Computational Linear Algebra

- **Course:** (MATH: 6800, CSCI: 6800)
- **Semester:** Fall 1998
- **Instructors:**
  - Joseph E. Flaherty, flaherje@cs.rpi.edu
  - Franklin T. Luk, luk@cs.rpi.edu
  - Wesley Turner, turnerw@cs.rpi.edu

Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180

OUTLINE

1. Introduction

   1.1 Notation

   1.2 Special matrices

2. Gaussian Elimination

   2.1 Vector and matrix norms

   2.2 Finite precision arithmetic

   2.3 Factorizations

   2.4 Pivoting

   2.5 Accuracy estimation

3. Special Linear Systems

   3.1 Symmetric positive definite systems

   3.2 Banded and profile systems

   3.3 Block tridiagonal systems

   3.4 Symmetric indefinite systems

   3.5 Profile methods

   3.6 Sparse Systems

4. Orthogonalization and Least Squares

   4.1 Orthogonality and the singular value decomposition

   4.2 Rotations and reflections

   4.3 **QR** factorizations

   4.4 Least squares problems

OUTLINE

5. The Algebraic Eigenvalue Problem

   5.1 Introduction

   5.2 Power methods

   5.3 Hessenberg and Schur forms. The **QR** algorithm

   5.4 Symmetric problems

   5.5 Jacobi iteration

6. Iterative Methods

   6.1 Fixed-point methods

   6.2 The basic conjugate gradient method

   6.3 Preconditioning

   6.4 Krylov and generalized-residual methods

7. Introduction to Parallel Computing

   7.1 Shared- and distributed-memory computation

   7.2 Matrix multiplication

   7.3 Gaussian elimination

References

1. O. Axelsson (1994), *Iterative Solution Methods*, Cambridge, Cambridge.

2. R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, V. Pozo, C. Romine, and H. van der Vorst (1994), *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia.

3. Å. Björck (1996), *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia.

4. W. Briggs (1987), *A Multigrid Tutorial*, SIAM, Philadelphia.

5. J.W. Demmel (1997), *Applied Numerical Linear Algebra*, SIAM, philadelphia.

6. A. George and J. Liu (1981), *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs.

7. G.H. Golub and C.F. Van Loan (1993), *Matrix Computations*, Third Edition, Johns Hopkins, Baltimore.

8. A. Greenbaum (1997), *Iterative Methods for Solving Linear Systems*, SIAM Philadelphia.

9. W. Hackbusch (1994), *Iterative Solution of Large Sparse Linear Systems of Equations*, Springer-Verlag, Berlin.

10. N. Higham (1996), *Accuracy and Stability of Numerical Algoritms*, SIAM Philadelphia.

11. B. Parlett (1980), *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs.

12. Y. Saad (1996), *Iterative Methods for Sparse Linear Systems*, PWS, Boston.

13. B. Smith, P. Bjorstad, and W. Gropp (1996), *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge, Cambridge.

14. G.W. Stewart (1973), *Introduction to Matrix Computations*, Academic Press, New York.

15. G.W. Stewart and J.-G. Sun (1990) *Matrix Perturbation Theory*, Academic Press, New York.

16. L.N. Trefethen and D. Bau, III (1997), *Numerical Linear Algebra*, SIAM, Philadelphia. [1]

17. P. Wesseling (1992), *An Introduction to Multigrid Methods*, Wiley, Chichester.

[1] Text

18. D. Young (1971), *Iterative Solution of Large Linear Systems*, Academic Press, New York.

1. Introduction

1.1. Notation

- *Motivation:*
  - Linear algebra is involved in approximately 75% of scientific computation
    - Circuit and network analysis
    - Potential theory
    - Signal processing
    - Acoustics and vibrations
    - Optimization
    - Finite difference and finite element methods
    - Tomography
    - Optimal control

- *Problems:*
  - Solve linear systems
  - Solve least squares problems
  - Solve eigenvalue-eigenvector problems

Notation

- Reading: Trefethen and Bau (1997), Lecture 1
- An $m \times n$ matrix

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \tag{1}$$

  - Bold letters denote vectors and matrices
  - Italic letters denote scalars
    - Lower case letters are elements of a matrix
  - $a_{ij} \in \Re$, $i = 1, 2, ..., m$, $j = 1, 2, ..., n$, unless stated
  - $\mathbf{A} \in \Re^{m \times n}$
    - $\Re^{m \times n}$ a vector space of $m \times n$ real matrices
  - A vector: $\mathbf{x} \in \Re^{m \times 1} := \Re^m$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}, \qquad \mathbf{x}^T = [x_1, x_2, \cdots, x_m]$$

MATLAB

- Specify Algorithms using *MATLAB*
- *Vector dot (scalar) product*

$$c = \mathbf{x}^T\mathbf{y} = \sum_{i=1}^{n} x_i y_i \tag{2}$$

function c = dot($\mathbf{x}$, $\mathbf{y}$)
% dot: compute a dot product of the n-vectors $\mathbf{x}$ and $\mathbf{y}$
  n = length($\mathbf{x}$)
  c = 0
  for i = 1:n
    c = c + x(i)*y(i)
  end

  - Subscripts are enclosed in parentheses
  - The MATLAB function *length* computes the dimension of a vector
  - The : indicates a range
    - In the for loop, i ranges over integers from 1 to n
  - dot($\mathbf{x}$, $\mathbf{y}$) is a library function in MATLAB

A SAXPY

- A *saxpy* ("scalar a x plus y") is the vector

$$\mathbf{z} = a\mathbf{x} + \mathbf{y} \tag{3}$$

function $\mathbf{z}$ = saxpy(a, $\mathbf{x}$, $\mathbf{y}$)
% saxpy: compute the vector $\mathbf{z}$ = a $*$ $\mathbf{x}$ + $\mathbf{y}$ where
% a is a scalar and $\mathbf{x}$ and $\mathbf{y}$ are vectors
  n = length($\mathbf{x}$)
  for i = 1:n
    z(i) = a*x(i) + y(i)
  end

  - $\mathbf{x}$ and $\mathbf{y}$ have the same dimensions n
    - Production software should check this

Matrix Multiplication

- Let $\mathbf{A} \in \Re^{m \times r}$, $\mathbf{B} \in \Re^{r \times n}$, and $\mathbf{C} \in \Re^{m \times n}$, then

$$c_{ij} = \sum_{k=1}^{r} a_{ik} b_{kj}, \qquad i = 1 : m, \qquad j = 1 : n \tag{4}$$

```
function C = matmy(A, B)
% matmy: compute the matrix product C = AB
   [m r] = size(A);
   [r n] = size(B);
   for i = 1:m
      for j = 1:n
         C(i,j) = 0;
         for k = 1:r
            C(i,j) = C(i,j) + A(i,k)*B(k,j);
         end
      end
   end
```

  - size(**A**) returns the row and column dimensions of **A**
  - The ;s suppress printing of results
  - The complexity is mnr multiplications and additions
  - Some details are omitted

Matrix Multiplication

- Matrix multiplication using the dot product
  - Let $\mathbf{a}_j$ denote the jth column of $\mathbf{A}$

$$\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_r] \tag{5}$$

  - Replace the loop in (4) by a dot product

$$c_{ij} = \mathbf{a}_i^T \mathbf{b}_j, \qquad i = 1 : m, \qquad j = 1 : n \tag{6}$$

    * $\mathbf{a}_i^T$ is the ith column of $\mathbf{A}^T$ or the ith row of $\mathbf{A}$

```
function C = matmy(A, B)
% matmy: compute the matrix product C = AB using
% the dot product
   [m r] = size(A);
   [r n] = size(B);
      for i = 1:m
      for j = 1:n
         C(i,j) = dot(A(i,1:r)',B(1:r,j));
      end
   end
```

  - The range 1:r indicates an operation for an entire row or column
  - A ' denotes transposition
  - $A(i, 1 : r)$ is the ith row of $\mathbf{A}$

Matrix Multiplication with Saxpys

- Reorder the loop structure: regard the columns of $\mathbf{C}$ as linear combinations of those of $\mathbf{A}$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 5 \begin{bmatrix} 1 \\ 3 \end{bmatrix} + 7 \begin{bmatrix} 2 \\ 4 \end{bmatrix} & 6 \begin{bmatrix} 1 \\ 3 \end{bmatrix} + 8 \begin{bmatrix} 2 \\ 4 \end{bmatrix} \end{bmatrix}$$

  - compute $\mathbf{C} = \mathbf{AB}$ as

$$\mathbf{c}_j = \sum_{k=1}^{r} b_{kj} \mathbf{a}_k, \qquad j = 1 : n \tag{7}$$

  - Implement the sum using saxpys

```
function C = smatmy(A, B)
% smatmy: compute the matrix product C = AB
% using saxpys
   [m r] = size(A);
   [r n] = size(B);
   C = zeros(m, n);
   for j = 1:n
      for k = 1:r
         C(1:m,j) = saxpy(B(k,j), A(1:m,k), C(1:m,j));
      end
   end
```

  - zeros(m,n) returns a $m \times n$ matrix of zeros.

Outer Products

- Reorder the loops to perform an outer product
  - An *outer product* of an m-vector $\mathbf{x}$ and an n-vector $\mathbf{y}$ is the $m \times n$ matrix

$$\mathbf{C} = \mathbf{x}\mathbf{y}^T \tag{8}$$

    or

$$\mathbf{C} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} [y_1 y_2 \cdots y_n] = \begin{bmatrix} x_1y_1 & x_1y_2 & \cdots & x_1y_n \\ x_2y_1 & x_2y_2 & \cdots & x_2y_n \\ \vdots & \vdots & \ddots & \vdots \\ x_my_1 & x_my_2 & \cdots & x_my_n \end{bmatrix}$$

  - *Problem 1*: do matrix multiplication using outer products
    - Show this for the 2×2 problem

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

Outer Product

$*$ Write an algorithm for a general system

- Different loop structures perform differently on different computers

dot product	i, j, k
saxpy	j, k, i
outer porduct	k, i, j

1.2. Special Matrices

- Complex Matrices
  - $\mathbf{A} \in C^{m \times n}$ implies that a_{ij} are complex numbers

$$a_{ij} = \alpha_{ij} + i\beta_{ij}, \qquad \alpha_{ij}, \beta_{ij} \in \Re,$$
$$i = 1, 2, ..., m, \qquad j = 1, 2, ..., n$$

  - Operations are the same except for:
    - Transposition: if $a_{ij} = \alpha_{ij} + i\beta_{ij}$ then

$$\mathbf{C} = \mathbf{A}^H = \bar{\mathbf{A}}^T, \qquad c_{ij} = \bar{a}_{ji} = \alpha_{ji} - i\beta_{ji}$$

    - Inner product: if $\mathbf{x}, \mathbf{y} \in C^n$ then

$$s = \mathbf{x}^H \mathbf{y} = \sum_{i=1}^{n} \bar{x}_i y_i$$

    - $\mathbf{x}^H \mathbf{x}$ is real

$$\mathbf{x}^H \mathbf{x} = \sum_{i=1}^{n} \bar{x}_i x_i = \sum_{i=1}^{n} |x_i|^2$$

Band Matrices

- **Definition 1**: $\mathbf{A} \in \Re^{m \times n}$ has *lower bandwidth* p if $a_{ij} = 0$ for $i > j + p$ and *upper bandwidth* q if $a_{ij} = 0$ for $j > i + q$.

$$\mathbf{A} = \begin{bmatrix} \times & \times & 0 & 0 & 0 & 0 \\ \times & \times & \times & 0 & 0 & 0 \\ \times & \times & \times & \times & 0 & 0 \\ \times & \times & \times & \times & \times & 0 \\ 0 & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \end{bmatrix}$$

  - The $\times$ denotes an arbitrary nonzero entry
  - This 8×6 matrix has lower bandwidth 3 and upper bandwidth 1
  - *Example 1.* A 5×7 upper triangular matrix $p = 0$, $q = 6$

$$\begin{bmatrix} \times & \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times & \times \end{bmatrix}$$

Some Band Matrices

Matrix	p	q
Diagonal	0	0
Upper Triangular	0	$n-1$
Lower Triangular	$m-1$	0
Tridiagonal	1	1
Upper bidiagonal	0	1
Lower bidiagonal	1	0
Upper Hessenberg	1	$n-1$
Lower Hessenberg	$m-1$	1

- *Example 2.* A 5×6 tridiagonal matrix $p = q = 1$

$$\mathbf{A} = \begin{bmatrix} \times & \times & 0 & 0 & 0 & 0 \\ \times & \times & \times & 0 & 0 & 0 \\ 0 & \times & \times & \times & 0 & 0 \\ 0 & 0 & \times & \times & \times & 0 \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix}$$

- *Example 3.* A 4×6 Lower Hessenberg matrix $p = 3$, $q = 1$

$$\mathbf{A} = \begin{bmatrix} \times & \times & 0 & 0 & 0 & 0 \\ \times & \times & \times & 0 & 0 & 0 \\ \times & \times & \times & \times & 0 & 0 \\ \times & \times & \times & \times & \times & 0 \end{bmatrix}$$

Lower by Upper Triangular Matrix Product

- *Example 4* (cf. Golub and Van Loan (1996), Problem 1.2.3, p. 23).
- $\mathbf{L}$ and $\mathbf{U}$ are $n \times n$ lower and upper triangular matrices
- Give a column saxpy algorithm for computing

$$\mathbf{C} = \mathbf{UL}$$

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} = [\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3]$$

- Express the columns of $\mathbf{C}$ as a linear combination of the columns of $\mathbf{U}$

$$\mathbf{c}_1 = \begin{bmatrix} u_{11} \\ 0 \\ 0 \end{bmatrix} l_{11} + \begin{bmatrix} u_{12} \\ u_{22} \\ 0 \end{bmatrix} l_{21} + \begin{bmatrix} u_{13} \\ u_{23} \\ u_{33} \end{bmatrix} l_{31}$$

$$\mathbf{c}_2 = \begin{bmatrix} u_{12} \\ u_{22} \\ 0 \end{bmatrix} l_{22} + \begin{bmatrix} u_{13} \\ u_{23} \\ u_{33} \end{bmatrix} l_{32}, \qquad \mathbf{c}_3 = \begin{bmatrix} u_{13} \\ u_{23} \\ u_{33} \end{bmatrix} l_{33}$$

Lower by Upper Triangular Matrix Product

- Use the saxpy matrix multiplication formula (1.7)

$$\mathbf{c}_j = \sum_{k=1}^{n} \mathbf{u}_k l_{kj}$$

- $l_{kj} = 0$ if $k < j$

$$\mathbf{c}_j = \sum_{k=j}^{n} \mathbf{u}_k l_{kj}$$

- $u_{ik} = 0$ if $i > k$

```
function C = suplow(U, L)
% suplow: compute the matrix product C = UL using saxpys
% where L and U are n × n lower and upper
% triangular matrices.
   [n n] = size(U);
   C = zeros(n);
   for j = 1:n
      for k = j:n
         C(1:k,j) = saxpy(L(k,j), U(1:k,k), C(1:k,j));
      end
   end
```

Lower by Upper Triangular Matrix Product

- Operation count:
  - A saxpy for a vector of length k requires k multiplications and k additions
  - The inner k loop has $n - j$ saxpys on vectors of length k and requires

$$\sum_{k=j}^{n} k$$

    multiplications and additions
    - Recall

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \tag{1}$$

    - The multiplications/additions in the inner loop are

$$\sum_{k=j}^{n} k = \sum_{k=1}^{n} k - \sum_{k=1}^{j-1} k = \frac{1}{2}[n(n+1) - (j-1)j]$$

  - The multiplications/additions in the outer loop are

$$\frac{1}{2}\sum_{j=1}^{n}[n(n+1) - (j-1)j]$$

    - Recall

$$\sum_{i=1}^{n} i^2 = \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6} \tag{2}$$

Lower by Upper Triangular Matrix Product

- Thus, the total operation count is

$$\frac{1}{2}[n^2(n+1) - \frac{n^3}{3} - \frac{n^2}{2} - \frac{n}{6} + \frac{n(n+1)}{2}]$$

  or

$$\frac{1}{2}[\frac{2n^3}{3} + n^2 + \frac{n}{3}] = \frac{(n+1)(2n+1)n}{6}$$

  - For large n, there are approximately $n^3/3$ multiplications and additions

- Multiplication of two $n \times n$ full matrices requires n^3 multiplications and additions

Band-Matrix Storage

- Let $\mathbf{A} \in \Re^{n \times n}$ have lower and upper bandwidth p and q
  - If $p, q \ll n$ then store $\mathbf{A}$ in either a $(p+q+1) \times n$ or a $n \times (p+q+1)$ matrix $\mathbf{A}^{band}$ to save storage
  - *Example 5.* Consider a 8×8 matrix with $p = 3$ and $q = 1$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & 0 & 0 & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} & 0 & 0 & 0 & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & 0 & 0 & 0 \\ 0 & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} & 0 & 0 \\ 0 & 0 & a_{63} & a_{64} & a_{65} & a_{66} & a_{67} & 0 \\ 0 & 0 & 0 & a_{74} & a_{75} & a_{76} & a_{77} & a_{78} \\ 0 & 0 & 0 & 0 & a_{85} & a_{86} & a_{87} & a_{88} \end{bmatrix}$$

  - *Row Storage*: "slide" all rows to the left or right to align the columns on anti-diagonals

$$\mathbf{A}^{band} = \begin{bmatrix} 0 & 0 & 0 & a_{11} & a_{12} \\ 0 & 0 & a_{21} & a_{22} & a_{23} \\ 0 & a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{63} & a_{64} & a_{65} & a_{66} & a_{67} \\ a_{74} & a_{75} & a_{76} & a_{77} & a_{78} \\ a_{85} & a_{86} & a_{87} & a_{88} & 0 \end{bmatrix}$$

    - $\mathbf{A}^{band} \in \Re^{n \times p+q+1}$ with

$$a_{ij} = a^{band}_{i,j-i+p+1}$$

    - Can also store $\mathbf{A}$ by columns

Band Matrix-Vector Multiplication

- *Example 6.* Multiply a band matrix $\mathbf{A}$ by a vector $\mathbf{x}$

$$y_i = \sum_{j=\max(1,i-p)}^{\min(n,i+q)} a_{ij}x_j = \sum_{j=\max(1,i-p)}^{\min(n,i+q)} a^{band}_{i,j-i+p+1}x_j$$

function $\mathbf{y}$ = bmatvec(p, q, $\mathbf{A}^{band}$, $\mathbf{x}$)
% bmatvec: compute the matrix-vector product $\mathbf{y} = \mathbf{Ax}$
% where $\mathbf{A}^{band}$ is an $n \times n$ matrix stored in banded
% form by rows and $\mathbf{x}$ is an n-vector. The scalars p and q
% are the lower and upper bandwidths of $\mathbf{A}$, respectively.

% start and stop indicate the column index of the first and
% last nonzero entries in row i of $\mathbf{A}^{band}$.
% jstart and jstop indicate similar column indices for $\mathbf{A}$.

```
[n n] = size(A);
y(1:n) = 0;
for i = 1:n
   jstart = max(1, i-p);
   start = jstart - i + p + 1;
   jstop = min(n, i+q);
   stop = jstop - i + p + 1;
   y(i) = dot(A^band(i,start:stop)', x(jstart:jstop));
end
```

Symmetric Matrics

- **Definition 2**: $\mathbf{A} \in \Re^{n \times n}$ is *symmetric* if $\mathbf{A}^T = \mathbf{A}$

$$\mathbf{A} = \begin{bmatrix} 13 & 9 & -1 \\ 9 & 4 & 7 \\ -1 & 7 & -8 \end{bmatrix}$$

  - Only half of the matrix need be stored
  - Store the lower or upper triangular part of $\mathbf{A}$ as a vector

$$\mathbf{a}^{sym} = [13, 9, -1, 4, 7, -8]$$

  - Store the upper triangular part of $\mathbf{A}$ by rows

$$a_{ij} = a^{sym}_{(i-1)n - i(i-1)/2 + j} \tag{3a}$$

  - Arrange storage so that inner loops access contiguous data
    - *FORTRAN* stores matrices by columns
    - *C* stores matrices by rows
  - Symmetric matrix by vector multiplication

$$y_i = \sum_{j=1}^{n} a_{ij} x_j = \sum_{j=1}^{i-1} a_{ji} x_j + \sum_{j=i}^{n} a_{ij} x_j \tag{3b}$$

- Can also store $\mathbf{A}$ by diagonals

Block Matrices

- Partition the rows and columns of $\mathbf{A} \in \Re^{m \times n}$ into submatrices such that

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \cdots & \mathbf{A}_{1q} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \cdots & \mathbf{A}_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{p1} & \mathbf{A}_{p2} & \cdots & \mathbf{A}_{pq} \end{bmatrix}$$

  - $\mathbf{A}_{ij} \in R^{m_i \times n_j}$, $i = 1 : m$, $j = 1 : n$
  - $m_1 + m_2 + ... + m_p = m$, $n_1 + n_2 + ... + n_q = n$
  - All operations on matrices with scalar components apply to those with matrix components
    - Matrix operations on the blocks are required
  - Let $\mathbf{B} \in \Re^{k \times l}$

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} & \cdots & \mathbf{B}_{1s} \\ \mathbf{B}_{21} & \mathbf{B}_{22} & \cdots & \mathbf{B}_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{B}_{r1} & \mathbf{B}_{r2} & \cdots & \mathbf{B}_{rs} \end{bmatrix}$$

    - $\mathbf{B}_{ij} \in \Re^{k_i \times l_j}$, $i = 1 : k$, $j = 1 : l$
    - $k_1 + k_2 + ... + k_r = k$, $l_1 + l_2 + ... + l_s = l$

Block Matrix Addition and Multiplication

- *Addition*: a partition is *conformable* for addition if
  - $m = k$, $n = l$, $p = r$, $q = s$
  - $m_i = k_i$, $n_j = l_j$, $i = 1 : m$, $j = 1 : l$
  - Then $\mathbf{C} = \mathbf{A} + \mathbf{B}$ with $\mathbf{C}_{ij} = \mathbf{A}_{ij} + \mathbf{B}_{\mathbf{ij}}$, $i = 1 : p$, $j = 1 : q$

- *Multiplication*: a partition is *conformable* for multiplication if
  - $n = k$, $q = r$, $n_j = k_j$, $j = 1 : q$
  - Then

$$\mathbf{C} = \mathbf{AB} = \begin{bmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} & \cdots & \mathbf{C}_{1s} \\ \mathbf{C}_{21} & \mathbf{C}_{22} & \cdots & \mathbf{C}_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_{p1} & \mathbf{C}_{p2} & \cdots & \mathbf{C}_{ps} \end{bmatrix} \tag{4a}$$

with

$$\mathbf{C}_{ij} = \sum_{t=1}^{q} \mathbf{A}_{it}\mathbf{B}_{tj}, \qquad i = 1 : p, \qquad j = 1 : s \tag{4b}$$

Sparse Matrices

- *Definition*
  - Sparse matrices have a "significant" number of zeros
  - A matrix is considered to be sparse if the zeros are neither to be stored nor operated upon
- *Motivation*
  - Sparse matrices with arbitrary zero structures arise in finite element and network problems

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 2 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ 6 & 0 & 7 & 8 & 9 \\ 0 & 0 & 10 & 11 & 0 \\ 0 & 0 & 0 & 0 & 12 \end{bmatrix}$$

- *Storage*
  - Store $\mathbf{A}$ as a vector $\mathbf{a}$ by rows or columns
  - An integer vector $\mathbf{ja}$ stores the column indices of $\mathbf{A}$
    - when $\mathbf{A}$ is stored by rows
  - A second vector $\mathbf{ia}$ stores the indices in $\mathbf{ja}$ of the first element of each row of $\mathbf{A}$
    - The last element of $\mathbf{ia}$, $ia_n + 1$, usually signals termination

Sparse Matrices

- For example

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \end{bmatrix} \qquad \mathbf{ja} = \begin{bmatrix} 1 \\ 4 \\ 1 \\ 2 \\ 4 \\ 1 \\ 3 \\ 4 \\ 5 \\ 3 \\ 4 \\ 5 \end{bmatrix} \qquad \mathbf{ia} = \begin{bmatrix} 1 \\ 3 \\ 6 \\ 10 \\ 12 \\ 13 \end{bmatrix}$$

- The dimension of **a** and **ja** is the number of nonzeros in **A**. That of **ia** is $n + 1$
  - $ia_n + 1$ is set to the beginning index of a ficticious row $n + 1$
- The storage scheme is called the *compressed sparse row (CSR)* format
  - CSC is the *compressed sparse column* format
- Linked structures can be used instead of arrays

Sparse Matrices

- *Example 7.* Multiply a sparse matrix **A** by a vector **x**

```
function y = smatvec(a, ja, ia, x)
% smatvec: compute the matrix-vector product y = Ax
% where a is the n-by-n matrix A stored in CSR
% format, ja is a vector of pointers to the nonzero elements
% of A, ia is a vector pointing to the first nonzero
% element in each row of A.

   n = length(ia) - 1;
   for i = 1:n

% start and stop locate the beginning
% and end of row i in a and ja.

   start = ia(i);
   stop = ia(i+1) - 1;
   y(i) = 0;
   for k = start:stop
      y(i) = y(i) + a(k)*x(ja(k));
   end
end
```

- cf. Y. Saad (1996), *Iterative Methods for Sparse Linear Systems,* PWS, Boston, Section 3.4.

2. Gaussian Elimination

2.1. Vector and Matrix Norms

- *(Linear) Vector Space*. Regard the columns (rows) of a matrix $\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\}$ as forming a vector space $\Re^{m \times n}$
  - The key properties of a vector space $\mathcal{V}$ are
    i. If $\mathbf{u}, \mathbf{v} \in V$ then $\mathbf{u} + \mathbf{v} \in \mathcal{V}$
    ii. If $\mathbf{u} \in V$ and $\alpha \in \Re$ then $\alpha \mathbf{u} \in \mathcal{V}$
  - The definition appears in, e.g., K. Hoffman and R. Kunze (1971), *Linear algebra*, Second Edition, Prentice Hall, Englewood Cliffs
  - A *subspace* of $\mathcal{V}$ is a subset that is also a vector space
- **Definition 1**: A set of vectors $\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\} \in \Re^{m \times n}$ is *linearly independent* if the only solution of

$$\sum_{j=1}^{n} \alpha_j \mathbf{a}_j = 0$$

is $\alpha_j = 0$, $j = 1 : n$

Subspaces

- **Definition 2**: The set of all linear combinations of

$$\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\} \in \Re^{m \times n}$$

  is a vector space called the *linear span* of $\Re^{m \times n}$:

$$\text{span}(\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\}) = \left\{ \sum_{j=1}^{n} \alpha_j \mathbf{a}_j | \alpha_j \in \Re \right\} \qquad (1)$$

  - **Theorem 1**: If $\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\}$ is linearly independent then $\mathbf{u} \in \text{span}(\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\})$ is unique

- **Definition 3**: Let $\mathcal{S}_1, \mathcal{S}_2 \subset \Re^{m \times n}$, then

  i. Their *sum* is the subspace $\{\mathbf{a}_1 + \mathbf{a}_2 | \mathbf{a}_1 \in \mathcal{S}_1, \mathbf{a}_2 \in \mathcal{S}_2\}$

  ii. If $\mathcal{S}_1 \cap \mathcal{S}_2$ is $\emptyset$ then the sum of $\mathcal{S}_1$ and $\mathcal{S}_2$ is called the *direct sum* and is written as $\mathcal{S}_1 \oplus \mathcal{S}_2$

- **Definition 4**: A subset $\mathcal{S}$ is a *maximal linearly independent subset* of $\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\}$ if:

  i. it is linearly independent

  ii. it is not properly contained in any linearly independent subset of $\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\}$

- **Definition 5**: A maximal linearly independent set $\{\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_k\}$ forms a *basis* for $\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\}$

Range, Null Space, and Rank

- Some properties:
  i. If $\{\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_k\}$ is a basis for $\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\}$ then
  $$\text{span}(\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\}) = \text{span}(\{\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_k\})$$
  ii. All bases for $\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\}$ have the same number of elements
  iii. This number is the *dimension* of $\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\}$ and is written as $\dim(\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\})$
- **Definition 6**: If $\mathbf{A} \in \Re^{m \times n}$ then:
  i. The *range* of $\mathbf{A}$ satisfies
  $$\text{range}(\mathbf{A}) = \{\mathbf{y} \in \Re^m | \mathbf{y} = \mathbf{A}\mathbf{x}, \mathbf{x} \in \Re^n\} \quad (2)$$
  ii. The *null space* of $\mathbf{A}$ satisfies
  $$\text{null}(\mathbf{A}) = \{\mathbf{x} \in \Re^n | \mathbf{A}\mathbf{x} = 0\} \quad (3)$$
- For $\mathbf{A} \in \Re^{m \times n}$
  i. $\text{range}(\mathbf{A}) = \text{span}(\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\})$
  ii. The *rank* of $\mathbf{A}$ satisfies $\text{rank}(\mathbf{A}) = \dim(\text{range}(\mathbf{A}))$
  iii. $\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{A}^T)$
    – $\text{rank}(\mathbf{A})$ is the number of independent rows or columns of $\mathbf{A}$
  iv. $\dim(\text{null}(\mathbf{A})) + \text{rank}(\mathbf{A})) = n$

Inverses

- The $n \times n$ identity matrix is

$$\mathbf{I} = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_n] = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{4}$$

- If $\mathbf{A}, \mathbf{X} \in \Re^{n \times n}$ and

$$\mathbf{AX} = \mathbf{I}$$

  then $\mathbf{X}$ is called the *inverse of* $\mathbf{A}$ and is written as $\mathbf{A}^{-1}$

  - If $\mathbf{A}^{-1}$ does not exist then $\mathbf{A}$ is called *singular*, otherwise *nonsingular*.
  - Some properties:

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1} \tag{5a}$$

$$(\mathbf{A}^{-1})^T = (\mathbf{A}^T)^{-1} := \mathbf{A}^{-T} \tag{5b}$$

Determinants

- The *determinant* of a matrix $\mathbf{A} \in \Re^{n \times n}$ satisfies

$$\det(\mathbf{A}) = \begin{cases} a_{11}, & \text{if } n = 1 \\ \sum_{j=1}^{n} (-1)^{j+1} a_{1j} \det(\mathbf{A_{1j}}), & \text{if } n > 1 \end{cases} \tag{6}$$

  where $\mathbf{A}_{1j}$ is an $(n-1) \times (n-1)$ matrix obtained by deleting the first row and column of $\mathbf{A}$

  - The definition is recursive
  - Some properties:

$$\det(\mathbf{AB}) = \det(\mathbf{A}) \det(\mathbf{B}) \tag{7a}$$

$$\det(\mathbf{A}^T) = \det(\mathbf{A}) \tag{7b}$$

$$\det(\alpha \mathbf{A}) = \alpha^n \det(\mathbf{A}), \qquad \alpha \in \Re \tag{7c}$$

    * $\det(\mathbf{A}) \neq 0$ if and only if $\mathbf{A}$ is nonsingular

Vector Norms

- Reading: Trefethen and Bau (1997), Lecture 3
- Norms provide a way to measure distances between vectors and matrices
  - They provide a measure of "closeness" that is used to understand convergence
- **Definition 7**: A *vector norm* $||\cdot||$ is a functional ($||\cdot|| : \Re^n \rightarrow \Re$) satisfying
  - i. $||\mathbf{x}|| \geq 0, \qquad ||\mathbf{x}|| = 0 \leftrightarrow \mathbf{x} = 0, \mathbf{x} \in \Re^n$
  - ii. $||\mathbf{x} + \mathbf{y}|| \leq ||\mathbf{x}|| + ||\mathbf{y}||, \mathbf{x}, \mathbf{y} \in \Re^n$
  - iii. $||\alpha\mathbf{x}|| = |\alpha|||\mathbf{x}||, \alpha \in \Re, \mathbf{x}, \mathbf{y} \in \Re^n$
- The p-norms are the most useful to us:

$$||\mathbf{x}||_p = (|x_1|^p + |x_2|^p + ... + |x_n|^p)^{1/p}, \qquad p \geq 1 \tag{8}$$

  - The most important of these are

$$||\mathbf{x}||_1 = |x_1| + |x_2| + ... + |x_n| \tag{9a}$$

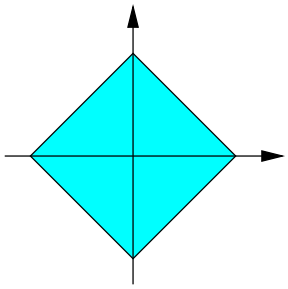
$$||\mathbf{x}||_2 = (|x_1|^2 + |x_2|^2 + ... + |x_n|^2)^{1/2} \tag{9b}$$

$$||\mathbf{x}||_\infty = \max_{1 \leq i \leq n} |x_i| \tag{9c}$$

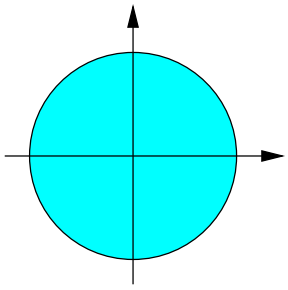
Vector Norms

- For $\mathbf{x} \in \Re^2$

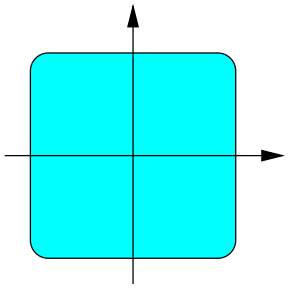
$||\mathbf{x}||_1 = |x_1| + |x_2|$



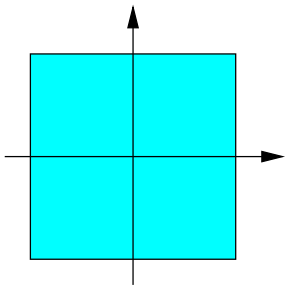
$||\mathbf{x}||_2 = (|x_1|^2 + |x_2|^2)^{1/2}$



$||\mathbf{x}||_p = (|x_1|^p + |x_2|^p)^{1/p}, 1 < p < \infty$



$||\mathbf{x}||_\infty = \max(|x_1|, |x_2|)$



Vector Norms

- Some properties of p-norms
  - The *Holder inequality:*

$$|\mathbf{x}^T\mathbf{y}| \leq ||\mathbf{x}||_p||\mathbf{y}||_q, \qquad \frac{1}{p}+\frac{1}{q}=1 \tag{10a}$$

  - The *Cauchy-Schwartz inequality* $(p = q = 2)$

$$|\mathbf{x}^T\mathbf{y}| \leq ||\mathbf{x}||_2||\mathbf{y}||_2 \tag{10b}$$

  - **Theorem 2**: p-norms are equivalent in the sense that there exist constants $c, C > 0$ such that

$$c||\mathbf{x}||_\alpha \leq ||\mathbf{x}||_\beta \leq C||\mathbf{x}||_\alpha \tag{11}$$

    * For example

$$||\mathbf{x}||_\infty \leq ||\mathbf{x}||_1 \leq n||\mathbf{x}||_\infty \tag{12a}$$

$$||\mathbf{x}||_\infty \leq ||\mathbf{x}||_2 \leq \sqrt{n}||\mathbf{x}||_\infty \tag{12b}$$

$$||\mathbf{x}||_2 \leq ||\mathbf{x}||_1 \leq \sqrt{n}||\mathbf{x}||_2 \tag{12c}$$

Vector Norms

- *Example 1* (cf. Trefethen and Bau (1997)), Problem 3.3). Verify (12)

Matrix Norms

- Matrix norms measure the sensitivity of a matrix to perturbations
- The definition of a matrix norm is identical to that of a vector norm
- **Definition 8**: A *matrix norm* $||\mathbf{A}||$, $\mathbf{A} \in \Re^{m \times n}$, is a functional satisfying

  i. $||\mathbf{A}|| \geq 0, \qquad ||\mathbf{A}|| = 0 \leftrightarrow \mathbf{A} = 0$

  ii. $||\mathbf{A} + \mathbf{B}|| \leq ||\mathbf{A}|| + ||\mathbf{B}||$

  iii. $||\alpha \mathbf{A}|| = |\alpha| ||\mathbf{A}||, \alpha \in \Re$

- Vector induced matrix norms are defined in terms of p-norms of vectors

$$||\mathbf{A}||_p = \max_{\mathbf{x} \neq 0} \frac{||\mathbf{A}\mathbf{x}||_p}{||\mathbf{x}||_p} \tag{13a}$$

  – Letting $\mathbf{y} = \mathbf{x}/||\mathbf{x}||_p$, yields the equivalent definition

$$||\mathbf{A}||_p = \max_{||\mathbf{x}||=1} ||\mathbf{A}\mathbf{x}||_p \tag{13b}$$

  – There are also pq-norms

$$||\mathbf{A}||_{pq} = \max_{\mathbf{x} \neq 0} \frac{||\mathbf{A}\mathbf{x}||_p}{||\mathbf{x}||_q} \tag{13c}$$

Matrix Norms

- Properties of matrix norms:
  - From (13a), it is clear that

$$||\mathbf{Ax}||_p \leq ||\mathbf{A}||_p ||\mathbf{x}||_p \tag{14a}$$

    and, more generally,

$$||\mathbf{AB}||_p \leq ||\mathbf{A}||_p ||\mathbf{B}||_p, \qquad \mathbf{A} \in \Re^{m \times n}, \qquad \mathbf{B} \in \Re^{n \times r} \tag{14b}$$

    * Matrix norms that satisfy (14b) are called *consistent.*
  - As a further consequence of (14b)

$$||\mathbf{A}^k||_p \leq ||\mathbf{A}||_p^k$$

- Matrix p norms corresponding to vector p-norms are

$$||\mathbf{A}||_1 = \max_{1 \leq j \leq n} \sum_{i=1}^{m} |a_{ij}| \tag{15a}$$

$$||\mathbf{A}||_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^{n} |a_{ij}| \tag{15b}$$

$$||\mathbf{A}||_2 = [\rho(\mathbf{A}^T \mathbf{A})]^{1/2} \tag{15c}$$

  - The *spectral radius* $\rho(\mathbf{A})$ is the magnitude of the largest eigenvalue of a matrix $\mathbf{A} \in \Re^{n \times n}$.

Frobenius Matrix Norm

- p norms
  - The 1-norm is the maximum absolute column sum
  - The ∞-norm is the maximum absolute row sum
  - The 2-norm requires an eigenvalue analysis
- The *Frobenius norm*:

$$||\mathbf{A}||_F = \left[\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right]^{1/2} \tag{16a}$$

  - It is not a p-norm
  - It can be viewed as the 2-norm of a vector in $\Re^{mn}$
  - It satisfies (14b)
  - It has the equivalent definition

$$||\mathbf{A}||_F = [\mathrm{tr}(\mathbf{A}^T\mathbf{A})]^{1/2} = [\mathrm{tr}(\mathbf{A}\mathbf{A}^T)]^{1/2} \tag{16b}$$

    where

$$\mathrm{tr}(\mathbf{A}) = \sum_{i=1}^{n} a_{ii}, \qquad \mathbf{A} \in \Re^{n \times n} \tag{16c}$$

Fun with Norms

- *Example 2.* Calculate the 1, 2, ∞, and F norms of

$$\mathbf{A} = \begin{bmatrix} 1 & -3 & 6 \\ 4 & 7 & -3 \\ -2 & 5 & 9 \\ 3 & -2 & 4 \end{bmatrix}$$

  - The column sums are

$$\sum_{i=1}^{4} |a_{i1}| = \quad , \quad \sum_{i=1}^{4} |a_{i2}| = \quad , \quad \sum_{i=1}^{4} |a_{i3}| =$$

    Thus,

$$||\mathbf{A}||_1 =$$

  - The row sums are

$$\sum_{j=1}^{3} |a_{1j}| = \quad , \quad \sum_{j=1}^{3} |a_{2j}| =$$

$$\sum_{j=1}^{3} |a_{3j}| = \quad , \quad \sum_{j=1}^{3} |a_{4j}| =$$

    Thus,

$$||\mathbf{A}||_\infty =$$

More Fun

- The product of $\mathbf{A}$ and $\mathbf{A}^T$ is

$$\mathbf{A}^T\mathbf{A} = \begin{bmatrix} 30 & 9 & -12 \\ 9 & 87 & -2 \\ -12 & -2 & 142 \end{bmatrix}$$

- The eigenvalues of $\mathbf{A}^T\mathbf{A}$ are 88.1293, 27.4447, and 143.4260
- Thus,

$$||\mathbf{A}||_2 = \sqrt{143.4260} = 11.9761$$

- Also $||\mathbf{A}||_F = 16.0935$
- **Theorem 3**: If $\mathbf{A} \in \Re^{m\times n}$ then

$$||\mathbf{A}||_2^2 \leq ||\mathbf{A}||_1||\mathbf{A}||_\infty \tag{17}$$

  - *Proof*: $\mu = ||\mathbf{A}||_2$ satisfies

$$\mathbf{A}^T\mathbf{A}\mathbf{z} = \mu^2\mathbf{z}$$

    μ^2 is the largest eigenvalue of $\mathbf{A}^T\mathbf{A}$ and $\mathbf{z}$ is the corresponding eigenvector

  - Take the 1 norm and use (14b) and (15a,b)

$$\begin{aligned} \mu^2||\mathbf{z}||_1 = ||\mathbf{A}^T\mathbf{A}\mathbf{z}||_1 &\leq ||\mathbf{A}^T||_1||\mathbf{A}||_1||\mathbf{z}||_1 \\ &= ||\mathbf{A}||_\infty||\mathbf{A}||_1||\mathbf{z}||_1 \end{aligned}$$

  - Divide by $||\mathbf{z}||_1$

- For Example 2, $||\mathbf{A}||_2 \leq \sqrt{(22)(16)} = 18.7617$

Spectral Radii and Norms

- **Theorem 4**: For $\mathbf{A} \in \Re^{n \times n}$

$$\rho(\mathbf{A}) \leq ||\mathbf{A}||_p \tag{18}$$

  - Let λ be the eigenvalue of $\mathbf{A}$ corresponding to the spectral radius and $\mathbf{z}$ be the corresponding eigenvector
    - $\rho(\mathbf{A}) = |\lambda|$
  - Consider

$$\mathbf{A}\mathbf{z} = \lambda\mathbf{z}$$

  - Take a p norm

$$||\mathbf{A}\mathbf{z}||_p = \rho(\mathbf{A})||\mathbf{z}||_p$$

    or

$$\rho(\mathbf{A}) = \frac{||\mathbf{A}\mathbf{z}||_p}{||\mathbf{z}||_p} \leq \max_{x \neq 0} \frac{||\mathbf{A}\mathbf{x}||_p}{||\mathbf{x}||_p} = ||\mathbf{A}||_p$$

2.2. Finite Precision Arithmetic

- Reading: Trefethen and Bau (1997), Lectures 12-15
- Real numbers are approximated by floating point numbers
- *Real number:*

$$x = \pm d_0.d_1 d_2 \cdots d_{t-1} d_t \cdots \times \beta^e \tag{1a}$$

  β The base of the number system
  d_i β-digits, $0 \le d_i < \beta$
  $d_0 \neq 0$ unless $x = 0$
  e Exponent

- *Floating point number:*

$$x \approx fl(x) = \pm d_0.d_1 d_2 \cdots d_{t-1} \times \beta^e \tag{1b}$$

  t Precision
  e $m \le e \le M$

  – Approximate x by
    Chopping: Set $d_t = d_{t+1} = \cdots = 0$
    Rounding: Add $\mathrm{sgn}(x)\beta/2 \times \beta^{e-t}$ to x and chop
  – The approximation satisfies

$$fl(x) = x(1+\epsilon), \qquad |\epsilon| \le u \tag{2a}$$

  – where u is the *unit roundoff error*

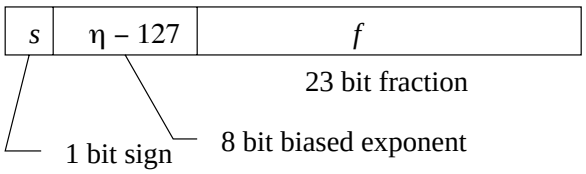
$$u = \begin{cases} \beta^{1-t}, & \text{with chopping} \\ \beta^{1-t}/2, & \text{with rounding} \end{cases} \tag{2b}$$

  – The *relative error* is

$$\frac{|fl(x) - x|}{|x|} \le u \tag{2c}$$

Floating Point Standards

- IEEE binary ($\beta = 2$) single-precision standard



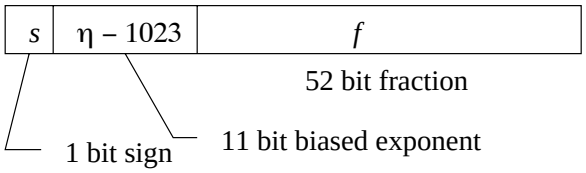
  - Represented number: $(-1)^s 2^{\eta-127}(1+f)$
    - s Sign (0 positive, 1 negative)
    - η Displaced exponent
    - f Fraction: bits $d_1 : d_t$ (d_0 is set to 1 and not stored)

Decimal	s	Exponent	Fraction
$+0$	0	00000000	00000000000000000000000
-0	1	00000000	00000000000000000000000
$+1$	0	01111111	00000000000000000000000
-1	1	01111111	00000000000000000000000
2^{-126}	0	00000001	00000000000000000000000
$2^{127}(2-2^{-23})$	0	11111110	11111111111111111111111
$+\infty$	0	11111111	00000000000000000000000
$-\infty$	1	11111111	00000000000000000000000
NaN	?	11111111	Anything but all zeros
Denormal	?	00000000	Anything but all zeros

  - *Underflow threshold*: $2^{-126} \approx 1.17 \times 10^{-38}$
  - *Overflow threshold*: $2^{127}(2-2^{-23}) \approx 2^{128} \approx 3.40 \times 10^{38}$
  - *NaN*: not a number
  - *Denormal*: un-normalized number ($d_0 = 0$)
  - $u = 2^{-24} \approx 5.96 \times 10^{-8}$

Floating Point Standards

- IEEE binary double-precision standard



  - Represented number: $(-1)^s 2^{\eta-1023}(1+f)$
  - *Underflow threshold*: $2^{-1022} \approx 2.23 \times 10^{-308}$
  - *Overflow threshold*: $2^{1023}(2-2^{-52}) \approx 2^{1024} \approx 1.80 \times 10^{308}$
  - $u = 2^{-53} \approx 1.11 \times 10^{-16}$

Stability

- Assume that an operation $\odot$ $(+, -, *, /, \sqrt{\ })$ satisfies

$$fl(a \odot b) = a \odot b(1 + \epsilon), \qquad |\epsilon| \leq u \tag{3a}$$

  - The *relative error* is

$$\frac{|fl(a \odot b) - a \odot b|}{|a \odot b|} \leq u \tag{3b}$$

- Analyze the roundoff error in computing a dot product

```
s = 0;
for k = 1:n
   s = s + x(k)*y(k);
end
```

- Let

$$s_p = fl\left(\sum_{k=1}^{p} x_k y_k\right)$$

- From the algorithm

$$s_p = (s_{p-1} + x_p y_p(1 + \delta_p))(1 + \epsilon_p), \qquad |\epsilon_j|, |\delta_j| \leq u$$

- With $p = 1$

$$s_1 = x_1 y_1(1 + \delta_1), \qquad |\delta_1| \leq u$$

  - Write this as

$$s_1 = x_1 y_1(1 + \delta_1)(1 + \epsilon_1), \qquad \epsilon_1 = 0$$

- With $p = 2$

$$s_2 = (s_1 + x_2 y_2(1 + \delta_2))(1 + \epsilon_2)$$

or

$$s_2 = x_1 y_1(1 + \delta_1)(1 + \epsilon_1)(1 + \epsilon_2) + x_2 y_2(1 + \delta_2)(1 + \epsilon_2)$$

Roundoff Error in a Dot Product

- With $p = 3$

$$s_3 = x_1y_1(1+\delta_1)(1+\epsilon_1)(1+\epsilon_2)(1+\epsilon_3)$$
$$+x_2y_2(1+\delta_2)(1+\epsilon_2)(1+\epsilon_3)$$
$$+x_3y_3(1+\delta_3)(1+\epsilon_3)$$

- With $p = n$

$$fl(\mathbf{x}^T\mathbf{y}) = s_n = \sum_{k=1}^{n} x_k y_k(1+\gamma_k) \tag{4a}$$

  where

$$1+\gamma_k = (1+\delta_k)\prod_{j=k}^{n}(1+\epsilon_j) \tag{4b}$$

- Expand (4b)

$$1+\gamma_k = 1+\delta_k+\sum_{j=k}^{n}\epsilon_j + O(u^2)$$

- Since $|\epsilon_j|$, $|\delta_j| \leq u$, $j = 1:n$

$$|\gamma_k| \leq (n-k+1)u + O(u^2)$$

  – Set $k = 1$ for simplicity

$$|\gamma_k| \leq nu + O(u^2)$$

Roundoff Error in a Dot Product

- Use (4a)

$$|fl(\mathbf{x}^T\mathbf{y}) - \mathbf{x}^T\mathbf{y}| \leq \sum_{k=1}^{n} |x_k y_k|(nu + O(u^2))$$

  - Let $|\mathbf{y}|$ be a vector with elements $|y_k|$, $k = 1:n$
  - Write

$$\sum_{k=1}^{n} |x_k y_k| := |\mathbf{x}|^T|\mathbf{y}|$$

- Then

$$|fl(\mathbf{x}^T\mathbf{y}) - \mathbf{x}^T\mathbf{y}| \leq nu|\mathbf{x}|^T|\mathbf{y}| + O(u^2)$$

- Dotting the is and crossing the ts

  - There is a $C \approx 1$

$$\begin{aligned} nu|\mathbf{x}|^T|\mathbf{y}| + O(u^2) &= nu|\mathbf{x}|^T|\mathbf{y}|(1 + O(u)) \\ &\leq Cnu|\mathbf{x}|^T|\mathbf{y}| \end{aligned}$$

  - Golub and van Loan (1996): $C = 1.01$ if $nu < 0.01$

- We have

$$\frac{|fl(\mathbf{x}^T\mathbf{y}) - \mathbf{x}^T\mathbf{y}|}{|\mathbf{x}^T\mathbf{y}|} \leq Cnu\frac{|\mathbf{x}|^T|\mathbf{y}|}{|\mathbf{x}^T\mathbf{y}|} \tag{5}$$

  - The relative error may not be small since $|\mathbf{x}^T\mathbf{y}|$ can be much less than $|\mathbf{x}|^T|\mathbf{y}|$

Backward Error Analysis

- Prior analysis is called *forward error analysis*
  - Relate rounding errors to the solution
- *Backward error analysis:*
  - Relate rounding errors to the original data
- Consider the dot-product computation
  - Write (4a) as

$$fl(\mathbf{x}^T\mathbf{y}) = \hat{\mathbf{x}}^T\hat{\mathbf{y}} \tag{6a}$$

    * $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$ are perturbed values of $\mathbf{x}$, $\mathbf{y}$
    * The rounded dot product $fl(\mathbf{x}^T\mathbf{y})$ is the exact dot product $\hat{\mathbf{x}}^T\hat{\mathbf{y}}$ of perturbed vectors
  - Using (4a)

$$\hat{\mathbf{x}} = \begin{bmatrix} x_1\sqrt{1+\gamma_1} \\ x_2\sqrt{1+\gamma_2} \\ \vdots \\ x_n\sqrt{1+\gamma_n} \end{bmatrix}, \qquad \hat{\mathbf{y}} = \begin{bmatrix} y_1\sqrt{1+\gamma_1} \\ y_2\sqrt{1+\gamma_2} \\ \vdots \\ y_n\sqrt{1+\gamma_n} \end{bmatrix} \tag{6b}$$

  - Let $\boldsymbol{\delta x}$, $\boldsymbol{\delta y}$ be perturbations, i.e.,

$$\hat{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta x}, \qquad \hat{\mathbf{y}} = \mathbf{y} + \boldsymbol{\delta y}$$

Backward Error Analysis

- Observe that

$$\sqrt{1+\gamma_k} = 1 + \gamma_k/2 + O(\gamma_k^2)$$

- Thus,

$$\boldsymbol{\delta x} = \begin{bmatrix} x_1(\gamma_1/2 + O(\gamma_1^2)) \\ x_2(\gamma_2/2 + O(\gamma_2^2)) \\ \vdots \\ x_n(\gamma_n/2 + O(\gamma_n^2)) \end{bmatrix}$$

- But $|\gamma_k| \leq nu + O(u^2)$, so

$$|\boldsymbol{\delta x}| \leq \frac{nu}{2}|\mathbf{x}| + O(u^2) \tag{7a}$$

and

$$|\boldsymbol{\delta y}| \leq \frac{nu}{2}|\mathbf{y}| + O(u^2) \tag{7b}$$

- These bounds are very conservative

2.3. Factorizations

- Reading: Trefethen and Bau (1997), Lecture 20
- Solve the $n \times n$ linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1}$$

  by Gaussian elimination
  - Gaussian elimination is a direct method
    - The solution is found after a finite number of operations
  - Iterative methods find a solution as the number of operations tend to infinity
    - Iteration is superior for large sparse systems
- Gaussian elimination is a factorization of $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{L}\mathbf{U} \tag{2a}$$

  - $\mathbf{L}$ is unit lower triangular and $\mathbf{U}$ is upper triangular

LU Factorization

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \tag{2b}$$

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ l_{31} & l_{32} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix}, \qquad \mathbf{U} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ & 0 & & u_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix} \tag{2c}$$

- Once $\mathbf{L}$ and $\mathbf{U}$ have been determined

$$\mathbf{Ax} = \mathbf{LUx} = \mathbf{b}$$

  - Let

$$\mathbf{Ux} = \mathbf{y} \tag{3a}$$

  - Then

$$\mathbf{Ly} = \mathbf{b} \tag{3b}$$

  - Solve (3b) for $\mathbf{y}$ by forward substitution and (3a) for $\mathbf{x}$ by backward substitution

LU Factorization

- *Classical Gaussian elimination*
  - Use row saxpys to reduce $\mathbf{A}$ to upper triangular form $\mathbf{U}$
  - Show that these row operations are the product of $\mathbf{A}$ and lower triangular matrices $\mathbf{L}_k$, $k = 1 : n-1$, i.e.,

$$\mathbf{L}_{n-1} \cdots \mathbf{L}_2 \mathbf{L}_1 \mathbf{A} = \mathbf{U} \tag{4a}$$

  - Thus

$$\mathbf{L} = \mathbf{L}_1^{-1} \mathbf{L}_2^{-1} \cdots \mathbf{L}_{n-1}^{-1} \tag{4b}$$

  - cf. Trefethen and Bau (1997), Lecture 20

- We use a direct factorization
  - Multiply the first row of $\mathbf{L}$ by $\mathbf{U}$ to obtain the first row of $\mathbf{A}$

$$a_{1j} = (1)u_{1j}, \qquad j = 1 : n$$

    * Thus

$$u_{1j} = a_{1j}, \qquad j = 1 : n$$

  - Multiply $\mathbf{L}$ by the first column of $\mathbf{U}$ to obtain the first column of $\mathbf{A}$

$$l_{i1} u_{11} = a_{i1}, \qquad i = 2 : n$$

    or

$$l_{i1} = a_{i1}/u_{11}, \qquad i = 2 : n$$

    * Redundant information with $i = 1$ is not written
    * The algorithm fails if $u_{11} = a_{11} = 0$

LU Factorization

- Continuing
  - Multiply the second row of **L** by **U** to obtain the second row of **A**

$$l_{21}u_{1j} + u_{2j} = a_{2j}, \qquad j = 2 : n$$

    * Thus,

$$u_{2j} = a_{2j} - l_{21}u_{1j}, \qquad j = 2 : n$$

  - The general procedure is

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj}, \qquad j = i : n, \tag{5a}$$

$$l_{ji} = \frac{1}{u_{ii}}(a_{ji} - \sum_{k=1}^{i-1} l_{jk}u_{ki}), \qquad j = i+1 : n, \quad i = 1 : n-1 \tag{5b}$$

    * The sums are zero when the lower limit exceeds the upper one
    * The procedure fails if $u_{ii} = 0$ for some i
    * The matrices **L** and **U** can be stored in the locations used for the same elements of **A**

LU Factorization

```
function A = lufactor(A)
% lufactor: Compute the LU decomposition of an n-by-n
% matrix A by direct factorization. On return, L is
% stored in the lower triangular part of A and U is
% stored in the upper triangular part.

   [n n] = size(A);
%                       Loop over the rows
   for i = 1: n - 1
%                       Calculate the i th column of L
      for j = i + 1: n
         for k = 1: i - 1
            A(j,i) = A(j,i) - A(j,k)*A(k,i);
         end
         A(j,i) = A(j,i)/A(i,i);
      end
%                       Calculate the (i + 1) th row of U
      for j = i+1: n
         for k = 1: i
            A(i+1,j) = A(i+1,j) - A(i+1,k)*A(k,j);
         end
      end
   end
```

LU Factorization

- **Observe:**
  i. The first row of $\mathbf{U}$ needs no explicit calculation
  ii. The 1s on the diagonal of $\mathbf{L}$ are not stored
  iii. MATLAB Loops do not execute when the lower index exceeds the upper one
  iv. Classical Gausian elimination corresponds to a different loop ordering
    – cf. Trefethen and Bau (1997), Lecture 20

- *Example 1.* Factor

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 0 & -4 \\ -1 & 0 & 6 & 2 \\ 3 & -2 & -25 & 0 \\ -2 & -3 & 4 & 4 \end{bmatrix}$$

  – $i = 1$: Calculate the first column of $\mathbf{L}$ and the second row of $\mathbf{U}$

$$\mathbf{A} \Rightarrow \begin{bmatrix} 1 & 2 & 0 & -4 \\ -1 & 2 & 6 & -2 \\ 3 & -2 & -25 & 0 \\ -2 & -3 & 4 & 4 \end{bmatrix}$$

LU Factorization

– $i = 2$: Calculate the second column of $\mathbf{L}$ and the third row of $\mathbf{U}$

$$\mathbf{A} \Rightarrow \begin{bmatrix} 1 & 2 & 0 & -4 \\ -1 & 2 & 6 & -2 \\ 3 & -4 & -1 & 4 \\ -2 & 1/2 & 4 & 4 \end{bmatrix}$$

– $i = 3$: Calculate the third column of $\mathbf{L}$ and the fourth row of $\mathbf{U}$

$$\mathbf{A} \Rightarrow \begin{bmatrix} 1 & 2 & 0 & -4 \\ -1 & 2 & 6 & -2 \\ 3 & -4 & -1 & 4 \\ -2 & 1/2 & -1 & 1 \end{bmatrix}$$

- The $\mathbf{L}$ and $\mathbf{U}$ factors

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 3 & -4 & 1 & 0 \\ -2 & 1/2 & -1 & 1 \end{bmatrix}, \qquad \mathbf{U} = \begin{bmatrix} 1 & 2 & 0 & -4 \\ 0 & 2 & 6 & -2 \\ 0 & 0 & -1 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

L Exposed

- Elementary transformations

$$\mathbf{L}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ -3 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{L}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 4 & 1 & 0 \\ 0 & -1/2 & 0 & 1 \end{bmatrix},$$

$$\mathbf{L}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

  - $\mathbf{L}_k$ is the identity matrix with its $k\ th$ column replaced by the $k\ th$ column of $\mathbf{L}$ with the negative of the subdiagonal elements

$$\mathbf{L}^{-1} = \mathbf{L}_3\mathbf{L}_2\mathbf{L}_1 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 2.5 & 3.5 & 1 & 1 \end{bmatrix}$$

Forward and Backward Substitution

- Solution of (3b) by forward substitution

$$y_i = b_i - \sum_{j=1}^{i-1} l_{ij} y_j \tag{6a}$$

```
function y = forward(L, b)
% forward: Solution of a n-by-n lower triangular system
% Ly = b by forward substitution.

   [n n] = size(L);
   y(1) = b(1);
   for i = 2:n
      y(i) = b(i) - dot(L(i,1:i-1)', y(1:i-1));
   end
```

- Solve (3a) by backward substitution

$$x_i = \frac{1}{u_{ii}}[y_i - \sum_{j=i+1}^{n} u_{ij} x_j] \tag{6b}$$

```
function x = backward(U, y)
% backward: Solution of a n-by-n upper triangular system
% Ux = y by backward substitution.

   [n n] = size(U);
   x(n) = y(n)/U(n,n);
   for i = n - 1:-1:1
      x(i) = (y(i) - dot(U(i,i+1:n)', x(i+1:n)))/U(i,i);
   end
```

Forward and Backward Substitution

- *Example 2.* Solve the linear system $\mathbf{Ax} = \mathbf{b}$ for $\mathbf{x}$ when $\mathbf{A}$ is as given in Example 1 and

$$\mathbf{b} = [-1,\ 7,\ -24,\ 3]^T$$

  - Using the forward substitution algorithm

$$\mathbf{y} = [-1,\ 6,\ 3,\ 1]^T$$

  - Using backward substitution

$$\mathbf{x} = [1,\ 1,\ 1,\ 1]^T$$

Operation Count

- *Factorization:* (cf. p. 5)
  - Inner column loop:
    - $i-1$ multiplications and subtractions
    - 1 division
  - Inner row loop: i multiplications and subtractions
  - The summations on j give
    - $(2i-1)(n-i)$ multiplications and subtractions
    - $n-i$ divisions
  - The summations on i give
    - Multiplications and subtractions:

$$\sum_{i=1}^{n-1}(2i-1)(n-i) = \frac{n(n-1)(2n-1)}{6}$$

    - Divisions:

$$\sum_{i=1}^{n-1}(n-1) = \frac{n(n-1)}{2}$$

    - Formulas (1.2.1, 2) were used to obtain the result
  - A floating point operation (FLOP) is any $+, -, *, /, \sqrt{\ }, \cdots$
    - For large n the factorization has about $2n^3/3$ FLOPs

Operation Count

- *Forward Substitution*: (cf. p. 9)
  - $i-1$ multiplications and additions/subtractions in the dot product
    - Total multiplications and additions/subtractions:

$$\sum_{i=2}^{n}(i-1) = \frac{n(n-1)}{2}$$

- *Backward Substitution:*
  - $n(n-1)/2$ multiplications and additions/subtractions and n divisions
- *Forward and Backward Substitution*:
  - For large n, there are about $2n^2$ FLOPs
- Factorization dominates forward and backward substitution for large n

2.4. Pivoting

- Reading: Trefethen and Bau (1997), Lecture 21
- The Gaussian factorization and backward substitution fail when $u_{ii} = 0$, $i = 1 : n$
  - The system need not be singular, e.g.,

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

  - The factorization can proceed upon a row interchange
    - i.e., upon exchanging equations
  - Small divisors with finite-precision arithmetic will also cause problems
- *Example 1.* Consider three-decimal floating-point arithmetic ($\beta = 10$, $t = 3$)

$$\begin{bmatrix} 1.00 \times 10^{-4} & 1.00 \\ 1.00 & 1.00 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.00 \\ 2.00 \end{bmatrix}$$

  - The exact solution is $x_1 = 10000/9999 = 1.00010$ $x_2 = 9998/9999 = 0.99990$
  - Factorization:

$$\mathbf{L} = \begin{bmatrix} 1.00 & 0.00 \\ 1.00 \times 10^4 & 1.00 \end{bmatrix}$$

$$\mathbf{U} = \begin{bmatrix} 1.00 \times 10^{-}4 & 1.00 \\ 0.00 & -1.00 \times 10^4 \end{bmatrix}$$

The need for Pivoting

  - Forward substitution

$$\begin{bmatrix} 1.00 & 0.00 \\ 1.00 \times 10^4 & 1.00 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1.00 \\ 2.00 \end{bmatrix}$$

    or $y_1 = 1.00$, $y_2 = -1.00 \times 10^4$

  - Backward substitution

$$\begin{bmatrix} 1.00 \times 10^{-4} & 1.00 \\ 0.00 & -1.00 \times 10^4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.00 \\ -1.00 \times 10^4 \end{bmatrix}$$

  - Thus, $x_2 = 1.00$, $x_1 = 0.00$

- This is awful!
- Interchanging rows

$$\begin{bmatrix} 1.00 & 1.00 \\ 0.00 & 1.00 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2.00 \\ 1.00 \end{bmatrix}$$

  - The system is in upper triangular form ($l_{21} = 0.00$), so $x_2 = 1.00$ and $x_1 = 1.00$
    * This is correct to three digits

Pivoting Strategies

- Row pivoting (partial pivoting): at stage i of the outer loop of the factorization (cf. Section 2.3, p. 5)
  1. Find r such that $|a_{ri}| = \max_{i \le k \le n} |a_{ki}|$
  2. Interchange rows i and r
- Column pivoting: Proceed as row pivoting but interchange columns
  - Column pivoting requires reordering the unknowns
  - Column pivoting does not work well with direct factorization
- Complete pivoting: Choose r and c such that
  1. Find $r,\ c$ such that $|a_{rc}| = \max_{i \le k,l \le n} |a_{kl}|$
  2. Interchange rows i and r and columns i and c
- Complete pivoting is less common than partial pivoting
  - Have to search a larer space
  - Have to reorder unknowns
- Row pivoting is usually adequate
- Row, column, and complete pivoting have $l_{ij} \le 1$, $i \ne j$

Scaled Partial Pivoting

- The equations and unknowns may be scaled differently
- *Example 2.* Multiply the first row of Example 1 by 10^5

$$\begin{bmatrix} 1.00 \times 10^1 & 1.00 \times 10^5 \\ 1.00 & 1.00 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.00 \times 10^5 \\ 2.00 \end{bmatrix}$$

  - Row pivoting would choose the first row as a pivot
    - This yields the result $x_2 = 1.00$ and $x_1 = 0.00$
- There is no general solution to this problem
  - One strategy is to "equilibrate" the matrix
    - Select all elements to have the same magnitude
- Scaled partial pivoting:
  - Select row pivots relative to the size of the row
    1. Before factorization select scale factors

$$s_i = \max_{1 \le j \le n} |a_{ij}|, \qquad i = 1 : n$$

    2. At stage i of the factorization, select r such that

$$\left| \frac{a_{ri}}{s_r} \right| = \max_{i \le k \le n} \left| \frac{a_{ki}}{s_k} \right|$$

    3. Interchange rows k and i

Factorization with Pivoting

- Gaussian elimination with partial pivoting always finds factors $\mathbf{L}$ and $\mathbf{U}$ of a nonsingular matrix
  - Neglecting roundoff errors
- **Theorem 1**: For any $n \times n$ matrix $\mathbf{A}$ of rank n, there is a reordering of rows such that

$$\mathbf{PA} = \mathbf{LU} \tag{1}$$

  where $\mathbf{P}$ is a permutation matrix that reorders the rows of $\mathbf{A}$
  - A permutation matrix is an identity matrix with its rows or columns interchanged
  - *Proof:* cf. Golub and Van Loan (1996), Section 3.4.4 □

Scaled Partial Pivoting

- It is not necessary to store the permutation matrix or rearrange the rows of $\mathbf{A}$
  - Row interchanges can be recorded in a vector $\mathbf{p}$
- *Example 3.* Consider

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 2 \\ 1 & -1 & 1 \\ 2 & 3 & -1 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 2 \\ 1 \\ 4 \end{bmatrix}$$

  - Compute the scale factors and initialize the interchange vector

$$\mathbf{s} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}, \qquad \mathbf{p} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

    * $p_i = i$, $i = 1 : n$, implies no rows have been interchanged
  - $i = 1$:
    * Scale factor: $|a_{11}/s_1| = 1/2, |a_{21}/s_2| = 1/1, |a_{31}/s_3| = 2/3$
    * The second row is the pivot

$$\mathbf{s} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}, \qquad \mathbf{A} \Rightarrow \begin{bmatrix} 1 & 0 & 1 \\ 1 & -1 & 1 \\ 2 & 5 & -3 \end{bmatrix}, \qquad \mathbf{p} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$$

  - $\mathbf{p}$ records the implicit row interchange

Scaled Partial Pivoting

– $i = 2$:

  * Scale factors: $|a_{12}/s_1| = 0/2$, $|a_{32}/s_3| = 5/3$
  * The third row is the pivot

$$\mathbf{s} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}, \qquad \mathbf{A} \Rightarrow \begin{bmatrix} 1 & 0 & 1 \\ 1 & -1 & 1 \\ 2 & 5 & -3 \end{bmatrix}, \qquad \mathbf{p} = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

– Construct $\mathbf{P}$, $\mathbf{L}$, and $\mathbf{U}$

  * $p_1 = 2$, so Row 1 of $\mathbf{L}$ and $\mathbf{U}$ is Row 2 of $\mathbf{A}$
  * $p_2 = 3$, so Row 2 of $\mathbf{L}$ and $\mathbf{U}$ is Row 3 of $\mathbf{A}$
  * $p_3 = 1$, so Row 3 of $\mathbf{L}$ and $\mathbf{U}$ is Row 1 of $\mathbf{A}$

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \qquad \mathbf{U} = \begin{bmatrix} 1 & -1 & 1 \\ 0 & 5 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

  * From $\mathbf{p}$, Row 1 of $\mathbf{P}$ is Row 2 of the identity matrix
  * Row 2 of $\mathbf{P}$ is Row 3 of the identity matrix
  * Row 3 of $\mathbf{P}$ is Row 1 of the identity matrix

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Scaled Partial Pivoting

- Check that $\mathbf{PA} = \mathbf{LU}$

$$\mathbf{PA} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & -1 & 2 \\ 1 & -1 & 1 \\ 2 & 3 & -1 \end{bmatrix} =$$

$$\mathbf{LU} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 1 \\ 0 & 5 & -3 \\ 0 & 0 & 1 \end{bmatrix} =$$

- Forward and backward substition.

$$\mathbf{PAx} = \mathbf{Pb}$$

  - Use (1)

$$\mathbf{LUx} = \mathbf{Pb}$$

- Forward substitution: $\mathbf{Ly} = \mathbf{Pb}$
  - $\mathbf{p}_1 = 2$, so $y_1 = b_2 = 1$ or $y_1 = 1$
  - $\mathbf{p}_2 = 3$, so $2y_1 + y_2 = b_3 = 4$ or $y_2 = 2$
  - $\mathbf{p}_3 = 1$, so $y_1 + y_3 = b_1 = 2$ or $y_3 = 1$
- Backward substitution: $\mathbf{Ux} = \mathbf{y}$
  - $\mathbf{p}_3 = 1$, so $x_3 = y_3 = 1$ or $x_3 = 1$
  - $\mathbf{p}_2 = 3$, so $5x_2 - 3x_3 = y_2 = 2$ or $x_2 = 1$
  - $\mathbf{p}_1 = 2$, so $x_1 - x_2 + x_3 = y_1 = 1$ or $x_1 = 1$

LU Factorization

```
function [Ap] = plufactor(A)
% plufactor: Factor the n-by-n matrix A into LU. On return, L – I
% is stored in the lower triangular part of A and U
% is stored in the upper triangular part. The vector p
% stores the permuted row indices using scaled partial pivoting.

   [n n] = size(A);
%                          Initialize p and compute the scale vector s
   for i = 1; n
      s(i) = norm(A(i,1:n), inf);
      p(i) = i;
   end
%                          Loop over the rows
   for i = 1: n - 1
%                          Find the best pivot row
      colmax = 0;
      for k = i: n
         srow = abs(A(p(k),i))/s(p(k));
         if colmax < srow;
            colmax = srow;
            index = k;
         end
      end
      temp = p(i);
      p(i) = index;
      p(index) = temp;
%                          Calculate the i th column of L
      for j = i + 1: n
         for k = 1: i - 1
            A(p(j),i) = A(p(j),i) - A(p(j),k)*A(p(k),i);
         end
         A(p(j),i) = A(p(j),i)/A(p(i),i);
      end
%                          Calculate the (i + 1) th row of U
      for j = i+1: n
         for k = 1: i
            A(p(i+1),j) = A(p(i+1),j) - A(p(i+1),k)*A(p(k),j);
         end
      end
   end
```

Forward and Backward Substitution

```
function y = pforward(L, b, p)
% pforward: Solution of a n-by-n lower triangular system
% Ly = Pb by forward substitution. Row permutations have been
% stored in the vector p.

   [n n] = size(L);
   y(1) = b(p(1));
   for i = 2:n
      y(i) = b(p(i)) - dot(L(p(i),1:i-1)', y(1:i-1));
   end
```

```
function x = backward(U, y, p)
% pbackward: Solution of a n-by-n upper triangular system
% Ux = y by backward substitution. Row permuations have been
% stored in the vector p.

   [n n] = size(U);
   x(n) = y(n)/U(p(n),n);
   for i = n - 1: -1: 1
      x(i) = (y(i) - dot(U(p(i),i+1:n)', x(i+1:n)))/U(p(i),i);
   end
```

- **Note**:

  i. No attempt has been made to check for failure

     – **A** is singular if $colmax = 0$ or $s_i = 0$ for some i

  ii. The MATLAB function *norm* computes vector and matrix norms.

2.5. Accuracy Estimation

- Reading: Trefethen and Bau (1997), Lecture 22
- *Accuracy:*
  - With roundoff, how accurate is a solution obtained by Gaussian elimination?
  - How sensitive is a solution to perturbations introduced by round off error (stability)?
  - How can we appraise the accuracy of a computed solution?
  - Can the accuracy of a computed solution be improved?
- Let $\hat{\mathbf{x}}$ be a computed solution of

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1}$$

  - Compute the residual

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\hat{\mathbf{x}} \tag{2}$$

  - Is $\mathbf{r}$ a good measure of the accuracy of $\hat{\mathbf{x}}$?
- *Example 1.* Solve (1) with

$$\mathbf{A} = \begin{bmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 0.8642 \\ 0.1440 \end{bmatrix}$$

  - An approximate solution is $\hat{\mathbf{x}} = [0.9911, -0.4870]^T$
  - The residual is $\mathbf{r} = [-10^{-8}, 10^{-8}]^T$
  - Exact solution:
  - $\det(\mathbf{A}) = 10^{-8}$

Sensitivity

- *Example 2.* Consider

$$\mathbf{A} = \begin{bmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{bmatrix}$$

  - $\mathbf{A}$ is symmetric and positive definite with $\det(\mathbf{A}) = 1$
  - Exact solutions

$\mathbf{b}^T$	$\mathbf{x}^T$
$[23, 32, 33, 31]$	$[1, 1, 1, 1]$
$[22.9, 32.1, 32.9, 31.1]$	$[-7.2, 6, 2.9, 0.1]$
$[22.99, 32.01, 32.99, 31.01]$	$[0.18, 1.5, 1.19, 0.89]$

Perturbations

- **Definition 1**: A matrix is *ill-conditioned* if small changes in $\mathbf{A}$ or $\mathbf{b}$ produce large changes in the solution.
  - Suppose that $\mathbf{b}$ is perturbed to $\boldsymbol{b} + \boldsymbol{\delta b}$
  - The solution $\boldsymbol{x} + \boldsymbol{\delta x}$ of the perturbed system would satisfy
$$\mathbf{A}(\mathbf{x} + \boldsymbol{\delta x}) = \mathbf{b} + \boldsymbol{\delta b}$$
  - Estimate the magnitude of $\boldsymbol{\delta x}$
$$\boldsymbol{\delta x} = \mathbf{A}^{-1}\boldsymbol{\delta b}$$
  - Taking a norm
$$||\boldsymbol{\delta x}|| = ||\mathbf{A}^{-1}\boldsymbol{\delta b}|| \leq ||\mathbf{A}^{-1}||||\boldsymbol{\delta b}||$$
  - In addition
$$||\mathbf{b}|| = ||\mathbf{A}\mathbf{x}|| \leq ||\mathbf{A}||^{|}|\mathbf{x}||$$
  - Multiplying the two inequalities
$$||\boldsymbol{\delta x}||||\mathbf{b}|| \leq ||\mathbf{A}||||\mathbf{A}^{-1}||||\boldsymbol{\delta b}||^{|}|\mathbf{x}||$$
    or
$$\frac{||\boldsymbol{\delta x}||}{||\mathbf{x}||} \leq \kappa(\mathbf{A})\frac{||\boldsymbol{\delta b}||}{||\mathbf{b}||} \tag{3a}$$
    where
$$\kappa = ||\mathbf{A}||||\mathbf{A}^{-1}|| \tag{3b}$$
    * $\kappa(\mathbf{A})$ is called the *condition number*

The Condition Number

- **Note**:

  i. The condition number relates the relative uncertainty of the data ($||\boldsymbol{\delta b}||/||\mathbf{b}||$) to the relative uncertainty of the solution ($||\boldsymbol{\delta x}||/||\mathbf{x}||$)

  ii. If $\kappa(\mathbf{A})$ is large, small changes in $\mathbf{b}$ may result in large changes in $\mathbf{x}$

  iii. If $\kappa(\mathbf{A})$ is large, $\mathbf{A}$ is *ill-conditioned*

  iv. The bound (3a) is sharp: there are $\mathbf{b}$s for which equality holds

  v. The $\det(\mathbf{A})$ has nothing to do with conditioning. The matrix

$$\begin{bmatrix} 10^{-30} & 0 \\ 0 & 10^{-30} \end{bmatrix}$$

  is well-conditioned

- Repeat the analysis for a perturbation in $\mathbf{A}$

$$(\mathbf{A} + \boldsymbol{\delta A})(\mathbf{x} + \boldsymbol{\delta x}) = \mathbf{b}$$

or

$$\mathbf{A}\mathbf{x} + \mathbf{A}\boldsymbol{\delta x} + \boldsymbol{\delta A}(\mathbf{x} + \boldsymbol{\delta x}) = \mathbf{b}$$

or

$$\boldsymbol{\delta x} = -\mathbf{A}^{-1}\boldsymbol{\delta A}(\mathbf{x} + \boldsymbol{\delta x})$$

or

$$||\boldsymbol{\delta x}|| \leq ||\mathbf{A}^{-1}||||\boldsymbol{\delta A}||||\mathbf{x} + \boldsymbol{\delta x}||$$

or

$$\frac{||\boldsymbol{\delta x}||}{||\mathbf{x} + \boldsymbol{\delta x}||} \leq \kappa(\mathbf{A})\frac{||\boldsymbol{\delta A}||}{||\mathbf{A}||} \tag{3c}$$

Condition Numbers

- *Example 3.* The matrix $\mathbf{A}$ of Example 1 and its inverse are

$$\mathbf{A} = \begin{bmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{bmatrix}, \qquad \mathbf{A}^{-1} = 10^8 \begin{bmatrix} 0.1441 & -0.8648 \\ -0.2161 & 1.2969 \end{bmatrix}$$

  – Calculate

$$||\mathbf{A}||_\infty = 2.1617, \qquad ||\mathbf{A}^{-1}||_\infty = 1.5130 \times 10^8$$

$$\kappa_\infty(\mathbf{A}) = 3.2707 \times 10^8$$

- *Example 4.* The matrix $\mathbf{A}$ of Example 2 and its inverse are

$$\mathbf{A} = \begin{bmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{bmatrix}, \qquad \mathbf{A}^{-1} = \begin{bmatrix} 68 & -41 & -17 & 10 \\ -41 & 25 & 10 & -6 \\ -17 & 10 & 5 & -3 \\ 10 & -6 & -3 & 2 \end{bmatrix}$$

  and

$$||\mathbf{A}||_\infty = 33, \qquad ||\mathbf{A}^{-1}||_\infty = 136$$

$$\kappa_\infty(\mathbf{A}) = 4488$$

Stability of Gaussian Elimination

- How do rounding errors propagate in Gaussian elimination?
  - Let's discuss this without explicit treatment of pivoting
    - Assume that $\mathbf{A}$ has its rows arranged so that pivoting is unnecessary

- **Theorem 1**: Let $\mathbf{A}$ be an $n \times n$ real matrix of floating-point numbers on a computer with unit round-off error u. Assume that no zero pivots are encountered. Let $\hat{\mathbf{L}}$ and $\hat{\mathbf{U}}$ be the computed triangular factors of $\mathbf{A}$ and let $\hat{\mathbf{y}}$ and $\hat{\mathbf{x}}$ be the computed solutions of

$$\hat{\mathbf{L}}\mathbf{y} = \mathbf{b}, \qquad \hat{\mathbf{U}}\mathbf{x} = \hat{\mathbf{y}} \tag{4a}$$

  Then

$$(\mathbf{A} + \boldsymbol{\delta A})\hat{\mathbf{x}} = \mathbf{b} \tag{4b}$$

  where

$$|\boldsymbol{\delta A}| \leq 3nu|\mathbf{L}||\mathbf{U}| + O(n^2u^2) \tag{4c}$$

  - *Remark 1*: Recall (Section 2.1) that $|\mathbf{A}|$ is a matrix with elements $|a_{ij}|$, $i, j = 1 : n$
  - *Remark 2*: The computed solution $\hat{\mathbf{x}}$ is the exact solution of a system $\mathbf{A} + \boldsymbol{\delta A}$
  - *Remark 3*: Additionally, $\hat{\mathbf{L}}$ and $\hat{\mathbf{U}}$ are the exact factors of a perturbed matrix $\mathbf{A} + \mathbf{E}$

- *Proof:* Follow the arguments used in the round-off error analysis of the dot product of Section 2.1 (cf. Demmel, Section 2.4) $\square$

Growth Factor

- *Remark*: For the infinity, one and Frobenius norms,

$$|| \, |\mathbf{z}| \, || = ||\mathbf{z}||$$

  - Taking a norm of (4c)

$$||\boldsymbol{\delta A}|| \leq 3nu||\mathbf{L}||||\mathbf{U}|| + O(n^2u^2) \tag{5}$$

- We would like to estimate $||\mathbf{L}||$ and $||\mathbf{U}||$
  - With partial pivoting, the elements of $\mathbf{L}$ are bounded by unity
    - Thus, $||\mathbf{L}||_\infty \leq n$
  - Bounding $||\mathbf{U}||$ is much more difficult
    - Define the *growth factor* as

$$\rho = \frac{\max_{i,j} |u|_{ij}}{||\mathbf{A}||_\infty} \tag{6}$$

    - Then $|\mathbf{U}| \leq \rho||\mathbf{A}||$ and

$$||\mathbf{U}||_\infty \leq \rho n||\mathbf{A}||_\infty$$

  - Using the bounds for $||\mathbf{L}||_\infty$ and $||\mathbf{U}||_\infty$ in (5), we find

$$||\boldsymbol{\delta A}||_\infty \leq n^3 u\rho C||\mathbf{A}||_\infty \tag{7}$$

    - The constant C accounts for our "casual" treatment of the $O(n^2u^2)$ term

Growth Factors

- **Note**:

  i. The bound (7) for $\boldsymbol{\delta A}$ is acceptable unless ρ is large

  ii. J.H. Wilkinson (1961) [1] shows that if $|a_{ij}| \leq 1$, $i, j = 1 : n$,

  - $\rho||\mathbf{A}||_\infty \leq 2^{n-1}$ with partial pivoting
  - $\rho||\mathbf{A}||_\infty \leq 1.8n^{0.25 \ln n}$ with complete pivoting
  - The bound with partial pivoting is sharp! Consider

$$\mathbf{A} = \begin{bmatrix} 1 & & & & 1 \\ -1 & 1 & & & 1 \\ -1 & -1 & 1 & & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix}$$

  which has

$$\mathbf{L} = \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ -1 & -1 & 1 & & \\ -1 & -1 & -1 & 1 & \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} 1 & & & & 1 \\ & 1 & & & 2 \\ & & 1 & & 4 \\ & & & 1 & 8 \\ & & & & 16 \end{bmatrix}$$

  Thus, $\rho = 16$. For an $n \times n$ matrix, $\rho = 2^{n-1}$

  iii. In most cases, $\rho||\mathbf{A}||_\infty \leq 8$ with partial pivoting

  iv. Our bounds are conservative. In most cases

$$||\boldsymbol{\delta A}||_\infty \leq nu||\mathbf{A}||_\infty$$

[1] J.H. Wilkinson, "Error analysis of direct methods of matrix inversion," *J. A.C.M.*, **8**, 281-330

Iterative Improvement

- Iterative improvement can enhance the accuracy of a computed solution
  - It works best when double precision arithmetic is available but costs much more than single precision arithmetic
- Let $\hat{\mathbf{x}}$ be an approximate solution of (1).
- Calculate the residual (2)

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\hat{\mathbf{x}} = \mathbf{A}(\mathbf{x} - \hat{\mathbf{x}})$$

- Solve

$$\boldsymbol{A\delta x} = \mathbf{r} \tag{8a}$$

  for $\boldsymbol{\delta x}$ and calculate an "improved" solution

$$\mathbf{x} = \hat{\mathbf{x}} + \boldsymbol{\delta x} \tag{8b}$$

- $\hat{\mathbf{x}}$ cannot be improved unless:
  - The residual is calculated in double precision
  - The original (not the factored) $\mathbf{A}$ is used to calculate $\mathbf{r}$
- The procedure is
  1. Calculate $\mathbf{r} = \mathbf{b} - \mathbf{A}\hat{\mathbf{x}}$ using double precision
  2. Solve $\hat{\mathbf{L}}\mathbf{y} = \mathbf{Pr}$ by forward substitution
  3. Solve $\hat{\mathbf{U}}\boldsymbol{\delta x} = \mathbf{y}$ by backward substitution
  4. $\mathbf{x} = \hat{\mathbf{x}} + \delta\mathbf{x}$

Iterative Improvement

- **Notes:**
  i. Round the double precision residual to single precision
  ii. The cost of iterative improvement:
    - One matrix-by-vector multiplication for the residual and one forward and backward substitution
    - Each cost n^2 multiplications if double precision hardware is available
    - The total cost of $2n^2$ multiplications is small relative to the factorization
  iii. Continue the iteration for further improvement
  iv. Have to save the original $\mathbf{A}$ and the factors $\hat{\mathbf{L}}$ and $\hat{\mathbf{U}}$
  v. Have to do mixed-precision arithmetic

- Accuracy of the improved solution
  - From (8)
$$\boldsymbol{\delta x} := \mathbf{x} - \hat{\mathbf{x}} = \mathbf{A}^{-1}(\mathbf{b} - \mathbf{A}\hat{\mathbf{x}}) = \mathbf{A}^{-1}\mathbf{r}$$
  - Take a norm
$$||\boldsymbol{\delta x}||_\infty \leq ||\mathbf{A}^{-1}||_\infty ||\mathbf{r}||_\infty$$
  - $\hat{\mathbf{x}}$ satisfies (4c), so
$$\mathbf{r} = (\mathbf{A} + \boldsymbol{\delta A})\hat{\mathbf{x}} - \mathbf{A}\hat{\mathbf{x}} = \boldsymbol{\delta A}\hat{\mathbf{x}}$$
  - Take a norm
$$||\mathbf{r}||_\infty \leq ||\boldsymbol{\delta A}||_\infty ||\hat{\mathbf{x}}||_\infty$$

Still Improving

- Combining results

$$||\boldsymbol{\delta x}||_\infty \leq ||\mathbf{A}^{-1}||_\infty ||\boldsymbol{\delta A}||_\infty ||\hat{\mathbf{x}}||_\infty$$

- Using (3b)

$$\frac{||\boldsymbol{\delta x}||_\infty}{||\hat{\mathbf{x}}||_\infty} \leq \kappa(\mathbf{A}) \frac{||\boldsymbol{\delta A}||_\infty}{||\mathbf{A}||_\infty} \tag{9a}$$

- We could estimate $||\boldsymbol{\delta A}||_\infty$ using (7)
  - Golub and Van Loan (1996) suggest

$$||\boldsymbol{\delta A}||_\infty = nu||\mathbf{A}||_\infty$$

  - Trefethen and Bau (1997) suggest $\rho = O(n^{1/2})$

$$\frac{||\boldsymbol{\delta x}||_\infty}{||\hat{\mathbf{x}}||_\infty} \leq nu\kappa(\mathbf{A}) \tag{9b}$$

- If $u \propto \beta^{1-t}$ and $\kappa(\mathbf{A}) \propto \beta^{s-1}$

$$\frac{||\boldsymbol{\delta x}||_\infty}{||\hat{\mathbf{x}}||_\infty} \leq nc\beta^{s-t} \tag{9c}$$

  - c represents constants arising from u and κ

- The computed solution $\hat{\mathbf{x}}$ will have about $t-s$ correct β-digits
  - If s is small, $\mathbf{A}$ is well conditioned and the error is small
  - If s is large, $\mathbf{A}$ is ill-conditioned and the error is large
  - If $s > t$, the error grows relative to $\hat{\mathbf{x}}$

It Don't Get Much Better

- Let $\hat{\mathbf{x}}^{(0)} = \hat{\mathbf{x}}$, $\boldsymbol{\delta x}^{(0)} = \boldsymbol{\delta x}$, and $\hat{\mathbf{x}}^{(1)} = \hat{\mathbf{x}}^{(0)} + \boldsymbol{\delta x}^{(0)}$ be the solution computed by iterative improvement.
- Assume $\boldsymbol{\delta x} \approx \boldsymbol{\delta x}^{(0)}$ and Calculate an estimate of $\kappa(\mathbf{A})$ from (9b) as

$$\frac{1}{nu}\frac{||\boldsymbol{\delta x}^{(0)}||_\infty}{||\hat{\mathbf{x}}^{(0)}||_\infty} \leq \kappa(\mathbf{A})$$

- An empirical result that holds widely is

$$\frac{1}{n}\kappa(\mathbf{A}) \leq \frac{1}{nu}\frac{||\boldsymbol{\delta x}^{(0)}||_\infty}{||\hat{\mathbf{x}}||_\infty} \leq \kappa(\mathbf{A}) \tag{10}$$

- **Theorem 2**: Suppose

$$\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\hat{\mathbf{x}}^{(k)}, \qquad \hat{\mathbf{x}}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta x}^{(k)} \tag{11}$$

  are calculated in double precision and $\boldsymbol{\delta x}^{(k)}$ is the solution of

$$\mathbf{A}\boldsymbol{\delta x}^{(k)} = \mathbf{r}^{(k)}$$

  in single precision. Thus,

$$(\mathbf{A} + \boldsymbol{\delta A}^{(k)})\boldsymbol{\delta x}^{(k)} = \mathbf{r}^{(k)}$$

  If $||\mathbf{A}^{-1}\boldsymbol{\delta A}^{(k)}|| \leq 1/2$ then $||x^{(k)} - \mathbf{x}|| \to 0$ as $k \to \infty$
- **Corollary 2**: If $||\boldsymbol{\delta A}^{(k)}|| \leq nu||\mathbf{A}||$ and $nu\kappa(\mathbf{A}) \leq 1/2$ then $||\mathbf{x}^{(k)} - \mathbf{x}|| \to 0$ as $k \to \infty$
  - *Proof*: cf. G.E. Forsythe and C. Moler (1967), *Computer Solution of Linear Algebraic Systems*, Prentice Hall, Englewood Cliffs.
- *Remark*: Iterative improvement converges rather generally and each iteration gives $t - s$ correct β digits

A Hilbert Matrix

- *Example 5.* An $n \times n$ Hilbert matrix has entries

$$h_{ij} = \frac{1}{i + j - 1}$$

  - With $n = 4$:

$$\mathbf{H} = \begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{bmatrix}$$

  - It is symmetric and positive definite
  - It arises in the polynomial approximation of functions

- Condition numbers of Hilbert matrices

n	$\kappa_2(\mathbf{H})$
2	1.9282e+01
4	1.5514e+04
8	1.5258e+10
16	6.5341e+17

- Solve $\mathbf{Hx} = \mathbf{b}$ with $n = 12$
  - Select $\mathbf{b}$ the first column of the identity matrix
  - $\mathbf{x}$ is the first column of $\mathbf{H}^{-1}$

- Solve by Gaussian elimination with one step of iterative improvement

$$\frac{||\hat{\mathbf{x}}^{(0)} - \mathbf{x}||_2}{||\mathbf{x}||_2} = 0.0784, \qquad \frac{||\hat{\mathbf{x}}^{(1)} - \mathbf{x}||_2}{||\mathbf{x}||_2} = 0.0086$$

3. Special Linear Systems
3.1. Symmetric Positive Definite Systems

- Reading: Trefethen and Bau (1997), Lecture 23
- Simplify Gaussian elimination when $\mathbf{A}$ has special properties
  - Symmetry
  - positive definiteness
  - Banded structure
  - Sparsity
  - Block structure
- **Definition 1:** $\mathbf{A} \in \Re^{n \times n}$ is symmetric if $\mathbf{A}^T = \mathbf{A}$
- **Definition 2:** $\mathbf{A} \in \Re^{n \times n}$ is *positive definite* if $\mathbf{x}^T \mathbf{A} \mathbf{x} > 0$ for all nonzero $\mathbf{x} \in \Re^n$
- If $\mathbf{A}$ is positive definite:
  - and $\mathbf{X} \in \Re^{n \times k}$ has rank k then $\mathbf{X}^T \mathbf{A} \mathbf{X}$ is positive definite
  - then all of its principal submatrices are positive definite
  - then all of its diagonal entries are positive
  - then it may be factored as

    $$\mathbf{A} = \mathbf{L}\mathbf{U} = \mathbf{L}\mathbf{D}\mathbf{M}^T$$

    where $\mathbf{L}$ and $\mathbf{M}$ are unit lower triangular and $\mathbf{D}$ is diagonal with positive entries
    - The entries of $\mathbf{D}$ are the diagonal entries of $\mathbf{U}$
  - These properties are proved in Golub and Van Loan (1993), Section 4.2 or Demmel (1997), Section 2.7

Symmetric Positive Definite Systems

- Symmetric positive definite systems:
  - Arise in the finite difference or finite element solution of elliptic PDEs
  - $\mathbf{A}$ may be factorized as

$$\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T \tag{1a}$$

with

$$\mathbf{L} = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ l_{31} & l_{32} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & & 1 \end{bmatrix}, \qquad \mathbf{D} = \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{bmatrix} \tag{1b}$$

    - Zero entries are not shown

  - By direct factorization without pivoting

$$d_j = a_{jj} - \sum_{k=1}^{j-1} d_k l_{jk}^2, \tag{2a}$$

$$l_{ij} = \frac{1}{d_j}(a_{ij} - \sum_{k=1}^{j-1} d_k l_{jk} l_{ik}), \qquad i = j+1 : n, \quad j = 1 : n \tag{2b}$$

    - Summations are zero when the lower limit exceeds the upper one

Symmetric Positive Definite Factorization

```
function A = ldlt(A)
% ldlt: Factorization of a symmetric positive definite n-by-n
% matrix A into the product LDL', where L
% is unit lower triangular and D is diagonal. On return,
% A(i,j) stores L(i,j) if i > j and D(i) if i = j.

   [n n] = size(A);
   for j = 1:n
%                          Compute v(k), k = 1: j - 1
      for k = 1: j - 1
         v(k) = A(j,k)*A(k,k);
      end
%                          Compute d(j)
      v(j) = A(j,j) - dot(A(j,1:j-1),v(1:j-1));
      A(j,j) = v(j);
%                          Compute l(i,j), i = j + 1: n
      for i = j + 1:n
         A(i,j) = (A(i,j) - dot(A(i,1:j-1),v(1:j-1)))/v(j);
      end
   end
```

- We let $v_k = d_k l_{jk}$, $k = 1 : j - 1$
- We should use a symmetric storage mode, cf. Section 1.2
- The algorithm requires approximately $n^3/3$ FLOPs (half that of Gaussian elimination)
- Pivoting is not necessary for stability

Forward and Backward Substitution

- From (1a)

$$\mathbf{Ax} = \mathbf{LDL}^T\mathbf{x} = \mathbf{b}$$

  - Let

$$\mathbf{L}^T\mathbf{x} = \mathbf{y}, \qquad \mathbf{Dy} = \mathbf{z}, \qquad \mathbf{Lz} = \mathbf{b} \tag{3}$$

  - Forward substitution:

$$z_i = b_i - \sum_{k=1}^{i-1} l_{ik} z_k, \qquad i = 1 : n \tag{4a}$$

  - Diagonal scaling

$$y_i = z_i / d_i, \qquad i = 1 : n \tag{4b}$$

  - Backward substitution

$$x_i = y_i - \sum_{k=i+1}^{n} l_{ki} x_k, \qquad i = n : -1 : 1 \tag{4c}$$

- *Cholesky Factorization:*

$$\mathbf{A} = \tilde{\mathbf{L}}\tilde{\mathbf{L}}^T, \qquad \tilde{\mathbf{L}} = \mathbf{L}\mathbf{D}^{1/2}$$

  - The diagonal entries of $\tilde{\mathbf{L}}$ are square roots of the entires of $\mathbf{D}$
  - cf. Trefethen and Bau (1997), Lecture 23
  - The n square roots may be expensive

3.2. Banded and Profile Systems

- If $\mathbf{A} \in \Re^{n \times n}$ has lower bandwidth p and upper bandwidth q then
  - $\mathbf{L}$ has lower bandwidth p
    - $l_{ij} = 0$ for $j > i$ and $i > j + p$
  - $\mathbf{U}$ has upper bandwidth q
    - $u_{ij} = 0$ for $i > j$ and $j > i + q$
- *Example 1.* A 6×6 matrix with $p = 2$ and $q = 1$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & & & & \\ a_{21} & a_{22} & a_{23} & & & \\ a_{31} & a_{32} & a_{33} & a_{34} & & \\ & a_{42} & a_{43} & a_{44} & a_{45} & \\ & & a_{53} & a_{54} & a_{55} & a_{56} \\ & & & a_{64} & a_{65} & a_{66} \end{bmatrix}$$

  - Zeros are not shown

$$\mathbf{LU} = \begin{bmatrix} 1 & & & & & \\ l_{21} & 1 & & & & \\ l_{31} & l_{32} & 1 & & & \\ & l_{42} & l_{43} & 1 & & \\ & & l_{53} & l_{54} & 1 & \\ & & & l_{64} & l_{65} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & & & & \\ & u_{22} & u_{23} & & & \\ & & u_{33} & u_{34} & & \\ & & & u_{44} & u_{45} & \\ & & & & u_{55} & u_{56} \\ & & & & & u_{66} \end{bmatrix}$$

LU Factorization

- Linear equations solution
  - Use direct factorization
  - Neglect pivoting
    - Dangerous unless $\mathbf{A}$ is symmetric and positive definite
- *Factorization:*
  - u_{ij} and l_{ij} involve inner products between
    - the ith row of $\mathbf{L}$
    - the jth column of $\mathbf{U}$
    - l_{ij} is nonzero for $\max(1, i-p) \le j \le i$
    - u_{ij} is nonzero for $\max(1, j-q) \le i \le j$
  - Let

$$k^* = \max(1, i-p, j-q) \tag{1a}$$

$$u_{ij} = a_{ij} - \sum_{k=k^*}^{i-1} l_{ik} u_{kj}, \qquad j = i : i+q, \tag{1b}$$

$$l_{ji} = \frac{1}{u_{ii}}(a_{ji} - \sum_{k=k^*}^{i-1} l_{jk} u_{ki}), \qquad j = i+1 : i+p, \quad i = 1 : n \tag{1c}$$

    - A procedure should use the banded data structure of Section 1.2

Forward and Backward Substitution

- *Forward substitution:*

$$y_i = b_i - \sum_{k=\max(1,i-p)}^{i-1} l_{ik} y_k, \qquad i = 1 : n \tag{2a}$$

- *Backward substitution:*

$$x_i = \frac{1}{u_{ii}}(y_i - \sum_{k=i+1}^{\min(n,i+q)} u_{ik} y_k), \qquad i = n : -1 : 1 \tag{2b}$$

- *Operations and Storage* (with $p = q$):
  - Storage is $2(p+1)(n-p/2)$ locations
  - Factorization with $n \gg 1$, $p/n \ll 1$
    - Each inner product requires $i - k^*$ multiplications and subtractions and one division
    - $k^* = i - p$ for most rows
    - Summing on i, there are approximately $np(2p+1)$ multiplications and subtractions and np divisions
    - Thus, there are approximately $np(4p+3)$ FLOPs
  - Forward and backward substitution
    - There are approximately $2np$ multiplications and subtractions and n divisions or $n(4p+1)$ FLOPs
  - A full matrix requires about $2n^3/3$ FLOPs to factor and $2n^2$ to solve

Symmetric Positive Definite Band Matrices

- Let $\mathbf{A}$ be symmetric and positive definite with $p = q$
  - *Factorization* (cf. (1.2)):

$$k^* = \max(1, i - p)$$

$$d_j = a_{jj} - \sum_{k=k^*}^{j-1} d_k l_j k^2, \tag{3a}$$

$$l_{ij} = \frac{1}{d_j}(a_{ij} - \sum_{k=k^*}^{j-1} d_k l_{jk} l_{ik}), \qquad i = j+1 : j+p, \quad j = 1 : n \tag{3b}$$

  - *Forward substitution:*

$$z_i = b_i - \sum_{k=\max(1,i-p)}^{i-1} l_{ik} z_k, \qquad i = 1 : n \tag{4a}$$

  - *Diagonal scaling:*

$$y_i = z_i / d_i, \qquad i = 1 : n \tag{4b}$$

  - *Backward substitution:*

$$x_i = y_i - \sum_{k=i+1}^{\min(n,i+p)} l_{ki} x_k, \qquad i = n : -1 : 1 \tag{4c}$$

Tridiagonal Matrices

- **A** is tridiagonal ($p = q = 1$)
  - Equation (1a) becomes $k^* = \max(1, i-1, j-1)$
    - Only two nonzero entries per row or column $j = i, i+1$
  - *Factorization* (cf. (1)):

$$u_{ii} = a_{ii} - l_{i,i-1}u_{i-1,i} \tag{5a}$$

$$u_{i,i+1} = a_{i,i+1} \tag{5b}$$

$$l_{i+1,i} = a_{i+1,i}/u_{ii}, \qquad i = 1 : n \tag{5c}$$

    - Terms that do not exist (e.g., u_{01}, $l_{n+1,n}$) are zero
  - *Forward and backward substitution* (cf. (2)):

$$y_1 = b_1 \tag{6a}$$

$$y_i = b_i - l_{i,i-1}y_{i-1}, \qquad i = 2 : n \tag{6b}$$

$$x_n = y_n/u_{nn} \tag{6c}$$

$$x_i = \frac{1}{u_{ii}}(y_i - u_{i,i+1}y_{i+1}), \qquad i = n-1 : -1 : 1 \tag{6d}$$

- Implementing the tridiagonal procedure
  - Store the matrix by diagonals

$$l_i = a_{i,i-1}, \qquad a_i = a_{ii}, \qquad u_i = a_{i,i+1}$$

  - On output, replace l_i, a_i, u_i by their factors

The Tridiagonal Algorithm

```
function x = tridi(l, a, u, b)
% tridi: Solution of a tridiagonal system by Gaussian
% elimination. The vectorsl, a, u contain the subdiagonal,
% diagonal, and superdiagonal entries of an n-by-n
% tridiagonal matrix A. The solution of the system
% Ax = b is returned in x. The lower and upper
% triangular factors are stored as l and u.

   n = length(a);
%                          Factorization
   for i = 2:n
      l(i) = l(i)/a(i-1);
      a(i) = a(i) - l(i)*u(i-1);
   end
%                          Forward substitution
   x(1) = b(1);
   for i = 2:n
      x(i) = b(i) - l(i)*x(i-1);
   end
%                          Backward substitution
   x(n) = x(n)/a(n);
   for i = n-1: -1: 1
      x(i) = (x(i) - u(i+1)*x(i))/a(i)
   end
```

Pivoting with Banded Systems

- Pivoting alters the bandwidth
  - The upper bandwidth is enlarged
  - *Example 2.* Interchange the first two rows of the matrix of Example 1

$$\mathbf{A} = \begin{bmatrix} a_{21} & a_{22} & a_{23} & & & \\ a_{11} & a_{12} & & & & \\ a_{31} & a_{32} & a_{33} & a_{34} & & \\ & a_{42} & a_{43} & a_{44} & a_{45} & \\ & & a_{53} & a_{54} & a_{55} & a_{56} \\ & & & a_{64} & a_{65} & a_{66} \end{bmatrix}$$

    - The upper bandwidth is now $q = 2$
  - Some data structures cannot handle bandwidth changes
    - Others (to be discussed) can

Finite Difference Computation

- *Example 3.* A two-point boundary value problem

$$-w'' + w = 0, \qquad 0 < x < 1, \qquad w(0) = 0, \qquad w(1) = 1$$

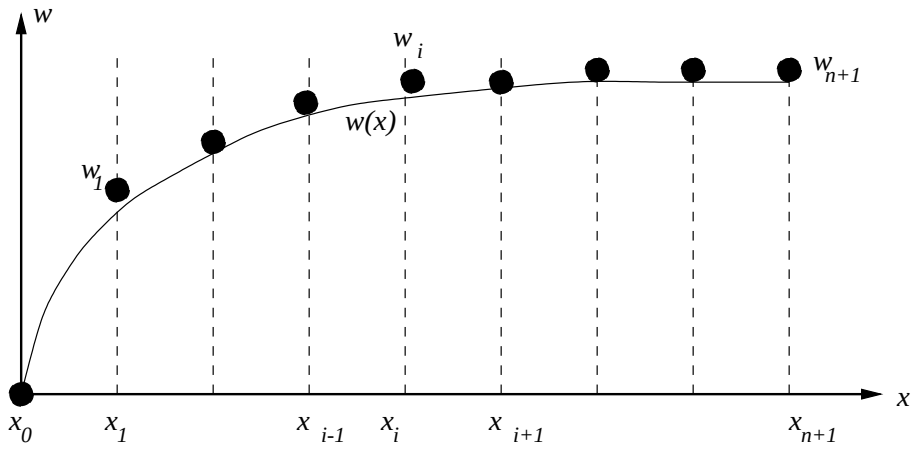
  - The exact solution is

$$w(x) = \frac{\sinh x}{\sinh 1}$$

- *Finite Difference Approximation:*

  - Introduce a mesh of spacing $h = 1/(n+1)$ on $[0, 1]$

    * Let $x_i = ih$, $i = 0 : n+1$



  - Approximate $w''(x_i)$ by central differences

$$w''(x_i) \approx \frac{w_{i-1} - 2w_i + w_{i+1}}{h^2}$$

    * w_i is the finite difference approximation of $w(x_i)$

  - Approximate the ODE at x_i

$$-\frac{w_{i-1} - 2w_i + w_{i+1}}{h^2} + w_i = 0$$

Finite Difference Computation

- Multiply by h^2

$$-w_{i-1} + (2 + h^2)w_i - w_{i+1} = 0$$

  - The discretization error is $O(h^2)$

- Use the difference equation at all interior mesh points

  - Use the boundary conditions $w_0 = 0$, $w_{n+1} = 1$

$$\begin{aligned}
(2 + h^2)w_1 - w_2 &= 0, \qquad && i = 1 \\
-w_{i-1} + (2 + h^2)w_i - w_{i+1} &= 0, \qquad && i = 2 : n - 1 \\
-w_n - 1 + (2 + h^2)w_n &= 1, \qquad && i = n
\end{aligned}$$

- The result is a tridiagonal linear system

$$\mathbf{Ax} = \mathbf{b}$$

  with

$$\mathbf{A} = \begin{bmatrix}
2 + h^2 & -1 & & & \\
-1 & 2 + h^2 & -1 & & \\
 & & \ddots & & \\
 & & -1 & 2 + h^2 & -1 \\
 & & & -1 & 2 + h^2
\end{bmatrix}$$

$$\mathbf{x} = [w_1,\ w_2,\ \cdots,\ w_n]^T, \qquad \mathbf{b} = [0,\ 0,\ \cdots,\ 1]^T$$

Finite Difference Computation

- The input to tridi is

$$\mathbf{a} = \begin{bmatrix} 2+h^2 \\ 2+h^2 \\ \vdots \\ 2+h^2 \end{bmatrix}, \qquad \mathbf{l} = \mathbf{u} = \begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

- Condition number of $\mathbf{A}$ as a function of n

n	h	$\kappa_2(\mathbf{A})$	$\kappa_2(\mathbf{A})h^2$
3	1/4	5.3629	0.3352
7	1/8	23.0147	0.3596
15	1/16	93.6675	0.3659
31	1/32	367.2897	0.3675
63	1/64	1506.8	0.3679
127	1/128	6028.7	0.3680

  - $\kappa_2(\mathbf{A}) = O(1/h^2)$
    - Accuracy and condition are at odds

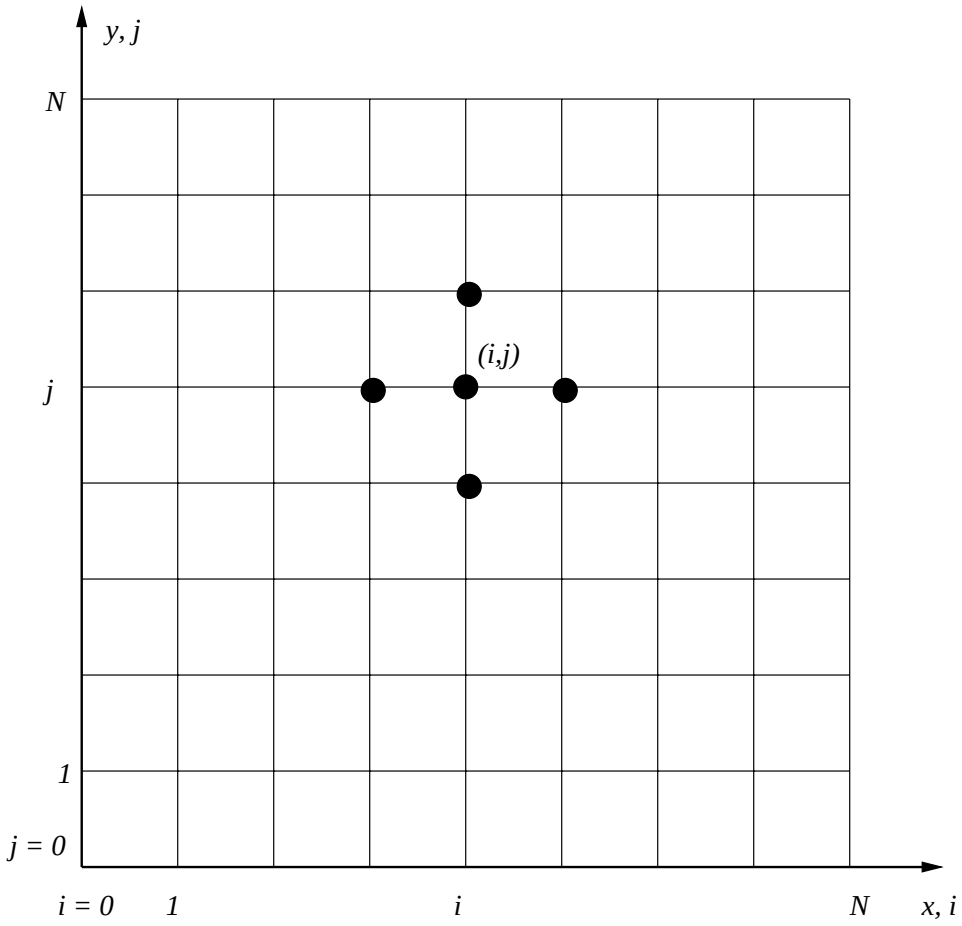
Two-Dimensional Finite Differences

- *Example 4.* Solving Poisson's equation on a square

$$-\Delta w := -w_{xx} - w_{yy} = f(x,y), \qquad (x,y) \in \Omega$$

$$w = g(x,y), \qquad (x,y) \in \partial\Omega$$

  - $\Omega = \{(x,y) | 0 < x, y < 1\}$
  - Divide Ω into N^2 squares of size $h \times h$ with $h = 1/N$
  - Let $x_i = ih$, $y_j = jh$



2D Finite Differences

- Approximate derivatives by central differences

$$w_{xx}(x_i, y_j) \approx \frac{w_{i-1,j} - 2w_{ij} + w_{i+1,j}}{h^2}$$

$$w_y y(x_i, y_j) \approx \frac{w_{i,j-1} - 2w_{ij} + w_{i,j+1}}{h^2}$$

  - w_{ij} is the finite difference approximation of $w(x_i, y_j)$
  - Approximate the PDE at (x_i, y_j)

$$-\frac{w_{i-1,j} - 2w_{ij} + w_{i+1,j}}{h^2} - \frac{w_{i,j-1} - 2w_{ij} + w_{i,j+1}}{h^2} = f(x_i, y_j)$$

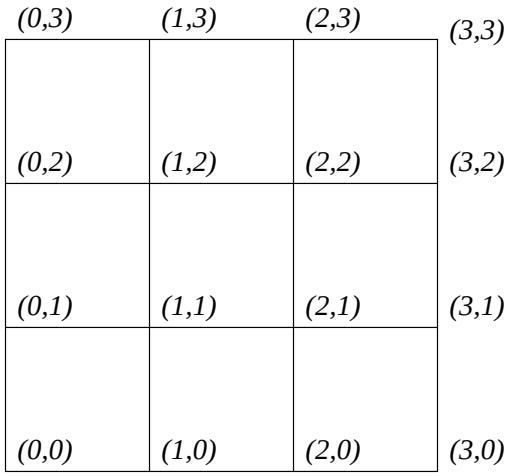
    or

$$-(w_{i-1,j} + w_{i+1,j} + w_{i,j-1} + w_{i,j+1}) + 4w_{ij} = h^2 f(x_i, y_j)$$

    * The difference equation at (x_i, y_j) connects w_{ij} to unknowns at its north, east, south, and west neighbors
    * The discretization error is $O(h^2)$

2D Finite Differences

- Use the difference equation at all interior nodes
  - Use the boundary conditions near edges and corners
    - For a 3×3 grid: write the difference equation at the points $(1,1)$, $(2,1)$, $(1,2)$, and $(2,2)$



$$4w_{11} - w_{21} - w_{12} = h^2 f_{11} + g_{01} + g_{10}$$
$$-w_{11} + 4w_{21} - w_{22} = h^2 f_{21} + g_{31} + g_{20}$$
$$-w_{11} + 4w_{12} - w_{22} = h^2 f_{12} + g_{02} + g_{13}$$
$$-w_{21} - w_{12} + 4w_{22} = h^2 f_{22} + g_{32} + g_{23}$$

    - $f_{ij} := f(x_i, y_j)$
    - The algebraic system is 4×4

- Order the equations and unknowns by rows

$$\begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} w_{11} \\ w_{21} \\ w_{12} \\ w_{22} \end{bmatrix} = \begin{bmatrix} h^2 f_{11} + g_{01} + g_{10} \\ h^2 f_{21} + g_{31} + g_{20} \\ h^2 f_{12} + g_{02} + g_{13} \\ h^2 f_{22} + g_{32} + g_{23} \end{bmatrix}$$

  - The structure of $\mathbf{A}$ is not apparent

2D Finite Differences

- The general finite difference system
  - Order the equations and unknowns by rows

$$\mathbf{x} = [\mathbf{x}_1^T,\ \mathbf{x}_2^T,\ \cdots,\ \mathbf{x}_{N-1}^T]^T \tag{7a}$$

    where

$$\mathbf{x}_j^T = [w_{1j},\ w_{2j},\ \cdots,\ 2_{N-1,j}] \tag{7b}$$

  - Write the difference equation at all interior nodes

$$\mathbf{A} = \begin{bmatrix}
4 & -1 & & & -1 & & & & & & & & & & \\
-1 & 4 & -1 & & & -1 & & & & & & & & & \\
 & & \ddots & & & & \ddots & & & & & & & & \\
 & & -1 & 4 & & & & -1 & & & & & & & \\
-1 & & & & 4 & -1 & & & -1 & & & & & & \\
 & -1 & & & -1 & 4 & -1 & & & -1 & & & & & \\
 & & \ddots & & & & \ddots & & & & \ddots & & & & \\
 & & & -1 & & & -1 & 4 & & & & -1 & & & \\
 & & & & & & & & & & & & & & \\
 & & & & & & & & & & \ddots & & & & \\
 & & & & & & & & & & & & & & \\
 & & & & & & & & -1 & & & & 4 & -1 & \\
 & & & & & & & & & -1 & & & -1 & 4 & -1 \\
 & & & & & & & & & & \ddots & & & & \ddots \\
 & & & & & & & & & & & -1 & & & -1 & 4
\end{bmatrix}$$

2D Finite Differences

- The algebraic system is *block tridiagonal*

$$\begin{bmatrix} \mathbf{C}_1 & \mathbf{D}_1 & & \\ \mathbf{D}_2 & \mathbf{C}_2 & \mathbf{D}_2 & \\ & & \ddots & \\ & & \mathbf{D}_{N-1} & \mathbf{C}_{N-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{N-1} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_{N-1} \end{bmatrix} \tag{8a}$$

where

$$\mathbf{C}_j = \begin{bmatrix} 4 & -1 & & \\ -1 & 4 & -1 & \\ & & \ddots & \\ & & -1 & 4 \end{bmatrix}, \qquad \mathbf{D}_j = \begin{bmatrix} -1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{bmatrix} \tag{8b}$$

and

$$\begin{aligned} \mathbf{b}_j = & h^2 \begin{bmatrix} f_{1j} \\ f_{2j} \\ \vdots \\ f_{N-1,j} \end{bmatrix} + \begin{bmatrix} g_{0j} \\ 0 \\ \vdots \\ 0 \\ g_{Nj} \end{bmatrix} - \delta_{1,j} \begin{bmatrix} g_{10} \\ g_{20} \\ \vdots \\ g_{N-1,0} \end{bmatrix} \\ & - \delta_{N-1,j} \begin{bmatrix} g_{1N} \\ g_{2N} \\ \vdots \\ g_{N-1,N} \end{bmatrix} \end{aligned} \tag{8c}$$

2D Finite Differences

- **Note:**

  i. Matrix Dimensions:
  - $\mathbf{C}_j$ is a $(N-1) \times (N-1)$ tridiagonal matrix
  - $\mathbf{D}_j$ is a $(N-1) \times (N-1)$ diagonal matrix
  - $\mathbf{A}$ is a $(N-1)^2 \times (N-1)^2$ block tridiagonal matrix

  ii. The Kronecker delta δ_{ij} is unity when $j = k$ and zero otherwise

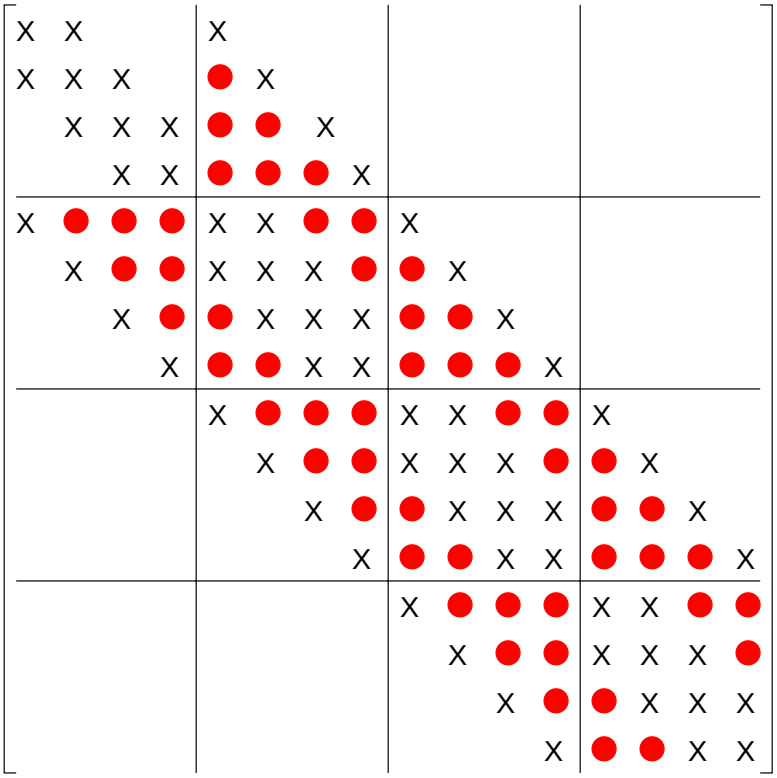
  iii. There are only 5 nonzero bands. The lower and upper bandwidths are $p = q = N - 1$

  iv. $\mathbf{D}_j$ is tridiagonal with finite element discretization
  - using a piecewise bilinear polynomial basis
  - The lower and upper bandwidths would be N

- *Fill In* for scalar band procedures
  - u_{ij} and l_{ij} involve inner products between the ith row of $\mathbf{L}$ and the jth column of $\mathbf{U}$
  - Zero elements within the band of $\mathbf{A}$ are nonzero in $\mathbf{L}$ and $\mathbf{U}$

Fill In

- Consider $N = 4$



X - *Original entry* ● - *Created by elimination*

  - **A** has about $5N^2$ nonzero entries
  - **L** and **U** require about $2N^3$ memory locations
  - With $n \approx N^2$ and $p = q \approx N$
    - Band factorization needs about N^4 multiplications
    - Full factorization needs about $N^6/3$ multiplications

Profile Schemes

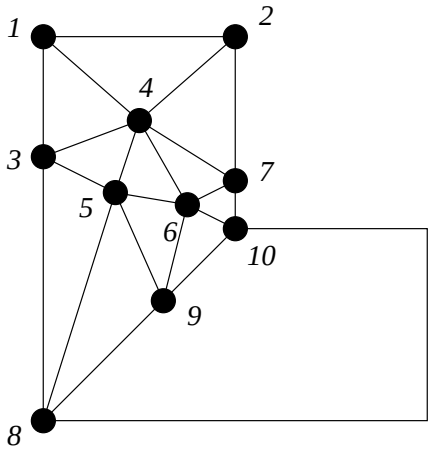
– Concentrate on symmetric positive definite systems

– There may be variations in the bandwidth

  * Due to pivoting
  * Due to nonuniform grids

– *Example 5.* Finite element matrix structure

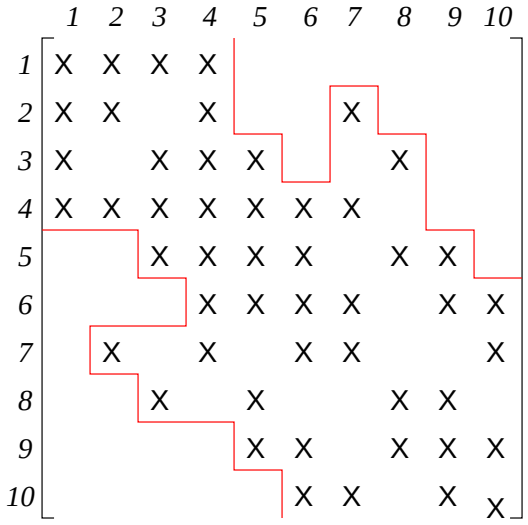
  * Unknowns at vertices are connected to unknowns on elements sharing the vertex
  * Equations and unknowns are ordered as shown



	1	2	3	4	5	6	7	8	9	10
1	X	X	X	X						
2	X	X		X			X			
3	X		X	X	X			X		
4	X	X	X	X	X	X	X			
5			X	X	X	X		X	X	
6				X	X	X	X		X	X
7		X		X		X	X			X
8			X		X			X	X	
9					X	X		X	X	X
10						X	X		X	X

Profile Schemes

- Account for local variation in bandwidth
  - The *skyline* or *profile* is the envelope formed by the local row or column bandwidths



  - **L** and **U** have the same profiles as **A**
  - Let m_i^* be the index of the first nonzero element of row i

$$l_{ij} = 0, \qquad 0 \le j < m_i^* \tag{9a}$$

and

$$k^* = \max(m_i^*, m_j^*) \tag{9b}$$

  - The profile form of the symmetric factorization (3) is

$$d_1 = a_{11} \tag{10a}$$

$$l_{ji} = \frac{1}{d_i}(a_{ji} - \sum_{k=k^*}^{i-1} d_k l_{jk} l_{ik}), \qquad i = m_j^* : j-1, \tag{10b}$$

$$d_j = a_{jj} - \sum_{k=m_j^*}^{j-1} d_k l_{jk}^2, \qquad j = 2 : n \tag{10c}$$

Profile Schemes

- The profile form of forward and backward substitution is

$$z_i = b_i - \sum_{k=m_i^*}^{i-1} l_{ik} z_k, \qquad i = 1 : n \tag{11a}$$

$$y_i = z_i / d_i, \qquad i = 1 : n \tag{11b}$$

$$x_i = y_i - \sum_{k=i+1}^{i+m_i^*} l_{ki} x_k, \qquad i = n : -1 : 1 \tag{11c}$$

  - Sums in (10, 11) are zero when a lower index exceeds an upper
  - $\mathbf{L}$ can be stored by rows (cf. Section 1.2)
    - Zeros within the profile become nonzero and are stored
  - The inner products in (10a,b) and (11a) involve row operations
    - All elements are within the profile
    - l_{ji} in (10b) is computed as u_{ij} and reflected
  - Backward substitution involves a column operation
    - Must test if an operation is inside the profile
    - Rewrite the operation using (row) saxpys

Profile Schemes

- *Example 6.* Column Identification

$$\mathbf{L} = \begin{bmatrix} \times & & & & & & & \\ \times & \times & & & & & & \\ \times & \times & \times & & & & & \\ \times & \times & \times & \times & & & & \\ & & & \times & \times & & & \\ & \times & \times & \times & \times & \times & & \\ & & & \times & \times & \times & \times & \\ & & \times & \times & \times & \times & \times & \times \end{bmatrix}$$

- The profile
  - The local (lower and upper) bandwidth of row i is $p_i = i - m_i^*$
  - The bandwidth of $\mathbf{A}$ is $p = \max_{1 \le i \le n} p_i$
  - The number of multiplications and divisions to factor $\mathbf{A}$ is

$$\Theta = \sum_{i=1}^{N-1} p_i(p_i + 3) \tag{12a}$$

  - The number of elements within the profile is

$$P = \sum_{i=1}^{N} p_i \tag{12b}$$

Profile Reduction

- Zero elements within the profile become nonzero during the factorization
  - Fill-in depends on the ordering of equations and unknowns
  - Methods for reducing bandwidth will reduce profile
    - Methods of reducing bandwidth are simpler
    - Methods for reducing profile are more effective
  - *Example 7.* Consider the 5×5 matrix

$$\mathbf{A} = \begin{bmatrix} \times & & & & \\ \times & \times & & & \\ \times & \bullet & \times & & \\ \times & \bullet & \bullet & \times & \\ \times & \bullet & \bullet & \bullet & \times \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix}$$
$$\begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix}$$

    - $\mathbf{A}$ is symmetric but only the lower triangular portion is shown
    - All elements within the profile will fill in ($\bullet$)
    - Reversing the order of equations and unknowns produces a system with no fill in

$$\mathbf{A} = \begin{bmatrix} \times & & & & \\ & \times & & & \\ & & \times & & \\ & & & \times & \\ \times & \times & \times & \times & \times \end{bmatrix} \begin{matrix} 5 \\ 4 \\ 3 \\ 2 \\ 1 \end{matrix}$$
$$\begin{matrix} 5 & 4 & 3 & 2 & 1 \end{matrix}$$

Combat Graph Theory

- Reading: Saad (1996)
- A graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is defined by sets of vertices and edges

$$\mathcal{V} = \{v_1, v_2, \cdots, v_{N_v}\}, \qquad \mathcal{E} = \{e_1, e_2, \cdots, e_{N_e}\}$$

  with e_k consisting of pairs (v_i, v_j), $v_i, v_j \in \mathcal{V}$

- The *adjacency graph* representation of a symmetric matrix $\mathbf{A}$:

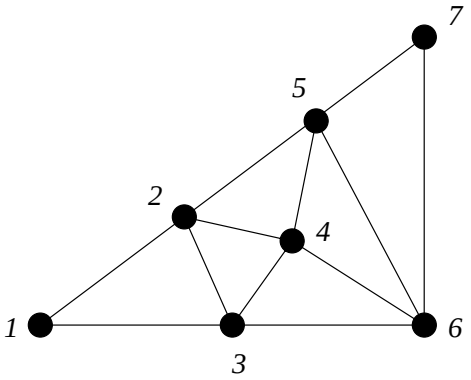
  i. Vertices of the graph correspond to rows of $\mathbf{A}$ (unknowns)

  ii. Edges of the graph correspond to nonzero elements of $\mathbf{A}$

  – Graph vertex i and j are connected if and only if $a_{ij} \neq 0$, $i \neq j$

- *Example 8.* The graph of a symmetric matrix $\mathbf{A}$ arising from a scalar PDE is often the mesh

$$\mathbf{A} = \begin{array}{c} \left[\begin{array}{ccccccc} \times & & & & & & \\ \times & \times & & & & & \\ \times & \times & \times & & & & \\ & \times & \times & \times & & & \\ & \times & & \times & \times & & \\ & & \times & \times & \times & \times & \\ & & & & \times & \times & \times \end{array}\right] \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \\ \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \end{array}$$



Combat Graph Theory

- *Example 9.* Draw graphs of the matrices of Example 7

  - A symmetric permutation is equivalent to reordering the vertices of a graph
  - A symmetric permutation changes the rows and columns of a matrix so that the eigenvalues do not change

- The *degree of a vertex* of a graph is the number of edges incident on it

  - *Example 10.* The degree of Node 1 of Example 8 is 2, that of Node 4 is 4

- A *level set* consists of all unmarked neighbors of all nodes visited in a previous level set

Cuthill-McKee Algorithm

- Cuthill-McKee (CM) profile minimization technique:
  - Traverse a graph in level sets
  - Traverse the level set in order of increasing degree
  - Select the initial level set to consist of a single vertex v_1
    - Select v_1 on the periphery of $\mathcal{G}$

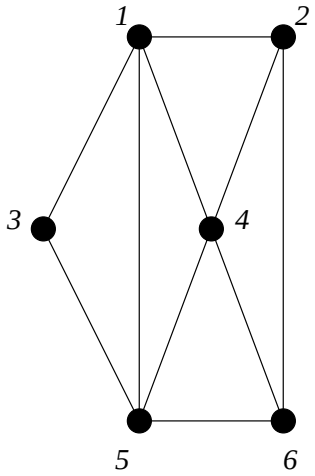
function $\mathcal{Q} = \mathrm{cm}(\mathcal{V})$
% cm: Cuthill-McKee procedure to number a set of vertices
% $\mathcal{V}$ to reduce the profile of a sparse matrix. The procedure
% begins with a starting vertex $v_1 \in \mathcal{V}$. On return, $\mathcal{Q}$
% stores the renumbered vertices.

$\mathcal{L} = \{v_1\}$;
$\mathcal{Q} = \mathcal{L}$;
while $\tilde{(}\mathcal{L} \in \emptyset)$
  For each vertex in $\mathcal{L}$ (in the order in which they are numbered)
    Append their unnumbered neighbors to $\mathcal{Q}$
    in order of increasing degree;
    $\mathcal{L} =$ { vertices just numbered };
  end
end

- This algorithm is slightly different than Saad's (1996)

Cuthill-McKee Algorithm

- *Example 11.* Consider the mesh shown

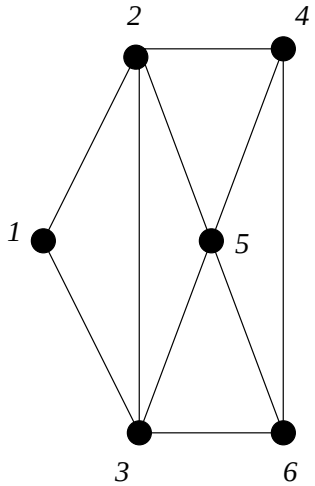


	1	2	3	4	5	6
1	X					
2	X	X				
3	X		X			
4	X	X		X		
5	X		X	X	X	
6		X		X	X	X

- Cuthill-McKee procedure with starting vertex 3

Level Set	Vertex	Neighbor	Degree	$\mathcal{Q}$
1	3	1	4	{3,
		5	4	1, 5,
2	1	2	3	
		4	4	2, 4,
	5	6	3	6}

- New numbering



	1	2	3	4	5	6
1	X					
2	X	X				
3	X	X	X			
4		X		X		
5		X	X	X	X	
6			X	X	X	X

Cuthill-McKee Algorithm

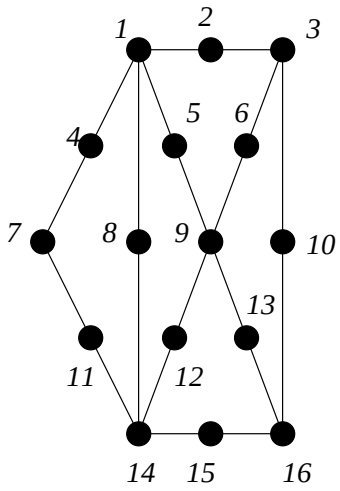
- *Example 12.* Consider the mesh and graph on the next page
  - Unknowns at midpoint nodes are only connected to those on elements containing the node
  - e.g., quadratic finite element approximation
- With row-by-row ordering: $p = 13$, $P = 106$
- First level set
  - Start at Node 7
    - $\mathcal{Q} = \{7\}$
  - Traverse the first level set

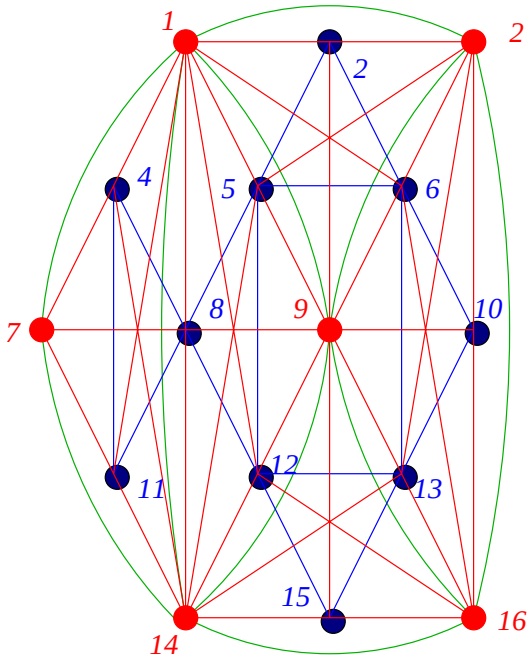
Vertex	Neighbors	Degree	$\mathcal{Q}$
7	1	11	$\{7,$
	4	5	
	8	8	
	11	5	
	14	11	4, 11, 8, 1, 14$\}$

  - After the first level-set traversal:

$$\mathcal{L} = \{4,\ 11,\ 8,\ 1,\ 14\}, \qquad \mathcal{Q} = \{7,\ 4,\ 11,\ 8,\ 1,\ 14\}$$

Example 12: Row-by-Row Order





	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	X															
2	X	X														
3	X	X	X													
4	X			X												
5	X	X	X		X											
6	X	X	X		X	X										
7	X			X			X									
8	X			X	X		X	X								
9	X	X	X		X	X		X	X							
10			X			X			X	X						
11	X			X			X	X			X					
12	X				X			X	X			X				
13			X			X			X	X		X	X			
14	X			X	X		X	X	X		X	X	X	X		
15									X			X	X	X	X	
16			X			X			X	X		X	X	X	X	X

Example 12: Cuthill-McKee Order

- Level Set 2:

$$\mathcal{L} = \{4,\ 11,\ 8,\ 1,\ 14\}, \qquad \mathcal{Q} = \{7,\ 4,\ 11,\ 8,\ 1,\ 14\}$$

Vertex	Neighbors	Degee
4		
11		
8	5	8
	9	12
	12	8
1	2	5
	3	8
	6	8
14	13	8
	15	5
	16	8

  - The new order

$$\mathcal{L} = \{5,\ 12,\ 9,\ 2,\ 3,\ 6,\ 15,\ 13\}$$

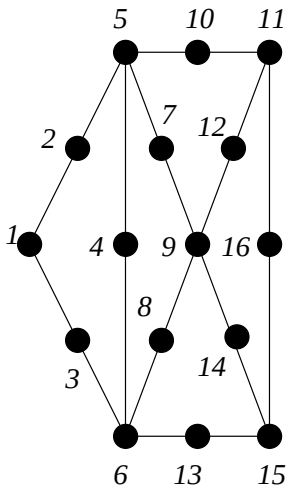
$$\mathcal{Q} = \{7,\ 4,\ 11,\ 8,\ 1,\ 14,\ 5,\ 12,\ 9,\ 2,\ 3,\ 6,\ 15,\ 13,\ 16\}$$

- Third level set: The only unnumbered vertex is 10

$$\mathcal{Q} = \{7,\ 4,\ 11,\ 8,\ 1,\ 14,\ 5,\ 12,\ 9,\ 2,\ 3,\ 6,\ 15,\ 13,\ 16,\ 10\}$$

Example 12: Cuthill-McKee Order

- Bandwidth and profile: $p = 9$, $P = 76$



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	X															
2	X	X														
3	X	X	X													
4	X	X	X	X												
5	X	X	X	X	X											
6	X	X	X	X	X	X										
7				X	X	X	X									
8				X	X	X	X	X								
9				X	X	X	X	X	X							
10					X		X		X	X						
11					X		X		X	X	X					
12					X		X		X	X	X	X				
13						X		X	X				X			
14						X		X	X		X	X	X	X		
15						X		X	X		X	X	X	X	X	
16									X		X	X		X	X	X

eeKcM-llihtuC Algorithm

- George (1971) [1] reversed the numbering given by the Cuthill-McKee algorithm
  - This is called the *Reverse Cuthill-McKee (RCM)* algorithm
  - George proves that the profile of RCM is never larger than that of CM
- The RCM ordering for Example 12 is

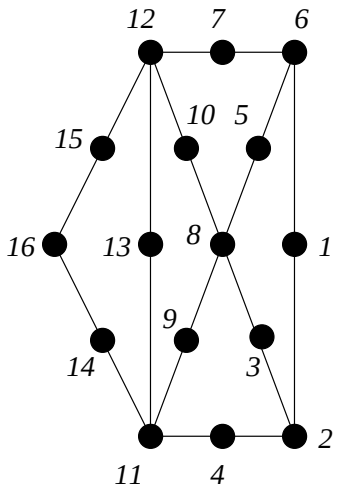
  $$\{10,\ 16,\ 13,\ 15,\ 6,\ 3,\ 2,\ 9,\ 12,\ 5,\ 14,\ 1,\ 8,\ 11,\ 4,\ 7\}$$

- The MATLAB procedure *symrcm* performs RCM on symmetric matrics
  - It produced the same ordering as above with 12 and 5 interchanged and 4 and 7 interchanged
  - Evidently, 4 was the starting node
  - This gave the same profile as the above ordering
- The MATLAB procedure *spy* will plot the nonzero structure of a sparse matrix

[1] J.A. George (1971), "Computer Implementation of the Finite Element Method," Rep. STAN-CS-71-208, Dept. Comp. Sci., Stanford University, Palo Alto

Example 12: RCM Oder

- Bandwidth and profile: $p = 9$, $P = 68$

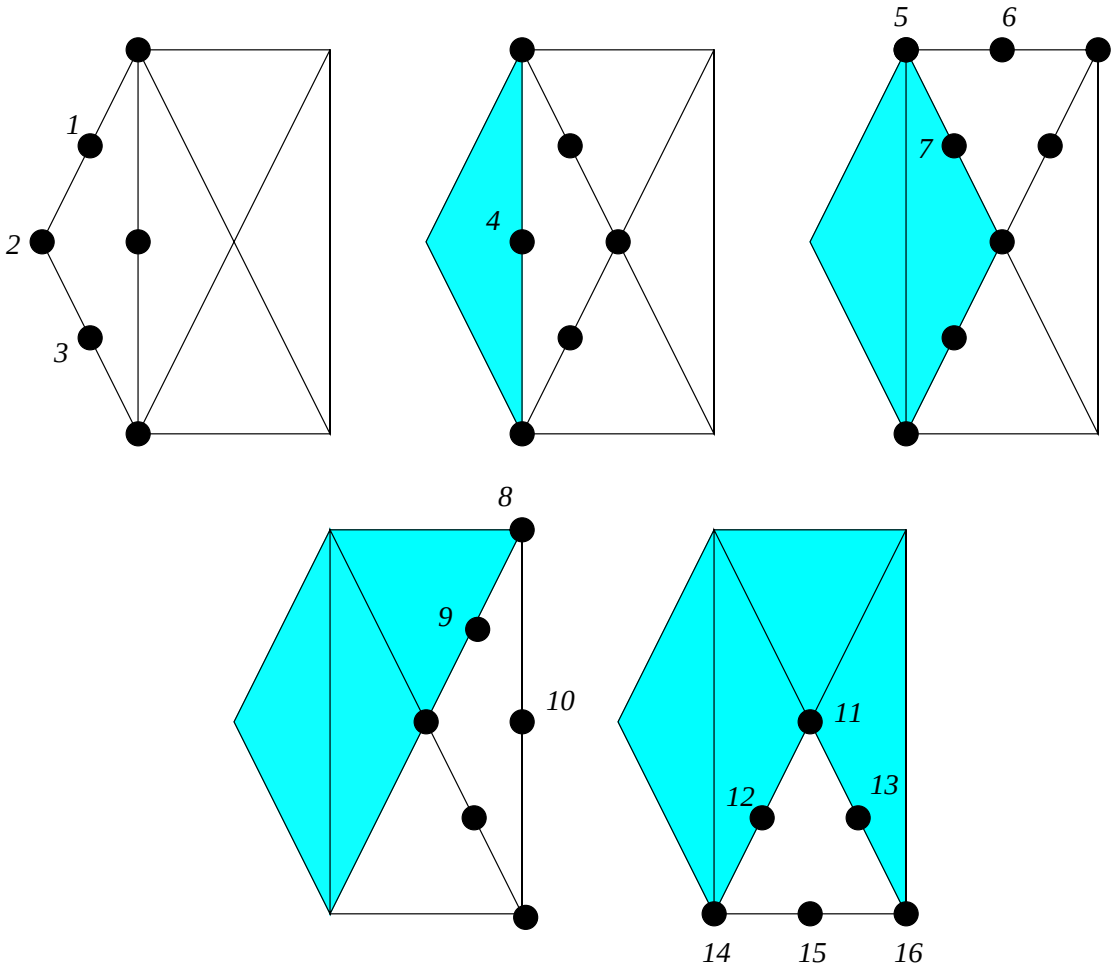


	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	X															
2	X	X														
3	X	X	X													
4		X	X	X												
5	X	X	X		X											
6	X	X	X		X	X										
7					X	X	X									
8	X	X	X	X	X	X	X	X								
9		X	X	X				X	X							
10					X	X	X	X	X	X						
11		X	X	X				X	X	X	X					
12					X	X	X	X	X	X	X	X				
13								X	X	X	X	X	X			
14											X	X	X	X		
15											X	X	X	X	X	
16											X	X	X	X	X	X

Wave-Front Ordering

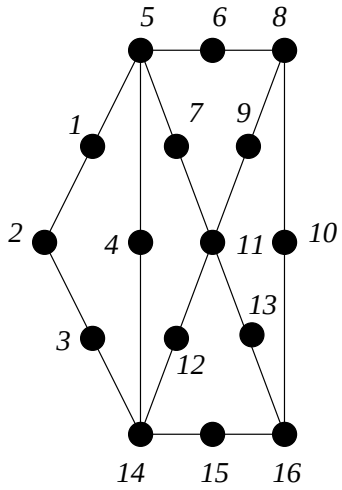
- "Frontal" or "wave front" techniques:
  - Select unknowns x_i that only depend on themselves
    - They form the initial level set
    - If none, select an unknown on the periphery of the graph
  - All unknowns that only involve those in Level Set 1 form Level Set 2
  - All unknowns that only involve those in Level Sets 1 and 2 form Level Set 3, etc.
- For finite element problems, this can be regarded as element annihilation
  - cf. J.A. George and J.W. Liu (1981)
- *Example 13.* Consider the problem of Example 12
  - Select an element and number the unknowns that are not connected to unknowns on other elements
  - Remove the element
  - Proceed to an adjacent element and repeat the process

Wave-Front Ordering



Wave-Front Ordering

- Wave-Front bandwidth and profile: $p = 13$, $P = 68$



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	X															
2	X	X														
3	X	X	X													
4	X	X	X	X												
5	X	X	X	X	X											
6					X	X										
7				X	X	X	X									
8					X	X	X	X								
9					X	X	X	X	X							
10								X	X	X						
11				X	X	X	X	X	X	X	X					
12				X	X		X				X	X				
13								X	X	X	X	X	X			
14	X	X	X	X	X		X				X	X	X	X		
15											X	X	X	X	X	
16								X	X	X	X	X	X	X	X	X

3.3. Block Tridiagonal Systems

- Example 2.4 produced the block tridiagonal system of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1a}$$

or

$$\begin{bmatrix} \mathbf{A}_1 & \mathbf{B}_1 & & \\ \mathbf{C}_2 & \mathbf{A}_2 & \mathbf{B}_2 & \\ & & \ddots & \\ & & \mathbf{C}_n & \mathbf{A}_n \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_n \end{bmatrix} \tag{1b}$$

  - $\mathbf{A}_k$, $\mathbf{C}_k$, $\mathbf{B}_k$, $k = 1 : n$, are $q \times q$
  - The blocks of Example 2.4 were symmetric

$$\mathbf{A}_j = \begin{bmatrix} 4 & -1 & & \\ -1 & 4 & -1 & \\ & & \ddots & \\ & & -1 & 4 \end{bmatrix}$$

$$\mathbf{B}_j = \mathbf{C}_j = \begin{bmatrix} -1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{bmatrix}$$

  - Do not assume symmetry ($\mathbf{B}_j \neq \mathbf{C}_j + 1$)
    - But do not pivot (dangerous)

The Block Tridiagonal algorithm

- Assume a factorization of the form

$$\mathbf{L} = \begin{bmatrix} \mathbf{I} & & & \\ \mathbf{L}_2 & \mathbf{I} & & \\ & \ddots & & \\ & & \mathbf{L}_n & \mathbf{I} \end{bmatrix}, \qquad \mathbf{U} = \begin{bmatrix} \mathbf{U}_1 & \mathbf{B}_1 & & \\ & \mathbf{U}_2 & \mathbf{B}_2 & \\ & & \ddots & \\ & & & \mathbf{U}_n \end{bmatrix} \tag{2}$$

  - Multiply

$$\mathbf{U}_1 = \mathbf{A}_1 \tag{3a}$$

$$\mathbf{L}_i \mathbf{U}_{i-1} = \mathbf{C}_i, \qquad i = 2 : n \tag{3b}$$

$$\mathbf{L}_i \mathbf{B}_{i-1} + \mathbf{U}_i = \mathbf{A}_i, \qquad i = 2 : n \tag{3c}$$

- Factorization

  $\mathbf{U}_1 = \mathbf{A}_1$;
  for i = 2:n
    Factor $\mathbf{U}_{i-1}^T$ and determine the rows of $\mathbf{L}_i$ from
      $\mathbf{U}_{i-1}^T \mathbf{L}_i^T = \mathbf{C}_i^T$ by forward and backward
      substitution;
    $\mathbf{U}_i = \mathbf{A}_i - \mathbf{L}_i \mathbf{b}_{i-1}$;
  end

- Forward and backward substitution: from (1, 2)

$$\mathbf{y}_1 = \mathbf{b}_1 \tag{4a}$$

$$\mathbf{y}_i = \mathbf{b}_i - \mathbf{L}_i \mathbf{y}_{i-1}, \qquad i = 2 : n \tag{4b}$$

$$\mathbf{U}_n \mathbf{x}_n = \mathbf{y}_n \tag{4c}$$

$$\mathbf{U}_i \mathbf{x}_i = \mathbf{y}_i - \mathbf{b}_i \mathbf{x}_{i+1}, \qquad i = n-1 : -1 : 1 \tag{4d}$$

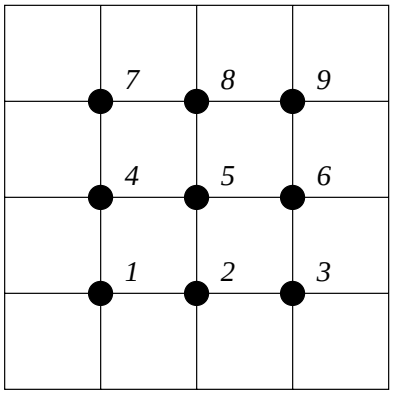
  - Save $\mathbf{U}_i$ in factored form from (3b)

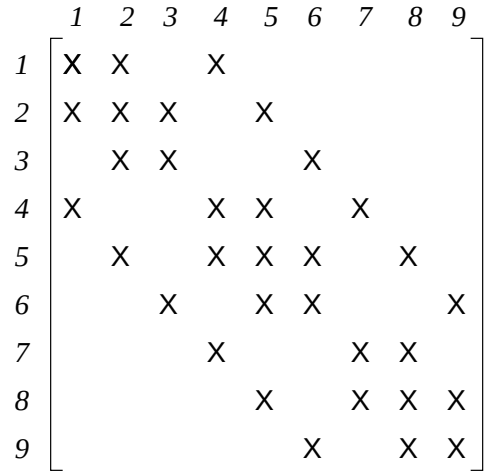
The Tridiagonal Algorithm

- Operation count
  - $2q^3/3$ FLOPs to factor $\mathbf{U}_{i-1}$ in (3b)
  - $2q^3$ FLOPs to determine $\mathbf{L}_i$ in (3b)
  - $2q^3$ FLOPs to determine $\mathbf{U}_i$ in (3c)
  - Summing on i, we have $14nq^3/3$ multiplications to factor
  - The forward and backward substitution takes $O(nq^2)$ FLOPs
  - $\mathbf{U}_i$ and $\mathbf{L}_i$ are not tridiagonal even if $\mathbf{A}_i$ is
- Band vs Block
  - Depends on the sizes of p (the bandwidth) and q (the block size)
  - For the 2D Poisson equation on an $N \times N$ grid (Example 2.4), $p \approx q$
    - Banded factorization has approximately $2N^4$ FLOPs
    - Block factorization has $n \approx N^2$ and $q \approx N$ for approximately $14N^5/3$ FLOPs
    - Some reductions in the block operation count are possible by accounting for the sparsity of $\mathbf{B}_i$ and $\mathbf{C}_i$

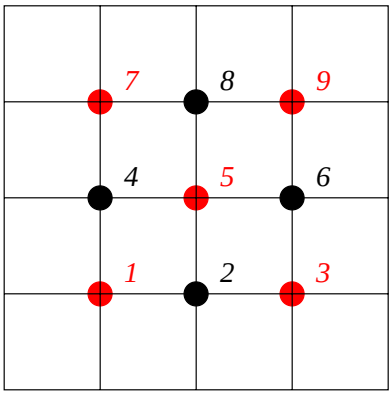
Two-Dimensional Poisson Equation

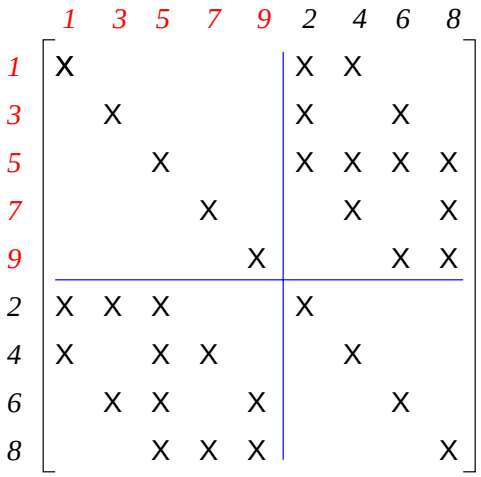
- *Example 1.* Consider Poisson's equation on an $N \times N$ grid (cf. Example 2.4)
  - Finite differencing for a 4×4 problem
  - Row-by-row numbering with a single index





- Write equations at the odd points first
  - Order the odd unknowns first





Odd-Even Ordering

- The structure of the matrix with odd-even ordering is

$$\mathbf{A} = \begin{bmatrix} \mathbf{D}_1 & \mathbf{V}_1 \\ \mathbf{V}_2 & \mathbf{D}_2 \end{bmatrix}$$

- To solve

$$\begin{bmatrix} \mathbf{D}_1 & \mathbf{V}_1 \\ \mathbf{V}_2 & \mathbf{D}_2 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}$$

  - Factor

$$\begin{bmatrix} \mathbf{D}_1 & \mathbf{V}_1 \\ \mathbf{V}_2 & \mathbf{D}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{L}_2 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{U}_1 & \mathbf{B}_1 \\ \mathbf{0} & \mathbf{U}_2 \end{bmatrix}$$

  - Expand the factorization

$$\mathbf{U}_1 = \mathbf{D}_1, \qquad \mathbf{B}_1 = \mathbf{V}_1$$

$$\mathbf{L}_2 = \mathbf{V}_2\mathbf{U}_1^{-1}, \qquad \mathbf{U}_2 = \mathbf{D}_2 - \mathbf{L}_2\mathbf{V}_1$$

  - Forward and backward substitution

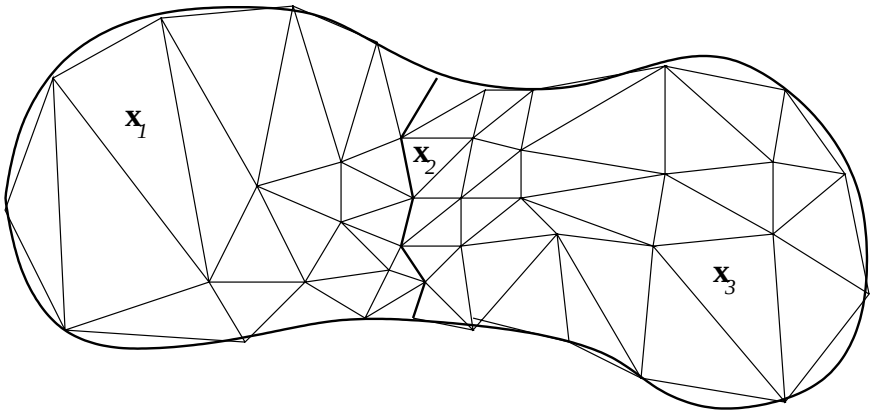
$$\mathbf{y}_1 = \mathbf{b}_1, \qquad \mathbf{y}_2 = \mathbf{b}_2 - \mathbf{V}_2\mathbf{U}_1^{-1}\mathbf{y}_1$$

$$\mathbf{U}_2\mathbf{x}_2 = \mathbf{y}_2, \qquad \mathbf{D}_1\mathbf{x}_1 = \mathbf{y}_1 - \mathbf{V}_1\mathbf{x}_2$$

  - $\mathbf{U}_2$ is penta-diagonal

Domain Decomposition

- "Dissection," "substructuring," and "domain decomposition"
  - Partition $\mathbf{A}$ into three pieces so that
    - The unknowns $\mathbf{x}_1$ are only connected to $\mathbf{x}_3$
    - The unknowns $\mathbf{x}_2$ are only connected to $\mathbf{x}_3$
  - Occurs "naturally" in finite difference and finite element applications



  - The structure of the system is

$$\mathbf{A}\mathbf{x} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{A}_{13} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{13}^T & \mathbf{A}_{23}^T & \mathbf{A}_{33} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{bmatrix} \qquad (5)$$

- Suppose $\mathbf{A}$ is symmetric
  - Factor $\mathbf{A}$ into $\mathbf{L}\mathbf{D}\mathbf{L}^T$

$$\mathbf{L} = \begin{bmatrix} \mathbf{I} & & \\ & \mathbf{I} & \\ \boldsymbol{\omega}_{13}^T & \boldsymbol{\omega}_{23}^T & \mathbf{I} \end{bmatrix}, \qquad \mathbf{D} = \begin{bmatrix} \mathbf{D}_1 & & \\ & \mathbf{D}_2 & \\ & & \mathbf{D}_3 \end{bmatrix}$$

Domain Decomposition

- Factorization procedure

  function $[\mathbf{L}, \mathbf{D}] = \mathrm{dd}(\mathbf{A})$
  % dd: block factorization of a matrix $\mathbf{A}$ into $\mathbf{LDL}^T$

  for k = 1:2
    $\mathbf{D}_k = \mathbf{A}_{kk}$;
    Solve $\mathbf{D}_k \boldsymbol{\omega}_{k3} = \mathbf{A}_{k3}$ for $\boldsymbol{\omega}_{k3}$;
  end
  $\mathbf{D}_3 = \mathbf{A}_{33} - \boldsymbol{\omega}_{13}^T \mathbf{D}_1 \boldsymbol{\omega}_{13} - \boldsymbol{\omega}_{23}^T \mathbf{D}_2 \boldsymbol{\omega}_{23}$;

- **Note:**

  i. $\mathbf{D}_1$ and $\mathbf{D}_2$ can be factored without knowledge of each other or $\mathbf{D}_3$, $\boldsymbol{\omega}_{13}$, and $\boldsymbol{\omega}_2 3$

  ii. The factorizations of $\mathbf{D}_1$ and $\mathbf{D}_2$ can be done in parallel

  iii. Design modifications only change those matrices affected

  iv. The procedure can be done recursively

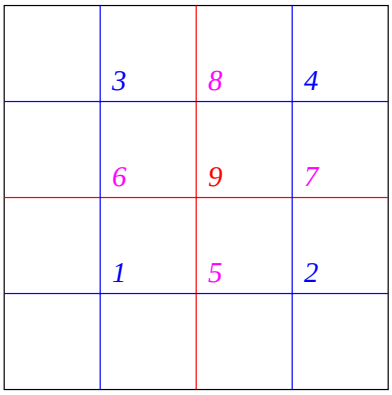
  v. $\mathbf{D}_3$ is called the *Schur complement* of $\mathbf{A}$

  - For $N \times N$ grid problems, George (1973) [1] shows that factorization takes $O(N^3)$ FLOPs
  - Banded and profile techniques require $O(N^4)$ FLOPs
  - Nested dissection has the optimal order of operations

[1] J.A. George (1973), Nested dissection of a regular finite element mesh, *SIAM J. Numer. Anal.*, **10**, 345-363

Nested Dissection

- Two-dimensional nested dissection
  - Bisect the domain
  - Number unknowns on the fine level first
  - Number unknowns on the interface between two regions next
  - Number unknowns at the juncture of four regions last
- Nested ordering of a 4×4 mesh



	1	2	3	4	5	6	7	8	9
1	X				X	X			
2		X			X		X		
3			X			X		X	
4				X			X	X	
5	X	X			X				X
6	X		X			X			X
7		X		X			X		X
8			X	X				X	X
9					X	X	X	X	X

4. Orthogonalization and Least Squares
4.1. Orthogonality and the Singular Value Decomposition

- Reading Trefethen and Bau (1997), Lecture 2
- *Orthogonal vectors:*
  - Orthogonal bases are better conditioned than others
- **Definition 1:** $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \in \Re^m$ is *orthogonal* if

$$\mathbf{x}_i^T \mathbf{x}_j = 0, \qquad i \neq j \tag{1a}$$

  - The vectors are *orthonormal* if

$$\mathbf{x}_i^T \mathbf{x}_j = \delta_{ij}, \qquad i \neq j \tag{1b}$$

- **Definition 2:** Two subspaces $\mathcal{S}_1, \mathcal{S}_2 \in \Re^m$ are orthogonal if $\mathbf{x}^T\mathbf{y} = 0$ for all $\mathbf{x} \in \mathcal{S}_1$ and all $\mathbf{y} \in \mathcal{S}_2$
  - $\mathcal{S}_2$ is called the *orthogonal complement* of $\mathcal{S}_1$
- **Theorem 1:** If $\mathbf{A} \in \Re^{m\times n}$ then range($\mathbf{A}$) is the orthogonal complement of null($\mathbf{A}^T$)
  - *Proof:* Let $\mathbf{y} \in$ range($\mathbf{A}$) and $\mathbf{x} \in$ null($\mathbf{A}^T$)
  - With $\mathbf{z} \in \Re^n$, let $\mathbf{y} = \mathbf{Az}$

$$\mathbf{y}^T\mathbf{x} = (\mathbf{Az})^\mathbf{T}\mathbf{x} = \mathbf{z}^\mathbf{T}\mathbf{A}^\mathbf{T}\mathbf{x} = \mathbf{z}^\mathbf{T}(\mathbf{A}^\mathbf{T}\mathbf{x}) = \mathbf{0}$$

  - Because $\mathbf{A}^T\mathbf{x} = 0$ $\square$

Orthogonality

- **Definition 3:** Vectors $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k\}$ form an *orthonormal basis* for a subspace $\mathcal{S} \in \Re^m$ if they are orthonormal and span $\mathcal{S}$
- **Definition 4:** A matrix $\mathbf{Q} \in \Re^{m \times m}$ is *orthogonal* if

$$\mathbf{Q}^T\mathbf{Q} = \mathbf{I} \tag{2}$$

  - Thus, $\mathbf{Q}^{-1} = \mathbf{Q}^T$
  - The columns of $\mathbf{Q}$ form an orthonormal basis for $\Re^m$
- $||\ ||_2$ is invariant under orthogonal transformations
  - If $\mathbf{Q}$ is orthogonal

$$||\mathbf{Qx}||_2^2 = \mathbf{x}^T\mathbf{Q}^T\mathbf{Qx} = \mathbf{x}^T\mathbf{x} = ||\mathbf{x}||_2^2$$

- The matrix 2 and F norms are also invariant under orthogonal transformations
  - If $\mathbf{Q}$ and $\mathbf{z}$ are orthogonal

$$||\mathbf{QAZ}||_\mathbf{2} = ||\mathbf{A}||_\mathbf{2}, \qquad ||\mathbf{QAZ}||_\mathbf{F} = ||\mathbf{A}||_\mathbf{F} \tag{3}$$

    * The proof follows that for a vector norm

The Singular Value Decomposition

- Reading: Trefethen and Bau (1997), Lectures 4 and 5
- **Theorem 2 (Singular Value Decomposition (SVD):)** Let $\mathbf{A} \in \Re^{m \times n}$, then there exists orthogonal matrices $\mathbf{U} \in \Re^{m \times m}$ and $\mathbf{V} \in \Re^{n \times n}$ such that

$$\mathbf{U}^T \mathbf{A} \mathbf{V} = \mathbf{\Sigma} \tag{4a}$$

  - $\mathbf{\Sigma} \in \Re^{m \times n}$

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_p \\ & & & \\ & & & \end{bmatrix} \tag{4b}$$

  - $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p$, $p = min(m, n)$

- *Proof:* Let

$$\mathbf{A}\mathbf{x} = \sigma \mathbf{y}, \qquad ||\mathbf{x}||_2 = ||\mathbf{y}||_2 = 1, \qquad ||\mathbf{A}||_2 = \sigma \tag{5}$$

  - Define orthogonal matrices $\mathbf{U}$ and $\mathbf{V}$ such that

$$\mathbf{U} = [\mathbf{y}, \mathbf{u}_2, \cdots, \mathbf{u}_m] = [\mathbf{y}, \mathbf{U}_1]$$

$$\mathbf{V} = [\mathbf{x}, \mathbf{v}_2, \cdots, \mathbf{v}_n] = [\mathbf{x}, \mathbf{V}_1]$$

    * Any orthonormal set of vectors can be extended to form an orthonormal basis

SVD

– Construct

$$\mathbf{U}^T\mathbf{A}\mathbf{V} = \begin{bmatrix} \mathbf{y}^T \\ \mathbf{U}_1^T \end{bmatrix} \mathbf{A} \begin{bmatrix} \mathbf{x} & \mathbf{V}_1 \end{bmatrix} = \begin{bmatrix} \mathbf{y}^T\mathbf{A}\mathbf{x} & \mathbf{y}^T\mathbf{A}\mathbf{V}_1 \\ \mathbf{U}_1^T\mathbf{A}\mathbf{x} & \mathbf{U}_1^T\mathbf{A}\mathbf{V}_1 \end{bmatrix}$$

– Use (5)

* $\mathbf{y}^T\mathbf{A}\mathbf{x} = \sigma$
* $\mathbf{U}_1^T\mathbf{A}\mathbf{x} = \sigma\mathbf{U}_1^T\mathbf{y} = 0$

– Thus

$$\mathbf{U}^T\mathbf{A}\mathbf{V} = \begin{bmatrix} \sigma & \mathbf{y}^T\mathbf{A}\mathbf{V}_1 \\ \mathbf{0} & \mathbf{U}_1^T\mathbf{A}\mathbf{V}_1 \end{bmatrix} := \begin{bmatrix} \sigma & \mathbf{w}^T \\ \mathbf{0} & \mathbf{B} \end{bmatrix} := \mathbf{A_1}$$

– Select $\mathbf{z}^T = [\sigma, \mathbf{w}^T]$ and compute

$$\mathbf{A}_1\mathbf{z} = \begin{bmatrix} \sigma & \mathbf{w}^T \\ \mathbf{0} & \mathbf{B} \end{bmatrix} \begin{bmatrix} \sigma \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} \sigma^2 + \mathbf{w}^T\mathbf{w} \\ \mathbf{B}\mathbf{w} \end{bmatrix}$$

– Then

$$||\mathbf{A}_1\mathbf{z}||_2^2 = (\sigma^2 + \mathbf{w}^T\mathbf{w})^2 + \mathbf{w}^T\mathbf{B}^T\mathbf{B}\mathbf{w} \geq (\sigma^2 + \mathbf{w}^T\mathbf{w})^2$$

– Thus,

$$||\mathbf{A}_1||_2^2 \geq \frac{||\mathbf{A}_1\mathbf{z}||_2^2}{||\mathbf{z}||_2^2} \geq \sigma^2 + \mathbf{w}^T\mathbf{w}$$

SVD

– Since $\mathbf{U}$ and $\mathbf{V}$ are orthogonal, use (3) to obtain

$$\sigma^2 = ||\mathbf{A}||_2^2 = ||\mathbf{U}^T\mathbf{A}\mathbf{V}||_2^2 = ||\mathbf{A}_1||_2^2$$

– Thus, $\mathbf{w} = 0$

– An induction argument completes the proof □

- $\sigma_1, \sigma_2, \cdots, \sigma_p$ are the *singular values* of $\mathbf{A}$
  - Let $\mathbf{u}_i$ and $\mathbf{v}_i$ be the $i\ th$ columns of $\mathbf{U}$ and $\mathbf{V}$
    - $\mathbf{u}_i$ and $\mathbf{v}_i$ are the left and right *singular vectors*
    - The singular vectors satisfy

$$\mathbf{A}\mathbf{v}_i = \sigma_i\mathbf{u}_i, \qquad \mathbf{A}\mathbf{u}_i = \sigma_i\mathbf{v}_i, \qquad i = 1:p \tag{6}$$

- Some properties of singular values
  - If $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_p = 0$, then

$$\text{rank}(\mathbf{A}) = r \tag{7a}$$

$$\text{null}(\mathbf{A}) = \text{span}\{\mathbf{v}_{r+1}, \mathbf{v}_{r+2}, \cdots, \mathbf{v}_n\} \tag{7b}$$

$$\text{range}(\mathbf{A}) = \text{span}\{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_r\} \tag{7c}$$

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_i\mathbf{u}_i\mathbf{v}_i^T \tag{7d}$$

$$||\mathbf{A}||_F^2 = \sigma_1^2 + \sigma_2^2 + \cdots + \sigma_p^2, \qquad p = \min(m,n) \tag{7e}$$

$$||\mathbf{A}||_2 = \sigma_1 \tag{7f}$$

Orthogonal Projections

- Reading: Trefethen and Bau (1997), Lecture 6
- **Definition 5:** $\mathbf{P} \in \Re^{n \times n}$ is the *orthogonal projection* onto a subspace $\mathcal{S} \subset \Re^{n \times n}$ if

$$\text{range}(\mathbf{P}) = \mathcal{S}, \qquad \mathbf{P}^2 = \mathbf{P}, \qquad \mathbf{P}^T = \mathbf{P} \tag{8}$$

  - If $\mathbf{x} \in \Re^n$ and $\mathbf{P}\mathbf{x} \in \mathcal{S}$ then $(\mathbf{I} - \mathbf{P})\mathbf{x}$ is in the orthogonal complement of $\mathcal{S}$
  - The orthogonal projection onto a subspace is unique
- Let the columns of $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_k]$ form an orthonormal basis for a subspace $\mathcal{S}$
  - $\mathbf{P} = \mathbf{V}\mathbf{V}^T$ is the unique orthogonal projection onto $\mathcal{S}$
    - $\mathbf{V}\mathbf{V}^T$ is symmetric
    - $\mathbf{P}^2 = \mathbf{V}\mathbf{V}^T\mathbf{V}\mathbf{V}^T = \mathbf{V}\mathbf{V}^T$ by orthogonality
    - If $\mathbf{x} \in \Re^n$ then $\mathbf{y} = \mathbf{P}\mathbf{x} = \mathbf{V}(\mathbf{V}^T\mathbf{x})$
    - $\mathcal{S} = \text{span}\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_k\}$, $\text{range}(\mathbf{P}) = \mathcal{S}$

Orthogonal Projections

- Some Properties:
  - Let rank($\mathbf{A}$) = r and partition $\mathbf{U}$ and $\mathbf{V}$

$$\mathbf{U} = [\mathbf{U}_r, \mathbf{U}_{m-r}], \qquad \mathbf{V} = [\mathbf{V}_r, \mathbf{V}_{n-r}] \tag{9}$$

    - $\mathbf{U}_r \in \Re^{m \times r}$, $\mathbf{U}_{m-r} \in \Re^{m \times (m-r)}$
    - $\mathbf{V}_r \in \Re^{n \times r}$, $\mathbf{V}_{n-r} \in \Re^{n \times (n-r)}$
  - $\mathbf{U}_r \mathbf{U}_r^T$ is a projection onto range($\mathbf{A}$)
  - $\mathbf{U}_{m-r} \mathbf{U}_{m-r}^T$ is a projection onto null($\mathbf{A}^T$)
    - the orthogonal complement of range($\mathbf{A}$)
  - $\mathbf{V}_r \mathbf{V}_r^T$ is a projection onto range($\mathbf{A}^T$)
    - the orthogonal complement of null($\mathbf{A}$)
  - $\mathbf{V}_{n-r} \mathbf{V}_{n-r}^T$ is a projection onto null($\mathbf{A}$)

Orthogonal Projections

- *Example 1.* The singular value decomposition of

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 3 \\ 1 & 0 & 5 \\ 3 & 1 & 8 \end{bmatrix}$$

is

$$\mathbf{U} = \begin{bmatrix} 0.3393 & -0.7427 & -0.5774 \\ 0.4735 & 0.6652 & -0.5774 \\ 0.8128 & -0.0775 & 0.5774 \end{bmatrix}$$

$$\mathbf{\Sigma} = \begin{bmatrix} 10.5826 & 0 & 0 \\ 0 & 1.4174 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} 0.3393 & -0.7427 & 0.5774 \\ 0.1089 & -0.5786 & -0.8083 \\ 0.9344 & 0.3371 & -0.1155 \end{bmatrix}$$

- With $\sigma_3 = 0$ we have $r = 2$
- Verify that $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, $\mathbf{V}^T\mathbf{V} = \mathbf{I}$

Orthogonal Projections

- Partition $\mathbf{U}$ and $\mathbf{V}$

$$\mathbf{U}_r = \begin{bmatrix} 0.3393 & -0.7427 \\ 0.4735 & 0.6652 \\ 0.8128 & -0.0775 \end{bmatrix}, \qquad \mathbf{U}_{m-r} = \begin{bmatrix} -0.5774 \\ -0.5774 \\ 0.5774 \end{bmatrix}$$

$$\mathbf{V}_r = \begin{bmatrix} 0.3393 & -0.7427 \\ 0.1089 & -0.5786 \\ 0.9344 & 0.3371 \end{bmatrix}, \qquad \mathbf{V}_{n-r} = \begin{bmatrix} 0.5774 \\ -0.8083 \\ -0.1155 \end{bmatrix}$$

- The projections

$$\mathbf{U}_r\mathbf{U}_r^T = \begin{bmatrix} 0.6667 & -0.3333 & 0.3333 \\ -0.3333 & 0.6667 & 0.3333 \\ 0.3333 & 0.3333 & 0.6667 \end{bmatrix}$$

$$\mathbf{U}_{m-r}\mathbf{U}_{m-r}^T = \begin{bmatrix} 0.3333 & 0.3333 & -0.3333 \\ 0.3333 & 0.3333 & -0.3333 \\ -0.3333 & -0.3333 & 0.3333 \end{bmatrix}$$

$$\mathbf{V}_r\mathbf{V}_r^T = \begin{bmatrix} 0.6667 & 0.4667 & 0.0667 \\ 0.4667 & 0.3467 & -0.0933 \\ 0.0667 & -0.0933 & 0.9867 \end{bmatrix}$$

$$\mathbf{V}_{n-r}\mathbf{V}_{n-r}^T = \begin{bmatrix} 0.3333 & -0.4667 & -0.0667 \\ -0.4667 & 0.6533 & 0.0933 \\ -0.0667 & 0.0933 & 0.0133 \end{bmatrix}$$

Orthogonal Projections

- Verify that $\mathbf{U}_r^T\mathbf{U}_r = \mathbf{I}$, etc
- Verify that
  - $\mathbf{U}_r\mathbf{U}_r^T + \mathbf{U}_{n-r}\mathbf{U}_{n-r}^T = \mathbf{I}$
  - $\mathbf{V}_r\mathbf{V}_r^T + \mathbf{V}_{n-r}\mathbf{V}_{n-r}^T = \mathbf{I}$
- Choose $\mathbf{x} = [1, 0, 0]^T$ and calculate

$$\mathbf{y} = \mathbf{V}_{n-r}\mathbf{V}_{n-r}^T\mathbf{x} = \begin{bmatrix} 0.3333 \\ -0.4667 \\ -0.0667 \end{bmatrix}$$

  - Verify that $\mathbf{A}\mathbf{y} = 0$
  - Verify that $\mathbf{A}\mathbf{V}_{n-r}\mathbf{V}_{n-r}^T = \mathbf{0}$
- Similarly, verify that $\mathbf{A}^T\mathbf{U}_{m-r}\mathbf{U}_{m-r}^T = \mathbf{0}$

4.2. Rotations and Reflections

- Reading: Trefethen and Bau (1997), Lecture 10
- *Example 1.* Consider

$$\mathbf{Q} = \begin{bmatrix} \cos\theta & -\sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

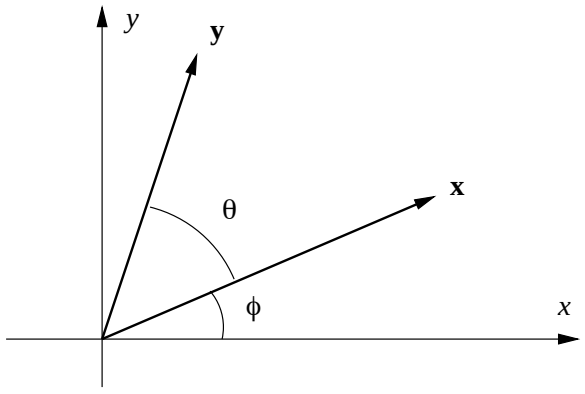
  - $\mathbf{Q}$ is orthogonal since $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$
  - $\mathbf{y} = \mathbf{Q}^T\mathbf{x}$ is a rotation of $\mathbf{x}$ through θ degrees
    - Let

$$\mathbf{x} = ||\mathbf{x}||_2[\cos\phi,\ \sin\phi]^T$$

    - Multiply

$$\begin{aligned} \mathbf{y} &= ||\mathbf{x}||_2 \begin{bmatrix} \cos\theta & -\sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} \\ &= ||\mathbf{x}||_2 \begin{bmatrix} \cos(\theta+\phi) \\ \sin(\theta+\phi) \end{bmatrix} \end{aligned}$$

  - $\mathbf{Q}$ is called a *rotation*



Reflections

- *Example 2.* Consider

$$\mathbf{Q} = \begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix}$$

  - $\mathbf{Q}$ is orthogonal since $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$
  - $\mathbf{y} = \mathbf{Q}^T\mathbf{x} = \mathbf{Q}\mathbf{x}$ is a reflection of $\mathbf{x}$ across the line

$$\mathbf{z} = C[\cos\theta/2,\ \sin\theta/2]^T$$

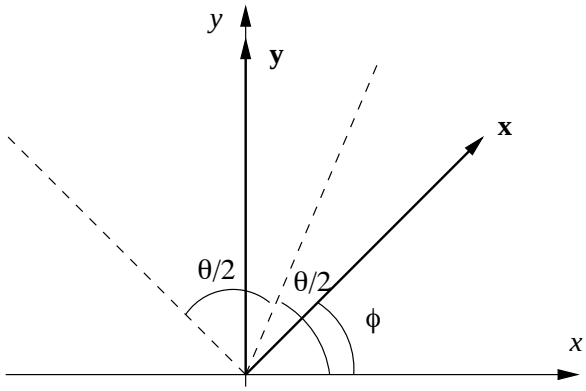
    * Let

$$\mathbf{x} = ||\mathbf{x}||_2[\cos\phi,\ \sin\phi]^T$$

    * Multiply

$$\begin{aligned} \mathbf{y} &= ||\mathbf{x}||_2 \begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix} \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} \\ &= ||\mathbf{x}||_2 \begin{bmatrix} \cos(\theta-\phi) \\ \sin(\theta-\phi) \end{bmatrix} \end{aligned}$$

  - $\mathbf{Q}$ is called a *reflection*



Householder Reflections

- **Definition 1:** A *reflection* $\mathbf{P} \in \Re^{n \times n}$ has the form

$$\mathbf{P} = \mathbf{I} - 2\frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}}, \qquad \mathbf{v} \in \Re^n \tag{1}$$

  - Also called a
    - Plane reflection
    - Householder reflection
    - Householder matrix

- Some Properties of Householder Matrices:
- $p_{ij} = \delta_{ij} - 2v_iv_j/(\mathbf{v}^T\mathbf{v})$, δ_{ij} is the Kronecker delta
- $\mathbf{P}$ is symmetric
- $\mathbf{P}$ is orthogonal

$$\mathbf{P}^T\mathbf{P} = \mathbf{P}^2 = (\mathbf{I} - 2\frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}})(\mathbf{I} - 2\frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}})$$

$$= \mathbf{I} - 4\frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}} + 4\frac{\mathbf{v}(\mathbf{v}^T\mathbf{v})\mathbf{v}^T}{(\mathbf{v}^T\mathbf{v})^2} = \mathbf{I}$$

  - Thus, $\mathbf{P} = \mathbf{P}^T = \mathbf{P}^-1$

- $\mathbf{P}$ is a rank-1 modification of $\mathbf{I}$

Reflections

- $\mathbf{P}$ can be constructed annihilate components of a vector
  - Given $\mathbf{x}$, construct $\mathbf{P}$ such that $\mathbf{Px} = c\mathbf{e}_1$
  - Consider

$$\mathbf{Px} = \left(\mathbf{I} - 2\frac{\mathbf{vv}^T}{\mathbf{v}^T\mathbf{v}}\right)\mathbf{x} = \mathbf{x} - 2\frac{\mathbf{v}^T\mathbf{x}}{\mathbf{v}^T\mathbf{v}}\mathbf{v}$$

  - Select $\mathbf{v} = \mathbf{x} + \alpha\mathbf{e}_1$

$$\mathbf{v}^T\mathbf{x} = \mathbf{x}^T\mathbf{x} + \alpha x_1$$

$$\mathbf{v}^T\mathbf{v} = \mathbf{x}^T\mathbf{x} + 2\alpha x_1 + \alpha^2$$

  - Thus,

$$\mathbf{Px} = \mathbf{x} - 2\frac{\mathbf{x}^T\mathbf{x} + \alpha x_1}{\mathbf{x}^T\mathbf{x} + 2\alpha x_1 + \alpha^2}(\mathbf{x} + \alpha\mathbf{e}_1)$$

  - For $\mathbf{Px}$ to be in the direction of $\mathbf{e}_1$ we want

$$1 - 2\frac{\mathbf{x}^T\mathbf{x} + \alpha x_1}{\mathbf{x}^T\mathbf{x} + 2\alpha x_1 + \alpha^2} = \frac{\alpha^2 - \mathbf{x}^T\mathbf{x}}{\mathbf{x}^T\mathbf{x} + 2\alpha x_1 + \alpha^2} = \mathbf{0}$$

  - Thus, $\alpha = \pm||\mathbf{x}||_2$

Reflections

- In summary, selecting

$$\mathbf{v} = \mathbf{x} \pm ||\mathbf{x}||_2 \mathbf{e}_1 \tag{2a}$$

  Makes

$$\mathbf{P}\mathbf{x} = -2\alpha \frac{\mathbf{x}^T\mathbf{x} + \alpha x_1}{\mathbf{x}^T\mathbf{x} + 2\alpha x_1 + \alpha^2}\mathbf{e}_1 = -(\pm||\mathbf{x}||_2)\mathbf{e}_1 \tag{2b}$$

- What's this $\pm$ thing?
  - If $\mathbf{x}$ is nearly parallel to $\mathbf{e}_1$ then

$$\mathbf{v} = \mathbf{x} - ||\mathbf{x}||_2 \mathbf{e}_1$$

    has a small norm which could magnify roundoff errors
  - To minimize this, choose

$$\mathbf{v} = \mathbf{x} + \text{sign}(x_1)||\mathbf{x}||_2 \mathbf{e}_1 \tag{2c}$$

- A program for Householder vectors

```
function v = house(x)
% house: Compute the Householder vector v that
% annihilates all but the first component of x.

  n = length(x)
  v = x;
  v(1) = x(1) + sign(x(1))*norm (x);
```

Reflections

- *Example 3* (cf. Golub and Van Loan (1993), Problem 5.1.1). Let $\mathbf{x} = [1,\ 7,\ 2,\ 3,\ -1]^T$. Find $\mathbf{P}$
  - $||\mathbf{x}||_2 = 8$
  - $\mathbf{v} = [9,\ 7,\ 2,\ 3,\ -1]^T$
  - The reflection is

$$\mathbf{P} = \mathbf{I} - 2\frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}} = \mathbf{I} - \frac{1}{72}\begin{bmatrix} 81 & 63 & 18 & 27 & -9 \\ 63 & 49 & 14 & 21 & -7 \\ 18 & 14 & 4 & 6 & -2 \\ 27 & 21 & 6 & 9 & -3 \\ -9 & -7 & -2 & -3 & 1 \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} -0.1250 & -0.8750 & -0.2500 & -0.3750 & 0.1250 \\ -0.8750 & 0.3194 & -0.1944 & -0.2917 & 0.0972 \\ -0.2500 & -0.1944 & 0.9444 & -0.0833 & 0.0278 \\ -0.3750 & -0.2917 & -0.0833 & 0.8750 & 0.0417 \\ 0.1250 & 0.0972 & 0.0278 & 0.0417 & 0.9861 \end{bmatrix}$$

  - The result is $\mathbf{P}\mathbf{x} = [-8,\ 0,\ 0,\ 0,\ 0]^T$

Reflections

- $\mathbf{P}$ never needs explicit generation and storage
  - Application of $\mathbf{P}$ to an $n \times n$ matrix $\mathbf{A}$ would require $O(n^3)$ FLOPs
  - Compute

$$\mathbf{PA} = \left(\mathbf{I} - 2\frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}}\right)\mathbf{A} = \mathbf{A} + \mathbf{v}\mathbf{w}^T \tag{3a}$$

    where

$$\mathbf{w} = \beta\mathbf{A}^T\mathbf{v}, \qquad \beta = -2/\mathbf{v}^T\mathbf{v} \tag{3b}$$

    * This requires $O(n^2)$ operations

- Function for a pre-Householder update

```
function A = row.house(A, v)
% row.house: Overwrite A ∈ ℜ^{m×n} with the product
% PA where P = I − 2vv^T/(v^Tv) is a Householder
% matrix and v is an m-vector.

β = −2/(v^T ∗ v);
w = β ∗ A^T ∗ v;
A = A + v ∗ w^T;
```

Reflections

- Function for a post-Householder update

  function $\mathbf{A} = \text{col.house}(\mathbf{A}, \mathbf{v})$
  % col.house: Overwrite $\mathbf{A} \in \Re^{m \times n}$ with the product
  % $\mathbf{AP}$ where $\mathbf{P} = \mathbf{I} - 2\mathbf{v}\mathbf{v}^T/(\mathbf{v}^T\mathbf{v})$ is a Householder
  % matrix and $\mathbf{v}$ is an n-vector.

  $\beta = -2/(\mathbf{v}^T * \mathbf{v})$;
  $\mathbf{w} = \beta * \mathbf{A} * \mathbf{v}$;
  $\mathbf{A} = \mathbf{A} + \mathbf{w} * \mathbf{v}^T$;

- *Example 4.* Let $\mathbf{A} \in \Re^{m \times n}$. Compute $\mathbf{Q}$ such that $\mathbf{B} = \mathbf{QA}$ has $b_{kj} = 0$, $k = j + 1 : m$
  - Partition $\mathbf{A}$

    $$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$$

    * where $\mathbf{A}_{11} \in \Re^{(j-1)\times(j-1)}$, $\mathbf{A}_{12} \in \Re^{(j-1)\times(n-j)}$, $\mathbf{A}_{21} \in \Re^{(m-j)\times(j-1)}$, and $\mathbf{A}_{22} \in \Re^{(m-j)\times(n-j)}$
  - Let

    $$\mathbf{Q} = \begin{bmatrix} \mathbf{I}_{j-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{P} \end{bmatrix} = \mathbf{I} - 2\frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}}, \qquad \mathbf{v} = \begin{bmatrix} \mathbf{0} \\ \tilde{\mathbf{v}} \end{bmatrix}$$

Reflections

- Multiply

$$\mathbf{B} = \mathbf{QA} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{PA}_{21} & \mathbf{PA}_{22} \end{bmatrix}$$

- A typical case is to apply the transform to each column with a goal of making $\mathbf{B}$ upper triangular
  - In this case, $\mathbf{A}_{21} = \mathbf{0}$
- The following code segment does the job

$\mathbf{v}(j:m) = \text{house}(\mathbf{A}(j:m, j));$
$\mathbf{A}(j:m, j, n) = \text{row.house}(\mathbf{A}(j:m, j:n), \mathbf{v}(j:m));$
$\mathbf{A}(j+1:m, j) = \mathbf{v}(j+1:m);$

  - The first line creates the nonzero portion of $\mathbf{v}$ that zeros b_{kj}, $k = j+1:m$
  - The second line applies the transformation $\mathbf{Q}$ to $\mathbf{A}_{22}$ and stores the result back in $\mathbf{A}$
  - The third line stores the nonzero portion of $\mathbf{v}$ in the zeroed portion of the $j\ th$ column of $\mathbf{A}$

Reflections

- Householder transformations are often applied in sequence

$$\mathbf{Q} = \mathbf{Q}_1 \mathbf{Q}_2 \cdots \mathbf{Q}_r$$

  where

$$\mathbf{Q}_j = \mathbf{I} - 2\frac{\mathbf{v}^{(j)}\left(\mathbf{v}^{(j)}\right)^T}{\left(\mathbf{v}^{(j)}\right)^T \mathbf{v}^{(j)}}, \qquad \mathbf{v}^{(j)} = \begin{bmatrix} \mathbf{0} \\ \tilde{\mathbf{v}}^{(j)} \end{bmatrix}$$

  - The dimension $\tilde{\mathbf{v}}^{(j)}$ is $m - j + 1$

- It is typically not necessary to store $\mathbf{Q}$, but rather store the Householder vectors $\mathbf{v}^{(j)}$, $j = 1 : r$
  - Suppose $\mathbf{v}^{(j)}$ is stored in the lower triangular part of a matrix $\mathbf{A}$
  - The following code segment overwrites $\mathbf{C}$ with $\mathbf{QC}$

```
for j = 1 : r
    v(j+1 : n) = A(j+1 : n, j);
    v(j) = 1;
    C(j : n, 1 : n) = row.house(C(j : n, j : n), v(j : n));
end
```

  - This can be done in $O(n^2 r)$ operations.
    - Direct computation of $\mathbf{Q}$ requires $O(n^3)$ operations

Rotations

- **Definition 2:** A *rotation* $\mathbf{G}(i,j,\theta) \in \Re^{n\times n}$ has the form

$$\mathbf{G}(i,j,\theta) = \begin{bmatrix} 1 & & 0 & & 0 & & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & & 0 & & 0 & & 1 \end{bmatrix} \begin{matrix} \\ \\ i \\ \\ j \\ \\ \\ \end{matrix} \qquad (4a)$$

$$\begin{matrix} & & i & & j & & \end{matrix}$$

$$c = \cos\theta, \qquad s = \sin\theta \qquad (4b)$$

  - The transformation is called a
    - plane rotation
    - Givens rotation

- Some Properties of Givens Rotations
  - $\mathbf{G}(i,j,\theta)$ is orthogonal
  - $\mathbf{G}(i,j,\theta)$ is a rank-2 modification of $\mathbf{I}$
  - The multiplication $\mathbf{G}(i,j,\theta)^T\mathbf{x}$ is a counterclockwise rotation of $\mathbf{x}$ in the i,j plane

Rotations

- If $\mathbf{x} \in \Re^n$ then

$$\mathbf{y} = \mathbf{G}(i,j,\theta)^T\mathbf{x}, \qquad y_k = \begin{cases} cx_i - sx_j, & \text{if } k = i \\ sx_i + cx_j, & \text{if } k = j \\ x_k, & \text{otherwise} \end{cases} \tag{5a}$$

- If we want $y_j = 0$ then

$$c = \frac{x_i}{\sqrt{x_i^2 + x_j^2}}, \qquad s = \frac{-x_j}{\sqrt{x_i^2 + x_j^2}} \tag{5b}$$

- Calculation of a Givens rotation

```
function [c, s] = givens(a, b)
% givens: Compute the Givens transformation c, s such
% that ca - sb = r and sa + cb = 0.

  if b = 0
     c = 1;
     s = 0;
  elseif | b | > | a |
     tau = -a/b;
     s = 1/ sqrt(1 + tau^2);
     c = s*tau;
  else
     tau = -b/a;
     c = 1/ sqrt(1 + tau^2);
     s = c*tau;
  end
```

Rotations

- Function for a pre-Givens rotation
  - The multiplication only affects rows i and j
  - Regard $\mathbf{A}$ as a $2 \times q$ matrix and compute

$$\mathbf{B} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \mathbf{A}$$

    - $\mathbf{B}$ is $2 \times q$
  - Overwrite $\mathbf{B}$ with $\mathbf{A}$

```
function A = row.rot(A, c, s)
% row.rot: Compute the rotation G(i,j,θ)A where
% A ∈ ℜ^{2×q}. The rotation is stored in A.

   [m, q] = size(A);
   for k = 1:q
      tau = A(1,k);
      sigma = A(2,k);
      A(1,k) = c*tau - s*sigma;
      A(2,k) = s*tau + c*sigma;
   end
```

Rotations

- Function for a post-Givens rotation
  - The multiplication only affects columns i and j

function $\mathbf{A}$ = col.rot($\mathbf{A}$, c, s)
% col.rot: Compute the rotation $\mathbf{A}\mathbf{G}^T(i, j, \theta)$ where
% $\mathbf{A} \in \Re^{q \times 2}$. The rotation is stored in $\mathbf{A}$.

```
[m,q] = size(A);
for k = 1:q
   tau = A(k,1);
   sigma = A(k,2);
   A(k,1) = c*tau - s*sigma;
   A(k,2) = s*tau + c*sigma;
end
```

- Each algorithm requires $4q$ multiplications and $2q$ additions
- Let $\mathbf{A} \in \Re^{m \times n}$ be such that $a_{kl} = 0$ $k = i-1 : i$, $l = 1 : j-1$
  - To make $a_{ij} = 0$ by rotating rows $i-1$ and i

```
[c,s] = givens(A(i-1,j), A(i,j))
A (i-1:i,j:n) = row.rot(A (i-1:i,j:n), c, s)
```

Rotations

- *Example 5.* Set $a_{33} = 0$ by a rotation of rows 2 and 3 when

$$\mathbf{A} = \begin{bmatrix} 3 & 2 & 6 & 7 \\ 0 & 0 & 5 & 1 \\ 0 & 0 & 12 & -3 \end{bmatrix}$$

  - Using the function [c, s] = givens(5,12) yields

$$c = -0.3846, \qquad s = 0.9231$$

    * Observe that $c^2 + s^2 = 1$

  - Performing $\mathbf{A}$ (2:3, 3:4) = row.rot ($\mathbf{A}$(2:3, 3:4), c, s) yields

$$\mathbf{A} = \begin{bmatrix} 3 & 2 & 6 & 7 \\ 0 & 0 & -13 & 2.3846 \\ 0 & 0 & 0 & 2.0769 \end{bmatrix}$$

4.3. The QR Reduction

- Reading Trefethen and Bau (1997), Lecture 7
- The **QR** factorization of a matrix $\mathbf{A} \in \Re^{m \times n}$ is

$$\mathbf{A} = \mathbf{QR} \tag{1}$$

  - $\mathbf{Q} \in \Re^{m \times m}$ is an orthogonal matrix
  - $\mathbf{R} \in \Re^{m \times n}$ is upper triangular
  - Assume (for the moment) $m \geq n$
- *Motivation:*
  - An alternate to Gaussian elimination when solving

$$\mathbf{Ax} = \mathbf{b}$$

    - More expensive than Gaussian elimination
  - Determining bases for orthogonal subspaces
  - Determining numerical rank through the SVD
  - Solving least squares problems

$$\min_{\mathbf{x}} ||\mathbf{Ax} - \mathbf{b}||_2$$

  - Solving algebraic eigenvalue problems

$$\mathbf{Ax} = \lambda \mathbf{x}$$

Householder QR Factorization

- Use Householder reflections to reduce successive columns of $\mathbf{A}$ to zero below their main diagonals
  - Suppose $m = 6$, $n = 4$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \\ a_{51} & a_{52} & a_{53} & a_{54} \\ a_{61} & a_{62} & a_{63} & a_{64} \end{bmatrix}$$

  - Construct a reflection $\mathbf{H}_1$ such that

$$\mathbf{H}_1\mathbf{A} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & a_{34}^{(1)} \\ 0 & a_{42}^{(1)} & a_{43}^{(1)} & a_{44}^{(1)} \\ 0 & a_{52}^{(1)} & a_{53}^{(1)} & a_{54}^{(1)} \\ 0 & a_{62}^{(1)} & a_{63}^{(1)} & a_{64}^{(1)} \end{bmatrix}$$

Householder QR Factorization

- Use a reflection to annihilate $a_{i2}^{(1)}$, $i = 3:6$

$$\mathbf{P}_2 \begin{bmatrix} a_{22}^{(1)} \\ a_{32}^{(1)} \\ a_{42}^{(1)} \\ a_{52}^{(1)} \\ a_{62}^{(1)} \end{bmatrix} = \begin{bmatrix} r_{22} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

  - Apply the transformation

$$\mathbf{H}_2 = \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_2 \end{bmatrix}$$

    to get

$$\mathbf{H}_2\mathbf{H}_1\mathbf{A} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ 0 & r_{22} & r_{23} & r_{24} \\ 0 & 0 & a_{33}^{(2)} & a_{34}^{(2)} \\ 0 & 0 & a_{43}^{(2)} & a_{44}^{(2)} \\ 0 & 0 & a_{53}^{(2)} & a_{54}^{(2)} \\ 0 & 0 & a_{63}^{(2)} & a_{64}^{(2)} \end{bmatrix}$$

Householder QR Factorization

- Repeat the process on the third column

$$\mathbf{H}_3\mathbf{H}_2\mathbf{H}_1\mathbf{A} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ 0 & r_{22} & r_{23} & r_{24} \\ 0 & 0 & r_{33} & r_{34} \\ 0 & 0 & 0 & a_{44}^{(2)} \\ 0 & 0 & 0 & a_{54}^{(2)} \\ 0 & 0 & 0 & a_{64}^{(2)} \end{bmatrix}$$

- One more should do it

$$\mathbf{H}_4\mathbf{H}_3\mathbf{H}_2\mathbf{H}_1\mathbf{A} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ 0 & r_{22} & r_{23} & r_{24} \\ 0 & 0 & r_{33} & r_{34} \\ 0 & 0 & 0 & r_{44} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Householder QR Factorization

- Algorithm for a Householder **QR** factorization

function $\mathbf{A}$ = houseqr($\mathbf{A}$)
% houseqr: Overwrite $\mathbf{A} \in \Re^{m \times n}$ with the product
% $\mathbf{QR}$ where $\mathbf{Q} \in \Re^{m \times m}$ is orthogonal and
% $\mathbf{R} \in \Re^{m \times n}$ is upper triangular. On return,
% the essential part of $\mathbf{Q}$ is stored in the lower triangular
% part of $\mathbf{A}$ and $\mathbf{R}$ is stored in the upper triangular part.

for $j = 1 : n$
  $\mathbf{v}$(j:m) = house($\mathbf{A}$(j:m,j));
  $\mathbf{A}$(j:m,j:n) = row.house($\mathbf{A}$(j:m,j:n), $\mathbf{v}$(j:m));
  if $j < m$
    $\mathbf{A}$(j+1:m,j) = $\mathbf{v}$(j+1:m);
  end
end

Householder QR Factorization

- As described in Example 2.4,

$$\mathbf{H}_j = \mathbf{I} - 2\frac{\mathbf{v}^{(j)}\left(\mathbf{v}^{(j)}\right)^T}{\left(\mathbf{v}^{(j)}\right)^T\mathbf{v}^{(j)}} \tag{2a}$$

  - $\mathbf{v}^{(j)}$ is the Householder vector $\mathbf{v}$ used at step j
  - $\mathbf{v}^{(j)}$ has the form

$$\mathbf{v}^{(j)} = [0, 0, ..., 0, 1, v_{j+1}^{(j)}, ..., v_m^{(j)}]^T \tag{2b}$$

- Normalize the Householder vector so that $v_j^{(j)} = 1$
  - This requires modification of the function *house* of Section 4.2. All components of $\mathbf{v}$ are divided by $v(1)$
  - The renormalization saves a storage location and allows $\mathbf{v}^{(j)}$ to be stored in the lower part of $\mathbf{A}$

- Upon completion of the algorithm

$$\mathbf{A} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ v_2^{(1)} & r_{22} & r_{23} & r_{24} \\ v_3^{(1)} & v_3^{(2)} & r_{33} & r_{34} \\ v_4^{(1)} & v_4^{(2)} & v_4^{(3)} & r_{44} \\ v_5^{(1)} & v_5^{(2)} & v_5^{(3)} & v_5^{(4)} \\ v_6^{(1)} & v_6^{(2)} & v_6^{(3)} & v_6^{(4)} \end{bmatrix} \tag{2c}$$

Householder QR Factorization

- $\mathbf{Q}^T = \mathbf{H}_n \mathbf{H}_{n-1} \cdots \mathbf{H}_1$ so

$$\mathbf{Q} = \mathbf{H}_1 \mathbf{H}_2 \cdots \mathbf{H}_n \tag{3}$$

  - $\mathbf{Q}$ rarely needs explicit construction

- The factorization requires about $2n^2(m - n/3)$ FLOPs (multiplications plus additions)
  - With $m = n$ we require about $4n^3/3$ FLOPs
  - Gaussian elimination requires about $2n^3/3$ FLOPs

- With roundoff, we compute

$$\hat{\mathbf{Q}}^T(\mathbf{A} + \delta\mathbf{A}) = \hat{\mathbf{R}}$$

  - $\hat{\mathbf{Q}}$ is *exactly* orthogonal
  - $\hat{\mathbf{R}}$ is upper triangular
  - The perturbation $||\delta\mathbf{A}||_2 \approx u||\mathbf{A}||_2$
  - This is acceptable

Givens QR Factorization

- We can also use Givens rotations to construct the **QR** decomposition
  - The order of the reduction is
  - $\times$ stands for a nonzero entry
  - The numbers indicate the rows that are rotated

$$\mathbf{A} = \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} = \begin{matrix} \\ \\ \\ 3 \\ 4 \end{matrix} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix} = \begin{matrix} 2 \\ 3 \\ \\ \\ \end{matrix} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix}$$

$$\mathbf{A} = \begin{matrix} 1 \\ 2 \\ \\ \\ \end{matrix} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix} = \begin{matrix} \\ \\ 3 \\ 4 \end{matrix} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix} = \begin{matrix} 2 \\ 3 \\ \\ \\ \end{matrix} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & \times \end{bmatrix}$$

$$\mathbf{A} = \begin{matrix} \\ \\ 3 \\ 4 \end{matrix} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & 0 \end{bmatrix}$$

Givens QR Factorization

- Algorithm for the Givens **QR** factorization

  function $\mathbf{A}$ = givensqr($\mathbf{A}$)
  % givensqr: Reduce $\mathbf{A} \in \Re^{m \times n}$ to upper triangular
  % form by the Givens **QR** procedure where $\mathbf{Q} \in \Re^{m \times m}$
  % is orthogonal and $\mathbf{R} \in \Re^{m \times n}$ is upper triangular.
  % On return, $\mathbf{R}$ is stored in the upper triangular part of $\mathbf{A}$.

  ```
  for j = 1:n
     for i = m: -1: j + 1
        [c,s] = givens(A(i-1,j), A(i,j));
        A(i-1:i,j:n) = row.rot(A(i-1:i,j:n), c, s);
     end
  end
  ```

  – The procedure does not save $\mathbf{Q}$

  – The factorization requires about $3n^2(m - n/3)$ multiplications and additions

    * Reflections require $2n^2(m - n/3)$ FLOPs
    * Reflections are generally preferred

  – Let $\mathbf{G}_i = \mathbf{G}(i, i-1, \theta)$, then

$$\mathbf{Q} = \underbrace{(\mathbf{G}_m \mathbf{G}_{m-1} \cdots \mathbf{G}_2)}_{1\ st\ col.} \underbrace{(\mathbf{G}_m \cdots \mathbf{G}_3)}_{2\ nd\ col.} \cdots \underbrace{(\mathbf{G}_m)}_{n\ th\ col.}$$

Givens QR Factorization

- *Example 1.* Obtain the **QR** factorization of

$$\mathbf{A} = \begin{bmatrix} 3 & 2 & 1 \\ 2 & -3 & 4 \\ 5 & 1 & -1 \\ 7 & 4 & 2 \end{bmatrix}$$

  - Rotating the third and fourth rows using givens(5,7)

$$c = -0.5812, \qquad s = 0.8137$$

  - Applying row.rot to $\mathbf{A}(3:4, 1:3)$ yields

$$\mathbf{A}^{(1)} = \begin{bmatrix} 3.0000 & 2.0000 & 1.0000 \\ 2.0000 & -3.0000 & 4.0000 \\ -8.6023 & -3.8362 & -1.0462 \\ 0 & -1.5112 & -1.9762 \end{bmatrix}$$

  - The Givens matrix corresponding to this rotation is

$$\mathbf{G}(3, 4, \theta)^T = \mathbf{G}_4^T$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & c & -s \\ 0 & 0 & s & c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -0.5812 & -0.8137 \\ 0 & 0 & 0.8137 & -0.5812 \end{bmatrix}$$

  - We may check that $\mathbf{G}_4^T \mathbf{A} = \mathbf{A}^{(1)}$

Givens QR Factorization

- Rotate the second and third rows using givens(2.0000, -8.6023)

$$c = 0.2265, \qquad s = 0.9740$$

  - Applying row.rot to $\mathbf{A}^{(1)}(2:3, 1:3)$ yields

$$\mathbf{A}^{(2)} = \begin{bmatrix} 3.0000 & 2.0000 & 1.0000 \\ 8.8318 & 3.0571 & 1.9249 \\ 0 & -3.7908 & 3.6592 \\ 0 & -1.5112 & -1.9762 \end{bmatrix}$$

  - The rotation matrix is

$$\mathbf{G}_3^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.2265 & -0.9740 & 0 \\ 0 & 0.9740 & 0.2265 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - We may check that $\mathbf{G}_3^T \mathbf{G}_4^T \mathbf{A} = \mathbf{A}^{(2)}$

Givens QR Factorization

- Rotate the first and second rows using givens(3.0000, 8.8318)

$$c = -0.3216, \qquad s = 0.9469$$

  - Applying row.rot to $\mathbf{A}^{(2)}(1:2, 1:3)$ yields

$$\mathbf{A}^{(3)} = \begin{bmatrix} -9.3274 & -3.5380 & -2.1442 \\ 0 & 0.9104 & 0.3278 \\ 0 & -3.7908 & 3.6592 \\ 0 & -1.5112 & -1.9762 \end{bmatrix}$$

  - The Givens matrix

$$\mathbf{G}_2^T = \begin{bmatrix} -0.3216 & -0.9469 & 0 & 0 \\ 0.9469 & -0.3216 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - The first column has been reduced

Givens QR Factorization

- Rotate the third and fourth rows using givens(-3.7908, -1.5112)

$$c = 0.9289, \qquad s = -0.3703$$

  - Applying row.rot to $\mathbf{A}^{(3)}(3:4, 2:3)$ yields

$$\mathbf{A}^{(4)} = \begin{bmatrix} -9.3274 & -3.5380 & -2.1442 \\ 0 & 0.9104 & 0.3278 \\ 0 & -4.0809 & 2.6672 \\ 0 & 0 & -3.1908 \end{bmatrix}$$

  - The Givens matrix

$$\mathbf{G}_4^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.9289 & -0.3703 \\ 0 & 0 & 0.3703 & 0.9289 \end{bmatrix}$$

- Rotate the second and third rows using givens(0.9104, -4.0809)

$$c = 0.2177, \qquad s = 0.9760$$

  - Applying row.rot to $\mathbf{A}^{(4)}(2:3, 2:3)$ yields

$$\mathbf{A}^{(5)} = \begin{bmatrix} -9.3274 & -3.5380 & -2.1442 \\ 0 & 4.1812 & -2.5318 \\ 0 & 0 & 0.9007 \\ 0 & 0 & -3.1908 \end{bmatrix}$$

  - The Givens matrix

$$\mathbf{G}_3^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.2177 & -0.9760 & 0 \\ 0 & 0.9760 & 0.2177 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - The second column is done

The Last Rotation

- Rotate the third and fourth rows using givens(0.9007, -3.1908)

$$c = 0.2717, \qquad s = 0.9624$$

  - Applying row.rot to $\mathbf{A}^{(5)}(3:4, 3)$ yields

$$\mathbf{A}^{(6)} = \begin{bmatrix} -9.3274 & -3.5380 & -2.1442 \\ 0 & 4.1812 & -2.5318 \\ 0 & 0 & 3.3154 \\ 0 & 0 & 0 \end{bmatrix}$$

  - The Givens matrix

$$\mathbf{G}_4^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.2717 & -0.9624 \\ 0 & 0 & 0.9624 & 0.2717 \end{bmatrix}$$

The Factored Form

- The factored matrix

$$\mathbf{R} = \begin{bmatrix} -9.3274 & -3.5380 & -2.1442 \\ 0 & 4.1812 & -2.5318 \\ 0 & 0 & 3.3154 \\ 0 & 0 & 0 \end{bmatrix}$$

- The orthogonal matrix
  - We have

$$\mathbf{Q}^T = \underbrace{(\mathbf{G}_4)}_{3\ rd\ col.}\ \underbrace{(\mathbf{G}_3^T\mathbf{G}_4^T)}_{2\ nd\ col.}\underbrace{(\mathbf{G}_2^T\mathbf{G}_3^T\mathbf{G}_4^T)}_{1\ st\ col.}$$

$$\mathbf{Q} = \begin{bmatrix} -0.3216 & 0.2062 & 0.2511 & 0.8894 \\ -0.2144 & -0.8989 & 0.3813 & 0.0232 \\ -0.5361 & -0.2144 & -0.8121 & 0.0851 \\ -0.7505 & 0.3216 & 0.3635 & -0.4486 \end{bmatrix}$$

  - Check that $\mathbf{QR} = \mathbf{A}$
  - Check that $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$

Properties of the QR Factorization

- **Theorem 1:** Let $\mathbf{A} \in \Re^{m\times n}$ have rank n with $m \geq n$. Let $\mathbf{A} = \mathbf{QR}$, then

$$\text{span}\{\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_k\} = \text{span}\{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_k\}, \qquad k = 1:n \tag{4a}$$

  - $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n]$ etc.
  - Let $\mathbf{Q}_n = [\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_n]$, $\mathbf{Q}_{m-n} = [\mathbf{q}_{n+1}, \mathbf{q}_{n+2}, \cdots, \mathbf{q}_m]$
  - Then

$$\text{range}(\mathbf{A}) = \text{range}(\mathbf{Q}_n) \tag{4b}$$

$$\text{range}(\mathbf{A})^{\perp} = \text{range}(\mathbf{Q}_{m-n}) \tag{4c}$$

    * The $\perp$ denotes the orthogonal complement
  - In addition

$$\mathbf{A} = \mathbf{Q}_n\mathbf{R}_n, \qquad \mathbf{R}_n = \mathbf{R}(1:n, 1:n) \tag{4d}$$

- *Remark:*

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_n \\ \mathbf{0} \end{bmatrix} \qquad \mathbf{Q} = \begin{bmatrix} \mathbf{Q}_n & \mathbf{Q}_{m-n} \end{bmatrix}$$

  - Thus,

$$\mathbf{A} = \mathbf{QR} = \begin{bmatrix} \mathbf{Q}_n & \mathbf{Q}_{m-n} \end{bmatrix} \begin{bmatrix} \mathbf{R}_n \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_n\mathbf{R}_n$$

- *Proof:* From (1)

$$\mathbf{a}_k = \sum_{j=1}^{k} r_{jk}\mathbf{q}_j \in \text{span}\{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_k\} \tag{5}$$

- Thus,

$$\text{span}\{\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_k\} \subseteq \text{span}\{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_k\}$$

Fat and Skinny QRs

- But rank($\mathbf{A}$) = n; thus,

$$\dim(\text{span}\{\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_k\}) = k$$

  So

$$\text{span}\{\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_k\} = \text{span}\{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_k\}$$

- Thus, (4a) is satisfied
- The rest of the Theorem follows directly □
- We call (4c) the *thin* or *reduced* **QR** *factorization* of $\mathbf{A}$
- **Theorem 2:** Let $\mathbf{A} \in \Re^{m \times n}$ have rank n, then the thin **QR** factorization (4c) is unique.
  - $\mathbf{R}_n$ is upper triangular with positive diagonal entries
  - Consider the Cholesky factorization of

$$\mathbf{A}^T\mathbf{A} = \mathbf{L}\mathbf{L}^T \tag{6a}$$

    * $\mathbf{L}$ is lower triangular
  - Then

$$\mathbf{R}_n = \mathbf{L}^T \tag{6b}$$

- *Proof:* Consider

$$\mathbf{A}^T\mathbf{A} = \mathbf{R}_n^T\mathbf{Q}_n^T\mathbf{Q}_n\mathbf{R}_n = \mathbf{R}_n^T\mathbf{R}_n$$

  - Comparing this with (6a) yields (6b)
  - The Cholesky factorization is unique
  - From (4d) $\mathbf{Q}_n = \mathbf{A}\mathbf{R}_n^{-1}$; hence, $\mathbf{Q}_n$ is unique □

The Condition Number of a Rectangular Matrix

- What is the condition number of an $m \times n$ matrix?
- **Theorem 3:** Let $\mathbf{A} \in \Re^{n \times n}$, then
  - The eigenvalues λ_i, $i = 1 : n$, of $\mathbf{A}^T\mathbf{A}$ and the singular values σ_i, $i = 1 : n$, satisfy

$$\lambda_i = \sigma_i^2, \qquad i = 1 : n \tag{7a}$$

  - The eigenvectors $\mathbf{x}_i$ of $\mathbf{A}^T\mathbf{A}$ and the singular vectors $\mathbf{v}_i$ of $\mathbf{A}$ satisfy

$$\mathbf{x}_i = \mathbf{v}_i, \qquad i = 1 : n \tag{7b}$$

  - The Euclidean norm

$$||\mathbf{A}||_2 = \sqrt{\rho(\mathbf{A}^T\mathbf{A})} = \sigma_1 \tag{7c}$$

    * The spectral radius $\rho(\mathbf{A}^T\mathbf{A})$ is the largest eigenvalue λ_1 of $\mathbf{A}^T\mathbf{A}$
    * σ_1 is the largest singular value of $\mathbf{A}$
  - The condition number of $\mathbf{A}$ in the Euclidean norm satisfies

$$\kappa_2(\mathbf{A}) = \frac{\sigma_1(\mathbf{A})}{\sigma_n(\mathbf{A})} \tag{7d}$$

The Condition Number of a Rectangular Matrix

- *Proof:*

- We can use this as a definition of the condition number for a rectangular matrix
  - An $O(\epsilon)$ perturbation in $\mathbf{A}$ introduces $O(\epsilon\kappa_2(\mathbf{A}))$ perturbations in $\mathbf{Q}_n$ and $\mathbf{R}_n$
    - cf. Stewart (1993), On the perturbation of LU Cholesky and QR factorizations, *SIAM J. Matrix. Anal.*, **14**, 1141-1145