5 Numerical computation of eigenvalues

The second basic problem of linear algebra is to compute the eigenvalues and eigenvectors of a square matrix. In other words, to diagonalize a square matrix.

In theory we know what to do. We have to compute the characteristic polynomial $p(\lambda)$, find the roots λ_i, then for each i we have the equation $(A - \lambda_i)\mathbf{x} = \mathbf{0}$ for the eigenvectors. However, this procedure is not satisfactory.

Newton's method give a very fast way to compute roots of polynomials $p(\lambda)$ (it can easily be extended to complex roots). But remember, that Newton's method usually works very well once you are reasonably close to the root, but it may diverge if you start far away. Still you need some *ad hoc* method to get "close" to the root. After we found one root λ_1, we can certainly factor out $(\lambda - \lambda_1)$ from the polynomial $p(\lambda)$, but this requires a polynomial division, which could be quite unstable (remember, division is dangerous). We get a polynomial of smaller degree, but then again we need some *ad hoc* method to get close to the next root before we can run Newton's method again. And even if we have found all eigenvalues, we still have to solve n different linear equations. We know how to do it in a reasonable way, but still the procedure is long. It would be nice to solve the full diagonalization problem at once. Moreover, we also have to care about the stability of the method, as the eigenvalue problem can be quite ill-conditioned (see an example below), so we do not want to make it much worse by an unstable algorithm.

We also remark, that the problem of finding eigenvalues cannot be "easier" than finding roots of a given polynomial (CONSTRUCT(*) a matrix with a given polynomial $p(\lambda) = \lambda^n + a_{n-1}\lambda^{n-1} + \dots$ as its characteristic polynomial). We know that these roots in general cannot be found with finitely many elementary operations if the degree of the polynomial is larger than four. Hence we cannot hope for *direct* methods in case of the eigenvalue problem. All algorithms will be iterative.

We discuss three methods. The power method and the QR iteration will find only the eigenvalues but for general matrices. For the eigenvectors we still have to solve the equations $(A - \lambda_i)\mathbf{x} = \mathbf{0}$ one by one. The Jacobi iteration will diagonalize a square matrix with one single (but not very fast) algorithm.

We remark that one should not underestimate the importance of the algorithms aiming only at the eigenvalues. One could think that there should be a simpler way to solve a characteristic equation than playing with the matrix. From theoretical point of view it is true that reducing the eigenvalue problem to a polynomial equation is a simplifying step (eventually a polynomial of one variable looks "simpler" than an $n \times n$ matrix). From numerical point of view sometimes it could be opposite. In fact matrix eigenvalue algorithms are actually used to find roots of given polynomials. In this case one is given a polynomial whose zeros are to be found and one constructs a matrix such that the characteristic polynomial be the given polynomial. Then one runs one of these algorithms to find the eigenvalues of the given matrix, which will give the solution to the original problem as well.

We close this section by an example showing that the eigenvalue problem can be ill-conditioned.

EXAMPLE. Consider the $n \times n$ matrix

$$A = \begin{pmatrix} 0 & & & & & & \varepsilon \\ 1 & 0 & & & & & \\ & 1 & 0 & & & & \\ & & 1 & 0 & & & \\ & & & 1 & 0 & & \\ & & & & \ddots & \ddots & \\ & & & & & 1 & 0 \end{pmatrix}$$

where ε is a tiny number (all other entries are zero). The characteristic polynomial of A is

$$p(\lambda) = \lambda^n - \varepsilon$$

(CHECK?(*)). Clearly all eigenvalues are 0 if $\varepsilon = 0$. However for $\varepsilon \neq 0$ there are n complex eigenvalues, the n-th roots of unity. Even if we focus on the real eigenvalue $\varepsilon^{1/n}$, it is clear that the problem is very ill-conditioned. Let $n = 40$ and choose $\varepsilon = 10^{-40}$ which is an extremely tiny relative error of order $10^{-40}/1 = 10^{-40}$ (you have to compare it with other entries of the matrix). However, one of the eigenvalue is $\lambda = 10^{-1} = 0.1$, which is at a distance 10^{-1} from the "unperturbed" eigenvalue 0. Hence the change in the eigenvalues is equal to the change in the perturbation parameter ε *multiplied by* 10^{39}!!

There is another disquieting aspect of this phenomenon. The number $\varepsilon = 10^{-40}$ is automatically replaced by 0 in the computer, and this rounding introduces an error of order 10^{-1} in the result.

Fortunately the eigenvalue problem is not so ill-conditioned for "most" matrices, this example was as particularly nasty one. However, it shows that we should aim at stable algorithms, which at least do not make "bad things much worse", since the problem itself can be quite "bad".

5.1 Power method for eigenvalues

This method is extremely simple. Let A be a square matrix. Pick any vector $\mathbf{u}_0$ and start successively multiplying it with A. We claim, that unless you are extremely unlucky with the choice of $\mathbf{u}_0$, we easily find the eigenvalue of largest modulus.

Theorem 5.1 *Suppose that the square matrix A has one eigenvalue λ_1 which is greater in absolute value than all other eigenvalues, and let $\mathbf{u}_{n+1} = A\mathbf{u}_n$. Then for randomly chosen nonzero vectors $\mathbf{u}_0$ and $\mathbf{w}$ we have*

$$\lim_{n\to\infty} \frac{\mathbf{w}^t \cdot \mathbf{u}_{n+1}}{\mathbf{w}^t \cdot \mathbf{u}_n} = \lambda_1.$$

"almost surely". Moreover, the sequence of normalized vectors $\widetilde{\mathbf{u}}_n := \frac{\mathbf{u}_n}{\|\mathbf{u}_n\|}$ *converges to the eigenvector of* λ_1*.*

REMARK: The probabilistic expression "almost surely" in the theorem can be made mathematically precise (it means "for almost all" vectors, i.e. with the exception of vectors lying in a smaller subspace) but we do not want to go into this detail. For the present purpose you should take it literally.

Proof: We give a proof only for the case when A is a symmetric k by k matrix i.e. $A = VDV^t$, and λ_1 is a simple eigenvalue. The proof for nonsymmetric diagonalizable matrices is slightly harder, and the case of a general matrix A is moderately harder.

We can use the spectral theorem, i.e.

$$A = \sum_{i=1}^{k} \lambda_i \mathbf{v}_i \mathbf{v}_i^t$$

Clearly

$$A^n = \sum_{i=1}^{k} \lambda_i^n \mathbf{v}_i \mathbf{v}_i^t$$

and

$$\mathbf{w}^t \cdot \mathbf{u}_n = \mathbf{w}^t \cdot A^n \mathbf{u}_0 = \sum_{i=1}^{k} \lambda_i^n (\mathbf{w}^t \cdot \mathbf{v}_i)(\mathbf{v}_i^t \cdot \mathbf{u}_0)$$

We compute

$$\lim_{n\to\infty} \frac{\mathbf{w}^t \cdot \mathbf{u}_n}{\lambda_1^n} = \lim_{n\to\infty} \sum_{i=1}^{k} \Big(\frac{\lambda_i}{\lambda_1}\Big)^n (\mathbf{w}^t \cdot \mathbf{v}_i)(\mathbf{v}_i^t \cdot \mathbf{u}_0) = (\mathbf{w}^t \cdot \mathbf{v}_1)(\mathbf{v}_1^t \cdot \mathbf{u}_0)$$

since $|\lambda_1| > |\lambda_i|$ for $i > 1$. Assume that $\mathbf{u}_0$ and $\mathbf{w}$ are not orthogonal to $\mathbf{v}_1$ (this is the case that one has to exclude by choosing the vectors "randomly"). Then

$$\lim_{n\to\infty} \frac{\mathbf{w}^t \cdot \mathbf{u}_{n+1}}{\mathbf{w}^t \cdot \mathbf{u}_n} = \lambda_1 \frac{\lim_{n\to\infty} \frac{\mathbf{w}^t \cdot \mathbf{u}_{n+1}}{\lambda_1^{n+1}}}{\lim_{n\to\infty} \frac{\mathbf{w}^t \cdot \mathbf{u}_n}{\lambda_1^n}} = \lambda_1 \frac{(\mathbf{w}^t \cdot \mathbf{v}_1)(\mathbf{v}_1^t \cdot \mathbf{u}_0)}{(\mathbf{w}^t \cdot \mathbf{v}_1)(\mathbf{v}_1^t \cdot \mathbf{u}_0)} = \lambda_1,$$

which completes the proof of the eigenvalue convergence.

To see the convergence of the normalized eigenvectors, notice that

$$\mathbf{u}_n = A^n \mathbf{u}_0 = \sum_{i=1}^{k} \lambda_i^n \mathbf{v}_i (\mathbf{v}_i^t \cdot \mathbf{u}_0)$$

hence $\|\mathbf{u}_n\| = \sqrt{\sum_{i=1}^{k} |\lambda_i|^{2n} (\mathbf{v}_i^t \cdot \mathbf{u}_0)^2}$, so

$$\widetilde{\mathbf{u}}_n = \frac{\mathbf{u}_n}{\|\mathbf{u}_n\|} = \frac{\sum_{i=1}^{k} \lambda_i^n \mathbf{v}_i (\mathbf{v}_i^t \cdot \mathbf{u}_0)}{\sqrt{\sum_{i=1}^{k} |\lambda_i|^{2n} (\mathbf{v}_i^t \cdot \mathbf{u}_0)^2}}$$

$$= \Big(\frac{\lambda_1 (\mathbf{v}_1^t \cdot \mathbf{u}_0)}{|\lambda_1 (\mathbf{v}_1^t \cdot \mathbf{u}_0)|} \Big)^n \frac{\mathbf{v}_1 + \sum_{i=2}^{k} \left(\frac{\lambda_i}{\lambda_1} \right)^n \frac{\mathbf{v}_i^t \cdot \mathbf{u}_0}{\mathbf{v}_1^t \cdot \mathbf{u}_0} \cdot \mathbf{v}_i}{\sqrt{1 + \sum_{i=2}^{k} \left| \frac{\lambda_i}{\lambda_1} \right|^{2n} \frac{(\mathbf{v}_i^t \cdot \mathbf{u}_0)^2}{(\mathbf{v}_1^t \cdot \mathbf{u}_0)^2}}}$$

and if $|\lambda_1| > |\lambda_i|$ for all other i, then this clearly converges to $\mathbf{v}_1$. $\square$.

This method gave only the eigenvalue of largest modulus. With a small trick, one can get all other eigenvalues,

Theorem 5.2 *Suppose that the square matrix A has an eigenvalue λ_p which is closer to p than all other eigenvalues. Run the*

$$\mathbf{u}_{n+1} = (A - pI)^{-1} \mathbf{u}_n$$

iteration with some initial vector $\mathbf{u}_0$. If the vectors $\mathbf{u}_0$ and $\mathbf{w}$ are chosen randomly, then

$$\lim_{n \to \infty} \frac{\mathbf{w}^t \cdot \mathbf{u}_{n+1}}{\mathbf{w}^t \cdot \mathbf{u}_n} = \frac{1}{\lambda_p - p}$$

"almost surely". Hence λ_p is obtained as

$$\lambda_p = \lim_{n \to \infty} \frac{\mathbf{w}^t \cdot \mathbf{u}_n}{\mathbf{w}^t \cdot \mathbf{u}_{n+1}} + p$$

(if $\lambda_p \neq p$). Moreover, again, the normalized vectors $\widetilde{\mathbf{u}}_n = \frac{\mathbf{u}_n}{\|\mathbf{u}_n\|}$ converge to the eigenvector belonging to λ_p.

Proof: Simply notice that the eigenvalues of $B := (A - pI)^{-1}$ are $(\lambda_1 - p)^{-1}$, $(\lambda_2 - p)^{-1}, \ldots$, where λ_i are the eigenvalues of A (WHY?(*) Hint: this statement does NOT require A to be symmetric, so spectral theorem cannot be used). From the condition it follows that $(\lambda_p - p)^{-1}$ is the largest in modulus among them, so we can apply Theorem 5.1. The statement on the eigenvector is straighforward from Theorem 5.1. $\square$.

REMARK: The power method looks very simple and elegant. However, notice it is really powerful only for the eigenvalue largest in modulus. Applying Theorem 5.2 already requires inverting a matrix, and more importantly it requires to know a point p "near" the eigenvalue. Hence we run into similar difficulties when applied Newton's iteration for finding the roots of the characteristic polynomial. In fact there is essentially a Newton's iteration behind Theorem 5.2.

Problem 5.3 *(i) Find the eigenvalues and the normalized eigenvectors of* $A = \begin{pmatrix} 3 & 1 \\ 2 & 4 \end{pmatrix}$ *with the standard method (characteristic polynomial)*

(ii) Find the larger (in modulus) eigenvalue λ_1 *of* $A = \begin{pmatrix} 3 & 1 \\ 2 & 4 \end{pmatrix}$ *and the corresponding eigenvector.*

(iii) Knowing that 3 is closer to the other eigenvalue λ_2 *than to* λ_1*, find it and find an eigenvector.*

SOLUTION: (i) This is standard: $\lambda_1 = 5$, $\lambda_2 = 2$, the corresponding eigenvectors are $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$, i.e., after normalization they are $\begin{pmatrix} .4472 \\ .8944 \end{pmatrix}$ and $\begin{pmatrix} .707 \\ -.707 \end{pmatrix}$

(ii) We can start from any nonzero vector $\mathbf{u}_0$, e.g $\mathbf{u}_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. We can also choose $\mathbf{w}$ arbitrarily, for example we choose $\mathbf{w} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, i.e. $\mathbf{w}^t \cdot \mathbf{u}_n$ will test the first entry. The iteration gives:

$$\mathbf{u}_1 = A\mathbf{u}_0 = \begin{pmatrix} 3 & 1 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 6 \end{pmatrix}$$

$$\mathbf{u}_2 = A\mathbf{u}_1 = \begin{pmatrix} 3 & 1 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} 4 \\ 6 \end{pmatrix} = \begin{pmatrix} 18 \\ 32 \end{pmatrix}$$

etc:

$$\mathbf{u}_3 = \begin{pmatrix} 86 \\ 164 \end{pmatrix} \qquad \mathbf{u}_4 = \begin{pmatrix} 422 \\ 828 \end{pmatrix} \qquad \mathbf{u}_5 = \begin{pmatrix} 2094 \\ 4156 \end{pmatrix} \qquad \mathbf{u}_6 = \begin{pmatrix} 10438 \\ 20812 \end{pmatrix} \qquad \mathbf{u}_7 = \begin{pmatrix} 52126 \\ 104124 \end{pmatrix} \dots$$

Now we compute the consecutive ratios of the numbers $\mathbf{w}^t \cdot \mathbf{u}_n$, i.e. the first entries:

$$\frac{4}{1} = 4 \qquad \frac{18}{4} = 4.5 \qquad \frac{86}{18} = 4.77 \qquad \frac{422}{86} = 4.9 \qquad \frac{2094}{422} = 4.96$$

$$\frac{10438}{2094} = 4.984 \qquad \frac{52126}{10438} = 4.993$$

i.e. $\lambda_1 \approx 5$ and you see that the iteration is fairly fast.

Just for curiosity, let's see what happens if you choose $\mathbf{w} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, i.e. you test the second entries:

$$\frac{6}{1} = 6 \qquad \frac{32}{6} = 5.33 \qquad \frac{164}{32} = 5.12 \qquad \frac{828}{164} = 5.04$$

$$\frac{4156}{828} = 5.02 \qquad \frac{20812}{4156} = 5.007 \qquad \frac{104124}{52126} = 5.003$$

i.e. the conclusion is the same.

To find the eigenvector, just normalize $\mathbf{u}_n$'s

$$\mathbf{v}_0 = \frac{\mathbf{u}_0}{\|\mathbf{u}_0\|} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} .707 \\ .707 \end{pmatrix} \qquad \mathbf{v}_1 = \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} = \frac{1}{\sqrt{60}} \begin{pmatrix} 4 \\ 6 \end{pmatrix} = \begin{pmatrix} .55 \\ .83 \end{pmatrix} \qquad \mathbf{v}_2 = \begin{pmatrix} .49 \\ .87 \end{pmatrix}$$

$$\mathbf{v}_3 = \begin{pmatrix} .464 \\ .885 \end{pmatrix} \qquad \mathbf{v}_4 = \begin{pmatrix} .454 \\ .89 \end{pmatrix} \qquad \mathbf{v}_5 = \begin{pmatrix} .45 \\ .893 \end{pmatrix} \qquad \mathbf{v}_6 = \begin{pmatrix} .448 \\ .8938 \end{pmatrix} \qquad \mathbf{v}_7 = \begin{pmatrix} .4476 \\ .8942 \end{pmatrix}$$

etc. so this is the (approximate) normalized eigenvector corresponding to $\lambda_1 \approx 5$. Compare it with the exact solution.

(iii) To find the smaller eigenvalue from the given information, we form the matrix

$$(A - 3)^{-1} = \begin{pmatrix} 0 & 1 \\ 2 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} -1/2 & 1/2 \\ 1 & 0 \end{pmatrix}$$

and run the same procedure as before. You can choose again $\mathbf{u}_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and test the first entry. The iteration gives

$$\mathbf{u}_1 = \begin{pmatrix} -1/2 & 1/2 \\ 1 & 0 \end{pmatrix} \mathbf{u}_0 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad \mathbf{u}_2 = \begin{pmatrix} -1/2 & 1/2 \\ 1 & 0 \end{pmatrix} \mathbf{u}_1 = \begin{pmatrix} 1/2 \\ 0 \end{pmatrix} \qquad \mathbf{u}_3 = \begin{pmatrix} -.25 \\ .5 \end{pmatrix}$$

$$\mathbf{u}_4 = \begin{pmatrix} .37 \\ -.25 \end{pmatrix} \qquad \mathbf{u}_5 = \begin{pmatrix} -.31 \\ .37 \end{pmatrix} \qquad \mathbf{u}_6 = \begin{pmatrix} .343 \\ -.312 \end{pmatrix} \qquad \mathbf{u}_7 = \begin{pmatrix} -.328 \\ .343 \end{pmatrix}$$

$$\mathbf{u}_8 = \begin{pmatrix} .335 \\ -.328 \end{pmatrix} \qquad \mathbf{u}_9 = \begin{pmatrix} -.332 \\ .335 \end{pmatrix}$$

The ratios of the first entries are

$$\frac{0}{1} = 0 \qquad \frac{1/2}{0} = \infty \qquad \frac{-.25}{1/2} = -.5 \qquad \frac{.37}{-.25} = -1.48 \qquad \frac{-.31}{.37} = -.83$$

$$\frac{.343}{-.31} = -1.1 \qquad \frac{-.328}{.343} = -.95 \qquad \frac{.335}{-.328} = -1.02 \qquad \frac{-.332}{-.335} = -.99$$

etc. The convergence is slower, but the eigenvalue is around -1. Don't worry if you got ∞ at some point: it usually disappears in the next step. But don't forget that this is the eigenvalue of $(A-3)^{-1}$, i.e. the corresponding eigenvalue of A is found from solving $(\lambda - 3)^{-1} \approx -1$, i.e. $\lambda \approx 2$.

The normalized vectors are

$$\mathbf{v}_0 = \begin{pmatrix} .707 \\ .707 \end{pmatrix} \qquad \mathbf{v}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad \mathbf{v}_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \mathbf{v}_3 = \begin{pmatrix} -.44 \\ .89 \end{pmatrix} \qquad \mathbf{v}_4 = \begin{pmatrix} .83 \\ -.55 \end{pmatrix}$$

$$\mathbf{v}_5 = \begin{pmatrix} -.64 \\ .76 \end{pmatrix} \qquad \mathbf{v}_6 = \begin{pmatrix} .74 \\ -.67 \end{pmatrix} \qquad \mathbf{v}_7 = \begin{pmatrix} -.69 \\ .723 \end{pmatrix} \qquad \mathbf{v}_8 = \begin{pmatrix} .715 \\ -.698 \end{pmatrix} \qquad \mathbf{v}_9 = \begin{pmatrix} -.702 \\ .711 \end{pmatrix}$$

etc. Notice that literally the vectors do not converge to anything because of a sign oscillation. This is due to the fact that we did not fix the normalization convention: even a normalized eigenvector has an ambiguity; it can be multiplied by -1. To remove this ambiguity we can require, for example, that we choose such normalization so that the first entry be always positive. In that case the approximate eigenvectors are

$$\begin{pmatrix} .707 \\ .707 \end{pmatrix} \qquad \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \begin{pmatrix} .44 \\ -.89 \end{pmatrix} \qquad \begin{pmatrix} .83 \\ -.55 \end{pmatrix} \qquad \begin{pmatrix} .64 \\ -.76 \end{pmatrix} \qquad \begin{pmatrix} .74 \\ -.67 \end{pmatrix}$$

$$\begin{pmatrix} .69 \\ -.723 \end{pmatrix} \qquad \begin{pmatrix} .715 \\ -.698 \end{pmatrix} \qquad \begin{pmatrix} .702 \\ -.711 \end{pmatrix}$$

and they clearly converge to the true eigenvector.

5.2 Jacobi method for eigenvalues

This method finds the spectral decomposition $A = VDV^t$ of a symmetric matrix. In one single algorithm, it finds the eigenvalues and eigenvectors. The algorithm is stable, but quite slow.

We have to recall the Givens rotations from Section 3.4. Remember that multiplying by an appropriately chosen rotation $R(i, j, \theta)$ was able to zero off the a_{ij} element of A. However, to reach the $A = VDV^t$ form, one would like to do everything "symmetrically". I.e. we are looking for a Givens rotation $G = R(i, j, \theta)$ such that the (i, j)-th entry of the matrix G^tAG be zero. Then, by symmetry, the (j, i)-th entry will be zero as well. Hence we have

$$B = G^tAG = \begin{pmatrix} & \vdots & & \vdots & \\ \dots & b_{ii} & \dots & b_{ij} & \dots \\ & \vdots & & \vdots & \\ \dots & b_{ji} & \dots & b_{jj} & \dots \\ & \vdots & & \vdots & \end{pmatrix}$$

$$= \begin{pmatrix} 1 & & & & \\ & c & & s & \\ & & 1 & & \\ & & & 1 & \\ & -s & & c & \\ & & & & 1 \end{pmatrix} \begin{pmatrix} & \vdots & & \vdots & \\ \dots & a_{ii} & \dots & a_{ij} & \dots \\ & \vdots & & \vdots & \\ \dots & a_{ji} & \dots & a_{jj} & \dots \\ & \vdots & & \vdots & \end{pmatrix} \begin{pmatrix} 1 & & & & \\ & c & & -s & \\ & & 1 & & \\ & & & 1 & \\ & s & & c & \\ & & & & 1 \end{pmatrix}$$

and we want to make sure that $b_{ij} = b_{ji} = 0$. One can easily express $b_{ij} = b_{ji}$ in terms of the elements of A and c, s. After some computation (DO IT FOR YOURSELF(*)) you arrive at

$$s = \left(\frac{1}{2} - \frac{\beta}{2\sqrt{1+\beta^2}} \right)^{1/2}$$

$$c = \left(\frac{1}{2} + \frac{\beta}{2\sqrt{1+\beta^2}} \right)^{1/2} \tag{5.1}$$

with

$$\beta := \frac{a_{ii} - a_{jj}}{2a_{ij}}$$

Thus by proper choice of G we can introduce a zero into any specified offdiagonal position, while preserving the eigenvalues and the symmetry of A (CHECK(*) that A and $G^t AG$ have the same eigenvalues!)

[REMARK: The idea is very similar to the QR-factorization with Givens matrices in Section 3.5, but the actual Givens matrix is not the same. In Section 3.5 the goal was to kill one element below the diagonal just by a single Givens multiplication from the left. This requirement set the choice of the elements in the Givens matrix. In the Jacobi algorithm we choose the Givens matrix such that after sandwiching the original matrix between the Givens matrix and its transpose, the result has a zero at the required position. This is why the formulas in (5.1) are different from (3.2).]

It would be nice if we could zero off all the off-diagonal elements of A in succession and after $n(n-1)/2$ transformations have a diagonal matrix which is conjugated to A.

But there must be a catch, because it is not possible to devise an algorithm which finds the eigenvalues of a matrix exactly in a finite number of steps. The catch is that when we zero an element of A, a previously zeroed element may become nonzero again. Every time we knock one off-diagonal element out, others pop back up, so it might seem the algorithm is useless. Fortunately, although the transformed matrices never become exactly diagonal, they do make a steady progress toward that goal. This is the content of the following theorem (the proof is easy calculation)

Theorem 5.4 *Let $B = G^tAG$ be a Jacobi iteration. Then*

$$\sum_{i=1}^{n} b_{ii}^2 = 2a_{ij}^2 + \sum_{i=1}^{n} a_{ii}^2$$

i.e. the sum of the squares of the diagonal elements increases by $2a_{ij}^2$.

Moreover, the sum of squares of all elements in A and B coincide. Hence the sum of squares of the offdiagonal elements of B decreases by $2a_{ij}^2$.

This theorem says that measured the entries in an *average sense* the transformed matrix B is closer to a diagonal matrix than A. Moreover, larger a_{ij}^2 means faster progress.

Now the Jacobi iteration is straighforward. Consider the offdiagonal element a_{ij} largest in modulus. Construct the suitable G matrix given by the formulas (5.1) above. Compute $A_1 = G^tAG$. Continue the process with the new A_1 matrix; consider its largest offdiagonal element, construct suitable Givens matrices etc. I.e. we have the iteration

$$A_0 = A$$

$$A_{n+1} = G_n^t A_n G_n$$

where G_n is the Givens matrix zeroing the biggest element of A_n. Then

$$\lim_{n\to\infty} A_n \to D$$

a diagonal matrix. Of course we never reach it exactly, but we can run the iteration for N steps, until it is acceptably nearly diagonal. Then we have

$$A \approx QDQ^t$$

with

$$Q = G_1 G_2 \dots G_N$$

This gives the spectral decompostion of the matrix A.

It is again important to emphasize that we manipulated with orthogonal matrices, which are stable, hence errors do not amplify. This method is very elegant and powerful, the only problem is that it is a bit slow and it works only for symmetric matrices.

However, we point out that the singular value decomposition for a general $n \times k$ matrix A actually was equivalent to diagonalizing the symmetric matrices AA^t and A^tA. Hence this algorithm is crucial even in the numerical analysis of general matrices.

5.3 QR iteration for eigenvalues

Finally we discuss the QR algorithm which is the one most frequently used for calculation of the set of eigenvalues of a *general* matrix. However it does not compute eigenvectors unless the matrix is symmetric.

The algorithm is very simple. Given a square matrix $A_0 = A$, we compute its QR factorization (we have learned effective algorithms for that in Section 3). Then we realize that

$$RQ = Q^t(QR)Q = Q^tAQ$$

matrix is actually conjugated to the original matrix A, hence RQ has the same set of eigenvalues as A. So we compute $A_1 = RQ$ and run the same steps again: compute the QR factorization of $A_1 = Q_1R_1$ (of course $Q_1 \neq Q$ and $R_1 \neq R$ since Q, R do not commute), then compute $A_2 = R_1Q_1$ and repeat.

In the general n-th step, we consider the QR-factorization

$$A_n = Q_nR_n$$

of A_n then we let

$$A_{n+1} = R_nQ_n$$

It is clear that the set of eigenvalues of all the A_n matrices are the same.

It is not true that A_n converges to a diagonal matrix, in fact the upper off-diagonal matrix elements may not converge at all. However, the sequence becomes more and more upper triangular, i.e. the lower off-diagonal elements go to zero, and more importantly, its diagonal elements converge to the eigenvalues. The precise theorem goes as follows, which we state without proof:

Theorem 5.5 *Suppose that the real valued square matrix A is invertible and all eigenvalues $\lambda_1, \lambda_2, \ldots$ are distinct in modulus. Then*

$$\lim_{n\to\infty} (A_n)_{ii} = \lambda_i$$

$$\lim_{n\to\infty} (A_n)_{ij} = 0 \qquad i > j$$

If you start with a symmetric matrix A, then clearly the sequence A_n consists of symmetric matrices as well (WHY???). From the theorem we know that A_n is (almost) upper triangular, but A is also symmetric, hence it is (almost) diagonal. We obtain

$$A_n = Q_n^t Q_{n-1}^t \ldots Q_1^t A Q_1 Q_2 \ldots Q_n = Q^t A Q$$

with the choice $Q = Q_1 Q_2 \ldots Q_n$, i.e. $A = Q A_n Q^t$ is almost the spectral decomposition of A since A_n is almost diagonal.

REMARK: Notice that the QR-iteration uses only real numbers; every step of the procedure you never see any genuine complex number. Hence it is unable to find complex eigenvalues. This is not in contradiction to the theorem, since for a real matrix the complex eigenvalues come in conjugate pairs and conjugate pairs have the same modulus. (If λ_1 is a solution to $p(\lambda) = 0$ characteristic equation, then so is $\lambda_2 = \overline{\lambda_1}$, its complex conjugate. To

see this, simply take the complex conjugate of the characteristic equation and use that all coefficients are real).

There are improvements of the QR-method which finds complex eigenvalues as well, but we do not discuss them here.

As a closing remark, let us emphasize again, that we just scratched the surface of an enormously large and well-developed field of numerical mathematics. We just presented the simplest versions of a few simple algorithm. They all have several theoretical improvements. In addition, there are several computer implementations of the same algorithm, designed for optimizing number of steps, necessary memory etc. And in addition, there are special algorithms for special but important class of matrices (mainly sparse matrices). The moral of the story is, however, that the theoretically most convenient algorithms might fail terribly in applications. Special care is needed to deal with stability issues and ill-conditioned problems. In the numerical analysis of the problems from linear algebra, the most important tool is the orthogonal transformations, as they are stable. Hence the conclusion is: Use orthogonal matrices, bases, transformations etc. whenever possible!!