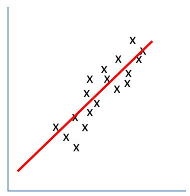
  - The orthogonal matrix: $P_{n-1} \ldots P_1 \cdot A = R$

$$Q = (P_{n-1} \ldots P_1)^T$$

Matrix Decomposition-QR

- Application: Regression line (least square sol.)

$r: \quad ax + by + 1 = 0$



$$ax_i + by_i + 1 = 0$$

$$\begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ \vdots & \vdots \\ \vdots & \vdots \\ x_n & y_n \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} -1 \\ \vdots \\ \\ \vdots \\ -1 \end{pmatrix}$$

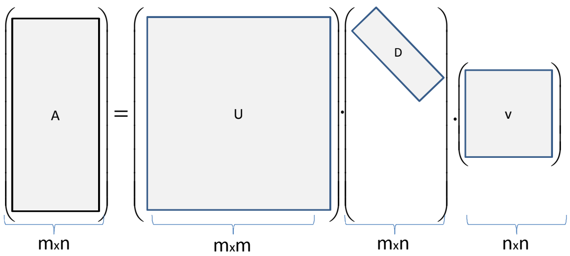
DIRECT METHODS

- Gaussian Elimination-LU
  - Inverse matrix
  - Determinant
- Cholesky
- QR
- SVD

$$Ax = b$$

Singular Value Decomposition-SVD

- $A = UDV^T$, with *U, V* orthogonals and D diagonal



Singular Value Decomposition-SVD

- Solution from SVD:

$$A = UDV^T$$
$$Ax = b \longrightarrow UDV^T x = b$$
$$DV^T x = U^T b$$
$$V^T x = D^{-1} U^T b$$
$$\boxed{x = VD^{-1} U^T b}$$

- >>svd

Singular Value Decomposition-SVD

- Singular values: $D = diag(\sigma_1, \sigma_2, \ldots\ldots, \sigma_n)$

  $\sigma_1 > \sigma_2 > \ldots\ldots > \sigma_n$
- rank(*A*) is the number of $\sigma_i \neq 0$
- Norm $\|A\|_2 = \sigma_1$
- Decomposition $A = \sum_{i=1}^{n} \sigma_i u_i \cdot v_i^T$

$$A = \sigma_1 \begin{pmatrix} u_1 \end{pmatrix} \begin{pmatrix} v_1 \end{pmatrix}^T + \sigma_2 \begin{pmatrix} u_2 \end{pmatrix} \begin{pmatrix} v_2 \end{pmatrix}^T + \ldots + \sigma_n \begin{pmatrix} u_n \end{pmatrix} \begin{pmatrix} v_n \end{pmatrix}^T$$

Singular Value Decomposition-SVD

- For SVD we have $Av = \sigma u$
- In general singular values $eig(A^T A) = \sigma^2$
- If *A* is symmetric $\sigma_i = |\lambda_i|$ and eigenvectors are the columns of V
- For square *A*

$$|\det(A)| = \sum_{i=1}^{n} \sigma_i$$

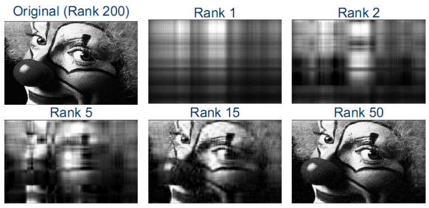
Singular Value Decomposition-SVD

- Stability: $(A+\delta A)(x+\delta x)=b+\delta b$
- Condition number: $\kappa = \sup \dfrac{\|\delta x\|/\|x\|}{\|\delta A\|/\|A\|}$
  - Better if $\kappa \approx 1$
  - In general $\kappa \le \|A\|\|A^{-1}\|$
  - Using Norm2 $\kappa = \dfrac{\sigma_1}{\sigma_n}$
  - >> Hilbert Matrix

Singular Value Decomposition-SVD

- Low rank approximation: $A_\kappa = \sum_{i=1}^{k} \sigma_i u_i \cdot v_i^T$

  $\|A - A_\kappa\|_2 = \sigma_{k+1}$
- Compression: k×(m+n) instead of m×n



Backslash in MATLAB

`x=A\b` for dense `A` performs these steps (stopping when successful):

1. If `A` is upper or lower triangular, solve by back/forward substitution
2. If `A` is permutation of triangular matrix, solve by permuted back substitution (useful for `[L,U]=lu(A)` since `L` is permuted)
3. If `A` is symmetric/hermitian
   - Check if all diagonal elements are positive
   - Try Cholesky, if successful solve by back substitutions
4. If `A` is Hessenberg (upper triangular plus one subdiagonal), reduce to upper triangular then solve by back substitution
5. If `A` is square, factorize $PA = LU$ and solve by back substitutions
6. If `A` is not square, run Householder QR, solve least squares problem

GOOGLE RANK

- H has entries non-negative and the sum of entries for each column is 1 are named stochastic matrices.
- The final order correspond to vector

$$Hw = w$$

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 0 \\ 1/2 & 0 & 1/2 & 1/3 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/3 & 1 & 1/3 & 0 \end{bmatrix}$$

(eigenvector associated to eigenvalue=1)

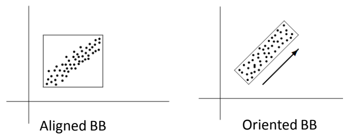
http://www.ams.org/samplings/feature-column/fcarc-pagerank

GOOGLE RANK

$$Hw = w \longrightarrow w = \begin{bmatrix} 0.0600 \\ 0.0675 \\ 0.0300 \\ 0.0675 \\ 0.0975 \\ 0.2025 \\ 0.1800 \\ 0.2950 \end{bmatrix}$$

Oriented Bounding Box (OBB)

- Given a set of 3D points, how can we compute their associated Oriented Bounding Box.
  - Compute the centroid
  - Compute the Covariance Matrix: C
  - Eigenvectors of C
  - >>OBB

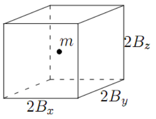


Aligned BB Oriented BB

Oriented Bounding Box (OBB)

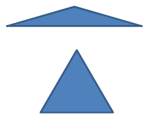
- Centroid: $m_x = \frac{1}{N}\sum_{i=1}^{N} x_i \quad m_y = \frac{1}{N}\sum_{i=1}^{N} y_i \quad m_z = \frac{1}{N}\sum_{i=1}^{N} z_i$
- Local coordinates: $(\tilde{x}_i, \tilde{y}_i, \tilde{z}_i) = (x_i - m_x, y_i - m_y, z_i - m_z)$
- Covariance Matrix:
  - C is symmetric
- Principal Directions
  - Eigenvectors

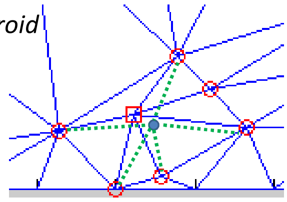
$$C = \frac{1}{N}\begin{pmatrix} \sum_{i=1}^{N} \tilde{x}_i^2 & \sum_{i=1}^{N} \tilde{x}_i\tilde{y}_i & \sum_{i=1}^{N} \tilde{x}_i\tilde{z}_i \\ \sum_{i=1}^{N} \tilde{x}_i\tilde{y}_i & \sum_{i=1}^{N} \tilde{y}_i^2 & \sum_{i=1}^{N} \tilde{y}_i\tilde{z}_i \\ \sum_{i=1}^{N} \tilde{x}_i\tilde{z}_i & \sum_{i=1}^{N} \tilde{y}_i\tilde{z}_i & \sum_{i=1}^{N} \tilde{z}_i^2 \end{pmatrix}$$



Regularization of a Mesh

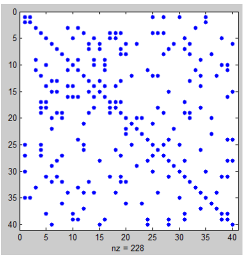
- Numerical quality of a mesh depends on non-degenerate triangles
- Move to the *centroid*
- Equilateral 'best'

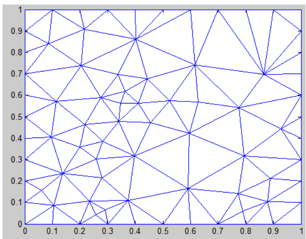




Regularization of a Mesh

- Matrix connectivity

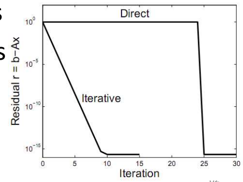




ITERATIVE METHODS

- Compute successive approximations of the solution $x^{(1)}, x^{(2)}, \ldots\ldots, x^{(n)}$ that *"hopefully"* converges to the real solution
- Appropriate for large systems (n>1000)
- Faster than direct methods
- Less memory requirements
- Handle special structures

(sparsity)



ITERATIVE METHODS

- <u>Stationary Methods</u>: Finds a splitting of the matrix $A = M + K$, with *M* invertible

  and iterate

$$Ax = b$$
$$(M+K)x = b$$
$$Mx = -Kx + b$$
$$\boxed{x^{(k+1)} = M^{-1}(-Kx^{(k)} + b) = Rx^{(k)} + c}$$

$x^{(k+1)}$ converges iff $\rho(R) < 1$

– Jacobi, Gauss-Seidel, Successive Overrelaxation (SOR)

ITERATIVE METHODS

- <u>Krylov subspace methods</u>: use only multiplication by *A* (or A^T) and find solutions in the Krylov subspace generated by

$$b, Ab, A^2b, A^3b, \ldots\ldots, A^{k-1}b$$

  – Conjugate Gradient (CG)
  – Generalized Minimal Residual (GMRES)

STATIONARY METHODS

- Jacobi Method:
  - Idea: to solve ith-unknown for the ith-equation

$$\left.\begin{aligned}10x_1+x_2+x_3&=12\\x_1+10x_2+x_3&=12\\x_1+x_2+10x_3&=12\end{aligned}\right\}\Longrightarrow\begin{aligned}x_1&=\frac{1}{10}(-x_2-x_3+12)\\x_2&=\frac{1}{10}(-x_1-x_3+12)\\x_3&=\frac{1}{10}(-x_1-x_2+12)\end{aligned}$$

STATIONARY METHODS

- Jacobi Method:
  - Idea: to solve *ith-unknown* for the *ith-equation*

$$\left.\begin{aligned}10x_1+x_2+x_3&=12\\x_1+10x_2+x_3&=12\\x_1+x_2+10x_3&=12\end{aligned}\right\}\Longrightarrow\begin{aligned}x_1^{(k+1)}&=\frac{1}{10}(-x_2^{(k)}-x_3^{(k)}+12)\\x_2^{(k+1)}&=\frac{1}{10}(-x_1^{(k)}-x_3^{(k)}+12)\\x_3^{(k+1)}&=\frac{1}{10}(-x_1^{(k)}-x_2^{(k)}+12)\end{aligned}$$

STATIONARY METHODS

- Jacobi: Split the original matrix $A=M+K$

$$\begin{pmatrix}a_{11}&a_{12}&\cdots&\cdots&a_{1n}\\a_{21}&a_{22}&a_{23}&\cdots&a_{2n}\\\vdots&\vdots&a_{33}&\cdots&a_{3n}\\\vdots&\vdots&\vdots&\ddots&\vdots\\a_{n1}&\cdots&\cdots&\cdots&a_{nn}\end{pmatrix}=\begin{pmatrix}0&0&\cdots&\cdots&0\\a_{21}&0&0&&\vdots\\a_{31}&a_{32}&0&\ddots&\vdots\\\vdots&\vdots&\vdots&\ddots&0\\a_{n1}&\cdots&\cdots&a_{nn-1}&0\end{pmatrix}+\begin{pmatrix}a_{11}&&&\\&a_{22}&&\\&&\ddots&\\&&&\ddots&\\&&&&a_{nn}\end{pmatrix}+\begin{pmatrix}0&a_{12}&\cdots&\cdots&a_{1n}\\0&0&a_{23}&\cdots&a_{2n}\\\vdots&0&0&\cdots&a_{3n}\\\vdots&\vdots&0&\ddots&\vdots\\0&\cdots&\cdots&0&0\end{pmatrix}$$

$$A=A_L+A_D+A_U$$

$$M=A_D,\quad K=A_L+A_U$$

$$x^{(k+1)}=M^{-1}(-Kx^{(k)}+b)=$$

$$=A_D^{-1}(-(A_L+A_U)x^{(k)}+b)$$

STATIONARY METHODS

- Jacobi: Split the original matrix $A=M+K$

$$\begin{pmatrix}a_{11}&a_{12}&\cdots&\cdots&a_{1n}\\a_{21}&a_{22}&a_{23}&\cdots&a_{2n}\\\vdots&\vdots&a_{33}&\cdots&a_{3n}\\\vdots&\vdots&\vdots&\ddots&\vdots\\a_{n1}&\cdots&\cdots&\cdots&a_{nn}\end{pmatrix}=\begin{pmatrix}0&0&\cdots&\cdots&0\\a_{21}&0&0&&\vdots\\a_{31}&a_{32}&0&\ddots&\vdots\\\vdots&\vdots&\vdots&\ddots&0\\a_{n1}&\cdots&\cdots&a_{nn-1}&0\end{pmatrix}+\begin{pmatrix}a_{11}&&&\\&a_{22}&&\\&&\ddots&\\&&&\ddots&\\&&&&a_{nn}\end{pmatrix}+\begin{pmatrix}0&a_{12}&\cdots&\cdots&a_{1n}\\0&0&a_{23}&\cdots&a_{2n}\\\vdots&0&0&\cdots&a_{3n}\\\vdots&\vdots&0&\ddots&\vdots\\0&\cdots&\cdots&0&0\end{pmatrix}$$

```
>>AL=tril(A,-1)
>>AU=triu(A,1)
>>AD=diag(diag(A))
>>Xkp1=inv(AD)*(-(AL+AU)*Xk+b)
```


STATIONARY METHODS

- Gauss-Seidel Method:
  - Idea: to solve *ith-unknown* for the *ith-equation* using the previous updated unknowns

$$\left.\begin{aligned}10x_1+x_2+x_3&=12\\x_1+10x_2+x_3&=12\\x_1+x_2+10x_3&=12\end{aligned}\right\}\Longrightarrow\begin{aligned}x_1^{(k+1)}&=\frac{1}{10}(-x_2^{(k)}-x_3^{(k)}+12)\\x_2^{(k+1)}&=\frac{1}{10}(-x_1^{(k)}-x_3^{(k)}+12)\\x_3^{(k+1)}&=\frac{1}{10}(-x_1^{(k)}-x_2^{(k)}+12)\end{aligned}$$

STATIONARY METHODS

- Gauss-Seidel Method:
  - Idea: to solve *ith-unknown* for the *ith-equation* using the previous updated unknowns

$$\left.\begin{aligned}10x_1+x_2+x_3&=12\\x_1+10x_2+x_3&=12\\x_1+x_2+10x_3&=12\end{aligned}\right\}\Longrightarrow\begin{aligned}x_1^{(k+1)}&=\frac{1}{10}(-x_2^{(k)}-x_3^{(k)}+12)\\x_2^{(k+1)}&=\frac{1}{10}(-x_1^{(k+1)}-x_3^{(k)}+12)\\x_3^{(k+1)}&=\frac{1}{10}(-x_1^{(k+1)}-x_2^{(k+1)}+12)\end{aligned}$$

STATIONARY METHODS

- Gauss-Seidel: Split the original matrix

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & 0 & \cdots & \cdots & 0 \\ a_{21} & a_{22} & 0 & & \vdots \\ a_{31} & a_{32} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ a_{n1} & \cdots & \cdots & a_{nn-1} & a_{nn} \end{pmatrix} + \begin{pmatrix} 0 & a_{12} & \cdots & \cdots & a_{1n} \\ 0 & 0 & a_{23} & \cdots & a_{2n} \\ \vdots & 0 & 0 & \cdots & a_{3n} \\ \vdots & \vdots & 0 & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & 0 \end{pmatrix}$$

$$A = A_{LD} + A_U$$
$$M = A_{LD}, \; K = A_U$$
$$x^{(k+1)} = M^{-1}(-Kx^{(k)} + b) =$$
$$= A_{LD}^{-1}(-A_U x^{(k)} + b)$$

STATIONARY METHODS

- Gauss-Seidel: Split the original matrix

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & 0 & \cdots & \cdots & 0 \\ a_{21} & a_{22} & 0 & & \vdots \\ a_{31} & a_{32} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ a_{n1} & \cdots & \cdots & a_{nn-1} & a_{nn} \end{pmatrix} + \begin{pmatrix} 0 & a_{12} & \cdots & \cdots & a_{1n} \\ 0 & 0 & a_{23} & \cdots & a_{2n} \\ \vdots & 0 & 0 & \cdots & a_{3n} \\ \vdots & \vdots & 0 & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & 0 \end{pmatrix}$$

```
>>AL=tril(A,-1)
>>AU=triu(A,1)
>>AD=diag(diag(A))
>>Xkp1=inv(AD+AL)*(-AU*Xk+b)
```


STATIONARY METHODS

- Successive Overrelaxation Method (SOR): The Gauss-Seidel step is extrapolated by a factor

$$x^{(k+1)} = \omega \tilde{x}_i^{(k+1)} + (1-\omega)x_i^{(k)}$$

where $\tilde{x}$ is the Gauss-Seidel iterate.

- If $\omega = 1$ Gauss-Seidel
- If $\omega > 1$ Overrrelaxation
- If $\omega < 1$ Underrelaxation.

STATIONARY METHODS

- Convergence:
  - If *A* is strictly row diagonally dominant $|a_{ii}| > \sum_{i \neq j} |a_{ij}|$ Jacobi and G-S converges.
  - If *A* is symmetric positive definite G-S and SOR converges for $0 < \omega < 2$
  - In general ω is hard to choose, if the spectral radius $\rho(R_J)$ of the Jacobi iteration matrix then the optimal ω is

$$\omega = \frac{2}{1 + \sqrt{1 - \rho(R_J)}}$$

  - G-S can be twice as fast as Jacobi

CONJUGATE GRADIENTS METHOD

- Krylov subspace method: Create the Krylov subspace spanned by

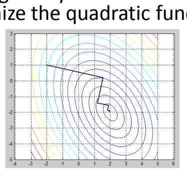
$$K = < b, Ab, A^2b, A^3b, \ldots.., A^{k-1}b >$$

and find solutions in this subspace.

- Only matrix-vector products involved
- For Symmetric Positive Definite matrices good convergence properties.

CONJUGATE GRADIENTS METHOD

- Optimization Problem:
  - Solving the system $Ax - b = 0$ is equivalent to minimize the quadratic function: $\varphi(x) = \frac{1}{2} x^T Ax - x^T b$



  - The minimization can be done by *line searches* where $\varphi(x_n)$ is minimized along search direction p_n

CONJUGATE GRADIENTS METHOD

– The α_{n+1} that minimizes $\varphi(x_n + \alpha_{n+1} p_n)$ is $\alpha_{n+1} = \frac{p_n^T r_n}{p_n^T A p_n}$ with residual $r_n = b - Ax_n$

– The residual is also the *Gradient* of $\varphi(x_n)$

$$\varphi'(x_n) = Ax_n - b = -r_n$$

– Simple approach: set the search direction p_n to the negative gradient r_n

CONJUGATE GRADIENTS METHOD

Algorithm: Steepest Descent

$x_0 = 0, r_0 = b$
for $k = 1, 2, 3, \ldots$

$\alpha_n = (r_{n-1}^T r_{n-1})/(r_{n-1}^T A r_{n-1})$	step length
$x_n = x_{n-1} + \alpha_n p_{n-1}$	approximate solution
$r_n = r_{n-1} - \alpha_n A p_{n-1}$	residual

– Corresponds to moving in the direction that $\varphi(x_n)$ changes the most

– Poor convergence, tends to move along previous directions

CONJUGATE GRADIENTS METHOD

– The optimization procedure can be improved by better search directions

– Let the search directions be A-conjugate: $p_i^T A p_k = 0$

– Then the algorithm converges in at most n steps.

CONJUGATE GRADIENTS METHOD

Algorithm: Conjugate Gradients Method

$x_0 = 0, r_0 = b, p_0 = r_0$
for $k = 1, 2, 3, \ldots$

$\alpha_n = (r_{n-1}^T r_{n-1})/(p_{n-1}^T A p_{n-1})$	step length
$x_n = x_{n-1} + \alpha_n p_{n-1}$	approximate solution
$r_n = r_{n-1} - \alpha_n A p_{n-1}$	residual
$\beta_n = (r_n^T r_n)/(r_{n-1}^T r_{n-1})$	improvement this step
$p_n = r_n + \beta_n p_{n-1}$	search direction

- Only few storage vectors needed
- Finds the best solution in norm $\|x\|_A = \sqrt{x^T A x}$

PRECONDITIONING

- The idea is to modify the initial system $Ax = b$ using a non-singular *preconditioner* matrix

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

- Convergence properties based on $M^{-1}A$
- Trade-off between the cost of applying M^{-1} and the improvement of the convergence properties.
- Very usual: $M = diag(A)$

PRECONDITIONING

Algorithm: Preconditioned Conjugate Gradients Method

$x_0 = 0, r_0 = b, p_0 = M^{-1}r_0, z_0 = p_0$
for $n = 1, 2, 3, \ldots$

$\alpha_n = (r_{n-1}^T z_{n-1})/(p_{n-1}^T A p_{n-1})$	step length
$x_n = x_{n-1} + \alpha_n p_{n-1}$	approximate solution
$r_n = r_{n-1} - \alpha_n A p_{n-1}$	residual
$z_n = M^{-1} r_n$	preconditioning
$\beta_n = (r_n^T z_n)/(r_{n-1}^T z_{n-1})$	improvement this step
$p_n = z_n + \beta_n p_{n-1}$	search direction