united nations
educational, scientific
and cultural
organization

international atomic
energy agency

the
abdus salam
international centre for theoretical physics

SMR.1492 - 7

Winter College on
NUMERICAL METHODS IN ELECTRONIC STRUCTURE THEORY

20 January – 4 February 2003

NUMERICAL LINEAR ALGEBRA I: factorization methods

Stefano FABRIS
INFM, SISSA & DEMOCRITOS, Trieste ITALY

NUMERICAL LINEAR ALGEBRA I

factorization methods

Stefano Fabris

INFM - DEMOCRITOS, Trieste, Italy

Overview

- Matrices in quantum mechanics
- Linear algebraic equations
  - solution of matrix equations
  - matrix inversion
  - determinant of a matrix
- Eigenvalue problem
- Libraries: LAPACK

Matrices in quantum mechanics

Schrödinger equation

$$H\ \Psi = \epsilon\ \Psi$$

Basis set

Orthogonal set of basis vectors:

$$\{\phi_i, i = 1, \ldots N\} \qquad\qquad \phi_i \cdot \phi_j = \delta_{ij}$$

We assume that Ψ can be expanded as:

$$\Psi = \sum_{i=1}^{N} c_i \phi_i$$

Matrices in quantum mechanics

Schrödinger equation

$$H\ \Psi = \epsilon\ \Psi$$

$$\Downarrow$$

$$H \sum_{i=1}^{N} c_i \phi_i = \epsilon \sum_{i=1}^{N} c_i \phi_i \qquad\qquad \cdot\ \phi_j$$

$$\sum_i c_i\ \phi_j \cdot H\phi_i = \epsilon \sum_i c_i\ \phi_j \cdot \phi_i = \epsilon\ c_j$$

Matrices in quantum mechanics

$$\sum_i c_i \; \phi_j \cdot H\phi_i = \epsilon \; c_i$$

Definitions: $\mathbf{H}$ $\quad H_{ij} = \phi_j \cdot H\phi_i$

$\mathbf{c}$ $\quad \mathbf{c} = (c_1, c_2, \ldots, c_N)$

Matrix formulation of the Schrödinger equation

$$\mathbf{H} \cdot \mathbf{c} = \epsilon \, \mathbf{c}$$

$\hookrightarrow$ eigenvalue problem

Overview 1

Matrix formulation of basic quantum mechanical problem
$\hookrightarrow$ efficient techniques to handle matrices

- Matrix equations: $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$
- Matrix inversion: $\mathbf{A}^{-1} \Rightarrow \mathbf{A} \cdot \mathbf{A}^{-1} = \mathbf{I}$
- Matrix determinant

$\hookrightarrow$ efficient techniques for eigenvalue problem

$$\mathbf{A} \cdot \mathbf{x} = \lambda \mathbf{x}$$

LIBRARIES OF ROUTINES: LAPACK

Linear algebraic equations

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N &= b_2 \\
\ldots \qquad\qquad \ldots &\quad \ldots \\
a_{M1}x_1 + a_{M2}x_2 + \ldots + a_{MN}x_N &= b_M
\end{aligned}
$$

or

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

Solution $\rightarrow \mathbf{x} = \{x_1, x_2, \ldots, x_N\}$

- $M < N \Rightarrow$ STOP!
- $M = N \Rightarrow$ solution is possible

Linear algebraic equations

$M = N \Rightarrow$ solution is possible ... BUT ...

1. Degeneracy $\Rightarrow$ STOP!
2. Numerical problems:
   - set is *nearly* degenerate $\Rightarrow$ STOP!
   - accumulated round-off errors
     $\Rightarrow$ solution is possible but can be <u>wrong</u>

Computer representation of a matrix

A is a two-dimensional matrix:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \dots & & & \\ a_{M1} & a_{M2} & \dots & a_{MN} \end{pmatrix}$$

The computer memory is numbered sequentially $\rightarrow$ the two-dimensional matrix is physically stored in a *unidimensional* array

Computer representation of a matrix

by columns (Fortran) $a_{11}a_{21}\dots a_{M1}a_{12}a_{22}\dots a_{MN}$
by rows (Pascal) $a_{11}a_{12}\dots a_{1N}a_{21}a_{22}\dots a_{MN}$

logical dimension $\leftrightarrow$ physical dimension

2×2 $\qquad$ 3×3

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \qquad \begin{pmatrix} 1 & 2 & 0 \\ 3 & 4 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

```
call invmat(a,ai,nl,np)
```

`nl=2, np=3`

Linear algebraic equations

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \dots & & & \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_N \end{pmatrix}$$

<u>NUMERICAL ALGORITHMS</u>

- Gaussian elimination with back substitution
- LU decomposition

Gaussian elimination + back substitut

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

$\downarrow$ line operations $\downarrow$

$$\begin{pmatrix} a'_{11} & a'_{12} & a'_{13} & a'_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a'_{33} & a'_{34} \\ 0 & 0 & 0 & a'_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \end{pmatrix}$$

Gaussian elimination + backsubstituti

-Gaussian elimination $\rightarrow$ triangular matrix

$$\begin{pmatrix} a'_{11} & a'_{12} & a'_{13} & a'_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a'_{33} & a'_{34} \\ 0 & 0 & 0 & a'_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \end{pmatrix}$$

-Backsubstitution

$$a'_{44}x_4 = b'_4 \qquad \Rightarrow \qquad x_4 = b'_4/a'_{44}$$

$$a'_{33}x_3 + a'_{34}x_4 = b'_3 \qquad \Rightarrow \qquad x_3 = b'_3/a'_{33} - a'_{34}/a'_{33}x_4$$

Gaussian elimination

Allowed operations:

- multiply line by a costant
- sum or subtract two or more lines
- swap rows or columns of $\mathbf{A}$ (and the corresponding rows of $\mathbf{b}$, accordingly)

Gaussian elimination

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

multiply equation 2 by a_{11}/a_{21}:

$$a'_{21} = a_{11} \qquad a'_{2i} = a_{2i}\,\frac{a_{11}}{a_{21}} \qquad b'_2 = b_2\,\frac{a_{11}}{a_{21}}$$

Gaussian elimination

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b'_2 \\ b_3 \\ b_4 \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b'_2 \\ b_3 \\ b_4 \end{pmatrix}$$

multiply equation 3 by a_{11}/a_{31}:

$$a'_{31} = a_{11} \qquad a'_{3j} = a_{3j}\,\frac{a_{11}}{a_{31}} \qquad b'_3 = b_3\,\frac{a_{11}}{a_{31}}$$

Gaussian elimination

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & a'_{32} & a'_{33} & a'_{34} \\ 0 & a'_{42} & a'_{43} & a'_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b'_2 \\ b'_3 \\ b'_4 \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a'_{33} & a'_{34} \\ 0 & a'_{42} & a'_{43} & a'_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b'_2 \\ b'_3 \\ b'_4 \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \end{pmatrix} = \begin{pmatrix} b_1 \\ b'_2 \\ \end{pmatrix}$$

LU decomposition

Gaussian method

- very simple
- very stable
- not very efficient
- the right-hand side of the equations must be known in advance

LU decomposition

LU decomposition

Suppose you can decompose the matrix $\mathbf{A}$ as:

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{U}$$

$\mathbf{L} \rightarrow$ Lower diagonal $\qquad$ $\mathbf{U} \rightarrow$ Upper diagonal

$$\begin{pmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix}$$

$$\mathbf{A} \cdot \mathbf{x} = (\mathbf{L} \cdot \mathbf{U}) \cdot \mathbf{x} = \mathbf{L} \cdot (\mathbf{U} \cdot \mathbf{x}) = \mathbf{b}$$

LU decomposition

$$\mathbf{L} \cdot (\mathbf{U} \cdot \mathbf{x}) = \mathbf{b}$$

The LU decomposition breaks the problem in 2 parts:

1. find $\mathbf{y}$ such $\mathbf{L} \cdot \mathbf{y} = \mathbf{b}$
2. find $\mathbf{x}$ such $\mathbf{U} \cdot \mathbf{x} = \mathbf{y}$

$\mathbf{L}$ and $\mathbf{U}$ are triangular
$\longrightarrow$ solution of problems 1. and 2. is trivial

LU decomposition: L part

$$\begin{pmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

$\hookrightarrow$ forward substitution:

$$\begin{aligned} y_1 &= b_1/l_{11} \\ y_i &= \left(b_i - \sum_{j=1}^{i-1} l_{ij} y_j\right) / l_{ii} \end{aligned}$$

LU decomposition: U part

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

$\hookrightarrow$ back substitution:

$$\begin{aligned} x_4 &= y_N/u_{44} \\ x_i &= \left(y_i - \sum_{j=i+1}^{N} u_{ij} x_j\right) / u_{ii} \end{aligned}$$

LU factorization

Suppose you can decompose the matrix $\mathbf{A}$ as:

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{U}$$

OK, but HOW?

Crout's algorithm:
$\hookrightarrow$ find $\mathbf{L}$ and $\mathbf{U}$ given $\mathbf{A}$
CAUTION: PIVOTING IS ESSENTIAL

$\Rightarrow$ you are NOT decomposing matrix $\mathbf{A}$ but a row permutation of $\mathbf{A}$
$\Rightarrow$ keep track of permutations!

LU decomposition: LAPACK

Routine Name	Operation
sgetrf, dgetrf cgetrf, zgetrf	Computes an LU factorization of a general matrix, using partial pivoting with row interchanges
sgetrs, dgetrs cgetrs, zgetrs	Solves a general system of linear equations AX=B, A**T X=B or A**H X=B, using the LU factorizati computed by SGETRF/CGETRF
sgecon, dgecon cgecon, zgecon	Estimates the reciprocal of the condition number of a general matrix, in either the 1-norm or the infinity-norm, using the LU factorization computed by SGETRF/CGETRF
sgerfs, dgerfs cgerfs, zgerfs	Improves the computed solution to a general system of linear equations AX=B, A**T X=B or A**H X=B, and provides forward and backward error bounds for the solution
sgetri, dgetri cgetri, zgetri	Computes the inverse of a general matrix, using the LU factorization computed by SGETRF/CGETRF
sgeequ, dgeequ cgeequ, zgeequ	Computes row and column scalings to equilibrate a general rectangular matrix and reduce its condition number
sgbtrf, dgbtrf cgbtrf, zgbtrf	Computes an LU factorization of a general band matrix, using partial pivoting with row interchanges
sgbtrs, dgbtrs cgbtrs, zgbtrs	Solves a general banded system of linear equations AX=B, A**T X=B or A**H X=B, using the LU factorization computed by SGBTRF/CGBTRF
sgbcon, dgbcon cgbcon, zgbcon	Estimates the reciprocal of the condition number of a general band matrix, in either the 1-norm or the infinity-norm, using the LU factorization computed by SGBTRF/CGBTRF
sgbrfs, dgbrfs cgbrfs, zgbrfs	Improves the computed solution to a general banded system of linear equations AX=B, A**T X=B or A**H X=B, and provides forward and backward error bounds for the solution
sgbequ, dgbequ cgbequ, zgbequ	Computes row and column scalings to equilibrate a general band matrix and reduce its condition numbe

LU decomposition: LAPACK

```
SGETRF(l)              LAPACK routine (version 1.1)              SGETRF(l)

NAME
  SGETRF - compute an LU factorization of a general M-by-N matrix A using
  partial pivoting with row interchanges

SYNOPSIS

  SUBROUTINE SGETRF( M, N, A, LDA, IPIV, INFO )

      INTEGER        INFO, LDA, M, N

      INTEGER        IPIV( * )

      REAL           A( LDA, * )

PURPOSE
  SGETRF computes an LU factorization of a general M-by-N matrix A using par-
  tial pivoting with row interchanges.

  The factorization has the form
     A = P * L * U
  where P is a permutation matrix, L is lower triangular with unit diagonal
  elements (lower trapezoidal if m > n), and U is upper triangular (upper
  trapezoidal if m < n).

  This is the right-looking Level 3 BLAS version of the algorithm.

ARGUMENTS

  M       (input) INTEGER
          The number of rows of the matrix A.  M >= 0.

  N       (input) INTEGER
          The number of columns of the matrix A.  N >= 0.

  A       (input/output) REAL array, dimension (LDA,N)
          On entry, the M-by-N matrix to be factored.  On exit, the factors L
          and U from the factorization A = P*L*U; the unit diagonal elements
```


LU decomposition: LAPACK

```
SGETRS(l)              LAPACK routine (version 1.1)              SGETRS(l)

NAME
  SGETRS - solve a system of linear equations  A * X = B or A' * X = B with a
  general N-by-N matrix A using the LU factorization computed by SGETRF

SYNOPSIS

  SUBROUTINE SGETRS( TRANS, N, NRHS, A, LDA, IPIV, B, LDB, INFO )

      CHARACTER      TRANS

      INTEGER        INFO, LDA, LDB, N, NRHS

      INTEGER        IPIV( * )

      REAL           A( LDA, * ), B( LDB, * )

PURPOSE
  SGETRS solves a system of linear equations
     A * X = B  or  A' * X = B with a general N-by-N matrix A using the LU
  factorization computed by SGETRF.

ARGUMENTS

  TRANS   (input) CHARACTER*1
          Specifies the form of the system of equations:
          = 'N':  A * X = B  (No transpose)
          = 'T':  A'* X = B  (Transpose)
          = 'C':  A'* X = B  (Conjugate transpose = Transpose)

  N       (input) INTEGER
          The order of the matrix A.  N >= 0.

  NRHS    (input) INTEGER
          The number of right hand sides, i.e., the number of columns of the
          matrix B.  NRHS >= 0.

  A       (input) REAL array, dimension (LDA,N)
          The factors L and U from the factorization A = P*L*U as computed by
```

Matrix inversion

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b} \qquad \Rightarrow \qquad \mathbf{x} = \mathbf{A}^{-1} \cdot \mathbf{b}$$

$$\mathbf{b} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

solve the linear system $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$

$\mathbf{x}$ is the 1th column of $\mathbf{A}^{-1}$

$$\mathbf{b} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

Matrix inversion: LAPACK

```
do i=1,N
   do j=1,N
      b(i,j)=0.0
   enddo
   b(i,i)=1.0
enddo
! DECOMPOSE THE MATRIX "a" just once
call sgetrf(N,N,a,...)
! find inverse column by column
do j=1,N
   call sgetrs(..,,a,b(1,j),...)
enddo
```

Determinant of a matrix

$$\det(\mathbf{A}) = \det(\mathbf{L} \cdot \mathbf{U}) = \det(\mathbf{L}) \det(\mathbf{U})$$

- LU decompose matrix $\mathbf{A}$
  $\hookrightarrow$ `call sgetrf`
- The determinant of the LU decomposed matrix is the product of the diagonal elements:

$$\det(\mathbf{L}) \det(\mathbf{U}) = \det(\mathbf{U}) = \Pi_{i=1}^{N} u_{ii}$$

! REMEBER THE PIVOTING !

$$\det(\mathbf{A}) = \pm \Pi_{i=1}^{N} u_{ii}$$

LAPACK

sgesv, dgesv cgesv, zgesv	Solves a general system of linear equations AX=B	sgesvx, dges cgesvx, zges
sgbsv, dgbsv cgbsv, zgbsv	Solves a general banded system of linear equations AX=B	sgbsvx, dgbs cgbsvx, zgbs
sgtsv, dgtsv cgtsv, zgtsv	Solves a general tridiagonal system of linear equations AX=B	sgtsvx, dgtsv cgtsvx, zgtsv
sposv, dposv	Solves a symmetric positive definite system of linear equations AX=B	sposvx, dpos
cposv, zposv	Solves a Hermitian positive definite system of linear equations AX=B	cposvx, zpos
sppsv, dppsv	Solves a symmetric positive definite system of linear equations AX=B, where A is held in packed storage	sppsvx, dpps
cppsv, zppsv	Solves a Hermitian positive definite system of linear equations AX=B, where A is held in packed storage	cppsvx, zpps
spbsv, dpbsv	Solves a symmetric positive definite banded system of linear equations AX=B	spbsvx, dpbs
cpbsv, zpbsv	Solves a Hermitian positive definite banded system of linear equations AX=B	cpbsvx, zpbs
sptsv, dptsv	Solves a symmetric positive definite tridiagonal system of linear equations AX=B	sptsvx, dptsv
cptsv, zptsv	Solves a Hermitian positive definite tridiagonal system of linear equations AX=B	cptsvx, zptsv
ssysv, dsysv csysv, zsysv	Solves a real/complex symmetric indefinite system of linear equations AX=B	ssysvx, dsysv csysvx, zsysv
chesv, zhesv	Solves a complex Hermitian indefinite system of linear equations AX=B	chesvx, zhes
sspsv, dspsv cspsv, zspsv	Solves a real/complex symmetric indefinite system of linear equations AX=B, where A is held in packed storage	sspsvx, dspsv cspsvx, zspsv
chpsv, zhpsv	Solves a complex Hermitian indefinite system of linear equations AX=B, where A is held in packed storage	chpsvx, zhps
	Computes the least squares solution to an overdetermined system of linear equations, A	

Eigenvalue problem: introduction

$$\mathbf{A} \cdot \mathbf{x} = \lambda \mathbf{x}$$

$\mathbf{A}$	$N \times N$ matrix
$\mathbf{x}$	eigenvector
λ	eigenvalue

$$\det(\mathbf{A} - \lambda \mathbf{I}) = 0$$

$$\Downarrow$$

N^{th} degree polynomial in λ

ex: $(2-\lambda)(1+\lambda) = 0$

Root searching is <u>NOT!</u> the way to go

Eigenvalue problem: definitions

Symmetric	$\mathbf{A} = \mathbf{A}^T$	$a_{ij} = a_{ji}$
Hermitian	$\mathbf{A} = \mathbf{A}^\dagger$	$a_{ij} = a_{ji}^\star$
Orthogonal	$\mathbf{A}^{-1} = \mathbf{A}^T$	$\mathbf{A} \cdot \mathbf{A}^T = \mathbf{I}$
Unitary	$\mathbf{A}^{-1} = \mathbf{A}^\dagger$	$\mathbf{A} \cdot \mathbf{A}^\dagger = \mathbf{I}$
Normal	$\mathbf{A} \cdot \mathbf{A}^\dagger = \mathbf{A}^\dagger \cdot \mathbf{A}$	

- Hermitian: its eigenvalues are real
- Normal: its eigenvectors are complete and orthogonal (if eigenvalues are not degenerate)

Similarity transformation

$$\mathbf{A} \longrightarrow \mathbf{P}^{-1} \cdot \mathbf{A} \cdot \mathbf{P}$$

$\mathbf{P}$ is a transformation matrix

$$|\mathbf{P}| = 1$$

ST do not affect the eigenvalue problem

$$\left|\mathbf{P}^{-1} \cdot \mathbf{A} \cdot \mathbf{P} - \lambda \mathbf{I}\right| = 0$$

$$\left|\mathbf{P}^{-1} \cdot (\mathbf{A} - \lambda \mathbf{I}) \cdot \mathbf{P}\right| = \left|\mathbf{P}^{-1}\right| |\mathbf{A} - \lambda \mathbf{I}| \, |\mathbf{P}| = |\mathbf{A} - \lambda \mathbf{I}$$

Eigenvalue problem: main idea

$$\mathbf{A} \cdot \mathbf{x} = \lambda \mathbf{x}$$

define the eigenvectors matrix $\mathbf{X}$:

$$\mathbf{X} = (\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N})$$

eigenvalue problem (equivalent form):

$$\mathbf{A} \cdot \mathbf{X} = \mathrm{diag}(\lambda_1, \lambda_2, \lambda_3, ..., \lambda_N) \cdot \mathbf{X}$$

$$\mathbf{X}^{-1} \cdot \mathbf{A} \cdot \mathbf{X} = \mathrm{diag}(\lambda_1, \lambda_2, \lambda_3, ..., \lambda_N)$$

Eigenvalue problem: main idea

$$\mathbf{X}^{-1} \cdot \mathbf{A} \cdot \mathbf{X} = \mathrm{diag}(\lambda_1, \lambda_2, \lambda_3, ..., \lambda_N)$$

MAIN IDEA

Apply a sequence of similarity transformations which push $\mathbf{A}$ *towards* a diagonal form

$$\mathbf{A} \longrightarrow \mathbf{P_1}^{-1}\mathbf{A}\mathbf{P_1} \longrightarrow \mathbf{P_2}^{-1}\mathbf{P_1}^{-1}\mathbf{A}\mathbf{P_1}\mathbf{P_2} \longrightarrow \ldots$$

$\Downarrow$ at convergence $\Downarrow$

$$\mathbf{X} = \mathbf{P_1}\,\mathbf{P_2}\,\mathbf{P_3}\ldots$$

Eigenvalue problem: Jacobi method

- Simple and stable
- It is valid for symmetric matrices
- It consist in a sequence of plane rotations
- Each transformation annihilate a particular off-diagonal element
- $\mathbf{X} = \mathbf{P_1}\,\mathbf{P_2}\,\mathbf{P_3}\ldots$
- Eigenvalues are on the diagonal of the final matrix

Eigenvalue problem: Jacobi method

$$
\mathbf{A} = \begin{array}{c}
\begin{pmatrix}
a_{11} & \cdots & & \cdots & & \cdots & a_{1N} \\
\vdots & \ddots & \vdots & & \vdots & & \vdots \\
 & \cdots & a_{pp} & \cdots & a_{pq} & \cdots & \\
\vdots & & \vdots & \ddots & \vdots & & \vdots \\
 & \cdots & a_{qp} & \cdots & a_{qq} & \cdots & \\
\vdots & & \vdots & & \vdots & \ddots & \vdots \\
a_{N1} & \cdots & & \cdots & & \cdots & a_{NN}
\end{pmatrix}
\begin{array}{l}
\\ \\ \leftarrow p \\ \\ \leftarrow q \\ \\ \\
\end{array} \\
\begin{array}{ccccccc}
 & & \uparrow & & \uparrow & & \\
 & & p & & q & &
\end{array}
\end{array}
$$

Eigenvalue problem: Jacobi method

$$
\mathbf{J}_{pq} = \begin{array}{c}
\begin{pmatrix}
1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & & \vdots & & \vdots \\
0 & \cdots & c & \cdots & s & \cdots & 0 \\
\vdots & & \vdots & \ddots & \vdots & & \vdots \\
0 & \cdots & -s & \cdots & c & \cdots & 0 \\
\vdots & & \vdots & & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & \cdots & 0 & \cdots & 1
\end{pmatrix}
\begin{array}{l}
\\ \\ \leftarrow p \\ \\ \leftarrow q \\ \\ \\
\end{array} \\
\begin{array}{ccccccc}
 & & \uparrow & & \uparrow & & \\
 & & p & & q & &
\end{array}
\end{array}
$$

Eigenvalue problem: Jacobi matrix

The Jacobi matrix is chosen such that:

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}^T \begin{pmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{pmatrix} \begin{pmatrix} c & s \\ -s & c \end{pmatrix} = \begin{pmatrix} a'_{pp} & 0 \\ 0 & a'_{qq} \end{pmatrix}$$

$$\begin{aligned}
a'_{rp} &= ca_{rp} - sa_{rq} \\
a'_{rq} &= ca_{rq} + sa_{rp} \qquad r \neq p, r \neq q \\
a'_{pp} &= c^2 a_{pp} + s^2 a_{qq} - 2sca_{pq} \\
a'_{qq} &= s^2 a_{pp} + c^2 a_{qq} + 2sca_{pq} \\
a'_{pq} &= (c^2 - s^2) a_{pq} + sc(a_{pp} - a_{qq}) = 0
\end{aligned}$$

Eigenvalue problem: Jacobi method

- method is NOT exact
- Each rotation reduces the *norm* of the off diagonal elements

$$S = \sum_{j \neq i} {a'_{ij}}^2$$

- At convergence (S small):

$$\mathbf{D} = \mathbf{X}^{-1} \cdot \mathbf{A} \cdot \mathbf{X} = \mathrm{diag}(\lambda_1, \lambda_2, \lambda_3, ..., \lambda_N)$$

$$\mathbf{X} = \mathbf{J_{12}}\,\mathbf{J_{13}}\,\ldots\mathbf{J_{1N}}\,\mathbf{J_{23}}\ldots$$

Optimal strategy

The Jacobi method is simple
works well for real symmetric matrices
but it is not very efficient

A much more efficient strategy involves:

- reducing the matrix to a simple form (tridiagonal)
- diagonalizing the tridiagonal matrix (QR LR methods)

Tridiagonal form: Givens method

$$\begin{pmatrix} t_{11} & t_{12} & 0 & 0 \\ t_{21} & t_{22} & t_{23} & 0 \\ 0 & t_{32} & t_{33} & t_{34} \\ 0 & 0 & t_{43} & t_{44} \end{pmatrix}$$

<u>Givens method:</u> modification of Jacobi
$\hookrightarrow$ reduces the initial matrix to tridiagonal
$\hookrightarrow$ involves a *finite* number of steps

$$\mathbf{J}_{13}\,\mathbf{J}_{14}\,\ldots\mathbf{J}_{1N};\ \mathbf{J}_{24}\ldots\mathbf{J}_{2N};\ldots\mathbf{J}_{N-2,N}$$

Tridiagonal form: Householder metho

Each transformation annihilates the whole column (and row) below a certain element

$$\begin{pmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{23} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ t_{41} & t_{42} & t_{43} & t_{44} \end{pmatrix}$$

$$\begin{pmatrix} t_{11} & t_{12} & 0 & 0 \\ t_{21} & t_{22} & t_{23} & t_{23} \\ 0 & t_{32} & t_{33} & t_{34} \\ 0 & t_{42} & t_{43} & t_{44} \end{pmatrix}$$

Eigenvalue problem: LAPACK

ssyev, dsyev	Computes all eigenvalues and eigenvectors of a symmetric matrix	ssyevx, dsye
cheev, zheev	Computes all eigenvalues and eigenvectors of a Hermitian matrix	cheevx, zhee
sspev, dspev	Computes all eigenvalues and eigenvectors of a symmetric matrix in packed storage	sspevx, dspe
chpev, zhpev	Computes all eigenvalues and eigenvectors of a Hermitian matrix in packed storage	chpevx, zhpevx
ssbev, dsbev	Computes all eigenvalues and eigenvectors of a symmetric band matrix	ssbevx, dsbe
chbev, zhbev	Computes all eigenvalues and eigenvectors of a Hermitian band matrix	chbevx, zhbevx
sstev, dstev	Computes all eigenvalues and eigenvectors of a real symmetric tridiagonal matrix	sstevx, dstev
sgees, dgees cgees, zgees	Computes the eigenvalues and Schur factorization of a general matrix, and orders the factorization so that selected eigenvalues are at the top left of the Schur form	sgeesx, dgee cgeesx, zgee
sgeev, dgeev cgeev, zgeev	Computes the eigenvalues and left and right eigenvectors of a general matrix	sgeevx, dgee cgeevx, zgee
sgesvd, dgesvd cgesvd, zgesvd	Computes the singular value decomposition (SVD) of a general rectangular matrix	no
ssygv, dsygv	Computes all eigenvalues and the eigenvectors of a generalized symmetric-definite generalized eigenproblem, Ax= lambda Bx, ABx=lambda x, or BAx= lambda x	no
chegv, zhegv	Computes all eigenvalues and the eigenvectors of a generalized Hermitian-definite generalized eigenproblem, Ax= lambda Bx, ABx=lambda x, or BAx= lambda x	no
sspgv, dspgv chpgv, zhpgv	Computes all eigenvalues and eigenvectors of a generalized symmetric/Hermitian-definite generalized eigenproblem, Ax = lambda Bx, ABx= lambda x, or BAx= lambda x, where A and B are in packed storage	no
sgegs, dgegs cgegs, zgegs	Computes the generalized eigenvalues, Schur form, and left and/or right Schur vectors for a pair of nonsymmetric matrices	no

LAPACK

- Simple Driver Routines: solve a complete problem
  (e.g. eigenvalues of a matrix)
- Expert Driver Routines: provide more options/information
- Computational Routines: called by the driver routines
  perform a distinct computational task
  (LU factorization)

LAPACK: naming scheme

XYYZZZ

- X: data type
  - S REAL
  - D DOUBLE PRECISION
  - C COMPLEX
  - Z COMPLEX*16 or DOUBLE COMPLEX
- YY: matrix type
  - DI diagonal
  - GT general tridiagonal
  - HE (complex) Hermitian
  - OR (real) orthogonal
  - SB (real) symmetric band

LAPACK: naming scheme

XYYZZZ

- ZZZ: computation performed

  e.g. SSYTRD perfrom a tridiagonal reduction of a dense symmetric matrix of real numbers

Reference

NUMERICAL RECIPES
The Art of Scientific Computing

W. H. Press
B.P. Flannery
S. A. Teukolsky
W. T. Vetterling