Eigenvalue problems, basic definitions

Let us consider the matrix $A$ of dimension $n$. The eigenvalues of $A$ are defined through the matrix equation

$$Ax^{(ν)} = λ^{(ν)}x^{(ν)},$$

where $λ^{(ν)}$ are the eigenvalues and $x^{(ν)}$ the corresponding eigenvectors. Unless otherwise stated, when we use the wording eigenvector we mean the right eigenvector. The left eigenvalue problem is defined as

$$x^{(ν)}A = λ^{(ν)}x^{(ν)}.$$

The above right eigenvector problem is equivalent to a set of $n$ equations with $n$ unknowns $x_i$.

The eigenvalues of a matrix $A \in \mathbb{C}^{n \times n}$ are thus the roots of its characteristic polynomial

$$P(λ) = \det(A - λI),$$

or

$$P(λ) = \prod_{i=1}^{n}(λ_i - λ).$$

The set of these roots is called the spectrum and is denoted as $λ(A)$. If $λ(A) = \{λ_1, λ_2, \ldots, λ_n\}$ then we have

$$\det(A) = λ_1λ_2\ldotsλ_n,$$

and if we define the trace of $A$ as

$$Tr(A) = \sum_{i=1}^{n}a_{ii},$$

then

$$Tr(A) = λ_1 + λ_2 + \cdots + λ_n.$$

The above right eigenvector problem is equivalent to a set of $n$ equations with $n$ unknowns $x_i$.

Eigenvalue problems, basic definitions

The eigenvalue problem can be rewritten as

$$(A - λ^{(ν)}I)x^{(ν)} = 0,$$

with $I$ being the unity matrix. This equation provides a solution to the problem if and only if the determinant is zero, namely

$$\det(A - λ^{(ν)}I) = 0,$$

which in turn means that the determinant is a polynomial of degree $n$ in $λ$ and in general we will have $n$ distinct zeros.

Abel-Ruffini Impossibility Theorem

The Abel-Ruffini theorem (also known as Abel’s impossibility theorem) states that there is no general solution in radicals to polynomial equations of degree five or higher.

The content of this theorem is frequently misunderstood. It does not assert that higher-degree polynomial equations are unsolvable. In fact, if the polynomial has real or complex coefficients, and we allow complex solutions, then every polynomial equation has solutions; this is the fundamental theorem of algebra. Although these solutions cannot always be computed exactly with radicals, they can be computed to any desired degree of accuracy using numerical methods such as the Newton-Raphson method or Laguerre method, and in this way they are no different from solutions to polynomial equations of the second, third, or fourth degrees.

The theorem only concerns the form that such a solution must take. The content of the theorem is that the solution of a higher-degree equation cannot in all cases be expressed in terms of the polynomial coefficients with a finite number of operations of
The Abel-Ruffini theorem says that there are some fifth-degree equations whose solution cannot be so expressed. The equation $x^5 - x + 1 = 0$ is an example. Some other fifth degree equations can be solved by radicals, for example $x^4 - x^3 - x + 1 = 0$. The precise criterion that distinguishes between those equations that can be solved by radicals and those that cannot was given by Galois and is now part of Galois theory; a polynomial equation can be solved by radicals if and only if its Galois group is a solvable group.

Today, in the modern algebraic context, we say that second, third and fourth degree polynomial equations can always be solved by radicals if and only if its Galois group is a solvable group.

The variable $\lambda$ is an eigenvalue of $B$ as well, but with eigenvector $S^T x$.

The importance of a similarity transformation lies in the fact that the resulting matrix has the same eigenvalues, but the eigenvectors are in general different.

In the present discussion we assume that our matrix is real and symmetric, that is $A \in \mathbb{R}^{n \times n}$. The matrix $A$ has $n$ eigenvalues $\lambda_1, \ldots, \lambda_n$ (distinct or not). Let $D$ be the diagonal matrix with the eigenvalues on the diagonal

$$D = \begin{pmatrix}
\lambda_1 & 0 & 0 & \cdots & 0 \\
0 & \lambda_2 & 0 & \cdots & 0 \\
0 & 0 & \lambda_3 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \cdots & \cdots & \cdots & \lambda_n
\end{pmatrix}$$

If $A$ is real and symmetric then there exists a real orthogonal matrix $S$ such that

$$S^T AS = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n),$$

and for $j = 1 : n$ we have $A S(:,j) = \lambda_j S(:,j)$.

To prove this we start with the eigenvalue problem and a similarity transformed matrix $B$.

$$Ax = \lambda x \quad \text{and} \quad B = S^T AS.$$

We multiply the first equation on the left by $S^T$ and insert $S^T S = I$ between $A$ and $x$. Then we get

$$(S^T AS)(S^T x) = \lambda S^T x,$$

which is the same as

$$B (S^T x) = \lambda (S^T x).$$

The variable $\lambda$ is an eigenvalue of $B$ as well, but with eigenvector $S^T x$.

The general overview

One speaks normally of two main approaches to solving the eigenvalue problem.

- The first is the formal method, involving determinants and the characteristic polynomial. This proves how many eigenvalues there are, and is the way most of you learned about how to solve the eigenvalue problem, but for matrices of dimensions greater than 2 or 3, it is rather impractical.

- The other general approach is to use similarity or unitary transformations to reduce a matrix to diagonal form. This is normally done in two steps: first reduce to for example a tridiagonal form, and then to diagonal form. The main algorithms we will discuss in detail, Jacobi’s and Householder’s method, and then Lanczos algorithms (an iterative method) follow this methodology.
Discussion of Jacobi's method for eigenvalues

Direct or non-iterative methods require for matrices of dimensionality \( n \times n \) typically \( O(n^3) \) operations. These methods are normally called standard methods and are used for dimensionality \( n \leq 10^4 \) or smaller. A brief historical overview shows that in the course of 60 years the dimension that direct diagonalization methods can handle has increased by almost a factor of \( 10^4 \). However, it pales beside the progress achieved by computer hardware, from flops to petaflops, a factor of almost a factor of \( 10^{15} \). We see clearly played out in history the \( O(n^2) \) bottleneck of direct matrix algorithms. Sloppily speaking, when \( n \sim 10^4 \) is cubed we have \( O(10^{12}) \) operations, which is smaller than the \( 10^{18} \) increase in flops.

If the matrix to diagonalize is large and sparse, direct methods simply become impractical, also because many of the direct methods tend to destroy sparsity. As a result large dense matrices may arise during the diagonalization procedure. The idea behind iterative methods is to project the \( n \times n \) matrix \( A \) on smaller spaces, so-called Krylov subspaces. Given a matrix \( A \) and a vector \( v \), the associated Krylov sequences of vectors (and thereby Krylov subspaces) \( v, Av, A^2v, A^3v, \ldots \) represent successively larger Krylov subspaces.

The Numerical Recipes codes have been rewritten in Fortran 90/95 and C/C++ by us. The original source codes are taken from the widely used software package Lapack, which follows two other popular packages developed in the 1970s, namely EISPACK and LINPACK.

- Linpack: package for linear equations and least square problems.
- Lapack: package for solving symmetric, unsymmetric and generalized eigenvalue problems. From LAPACK's website http://www.netlib.org it is possible to download for free all source codes from this library. Both C/C++ and Fortran versions are available.
- BLAS (I, II and III): (Basic Linear Algebra Subprograms) are routines that provide standard building blocks for performing basic vector and matrix operations. Blas I is vector operations, II vector-matrix operations and III matrix-matrix operations.

It means that its matrix elements that differ from zero are given by

\[
\begin{align*}
\delta_{kk} &= c = \cos \theta, \\
\delta_{kl} &= s = \sin \theta, \\
\delta_{ij} &= \delta_{ji} = 0, \\
&\text{for } i \neq j
\end{align*}
\]

A similarity transformation

\[
B = S^TAS,
\]

results in

\[
\begin{align*}
\delta_{kk} &= -a_{kk} \cos \theta - a_{kk} \sin \theta, \\
\delta_{kl} &= -a_{kl} \cos \theta + a_{lk} \sin \theta, \\
\delta_{ij} &= \delta_{ji} = 0, \\
&\text{for } i \neq j
\end{align*}
\]

The angle \( \theta \) is arbitrary. The recipe is to choose \( \theta \) so that all non-diagonal matrix elements \( \delta_{kl} \) become zero.

Consider an example of an \((n \times n)\) orthogonal transformation matrix

\[
S = \begin{pmatrix}
1 & 0 & \ldots & 0 & 0 \\
0 & 1 & \ldots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \ldots & 0 & \cos \theta \\
0 & 0 & \ldots & 0 & \sin \theta
\end{pmatrix}
\]

with property \( S^T = S^{-1} \). It performs a plane rotation around an angle \( \theta \) in the Euclidean \( n \)-dimensional space.

The main idea is thus to reduce systematically the norm of the off-diagonal matrix elements of a matrix \( A \)

\[
\text{cond}(A) = \sum_{i \leq k < j \leq n} |a_{ij}|
\]

To demonstrate the algorithm, we consider the simple \( 2 \times 2 \) similarity transformation of the full matrix. The matrix is symmetric, so we single out \( 1 \leq k < j \leq n \) and use the abbreviations \( c = \cos \theta \) and \( s = \sin \theta \) to obtain

\[
\begin{pmatrix}
\delta_{kk} & 0 \\
0 & \delta_{jj}
\end{pmatrix} =
\begin{pmatrix}
c & -s \\
s & c
\end{pmatrix}
\begin{pmatrix}
a_{kk} & a_{kj} \\
a_{jk} & a_{jj}
\end{pmatrix}
\begin{pmatrix}
c & s \\
-s & c
\end{pmatrix}.
\]
Discussion of Jacobi’s method for eigenvalues

We require that the non-diagonal matrix elements $b_{kl} = b_{lk} = 0$, implying that
\[ a_{kl}(c^2 - s^2) + (a_{kk} - a_{ll})cs = b_{kl} = 0. \]
If $a_{kl} = 0$ one sees immediately that $\cos \theta = 1$ and $\sin \theta = 0$.

Discussion of Jacobi’s method for eigenvalues

Defining the quantities $\tan \theta = t = s/c$ and $\cot 2\theta = r = 2s - a_{kk}a_{ll}/2a_{kl}$, we obtain the quadratic equation (using $\cot 2\theta = 1/2(\cot \theta - \tan \theta)$)
\[
t^2 + 2rtc - 1 = 0,
\]
resulting in
\[
t = -r \pm \sqrt{1 + r^2},
\]
and $c$ and $s$ are easily obtained via
\[
c = \frac{1}{\sqrt{1 + r^2}}\theta
\]
and $s = tc$. Convince yourself that we have $|\theta| \leq \pi/4$. This has the effect of minimizing the difference between the matrices $B$ and $A$ since

Discussion of Jacobi’s method for eigenvalues

The convergence rate of the Jacobi method is however poor, one needs typically $3n^3 - 5n^2$ rotations and each rotation requires 4n operations, resulting in a total of $12n^3 - 20n^3$ operations in order to zero out non-diagonal matrix elements.

Discussion of Jacobi’s method for eigenvalues

The Frobenius norm of an orthogonal transformation is always preserved. The Frobenius norm is defined as
\[
\|A\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}|^2}.
\]
This means that for our $2 \times 2$ case we have
\[ 2a_{11}^2 + a_{22}^2 + s^2 c^2 - a_{12}^2 c^2 - b_{22} = 0, \]
which leads to
\[
\|B\|_F^2 = \|A\|_F^2 - \sum_{i=1}^{n} b_{ii}^2 - \|A\|^2 - 2a_{12}^2,
\]
since
\[
\|B\|_F^2 = \sum_{i=1}^{n} b_{ii}^2 - \|A\|_F^2 - \sum_{i=1}^{n} (a_{11} + a_{22}) c^2 - (a_{11} - a_{22}) s^2 - 2b_{12}.
\]

Choose a tolerance $\epsilon$, making it a small number, typically $10^{-8}$ or smaller.

Setup a while test where one compares the norm of the newly computed off-diagonal matrix elements
\[
\|B\|_F > \epsilon.
\]
Now choose the matrix elements $a_{kl}$ so that we have those with largest value, that is $|a_{kl}| = \max_{i,j \neq k,l} |a_{ij}|$.

Compute thereafter $r = (a_{kk} - a_{ll})/2a_{kl}$, $\tan \theta$ and $\sin \theta$.

Compute thereafter the similarity transformation for this set of values $(k,l)$, obtaining the new matrix
\[ B = S(k,l,\theta)^T AS(k,l,\theta). \]
Compute the new norm of the off-diagonal matrix elements and continue till you have satisfied $\|B\|_F \leq \epsilon$.

Discussion of Jacobi’s method for eigenvalues

We specialize to a symmetric $3 \times 3$ matrix $A$. We start the process as follows (assuming that $a_{23} = a_{32}$ is the largest non-diagonal) with $c = \cos \theta$ and $s = \sin \theta$
\[
B = \begin{pmatrix}
1 & 0 & 0 \\
0 & c & -s \\
0 & s & c
\end{pmatrix} \begin{pmatrix}
a_{21} & a_{22} & a_{23} \\
0 & a_{22} & a_{23} \\
0 & a_{23} & a_{22}
\end{pmatrix} \begin{pmatrix}
1 & 0 & 0 \\
0 & c & s \\
0 & -s & c
\end{pmatrix}.
\]
We will choose the angle $\theta$ in order to have $a_{23} = a_{32} = 0$. We get the symmetric matrix $B$.
\[
B = \begin{pmatrix}
a_{11} & a_{12}c - a_{13}s \\
a_{12}c + a_{13}s & a_{11} & a_{12}c - a_{13}s \\
a_{13}c - a_{12}s & a_{13}c + a_{12}s & a_{11}
\end{pmatrix}
\]
Note that $a_{11}$ is unchanged! As it should.
We will fix the angle \( \theta \) so that \( b_{23} = 0 \).

The more general expression for the new matrix elements are

\[
\begin{align*}
    b_{10} &= a_{10}, \ i \neq k, j \neq l \\
    b_{1i} &= a_{1i} \cos \theta - a_{1j} \sin \theta, \ i \neq k, i \neq j \\
    b_{2i} &= a_{2i} \cos \theta + a_{2j} \sin \theta, \ i \neq k, i \neq j \\
    b_{3i} &= a_{3i} \cos \theta - a_{3j} \sin \theta + a_{3j} \sin \theta + a_{3j} \sin \theta \\
    b_{3i} &= (a_{22} - a_{33}) \cos \theta \sin \theta + a_{23} \sin \theta + a_{33} \sin \theta + a_{33} \sin \theta
\end{align*}
\]

This is what we will need to code.

We repeat then assuming that \( b_{23} \) is the largest non-diagonal matrix element and get a new matrix

\[
C = \begin{pmatrix}
    c & s & 0 \\
    s & c & 0 \\
    0 & 0 & 1
\end{pmatrix}
\]

We continue this process till all non-diagonal matrix elements are zero (ideally). You will notice that performing the above operations that the matrix element \( b_{23} \) which was previous zero becomes different from zero. This is one of the problems which slows down the Jacobi procedure.

Finding the new matrix elements

```cpp
void Jacobi_rotate ( mat A, mat R, int k, ... = R(i,l);
R(i,k) = c*r_ik - s*r_il;
R(i,l) = c*r_il + s*r_ik;
}
return;
} // end of function jacobi_rotate
```

Finding the max nondiagonal element

```cpp
double jacobi_rotate ( mat A, mat R, int k, int l )
{
    double max;
    for (int i = 0; i < n; i++)
        for ( int j = i+1; j < n; j++)
            if ( A(i,j) > max )
            {
                max = A(i,j); p = i; q = j;
            }
    return;
}
```

Discussion of Jacobi’s method for eigenvalues

We get then a new matrix

\[
B = \begin{pmatrix}
    b_{11} & b_{12} & b_{13} \\
    b_{21} & b_{22} & b_{23} \\
    b_{31} & b_{32} & b_{33}
\end{pmatrix}
\]

This is what we will need to code.

Finding the max nondiagonal element

```cpp
void jacobi_rotate ( mat A, mat R, int k, int l )
{
    double max;
    for (int i = 0; i < n; i++)
        for ( int j = i+1; j < n; j++)
            if ( A(i,j) > max )
            {
                max = A(i,j); p = i; q = j;
            }
    return;
}
```

Discussion of Jacobi’s method for eigenvalues

Finding the new matrix elements

```cpp
void Jacobi_rotate ( mat A, mat R, int k, ... = R(i,l);
R(i,k) = c*r_ik - s*r_il;
R(i,l) = c*r_il + s*r_ik;
}
return;
} // end of function jacobi_rotate
```
We are first interested in the solution of the radial part of Schrödinger’s equation for one electron. This equation reads
\[
-\frac{\hbar^2}{2m} \frac{d^2}{d\rho^2} u(\rho) + \frac{1}{\rho} \frac{d}{d\rho} \left( \frac{l(l+1)}{\rho^2} \right) u(\rho) = E u(\rho).
\]
In our case \( V(\rho) \) is the harmonic oscillator potential \( (1/2)\kappa \rho^2 \) with \( \kappa = m \omega^2 \) and \( E \) is the energy of the harmonic oscillator in three dimensions. The oscillator frequency is \( \omega \) and the energies are
\[
E_n = \hbar \omega \left( 2n + \frac{3}{2} \right),
\]
with \( n = 0, 1, 2, \ldots \) and \( J = 0, 1, 2, \ldots \).

We introduce a dimensionless variable \( \rho = (1/\alpha) r \) where \( \alpha \) is a constant with dimension length and get
\[
-\frac{\hbar^2}{2m \alpha^2} \frac{d^2}{d\rho^2} u(\rho) + \left( \frac{V(\rho) + \frac{l(l+1)}{\rho^2}}{2 \hbar^2 / m \alpha^2} \right) u(\rho) = E u(\rho).
\]
In project 2 we chose \( l = 0 \). Inserting \( V(\rho) = (1/2) \kappa \alpha^2 \rho^2 \) we end up with
\[
-\frac{\hbar^2}{2m \alpha^2} \frac{d^2}{d\rho^2} u(\rho) + \frac{k}{2} \alpha^2 \rho^2 u(\rho) = E u(\rho).
\]
We multiply thereafter with \( 2m \alpha^2 / \hbar^2 \) on both sides and obtain
\[
-\frac{d^2}{d\rho^2} u(\rho) + \frac{mk}{\hbar^2} \alpha^2 \rho^2 u(\rho) = 2m \alpha^2 / \hbar^2 E u(\rho).
\]

We use the by now standard expression for the second derivative of a function \( u \)
\[
u'' = \frac{u(\rho + h) - 2u(\rho) + u(\rho - h)}{h^2} + O(h^2),
\]
where \( h \) is our step. Next we define minimum and maximum values for the variable \( \rho \), \( \rho_{\text{min}} = 0 \) and \( \rho_{\text{max}} \), respectively. You need to check your results for the energies against different values \( \rho_{\text{max}} \), since we cannot set \( \rho_{\text{max}} = \infty \).
With a given number of steps, $n_{\text{step}}$, we then define the step $h$ as
$$h = \frac{\rho_{\text{max}} - \rho_{\text{min}}}{n_{\text{step}}}.$$ 
Define an arbitrary value of $\rho$ as
$$\rho_i = \rho_{\text{min}} + i h, \quad i = 0, 1, 2, \ldots, n_{\text{step}}.$$ 
we can rewrite the Schrödinger equation for $\psi_i$ as
$$-\frac{\hbar^2}{2m} \frac{d^2}{dr^2} \psi_i(r) + \frac{\hbar^2}{2m} \frac{1}{4} k r^2 \psi_i(r) = E^{(1)} \psi_i(r),$$
where $E^{(1)}$ stands for the energy with one electron only. For two electrons with no repulsive Coulomb interaction, we have the following Schrödinger equation
$$\left( -\frac{\hbar^2}{2m} \frac{d^2}{dr^2} + \frac{1}{4} k r^2 + \frac{1}{2} k R^2 \right) \psi(r, R) = E^{(2)} \psi(r, R).$$
Note that we deal with a two-electron wave function $\psi(r_1, r_2)$ and two-electron energy $E^{(2)}$.
We add then the repulsive Coulomb interaction between two electrons, namely a term
$$V(r_1, r_2) = \frac{\beta e^2}{|r_1 - r_2|} \frac{|r_1 - r_2|}{r}$$
with $\beta e^2 = 1.44$ eVnm.

Define first the diagonal matrix element
$$d_i = \frac{2}{m} + V_i,$$
and the non-diagonal matrix element
$$e_i = -\frac{1}{m}$$
in this case the non-diagonal matrix elements are given by a mere constant. All non-diagonal matrix elements are equal. With these definitions the Schrödinger equation takes the following form
$$d_i u_i + e_i u_{i-1} + e_i u_{i+1} = \lambda u_i,$$
where $u_i$ is unknown. We can write the latter equation as a matrix eigenvalue problem
$$\begin{pmatrix} d_1 & e & 0 & 0 & \cdots & 0 & 0 \\ e & d_2 & e & 0 & \cdots & 0 & 0 \\ 0 & e & d_3 & e & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & d_{n_{\text{step}}} & e \\ e & 0 & 0 & 0 & \cdots & e & d_{n_{\text{step}}-1} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n_{\text{step}}-1} \\ u_{n_{\text{step}}} \end{pmatrix} = \lambda \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n_{\text{step}}-1} \\ u_{n_{\text{step}}} \end{pmatrix}.$$
Discussion of project 2

We want to manipulate this equation further to make it as similar to that in (a) as possible. We define a 'frequency'

$$\omega^2 = \frac{1}{m} \frac{\hbar^2}{\ell^2}$$

and fix the constant $a$ by requiring

$$\frac{mv^2}{\hbar^2} = 1$$

or

$$a = \frac{\hbar^2}{m/\ell^2}$$

Discussion of project 2

With no repulsive Coulomb interaction you should get a result which corresponds to the relative energy of a non-interacting system. Make sure your results are stable as functions of $\rho_{\text{max}}$ and the number of steps. We are only interested in the ground state with $J = 0$. We omit the center-of-mass energy.

For specific oscillator frequencies, the above equation has analytic answers, see the article by M. Taut, Phys. Rev. A 48, 3561 - 3566 (1993). The article can be retrieved from the following web address:

In order to determine each $S_i$, let us see what happens after the first multiplication, namely,

$$S_i^T A S_i = \begin{pmatrix} a_{11} & e_2 & 0 & \cdots & 0 \\ e_2 & a_{22} & \cdots & \cdots & \cdots \\ 0 & \cdots & a_{33} & \cdots & \cdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & a_{nn} \end{pmatrix},$$

where the primed quantities represent a matrix $A'$ of dimension $n - 1$ which will subsequently be transformed by $S_2$.

The factor $e_1$ is a possibly non-vanishing element. The next transformation produced by $S_2$ has the same effect as $S_1$ but now on the submatrix $A'$ only:

$$S_2^T A S_2 = \begin{pmatrix} a_{11} & e_2 & 0 & \cdots & 0 \\ e_2 & a_{22} & \cdots & \cdots & \cdots \\ 0 & \cdots & a_{33} & \cdots & \cdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & a_{nn} \end{pmatrix}.$$
Eigenvalues with the QR and Lanczos methods

Our Householder transformation has given us a tridiagonal matrix. We discuss here how one can use Jacobi’s iterative procedure to obtain the eigenvalues, although it may not be the best approach. Let us specialize to a $4 \times 4$ matrix. The tridiagonal matrix takes the form

$$A = \begin{pmatrix} d_1 & e_1 & 0 & 0 \\ e_1 & d_2 & e_2 & 0 \\ 0 & e_2 & d_3 & e_3 \\ 0 & 0 & e_3 & d_4 \end{pmatrix}$$

As a first observation, if any of the elements $e_i$ are zero the matrix can be separated into smaller pieces before diagonalization. Specifically, if $e_1 = 0$ then $d_1$ is an eigenvalue.

This procedure can be continued on the remaining three-dimensional submatrix for the next eigenvalue. Thus after few transformations we have the wanted diagonal form. What we see here is just a special case of the more general procedure developed by Francis in two articles in 1961 and 1962. The algorithm is based on the so-called QR method (or just QR-algorithm). It follows from a theorem by Schur which states that any square matrix can be written out in terms of an orthogonal matrix $Q$ and an upper triangular matrix $U$. Historically $U$ was used instead of $U$ since the wording right triangular matrix was first used. The method is based on an iterative procedure similar to Jacobi’s method, by a succession of planar rotations. For a tridiagonal matrix it is simple to carry out in principle, but complicated in detail! We will discuss this in more detail during week 38.

Solving the latter equation gives us $u$ and thus the needed transformation $P$. We do first however need to compute the scalar $k$ by taking the scalar product of the last equation with its transpose and using the fact that $P^2 = I$. We get then

$$(Pv)^T Pv - k^2 = v^T v - |v|^2 = \sum_{i=2}^{n} d_i^2,$$

which determines the constant $k = \pm v$.

Thus, let us introduce a transformation $S_1$ which operates like

$$S_1 = \begin{pmatrix} \cos \theta & 0 & 0 & \sin \theta \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \cos \theta & 0 & 0 & \cos \theta \end{pmatrix}$$

Then the similarity transformation

$$S_1^T A S_1 = A' = \begin{pmatrix} d_1' & e_1' & 0 & 0 \\ e_1' & d_2' & e_2' & 0 \\ 0 & e_2' & d_3' & e_3' \\ 0 & 0 & e_3' & d_4' \end{pmatrix}$$

produces a matrix where the primed elements in $A'$ have been changed by the transformation whereas the unprimed elements are unchanged. If we now choose $\theta$ to give the element $x_{22}' = e - 0$ then we have the first eigenvalue $= \lambda_1 = d_1'$. (This is actually what you are doing in project 2!!)

Now we can rewrite Eq. (6) as

$$v - k e = 2a(u^T v),$$

and taking the scalar product of this equation with itself and obtain

$$2(u^T v)^2 = (v^2 \pm a_21 v),$$

which finally determines

$$u = \frac{v \pm k e}{2(u^T v)}.$$
Eigenvalues with the QR algorithm and Lanczos' method

The method is based on an iterative procedure similar to Jacobi’s method, by a succession of planar rotations. For a tridiagonal matrix it is simple to carry out in principle, but complicated in detail!

Schur’s theorem

\( \hat{A} = Q \hat{U} \)

is used to rewrite any square matrix into a unitary matrix times an upper triangular matrix. We say that a square matrix is similar to a triangular matrix.

Householder’s algorithm which we have derived is just a special case of the general Householder algorithm. For a symmetric square matrix we obtain a tridiagonal matrix.

There is a corollary to Schur’s theorem which states that every Hermitian matrix is unitarily similar to a diagonal matrix.

Suppose \( \hat{A} \) is the triangular matrix we obtained after the Householder transformation,

\( \hat{A} = Q \hat{U} \)

and multiply from the left with \( \hat{Q}^{-1} \) resulting in

\( \hat{Q}^{-1} \hat{A} \hat{U} \).

Suppose that \( \hat{Q} \) consists of a series of planar Jacobi like rotations acting on sub blocks of \( \hat{A} \) so that all elements below the diagonal are zeroed out

\( \hat{Q} = \hat{R}_{12} \hat{R}_{23} \ldots \hat{R}_{n-1,n} \).

A transformation of the type \( \hat{R}_{12} \) looks like

\[
\begin{pmatrix}
  c & s & 0 & 0 & 0 & 0 & 0 \\
-\overline{s} & c & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\]

Eigenvalues with the QR and Lanczos methods

If we now choose \( \theta \) to give the element \( \hat{a}_{21}^\prime = e^\prime = 0 \) then we have the first eigenvalue \( \hat{\lambda}_1 = \hat{d}_1 \).

This procedure can be continued on the remaining three-dimensional submatrix for the next eigenvalue. Thus after few transformations we have the wanted diagonal form.

What we see here is just a special case of the more general Householder transformation, \( R \) was used instead of \( U \) since the wording right triangular matrix was first used.

There is a corollary to Schur’s theorem which states that every Hermitian matrix is unitarily similar to a diagonal matrix.

The algorithm is based on the so-called QR method (or just QR-algorithm). It follows from a theorem by Schur which states that any square matrix can be written out in terms of an orthogonal matrix \( Q \) and an upper triangular matrix \( \hat{U} \). Historically \( R \) was used instead of \( U \) since the wording right triangular matrix was first used.
The matrix $A$ is transformed into a tridiagonal form and the last step is to transform it into a diagonal matrix giving the eigenvalues on the diagonal. The eigenvalues of a matrix can be obtained using the characteristic polynomial

$$P(\lambda) = \det(A - \lambda I) = \prod_{i=1}^{n} (\lambda - \lambda_i).$$

which rewritten in matrix form reads

$$P(\lambda) = Q^{-1} \lambda Q - B,$$

where $Q$ and $\hat{U}$ and this allows us to find the matrix $B$ which is, due to Schur’s theorem, unitarily similar to a triangular matrix (upper in our case) since we have that

$$P_k(\lambda) = (d_k - \lambda)P_{k-1}(\lambda) - e_k^2 P_{k-2}(\lambda).$$

Together with the starting values $P_1(\lambda)$ and $P_2(\lambda)$ and good root searching methods we arrive at an efficient computational scheme for finding the roots of $P_n(\lambda)$. However, for large matrices this algorithm is rather inefficient and time-consuming.

The matrix $\hat{U}$ takes then the form

$$\hat{U} = \begin{pmatrix} x & x & 0 & 0 & \cdots & 0 & 0 \\ 0 & x & x & 0 & \cdots & 0 & 0 \\ 0 & 0 & x & x & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & x & x \\ 0 & 0 & 0 & 0 & \cdots & 0 & x \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \end{pmatrix},$$

which has a second superdiagonal.

Basic features with a real symmetric matrix (and normally huge $n > 10^6$ and sparse) $\hat{A}$ of dimension $n \times n$:

- Lanczos’ algorithm generates a sequence of real tridiagonal matrices $T_k$ of dimension $k \times k$ with $k \leq n$, with the property that the extremal eigenvalues of $T_k$ are progressively better estimates of $\hat{A}$ extremal eigenvalues.
- The similarity transformation is

$$T = Q^T \hat{A} Q,$$

with the first vector $Q_1 = q_1$. We are going to solve iteratively

$$T - Q^T \hat{A} Q = 0,$$

with the first vector $Q_1 = q_1$. We can write out the matrix $\hat{Q}$ in terms of its column vectors

$$\hat{Q} = \begin{pmatrix} \hat{q}_1 \\ \hat{q}_2 \\ \vdots \\ \hat{q}_n \end{pmatrix}.$$
Eigenvalues and Lanczos’ method, tridiagonal and orthogonal matrices

Using the fact that $\hat{Q} \hat{Q}^T = \hat{I}$, we can rewrite

$$\hat{T} = \hat{Q}^T \hat{A} \hat{Q},$$

as

$$\hat{Q}^T = \hat{A} \hat{Q}.$$

Eigenvalues and Lanczos’ method

If we equate columns

$$\hat{T} = \begin{pmatrix}
\alpha_1 & \beta_1 & 0 & \cdots & 0 \\
\beta_1 & \alpha_2 & \beta_2 & 0 & \cdots \\
0 & \beta_2 & \alpha_3 & \beta_3 & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots \\
0 & \vdots & 0 & \beta_{n-2} & \alpha_{n-1} \\
0 & \ddots & \ddots & \ddots & \ddots \\
0 & 0 & \beta_{n-1} & \alpha_n
\end{pmatrix},$$

we obtain

$$\hat{A} \hat{q}_k = \beta_{k-1} \hat{q}_{k-1} + \alpha_k \hat{q}_k + \beta_k \hat{q}_{k+1}.$$

Eigenvalues and Lanczos’ method, defining the Lanczos’ vectors

We have thus

$$\hat{A} \hat{q}_k = \beta_{k-1} \hat{q}_{k-1} + \alpha_k \hat{q}_k + \beta_k \hat{q}_{k+1},$$

with $\beta_k \hat{q}_0 = 0$ for $k = 1 : n - 1$. Remember that the vectors $\hat{q}_k$ are orthonormal and this implies

$$\alpha_k = \hat{q}_k^T \hat{A} \hat{q}_k,$$

and these vectors are called Lanczos vectors.

Eigenvalues and Lanczos’ method, basic steps

We have thus

$$\hat{A} \hat{q}_k = \beta_{k-1} \hat{q}_{k-1} + \alpha_k \hat{q}_k + \beta_k \hat{q}_{k+1},$$

with $\beta_k \hat{q}_0 = 0$ for $k = 1 : n - 1$ and

$$\alpha_k = \hat{q}_k^T \hat{A} \hat{q}_k.$$

If

$$\hat{r}_k = (\hat{A} - \alpha_k \hat{I}) \hat{q}_k - \beta_{k-1} \hat{q}_{k-1},$$

is non-zero, then

$$\hat{q}_{k+1} = \hat{r}_k / \beta_k,$$

with $\beta_k = \pm ||\hat{r}_k||_2$. 

If

$$\hat{r}_k = (\hat{A} - \alpha_k \hat{I}) \hat{q}_k - \beta_{k-1} \hat{q}_{k-1},$$

is non-zero, then

$$\hat{q}_{k+1} = \hat{r}_k / \beta_k,$$

with $\beta_k = \pm ||\hat{r}_k||_2$. 

If

$$\hat{r}_k = (\hat{A} - \alpha_k \hat{I}) \hat{q}_k - \beta_{k-1} \hat{q}_{k-1},$$

is non-zero, then

$$\hat{q}_{k+1} = \hat{r}_k / \beta_k,$$

with $\beta_k = \pm ||\hat{r}_k||_2$. 

If

$$\hat{r}_k = (\hat{A} - \alpha_k \hat{I}) \hat{q}_k - \beta_{k-1} \hat{q}_{k-1},$$

is non-zero, then

$$\hat{q}_{k+1} = \hat{r}_k / \beta_k,$$

with $\beta_k = \pm ||\hat{r}_k||_2$. 

If

$$\hat{r}_k = (\hat{A} - \alpha_k \hat{I}) \hat{q}_k - \beta_{k-1} \hat{q}_{k-1},$$

is non-zero, then

$$\hat{q}_{k+1} = \hat{r}_k / \beta_k,$$

with $\beta_k = \pm ||\hat{r}_k||_2$. 

If

$$\hat{r}_k = (\hat{A} - \alpha_k \hat{I}) \hat{q}_k - \beta_{k-1} \hat{q}_{k-1},$$

is non-zero, then

$$\hat{q}_{k+1} = \hat{r}_k / \beta_k,$$

with $\beta_k = \pm ||\hat{r}_k||_2$. 

If

$$\hat{r}_k = (\hat{A} - \alpha_k \hat{I}) \hat{q}_k - \beta_{k-1} \hat{q}_{k-1},$$

is non-zero, then

$$\hat{q}_{k+1} = \hat{r}_k / \beta_k,$$

with $\beta_k = \pm ||\hat{r}_k||_2$. 

If

$$\hat{r}_k = (\hat{A} - \alpha_k \hat{I}) \hat{q}_k - \beta_{k-1} \hat{q}_{k-1},$$

is non-zero, then

$$\hat{q}_{k+1} = \hat{r}_k / \beta_k,$$

with $\beta_k = \pm ||\hat{r}_k||_2$. 

If

$$\hat{r}_k = (\hat{A} - \alpha_k \hat{I}) \hat{q}_k - \beta_{k-1} \hat{q}_{k-1},$$

is non-zero, then

$$\hat{q}_{k+1} = \hat{r}_k / \beta_k,$$

with $\beta_k = \pm ||\hat{r}_k||_2$. 

If

$$\hat{r}_k = (\hat{A} - \alpha_k \hat{I}) \hat{q}_k - \beta_{k-1} \hat{q}_{k-1},$$

is non-zero, then

$$\hat{q}_{k+1} = \hat{r}_k / \beta_k,$$

with $\beta_k = \pm ||\hat{r}_k||_2$. 

If

$$\hat{r}_k = (\hat{A} - \alpha_k \hat{I}) \hat{q}_k - \beta_{k-1} \hat{q}_{k-1},$$

is non-zero, then

$$\hat{q}_{k+1} = \hat{r}_k / \beta_k,$$

with $\beta_k = \pm ||\hat{r}_k||_2$. 

If

$$\hat{r}_k = (\hat{A} - \alpha_k \hat{I}) \hat{q}_k - \beta_{k-1} \hat{q}_{k-1},$$

is non-zero, then

$$\hat{q}_{k+1} = \hat{r}_k / \beta_k,$$

with $\beta_k = \pm ||\hat{r}_k||_2$. 

If

$$\hat{r}_k = (\hat{A} - \alpha_k \hat{I}) \hat{q}_k - \beta_{k-1} \hat{q}_{k-1},$$

is non-zero, then

$$\hat{q}_{k+1} = \hat{r}_k / \beta_k,$$

with $\beta_k = \pm ||\hat{r}_k||_2$. 

If

$$\hat{r}_k = (\hat{A} - \alpha_k \hat{I}) \hat{q}_k - \beta_{k-1} \hat{q}_{k-1},$$

is non-zero, then

$$\hat{q}_{k+1} = \hat{r}_k / \beta_k,$$

with $\beta_k = \pm ||\hat{r}_k||_2$. 

If

$$\hat{r}_k = (\hat{A} - \alpha_k \hat{I}) \hat{q}_k - \beta_{k-1} \hat{q}_{k-1},$$

is non-zero, then

$$\hat{q}_{k+1} = \hat{r}_k / \beta_k,$$

with $\beta_k = \pm ||\hat{r}_k||_2$. 

If

$$\hat{r}_k = (\hat{A} - \alpha_k \hat{I}) \hat{q}_k - \beta_{k-1} \hat{q}_{k-1},$$

is non-zero, then

$$\hat{q}_{k+1} = \hat{r}_k / \beta_k,$$

with $\beta_k = \pm ||\hat{r}_k||_2$. 

If

$$\hat{r}_k = (\hat{A} - \alpha_k \hat{I}) \hat{q}_k - \beta_{k-1} \hat{q}_{k-1},$$

is non-zero, then

$$\hat{q}_{k+1} = \hat{r}_k / \beta_k,$$

with $\beta_k = \pm ||\hat{r}_k||_2$. 

If

$$\hat{r}_k = (\hat{A} - \alpha_k \hat{I}) \hat{q}_k - \beta_{k-1} \hat{q}_{k-1},$$

is non-zero, then

$$\hat{q}_{k+1} = \hat{r}_k / \beta_k,$$