Linear algebra & Numerical Analysis

Eigenvalues and Eigenvectors

Marta Jarošová
http://homel.vsb.cz/~dom033/
Outline

- Methods computing all eigenvalues
  - Characteristic polynomial
  - Jacobi method for symmetric matrices
  - QR iteration
    - Shifts
    - preliminary reduction

- Methods computing selected eigenvalues
  - Power method
    - normalization
    - Shifts
    - Rayleigh quotient

- Matlab functions
Methods computing all eigenvalues

- Characteristic polynomial
- Jacobi method for symmetric matrices
- QR iteration
  - Shifts
  - preliminary reduction
Characteristic polynomial

- $\det(A - \lambda I) = 0$
- This method is not recommended as a general numerical procedure
  - The coefficients are not well-determined numerically – the roots can be sensitive to perturbations in the coefficients
  - Root finding of a polynomial of high degree requires great deal of work
    - the problems are equivalent in theory, but in practice the solution is not preserved numerically;
    - and **computing the roots of the polynomial is no simpler than the original eigenvalue problem**
Important theoretical observations:

- Abel (1824):
  the roots of a polynomial of degree greater than 4 cannot always be expressed by a closed-form formula in the coefficients using ordinary arithmetic operations and root extractors.

- \( \Rightarrow \) computing the eigenvalues of matrices of the order greater than 4 requires a (theoretically infinite) iterative process.
Jacobi method for symmetric matrices

- $A_0 = A$
- Iteration: $A_{k+1} = J_k^T A_k J_k$
- a plane rotation $J_k$ is chosen to annihilate a symmetric pair of entries in the matrix $A_k$ (symmetry is preserved)

- **Recall**: plane rotation
  \[
  \begin{bmatrix}
  c & s \\
  -s & c
  \end{bmatrix}
  \]
- With $c$ and $s$ the cosine and sine of the angle of rotation
Jacobi method for symmetric matrices

Choice of \( c \) and \( s \) slightly more complicated than in Givens QR method – we are annihilating a symmetric pair of matrix entries by a similarity transformations (Givens QR: single entry by a one-sided transformation)

\[
J^T AJ = \begin{bmatrix}
c & -s \\
s & c
\end{bmatrix}
\begin{bmatrix}
a & b \\
b & d
\end{bmatrix}
\begin{bmatrix}
c & s \\
-s & c
\end{bmatrix}
\]

\[
= \begin{bmatrix}
c^2a - 2csb + s^2d & c^2b + cs(a - d) - s^2b \\
c^2b + cs(a - d) - s^2b & c^2d + 2csb + s^2a
\end{bmatrix}
\]

with \( b \neq 0 \) (else diagonal). The transformed matrix is diagonal if

\[
c^2b + cs(a - d) - s^2b = 0.
\]

\[
c = \frac{1}{\sqrt{1 + t^2}} \quad \text{and} \quad s = c \cdot t
\]
Dividing both sides of this equation by $c^2 b$, we obtain

$$1 + \frac{s}{c} \frac{(a - d)}{b} - \frac{s^2}{c^2} = 0.$$ 

Making the substitution $t = s/c$, we obtain a quadratic equation

$$1 + t \frac{(a - d)}{b} - t^2 = 0$$

for $t$, the tangent of the angle of rotation, from which we can recover $c = 1/\sqrt{1 + t^2}$ and $s = c \cdot t$. It is advantageous numerically to use the root of smaller magnitude of the equation for $t$. 
Example 4.6 Plane Rotation. To illustrate the use of a plane rotation to annihilate a symmetric pair of off-diagonal entries, we consider the $2 \times 2$ matrix

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}.$$  

The quadratic equation for the tangent reduces to $t^2 = 1$ in this case, so we have $t = \pm 1$. Since the two roots are of the same magnitude, we arbitrarily choose $t = -1$, which yields $c = 1/\sqrt{2}$ and $s = -1/\sqrt{2}$. Using the resulting plane rotation $J$, we then have

$$J^T AJ = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & -1 \end{bmatrix}. $$
Jacobi method for symmetric matrices

- Plate rotations are repeatedly applied from both sides of the matrix until the off-diagonal entries are reduced to zero (with some tolerance)
- Resulting approximately diagonal matrix is orthogonally similar to the original matrix
- \( \Rightarrow \) we have the approximation of eigenvalues on the diagonal and the product of all plane rotations gives the eigenvectors
Jacobi method for symmetric matrices

• **PROS:**
  - easy to implement (also in parallel)
  - high accuracy

• **CONS:**
  - slow convergence
  - only for symmetric matrices

Main source of inefficiency: annihilated elements can become nonzero again
Jacobi method
for symmetric matrices

- **Example:**
  
  $A_0 = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 2 & 1 \\ 2 & 1 & 1 \end{bmatrix}$

- Annihilate entries (1,3) and (3,1) using plane rotation

  
  $J_0 = \begin{bmatrix} 0.707 & 0 & -0.707 \\ 0 & 1 & 0 \\ 0.707 & 0 & 0.707 \end{bmatrix}$

  to obtain $A_1 = J_0^T A_0 J_0 = \begin{bmatrix} 3 & 0.707 & 0 \\ 0.707 & 2 & 0.707 \\ 0 & 0.707 & -1 \end{bmatrix}$

- Next annihilate entries (1,2) and (2,1) using plane rotation

  
  $J_1 = \begin{bmatrix} 0.888 & -0.460 & 0 \\ 0.460 & 0.888 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

  to obtain $A_2 = J_1^T A_1 J_1 = \begin{bmatrix} 3.366 & 0 & 0.325 \\ 0 & 1.634 & 0.628 \\ 0.325 & 0.628 & -1 \end{bmatrix}$
Jacobi method for symmetric matrices

(example)

- Next annihilate entries (2,3) and (3,2) using plane rotation

\[
J_2 = \begin{bmatrix}
1 & 0 & 0 \\
0 & 0.975 & -0.221 \\
0 & 0.221 & 0.975
\end{bmatrix}
\]

\[
to\ obtain\quad A_3 = J_2^T A_2 J_2 = \begin{bmatrix}
3.366 & 0.072 & 0.317 \\
0.072 & 1.776 & 0 \\
0.317 & 0 & -1.142
\end{bmatrix}
\]

- Beginning a new sweep, we again annihilate entries (1, 3) and (3,1)

\[
J_3 = \begin{bmatrix}
0.998 & 0 & -0.070 \\
0 & 1 & 0 \\
0.070 & 0 & 0.998
\end{bmatrix}
\]

\[
to\ obtain\quad A_4 = J_3^T A_3 J_3 = \begin{bmatrix}
3.388 & 0.072 & 0 \\
0.072 & 1.776 & 0.005 \\
0 & 0.005 & -1.164
\end{bmatrix}
\]
QR iteration

- makes repeated use of the QR factorization to produce an orthogonal similarity transformation of the matrix to diagonal or triangular form

\[
A_0 = A \quad \text{initialization}
\]
\[
A_k = Q_k R_k \quad \text{QR factorization in the k-th step}
\]
\[
A_{k+1} = R_k Q_k \quad \text{reverse product}
\]

- Since \( R_k Q_k = Q_k^T A_k Q_k \)

\( A_k \) are orthogonally similar to each other
QR iteration

- For all distinct eigenvalues $A_k$ converge to
  - **triangular** form for general initial matrix, or
  - **diagonal** form for a symmetric initial matrix

Example 4.8  QR Iteration. Let

$$A_0 = \begin{bmatrix} 7 & 2 \\ 2 & 4 \end{bmatrix}.$$  

We first compute the QR factorization $A_0 = Q_0 R_0$, obtaining

$$Q_0 = \begin{bmatrix} 0.962 & -0.275 \\ 0.275 & 0.962 \end{bmatrix} \quad \text{and} \quad R_0 = \begin{bmatrix} 7.28 & 3.02 \\ 0 & 3.30 \end{bmatrix}.$$
QR iteration

- We next form the reverse product
  \[ A_1 = R_0 Q_0 = \begin{bmatrix} 7.83 & 0.906 \\ 0.906 & 3.17 \end{bmatrix}. \]

- off-diagonal entries are smaller,
- so the matrix is closer to being triangular
- and the diagonal entries are now closer to the eigenvalues, which are 8 and 3.
- repeat this process until the convergence
- the diagonal entries would then closely approximate the eigenvalues, the product of the orthogonal matrices \( Q_k \) would yield the corresponding eigenvectors
QR iteration with shifts

• The convergence rate of QR iteration can be accelerated by introducing shifts:

\[ A_k - \sigma_k I = Q_k R_k, \]
\[ A_{k+1} = R_k Q_k + \sigma_k I, \]

where \( \sigma_k \) is a rough approximation to an eigenvalue.

• Choice of shift:
  • lower right corner entry
  • better shift: computing eigenvalues of the 2x2 submatrix in the lower right corner of the matrix
QR iteration with shifts

- **Example:**

\[
A_0 = \begin{bmatrix} 7 & 2 \\ 2 & 4 \end{bmatrix} \quad \text{shift} \quad \sigma_0 = 4
\]

Thus, we first compute the QR factorization \( A_0 - \sigma_0 I = Q_0 R_0 \) so

\[
Q_0 = \begin{bmatrix} 0.832 & 0.555 \\ 0.555 & -0.832 \end{bmatrix} \quad \text{and} \quad R_0 = \begin{bmatrix} 3.61 & 1.66 \\ 0 & 1.11 \end{bmatrix}.
\]

We next form the reverse product and add back the shift to obtain

\[
A_1 = R_0 Q_0 + \sigma_0 I = \begin{bmatrix} 7.92 & 0.615 \\ 0.615 & 3.08 \end{bmatrix}.
\]

- Compared with the unshifted algorithm, the off-diagonal entries are smaller after one iteration,
- and the diagonal entries are closer approximations to the eigenvalues.
- For the next iteration, we would use the new value of the lower right corner entry as the shift.
Each iteration of the QR method requires $O(n^3)$ operations.

This can be reduced if the matrix is initially transformed into a simpler form.

It is advantageous if the matrix is as close as possible to triangular or diagonal for symmetric matrix before the QR iterations begin.
Preliminary reduction

- **Hessenberg matrix** is triangular except one additional nonzero diagonal immediately adjacent to the main diagonal
- **Note:** a symmetric Hessenberg matrix is tridiagonal
Preliminary reduction

• Any matrix can be reduced to Hessenberg form in a **finite number of steps** by an orthogonal similarity transformations (e.g. using Householder rotations)

• Upper Hessenberg or tridiagonal form can then be preserved during the subsequent QR iterations

• Advantages:
  • Work per QR iteration is reduced to at most $O(n^2)$
  • The convergence rate of the QR iterations is enhanced
  • If there are any zero entries on the first subdiagonal, then the problem can be broken into two or more smaller subproblems
Two-stage process of QR method:
- symmetric $\rightarrow$ tridiagonal $\rightarrow$ diagonal
- general $\rightarrow$ Hessenberg $\rightarrow$ triangular

Preliminary reduction requires finite number of steps, whereas the subsequent iterative stage continues until convergence (in practice, only modest number of iterations is usually required)

$\Rightarrow O(n^3)$ cost of preliminary reduction is significant factor in total
QR iteration with preliminary reduction

**Total cost:** depends on if the eigenvectors are needed because their inclusion determines whether the orthogonal transformations must be accumulated

- **symmetric case:**
  - overall cost is roughly $4/3 \ n^3$ operations if only eigenvectors are needed
  - and about $9 \ n^3$ operations if the eigenvectors are also desired

- **general case:**
  - $9 \ n^3$ (only eigenvalues)
  - and about $25 \ n^3$ (also eigenvectors)
Methods computing selected eigenvalues

- Power method
  - normalization
  - Shifts
  - Rayleigh quotient
Power method

- The simplest method for computing a single eigenvalue and eigenvector
- Takes sufficiently higher powers of the matrix times an initial starting vector
- Assume that the matrix has a unique eigenvalue $\lambda_1$ of maximum modulus (dominate eigenvalue) with corresponding eigenvector $u_1$

- For given $x_0$, iterations converge to $u_1$. 
  \[ x_k = Ax_{k-1} \]
Power method

To see why, we first express the starting vector \( x_0 \) as a linear combination, \( x_0 = \sum_{i=1}^{n} \alpha_i u_i \), where the \( u_i \) are eigenvectors of \( A \). We then have

\[
x_k = A x_{k-1} = A^2 x_{k-2} = \cdots = A^k x_0
\]

\[
= A^k \sum_{i=1}^{n} \alpha_i u_i = \sum_{i=1}^{n} \alpha_i A^k u_i = \sum_{i=1}^{n} \lambda_i^k \alpha_i u_i
\]

\[
= \lambda_1^k (\alpha_1 u_1 + \sum_{i=2}^{n} (\lambda_i/\lambda_1)^k \alpha_i u_i).
\]

Since \( |\lambda_i/\lambda_1| < 1 \) for \( i > 1 \), successively higher powers go to zero, leaving only the component corresponding to \( u_1 \).
Example 4.10 Power Method. In the sequence of vectors produced by the power method, the ratio of the values of a given component of $x_k$ from one iteration to the next converges to the dominant eigenvalue $\lambda_1$. For example, if

$$A = \begin{bmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{bmatrix} \quad \text{and} \quad x_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

then we obtain the following sequence.

<table>
<thead>
<tr>
<th>$k$</th>
<th>$x_k^T$</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>1.5</td>
</tr>
<tr>
<td>2</td>
<td>1.5</td>
<td>2.5</td>
</tr>
<tr>
<td>3</td>
<td>3.5</td>
<td>4.5</td>
</tr>
<tr>
<td>4</td>
<td>7.5</td>
<td>8.5</td>
</tr>
<tr>
<td>5</td>
<td>15.5</td>
<td>16.5</td>
</tr>
<tr>
<td>6</td>
<td>31.5</td>
<td>32.5</td>
</tr>
<tr>
<td>7</td>
<td>63.5</td>
<td>64.5</td>
</tr>
<tr>
<td>8</td>
<td>127.5</td>
<td>128.5</td>
</tr>
</tbody>
</table>

The sequence of vectors $x_k$ is converging to a multiple of the eigenvector $[1 \ 1]^T$, and the ratio of successive iterates for each component is converging to the corresponding eigenvalue, 2, which we saw in Example 4.1 is indeed the largest eigenvalue of this matrix.
Power method with normalization

- normalization

\[ y_k = A x_{k-1}, \]
\[ x_k = y_k / \| y_k \|_\infty \]

\[ \| y_k \|_\infty \rightarrow |\lambda_1|, \text{ and } x_k \rightarrow u_1 / \| u_1 \|_\infty \]
Example 4.11 Power Method with Normalization. Repeating the previous example with this normalized scheme, we get the following sequence:

<table>
<thead>
<tr>
<th>$k$</th>
<th>$x_k^T$</th>
<th>$|y_k|_\infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.000</td>
<td>1.0</td>
</tr>
<tr>
<td>1</td>
<td>0.333</td>
<td>1.500</td>
</tr>
<tr>
<td>2</td>
<td>0.600</td>
<td>1.667</td>
</tr>
<tr>
<td>3</td>
<td>0.778</td>
<td>1.800</td>
</tr>
<tr>
<td>4</td>
<td>0.882</td>
<td>1.889</td>
</tr>
<tr>
<td>5</td>
<td>0.939</td>
<td>1.941</td>
</tr>
<tr>
<td>6</td>
<td>0.969</td>
<td>1.970</td>
</tr>
<tr>
<td>7</td>
<td>0.984</td>
<td>1.985</td>
</tr>
<tr>
<td>8</td>
<td>0.992</td>
<td>1.992</td>
</tr>
</tbody>
</table>

The eigenvalue estimates have not changed, but now the approximate eigenvector is normalized at each iteration, thereby avoiding geometric growth or shrinkage of its components.
**Convergence rate**: depends on the ratio $|\lambda_2/\lambda_1|$
the smaller this ratio, the faster the convergence

It may be possible to choose a shift, $A - \sigma I$, such that

$$\left|\frac{\lambda_2 - \sigma}{\lambda_1 - \sigma}\right| < \left|\frac{\lambda_2}{\lambda_1}\right|$$

and thus convergence is accelerated

the shift must then be added to the result to obtain the eigenvalue of the original matrix
Rayleigh quotient

- given approximate eigenvector $\mathbf{x}$ for a real matrix $A$,
- the best estimate for the corresponding eigenvalue can be considered as an $n \times 1$ linear least squares approximation problem

$$x\lambda \approx Ax$$

- with normal equation $x^T x\lambda = x^T Ax$
- the least squares solution is given by

$$\lambda = \frac{x^T Ax}{x^T x}$$
Example 4.13 Rayleigh Quotient. For Example 4.11 using the power method, the value of the Rayleigh quotient at each iteration is shown next.

<table>
<thead>
<tr>
<th>( k )</th>
<th>( x_k^T )</th>
<th>( | y_k |_{\infty} )</th>
<th>( x_k^T A y_k / x_k^T x_k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.000</td>
<td>1.0</td>
<td>1.500</td>
</tr>
<tr>
<td>1</td>
<td>0.333</td>
<td>1.0</td>
<td>1.500</td>
</tr>
<tr>
<td>2</td>
<td>0.600</td>
<td>1.0</td>
<td>1.667</td>
</tr>
<tr>
<td>3</td>
<td>0.778</td>
<td>1.0</td>
<td>1.800</td>
</tr>
<tr>
<td>4</td>
<td>0.882</td>
<td>1.0</td>
<td>1.889</td>
</tr>
<tr>
<td>5</td>
<td>0.939</td>
<td>1.0</td>
<td>1.941</td>
</tr>
<tr>
<td>6</td>
<td>0.969</td>
<td>1.0</td>
<td>1.970</td>
</tr>
</tbody>
</table>

Thus, the Rayleigh quotient converges to the dominant eigenvalue, 2, faster than the successive approximations produced by the power method alone.
Matlab functions

- \( \mathbf{E} = \text{eig}(\mathbf{X}) \) is a vector containing the eigenvalues of a square matrix \( \mathbf{X} \).

- \([\mathbf{V}, \mathbf{D}] = \text{eig}(\mathbf{X})\) produces a diagonal matrix \( \mathbf{D} \) of eigenvalues and a full matrix \( \mathbf{V} \) whose columns are the corresponding eigenvectors so that \( \mathbf{X} \mathbf{V} = \mathbf{V} \mathbf{D} \).

- \( \mathbf{D} = \text{eigs} (\mathbf{A}) \) returns a vector of \( \mathbf{A}'s 6 \) largest magnitude eigenvalues. \( \mathbf{A} \) must be square and should be large and sparse.

- \([\mathbf{V}, \mathbf{D}] = \text{eigs}(\mathbf{A})\) returns a diagonal matrix \( \mathbf{D} \) of \( \mathbf{A}'s 6 \) largest magnitude eigenvalues and a matrix \( \mathbf{V} \) whose columns are the corresponding eigenvectors.
References


- Lloyd Nicholas Trefethen, David Bau, Numerical linear algebra (available on Google books)