Eigenvalue problems

• Determination of eigenvalues and eigenvectors

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

where $\mathbf{A}$ is an $N \times N$ matrix,
eigenvector $\mathbf{x} \neq 0$,
and eigenvalues λ are in general complex numbers

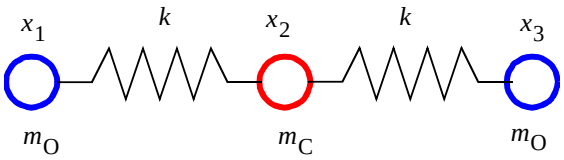
• In physics:

- Energy eigenvalues in a quantum mechanical system
  - Express wave function in terms of atom-like orbitals $|\psi\rangle = \sum_p a_p |p\rangle$
  - Minimize $\langle\psi|H|\psi\rangle - E\langle\psi|\psi\rangle$ ($H_{pq} = \langle p|H|q\rangle$) with respect to a_p: $\sum_q (H_{pq} - E\delta_{pq}) = 0$
  - In matrix form $\mathbf{H}\mathbf{a} = E\mathbf{a}$
- Vibration modes in molecules and solids
  - $\mathbf{K}\mathbf{u} = m\omega^2\mathbf{u}$ where $\mathbf{u}$ contains the displacement vectors of all atoms

  and $\mathbf{K}$ is the dynamical matrix $K_{ij} = \frac{\partial^2 E_{\text{pot}}}{\partial x_i \partial x_j}$

Eigenvalue problems

- A simple example: CO_2 molecule:



- Equations of motion (in 1D):

$$x_1'' = -\frac{k}{m_O}(x_1 - x_2)$$

$$x_2'' = -\frac{k}{m_C}(x_2 - x_1) - \frac{k}{m_C}(x_2 - x_3)$$

$$x_3'' = -\frac{k}{m_O}(x_3 - x_2)$$

- Assume all vibrate with the same frequency ω: trial solution $x_j = x_{j0}\cos\omega t$, $j = 1, 2, 3$ (x_{j0} are the initial positions; initial velocities are zero)
- Substitute these x_j to equations of motion we get

$$\begin{bmatrix} \frac{k}{m_O} & -\frac{k}{m_O} & 0 \\ -\frac{k}{m_C} & \frac{2k}{m_C} & -\frac{k}{m_C} \\ 0 & -\frac{k}{m_O} & \frac{k}{m_O} \end{bmatrix} \begin{bmatrix} x_{10} \\ x_{20} \\ x_{30} \end{bmatrix} = \omega^2 \begin{bmatrix} x_{10} \\ x_{20} \\ x_{30} \end{bmatrix}$$

Eigenvalue problems

- We want to find eigenvalues ω^2 and eigenvectors $[x_{10}\ x_{20}\ x_{30}]^{\mathrm{T}}$ of the matrix $\begin{bmatrix} a & -a & 0 \\ -b & 2b & -b \\ 0 & -a & a \end{bmatrix}$ where $a = \frac{k}{m_{\mathrm{O}}}$, $b = \frac{k}{m_{\mathrm{C}}}$

- Solution by Maxima:

```
(%i1) m:matrix([a,-a,0],[-b,2*b,-b],[0,-a,a]);
```

$$\begin{bmatrix} a & -a & 0 \\ -b & 2b & -b \\ 0 & -a & a \end{bmatrix}$$

```
(%i2) eigenvectors(m);
```

$$[[[2b+a,a,0],[1,1,1]],[[[1,-\frac{2b}{a},1]],[[1,0,-1]],[[1,1,1]]]]$$

eigenvalues multiplicities eigenvectors

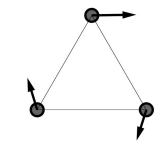
- Eigenvalues $\omega_1 = \sqrt{2b+a} = \sqrt{\frac{2k}{m_{\mathrm{C}}} + \frac{k}{m_{\mathrm{O}}}}$ $\omega_2 = \sqrt{a} = \sqrt{\frac{k}{m_{\mathrm{O}}}}$ $\omega_3 = 0$

- Eigenvectors $\mathbf{x}_1 = \begin{bmatrix} 1 & -\frac{2b}{a} & 1 \end{bmatrix}$ $\mathbf{x}_2 = [1\ \ 0\ \ -1]$ $\mathbf{x}_3 = [1\ \ 1\ \ 1]$

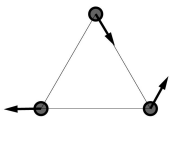
Eigenvalue problems

- Another vibration example: Cu trimer

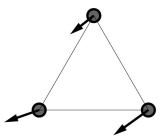
```
eval   0.0000 0.0000  0.0000  0.0000  0.0000  0.0000  7.6393  7.6393  20.3185

x1           -0.1520          -0.5432  0.3139          -0.5773 -0.0099  -0.5000
y1            0.4974          -0.1898 -0.5476          -0.0099  0.5773  -0.2887
z1      0.7137         0.5774                  -0.3966

x2           -0.1519          -0.5432  0.3139           0.2800  0.5049   0.5000
y2           -0.4770          -0.3920 -0.4496           0.5049 -0.2800  -0.2887
z2     -0.0134         0.5774                   0.8164

x3            0.6919          -0.3681  0.2290           0.2972 -0.4950
y3            0.0102          -0.2909 -0.4986          -0.4950 -0.2972   0.5774
z3     -0.7003         0.5774                  -0.4198
```



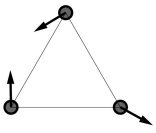
Eigenvalue 2



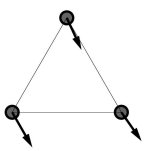
Eigenvalue 7: 7.6393



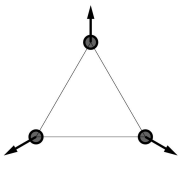
Eigenvalue 4



Eigenvalue 8: 7.6393



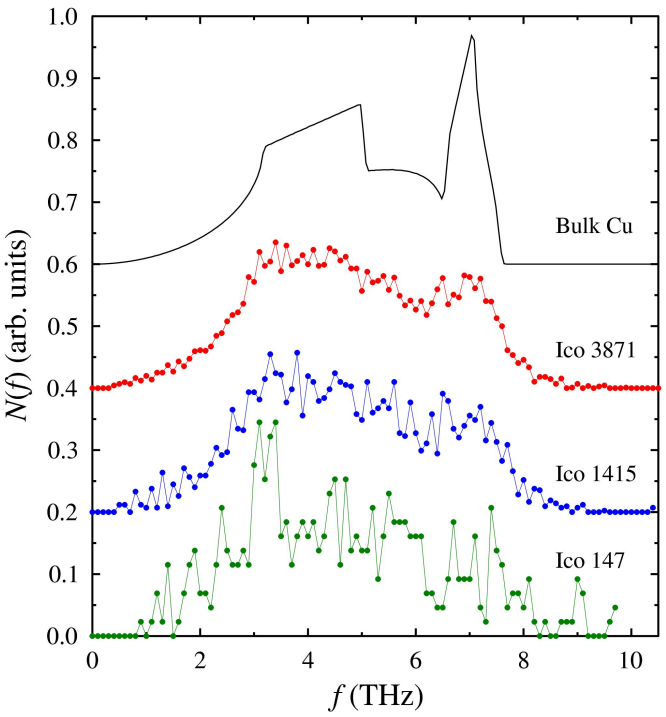
Eigenvalue 5



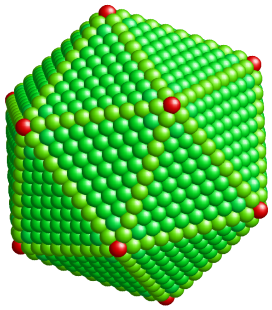
Eigenvalue 9: 20.3185

Eigenvalue problems

- Similar calculations for larger copper clusters than just a trimer (phonon density of states).



Icosahedral Cu with 3871 atoms



Eigenvalue problems

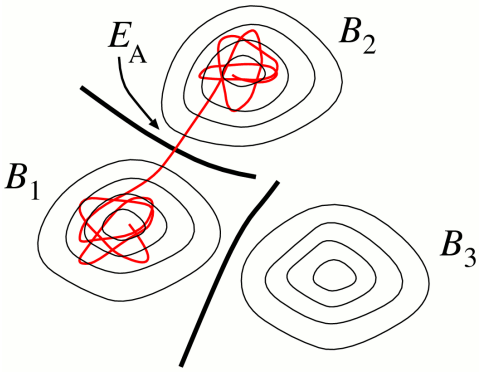
- All vibration modes of a atomic system can be found out by computing eigenvalues of the dynamical (Hessian) matrix $\mathbf{A}$

$$A_{ij} = \frac{\partial^2 V}{\partial x_i \partial x_j}$$

  - In potential energy minima *all eigenvalues* λ_i of the dynamical matrix are positive. (I.e. $\mathbf{A}$ is positive definite.)
    - Vibration frequencies $\omega_i = \sqrt{\lambda_i}$
    - If $\lambda_i < 0 \rightarrow$ no vibration. (May be a result of poor energy minimization.)

- Transition state theory: find saddle points in potential energy hypersurface

  - In saddle point *all but one eigenvalue* are positive.
    (Think of a saddle point on a 2D surface: curvature in one direction is positive in another negative.)



Eigenvalue problems

• Eigenvalues are those values of λ with which the equation $(\mathbf{A}-\lambda\mathbf{1})\mathbf{x} = 0$ has a nonzero solution $\mathbf{x}$.

$\rightarrow$ eigenvalues are the roots of the (characteristic) polynomial of Nth degree $\det(\mathbf{A}-\lambda\mathbf{1}) \equiv f_{\mathbf{A}}(\lambda) = 0$

- However, this is not a good method to solve λ's.

- Many methods for determining eigenvalues utilize the fact that the eigenvalues are invariant in *similarity transformations*:

$$\mathbf{A} \rightarrow \mathbf{Z}^{-1}\mathbf{A}\mathbf{Z} \quad (\mathbf{Z} \text{ is a nonsingular matrix})$$

- Eigenvectors of the transformed matrix are $\mathbf{Z}^{-1}\mathbf{x}$ ($\mathbf{x}$ are the eigenvectors of $\mathbf{A}$)

- By similarity transformations $\mathbf{A}$ is put into form that allows more or less easy determination of eigenvalues by e.g. various iterative methods.

- In the best cases $\mathbf{A}$ can be diagonalized:

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

- For every matrix it is possible to form Schur factorization: $\mathbf{Q}^*\mathbf{A}\mathbf{Q} = \mathbf{T}$, where $\mathbf{T}$ is an upper triangular matrix with λ's on the diagonal
- In many cases it is advisable to transform the matrix into e.g. tridiagonal or Hessenberg forms (upper triangular+one row below diagonal) before computations.
- What methods to use depends also whether the matrix is symmetric or real.

Eigenvalue problems

- Properties of the eigenvalues of a matrix depend on the type of the matrix:

Matrix	*Eigenvalues*
Upper or lower triangular	Diagonal elements of $\mathbf{A}$
Hermitian: $\mathbf{A}^* = \mathbf{A}$	Real
Positive definite: $\mathbf{x}^*\mathbf{A}\mathbf{x} > 0 \forall \mathbf{x} \in R^N$	Positive
Positive semidefinite: $\mathbf{x}^*\mathbf{A}\mathbf{x} \geq 0 \forall \mathbf{x} \in R^N$	Non-negative

- How to generate a positive definite matrix (e.g. when experimenting with Matlab)?
  If matrix $\mathbf{B}$ has all columns linearly independent then the following matrix $\mathbf{A}$ is positive definite[1]:
  $$\mathbf{A} = \mathbf{B}^T\mathbf{B}.$$

- Positive definiteness is an important concept in function minimization: the Hessian matrix of a multivariate function is positive definite at the minimum.

- A useful property of a *partitioned* matrix is that its eigenvalues can be computed from its partitions:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ 0 & \mathbf{A}_{22} \end{bmatrix} \begin{matrix} p \\ q \end{matrix}, \quad p+q = n \quad \rightarrow \lambda(\mathbf{A}) = \lambda(\mathbf{A}_{11}) \cup \lambda(\mathbf{A}_{22})$$
$$\begin{matrix} p & q \end{matrix}$$

1. See e.g. Golub-van Loan, section 4.2

Eigenvalue problems

• Canonical forms of square matrices: Relation of the structure of the matrix to its eigenvalues and eigenvectors:
(assume $\mathbf{A}$ and $\mathbf{B}$ are $n \times n$ square matrices)

- $\mathbf{A}$ and $\mathbf{B}$ are **similar** if there is a nonsingular matrix $\mathbf{P}$ such that

$$\mathbf{A} = \mathbf{P}^{-1}\mathbf{B}\mathbf{P}.$$

- If $\mathbf{A}$ and $\mathbf{B}$ are similar then they have the same characteristic polynomials

$$f_{\mathbf{A}}(\lambda) = f_{\mathbf{B}}(\lambda)$$

and, consequently, same eigenvalues and eigenvectors.

- **Schur normal form**: $\mathbf{A}$ can be expressed as

$$\mathbf{T} = \mathbf{U}^*\mathbf{A}\mathbf{U} \ (\rightarrow \mathbf{A} = \mathbf{U}\mathbf{T}\mathbf{U}^*)$$

where $\mathbf{U}$ is a unitary[1] matrix and $\mathbf{T}$ is upper triangular and

$$f_{\mathbf{A}}(\lambda) = f_{\mathbf{T}}(\lambda) = (\lambda - t_{11})(\lambda - t_{22})\ldots(\lambda - t_{nn}) \ ,$$

and the eigenvalues of $\mathbf{A}$ are the diagonal elements of $\mathbf{T}$.

It is easy to show that if matrix $\mathbf{A}$ is upper or lower triangular then its eigenvalues are simply the diagonal elements. This follows from the fact that the determinant of a triangular matrix $\mathbf{T}$ is simply

$$\det\mathbf{T} = \prod_{i=1}^{n} t_{ii}$$

- **Principal axis theorem**: If $\mathbf{A}$ is Hermitian[2] , then

1) it has real n eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ (not necessarily distinct) and

2) n corresponding eigenvectors $u^{(1)}, u^{(2)}, \ldots, u^{(n)}$ that form an orthonormal basis in C^n (or R^n if $\mathbf{A}$ real)

3) there is a unitary matrix $\mathbf{U}$ such that

$$\mathbf{U}^*\mathbf{A}\mathbf{U} = \mathbf{D} \equiv \mathrm{diag}[\lambda_1, \lambda_2, \ldots, \lambda_n]$$

---

1. $\mathbf{U}^{-1} = \mathbf{U}^*$, for real matrices $\mathbf{U}^{-1} = \mathbf{U}^T$. $\mathbf{U}^*$ is the *conjugate transpose* of $\mathbf{U}$; i.e. $(\mathbf{U}^*)_{ij} = \bar{u}_{ji}$, where $\bar{u}$ is the complex conjugate of u.
2. $\mathbf{A}^* = \mathbf{A}$

Eigenvalue problems

- **Singular value decomposition** (SVD): $\mathbf{A}$ (let it be a $m \times n$ matrix) can be expressed in form

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^*,$$

where $\mathbf{U}$ is a $m \times m$ unitary martix
$\mathbf{V}$ is a $n \times n$ unitary matrix
$\mathbf{S}$ is a "diagonal" $m \times n$ matrix:

$$\mathbf{S} = \mathrm{diag}[\sigma_1, \ldots, \sigma_p, 0, \ldots, 0] = \begin{bmatrix} \sigma_1 & & & & & \\ & \sigma_2 & & 0 & & \\ & & \sigma_3 & & & \\ & & & \ldots & & \\ & 0 & & & \sigma_p & \\ & & & & & 0 \\ & & & & & & \ldots \\ & & & & & & & 0 \end{bmatrix}, \quad p = \min\{m, n\}$$

- The real positive numbers σ_i are called the singular values of $\mathbf{A}$. They are arranged such that

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{p-1} \geq \sigma_p;$$

- If r is defined as $\sigma_1 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_p = 0$ then one can show that

$$r = \mathrm{rank}(\mathbf{A}).$$

Eigenvalue problems

- Moreover, if we write the matrices $\mathbf{U}$ and $\mathbf{V}$ in terms of their column vectors:

$$\mathbf{U} = [\mathbf{u}_1\mathbf{u}_2\ldots\mathbf{u}_m],\ \mathbf{V} = [\mathbf{v}_1\mathbf{v}_2\ldots\mathbf{v}_n],$$

one can prove the following SVD expansion of the matrix $\mathbf{A}$:

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

- The 2-norm and the Frobenius norms can be expressed in terms of the singular values:

$$\|\mathbf{A}\|_F^2 = \sigma_1^2 + \sigma_2^2 + \ldots + \sigma_p^2,\ p = \min\{m, n\}$$
$$\|\mathbf{A}\|_2 = \sigma_1.$$

- **QR** factorization is a common decomposition used in computing eigenvalues and eigenvectors.

- An $n \times n$ real matrix $\mathbf{A}$ can be written in form

$$\mathbf{A} = \mathbf{QR},$$

where $\mathbf{Q}$ is an orthogonal matrix ($\mathbf{Q}^T = \mathbf{Q}^{-1}$) and $\mathbf{R}$ is an upper triangular matrix.

Eigenvalue problems

• All methods to determine eigenvalues are iterative (cf. find zeroes of a polynomial)

• Power method

- Many more advanced methods are based on it.
- Finds λ with the largest absolute value and the corresponding eigenvector.
- Assume that $\mathbf{A}$ has eigenvalues $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_N|$ and linearly independent eigenvectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$.
- Further assume that $\lambda_i \in R$, and $\mathbf{x}_i \in R^N$; i.e. both eigenvalues and vectors are real.

- Basic algorithm:

1. Choose the initial vector $\mathbf{q}_0$

2. For $k = 1, 2, \ldots$ do

$\mathbf{z}_k = \mathbf{A}\mathbf{q}_{k-1}$

$\mathbf{q}_k = \mathbf{z}_k / \|\mathbf{z}_k\|$! *this keeps the eigenvector estimate normalized to one*

$\lambda_1^{(k)} = \mathbf{q}_k^* \mathbf{A} \mathbf{q}_k$! *eigenvalue estimate*

end

- It easy to show that $\lambda_1^{(k)} \to \lambda_1$ and $\mathbf{q}_k \to \mathbf{x}_1$ when $k \to \infty$:

Initial vector $\mathbf{q}_0 = a_1\mathbf{x}_1 + a_2\mathbf{x}_2 + \ldots + a_N\mathbf{x}_N$.

If $a_1 \neq 0 \quad \to \quad \mathbf{A}^k\mathbf{q}_0 = \sum_{i=1}^{N} a_i \mathbf{A}^k \mathbf{x}_i = \sum_{i=1}^{N} a_i \lambda_i^k \mathbf{x}_i = a_1 \lambda_1^k \left[\mathbf{x}_1 + \sum_{i=2}^{N} \frac{a_i}{a_1} \left(\frac{\lambda_i}{\lambda_1} \right)^k \mathbf{x}_i \right].$

Because $(\lambda_i / \lambda_1)^k \to 0 \ \ \forall i = 1, \ldots, N$ when $k \to \infty$ $\mathbf{q}_k$ approaches the eigenvector corresponding to λ_1.

Eigenvalue problems

- Convergence of the power method depends on the ratio $|\lambda_2| / |\lambda_1|$.

- If $\mathbf{A}$ has eigenvalues λ_i then $\mathbf{A}^{-1}$ has eigenvalues λ_i^{-1}. ($\mathbf{A}\mathbf{x}_i = \lambda_i \mathbf{x}_i \rightarrow \mathbf{A}^{-1}\mathbf{A}\mathbf{x}_i = \lambda_i \mathbf{A}^{-1}\mathbf{x}_i \rightarrow \mathbf{x}_i = \lambda_i \mathbf{A}^{-1}\mathbf{x}_i$)
  $\rightarrow$ Using the power method the smallest eigenvalue can be computed (**inverse iteration**).

- This can be combined with shift: $\mathbf{A}^{-1} \rightarrow (\mathbf{A} - \sigma \mathbf{1})^{-1} \rightarrow$ now we can determine the eigenvalue $\frac{1}{\lambda' - \sigma}$, where λ' is the eigenvalue of $\mathbf{A}$ that is closest to σ (**inverse iteration with shift)**.

- By applying the method with different values of σ all eigenvalues can in principle be determined.

- There are many ways to remove the effects of already determined eigenvalues from matrix $\mathbf{A}$
  - E.g. by doing the transformation $\mathbf{A} \rightarrow \mathbf{A} - \lambda_1 \mathbf{x}_1 \mathbf{x}_1^*$ the largest eigenvalue zeroed and the power method can be applied to determine the second largest one (**deflation**).

- A generalization of the power method is orthogonal iteration where a basis corresponding to $m < n$ largest eigenvalues is generated.

Eigenvalue problems

- **QR method** is based on the previous method and also uses the QR factorization to determine all eigenvalues.
  - Usually matrix $\mathbf{A}$ is similarity transformed to so called Hessenberg or tridiagonal form before the QR algorithm is applied.
  - This must be done by similarity transformations in order to preserve the eigenvalues of the original matrix.
  - Matrix $\mathbf{A}$ is a Hessenberg matrix if
    $a_{ij} = 0$, for all $i > j + 1$
    i.e. upper triangular except for a single nonzero subdiagonal.
    (Note that if the matrix is symmetric its Hessenberg form is tridiagonal.)
  - Applying the QR method to this matrix is much cheaper than to a general matrix.

  - This preliminary reduction is done by the *Householder transformation* or by *Givens rotation*.

  - Householder matrices $\mathbf{P}$ are orthogonal
    $\mathbf{P} = \mathbf{1} - 2\frac{\mathbf{u}\mathbf{u}^*}{\alpha}$, $\alpha = \|\mathbf{u}\|^2$,
    and the transformation is of the form
    $\tilde{\mathbf{A}} = \mathbf{P}_{n-2}\mathbf{P}_{n-1}\cdots\mathbf{P}_1\mathbf{A}\mathbf{P}_1^T\cdots\mathbf{P}_{n-1}^T\mathbf{P}_{n-2}^T$.

$$\mathbf{P}_1 \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} \mathbf{P}_1^* = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \end{bmatrix}$$

$$\mathbf{P}_2 \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \end{bmatrix} \mathbf{P}_2^* = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \end{bmatrix}$$

Eigenvalue problems

- In the QR algorithm a sequence of matrices $\mathbf{A}_i$, $\mathbf{Q}_i$, $\mathbf{R}_i$ is formed in the following fashion:

0. $\mathbf{A}_1 = \mathbf{A}$
1. Do the QR factorization: $\mathbf{A}_i = \mathbf{Q}_i\mathbf{R}_i$.
2. Compute the new iterate: $\mathbf{A}_{i+1} \leftarrow \mathbf{R}_i\mathbf{Q}_i$ (i.e. $\mathbf{A}_{i+1} = \mathbf{Q}_i^{-1}\mathbf{A}_i\mathbf{Q}_i = \mathbf{Q}_i^T\mathbf{A}_i\mathbf{Q}_i$)
3. $i \leftarrow i + 1$.
4. Go to 1.

- As a limit this iteration produces an upper triangular matrix $\mathbf{A}_i$.
- And remember: the eigenvalues of the original matrix $\mathbf{A}$ are the diagonal elements.
  - For proof see e.g. Golub-van Loan, Chapter 7.

- The iteration can be written as

$$\mathbf{A}_i = \mathbf{Q}_i^T\mathbf{Q}_{i-1}^T \ldots \mathbf{Q}_0^T\mathbf{A}\mathbf{Q}_1 \ldots \mathbf{Q}_{i-1}\mathbf{Q}_i.$$

- Because all $\mathbf{Q}_i$ are orthogonal then the above transformation is a similarity transformation and $\mathbf{A}_i$ has the same eigenvalues as the original matrix $\mathbf{A}$.

- When eigenvalues are known, *eigenvectors* can be obtained by e.g. inverse iteration with shift.

Eigenvalue problems

- LAPACK uses the QR method for eigenvalues and vectors

```
DGEEV - compute for an N-by-N real nonsymmetric matrix A, the eigenvalues
  and, optionally, the left and/or right eigenvectors
SYNOPSIS
  SUBROUTINE DGEEV( JOBVL, JOBVR, N, A, LDA, WR, WI, VL, LDVL, VR, LDVR,
                    WORK, LWORK, INFO )
      CHARACTER          JOBVL, JOBVR
      INTEGER            INFO, LDA, LDVL, LDVR, LWORK, N
      DOUBLE PRECISION   A(LDA,*), VL(LDVL,*), VR(LDVR,*), WI(*), WORK(*), WR(*)
```

- Matlab and Octave have function **eig**:

```
 EIG    Eigenvalues and eigenvectors.
    E = EIG(X) is a vector containing the eigenvalues of a square
    matrix X.

    [V,D] = EIG(X) produces a diagonal matrix D of eigenvalues and a
    full matrix V whose columns are the corresponding eigenvectors so
    that X*V = V*D.
```

Eigenvalue problems

- Example code using **DGEEV** (complex version of the routine):

```
! http://www.nacse.org/demos/lapack/codes/eigen-f.html

 program eigenvalue
   integer,parameter :: rk=8
   real(rk)::  A(3,3), DUMMY(3,3), WORK(9)
   real(rk) :: br(3),bi(3)
   integer i, ok

   A(1,1)=3.1; A(1,2)=1.3;  A(1,3)=-5.7
   A(2,1)=1.0; A(2,2)=-6.9; A(2,3)=5.8
   A(3,1)=3.4; A(3,2)=7.2;  A(3,3)=-8.8

   call DGEEV('N','N',3,A,3,br,bi,DUMMY,1,DUMMY,9,WORK,9,ok)

   if (ok==0) then
      do i=1, 3
           write(*,*) br(i),bi(i)
      enddo
   else
      write (*,*) "An error occured"
    endif

 end program eigenvalue
```

- Output:

```
f90> ifort -o eigendouble eigendouble.f90 -llapack
f90> ./eigendouble
  0.625482501680223        2.30344243268150
  0.625482501680223       -2.30344243268150
  -13.8509653848302       0.000000000000000E+000
```

Matlab:

```
A =
    3.1000    1.3000   -5.7000
    1.0000   -6.9000    5.8000
    3.4000    7.2000   -8.8000
>> [v,d]=eig(A)
v =
  -0.7160             -0.7160              0.2890
  -0.2686 + 0.3703i   -0.2686 - 0.3703i   -0.6375
  -0.3721 + 0.3738i   -0.3721 - 0.3738i    0.7142
d =
   0.6255 + 2.3034i         0                   0
        0              0.6255 - 2.3034i         0
        0                   0             -13.8510
```


Eigenvalue problems

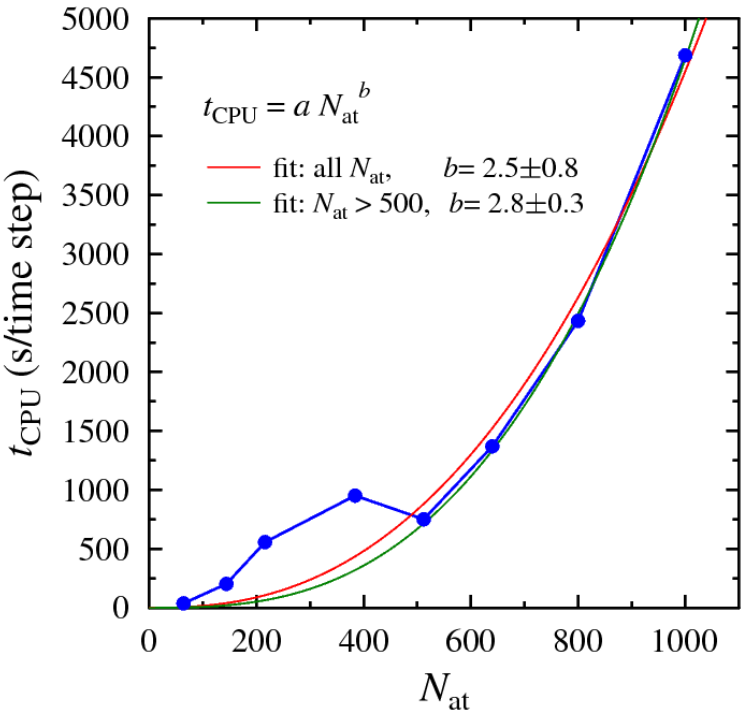
• Examples of execution times for calculating eigenvalues of large matrices using LAPACK:

```
Test AMD LAPACK and BLAS implementations in libacml.a
-----------------------------------------------------
Program: dynmat_eigen.f90
  Calculates all eigenvalues of a symmetric matrix (relaxat dynamical matrix, to be precise) using LAPACK
  routine dsyev.
A) LAPACK and BLAS in libacml.a used.
B) LAPACK and BLAS compiled from sources downloaded from netlib.
Compilation on ametisti and sepeli:
A) pathf90 -static -O3 -ipa -fno-math-errno -m64 -march=opteron -o dynmat_eigen_acml dynmat_eigen.f90 -L/home/akuronen/acml -lacml
B) pathf90 -static -O3 -ipa -fno-math-errno -m64 -march=opteron -o dynmat_eigen_default dynmat_eigen.f90 lapack.f blas.f dlamch.f
Run:
A) ./dynmat_eigen_acml    ${Nat} < dm_ico${Nat}.dat > ev_ico${Nat}.data
B) ./dynmat_eigen_default ${Nat} < dm_ico${Nat}.dat > ev_ico${Nat}.dat
```

```
Results from sepeli (AMD Opteron):
                          CPU time (s)
Nat     Matrix size         A             B          A/B
-----------------------------------------------------------
 147    441 x   441     0.078988     0.077989     1.012
1415   4245 x  4245    94.133690   159.87070      0.589
3871  11613 x 11613  1703.0691    3127.5565       0.545

Home computer (AMD Athlon):
(liblapack compiled from f90 sources at
                          CPU time (s)
Nat    Matrix size          A             B          A/B
-----------------------------------------------------------
 147   441 x   441      0.14997800   0.12798100   1.172
1415  4245 x  4245     81.391627    132.44986     0.615
3871 11613 x 11613   1899.9832     2682.6172      0.708
```

Eigenvalue problems

• Below is shown the CPU time usage of an ab inito code SIESTA for a silicon system.
  - Solution is performed by so called direct diagonalization; i.e. calculating eigenvalues.



Simulations by E. Holmström

Eigenvalue problems

• Summary:

- If you want all eigenvalues of a matrix that fits into your computers memory the QR method is a good choice.
  - I.e. use LAPACK or Matlab/Octave

- For computing only a few eigenvalues there are efficient methods that are also suitable for large sparse matrices.
  - Check out e.g. J. Haataja et al., *Numeeriset menetelmät käytännössä*, CSC, 1999.

- If you know the appoximate values of the eigenvalues then the power method with shift and inversion is a good method to improve the approximations.