Computing Eigenvalues and/or Eigenvectors;Part 2, The Power method and QR-algorithm

Tom Lyche

Centre of Mathematics for Applications,
Department of Informatics,
University of Oslo

November 16, 2008

Today

- The power method to find the dominant eigenvector
- The shifted power method to speed up convergence
- The inverse power method
- The Rayleigh quotient iteration
- The QR-algorithm

The Power Method

- Find the eigenvector corresponding to the dominant (largest in absolute value) eigenvalue.
- With a simple modification we can also find the corresponding eigenvalue

Assumptions

- Let $\mathbf{A} \in \mathbb{C}^{n,n}$ have eigenpairs $(\lambda_j, \mathbf{v}_j)$, $j = 1, \ldots, n$.
- Given $\mathbf{z}_0 \in \mathbb{C}^n$ we assume that

$$\begin{aligned} (i) &\quad |\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|, \\ (ii) &\quad \mathbf{A} \text{ has linearly independent eigenvectors }, \\ (iii) &\quad \mathbf{z}_0 = c_1\mathbf{v}_1 + \cdots + c_n\mathbf{v}_n \text{ with } c_1 \neq 0. \end{aligned}$$

- The first assumption means that $\mathbf{A}$ has a dominant eigenvalue λ_1 of algebraic multiplicity one.
- The second assumption is not necessary. It is included to simplify the analysis.
- The third assumption says that $\mathbf{z}_0^T\mathbf{v}_1 \neq 0$, i. e., $\mathbf{z}_0$ has a component in the direction $\mathbf{v}_1$.

Powers

- Given $\mathbf{A} \in \mathbb{C}^{n,n}$, a vector $\mathbf{z}_0 \in \mathbb{C}^n$, and assume that i),ii), iii) hold.
- Define a sequence $\{\mathbf{z}_k\}$ of vectors in $\mathbb{C}^n$ by $\mathbf{z}_k := \mathbf{A}^k \mathbf{z}_0 = \mathbf{A}\mathbf{z}_{k-1}, \quad k = 1, 2, \ldots.$
- Assume $\mathbf{z}_0 = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_n\mathbf{v}_n$.
- Then $\mathbf{z}_k = c_1\lambda_1^k\mathbf{v}_1 + c_2\lambda_2^k\mathbf{v}_2 + \cdots + c_n\lambda_n^k\mathbf{v}_n, \quad k = 0, 1, 2, \ldots.$
- $\frac{\mathbf{z}_k}{\lambda_1^k} = c_1\mathbf{v}_1 + c_2\left(\frac{\lambda_2}{\lambda_1}\right)^k\mathbf{v}_2 + \cdots + c_n\left(\frac{\lambda_n}{\lambda_1}\right)^k.$
- $\mathbf{z}_k/\lambda_1^k, \rightarrow c_1\mathbf{v}_1, \ k \rightarrow \infty$

The Power method

Need to normalize the vectors $\mathbf{z}_k$.

- Choose a norm on $\mathbb{C}^n$, set $\mathbf{x}_0 = \mathbf{z}_0/\|\mathbf{z}_0\|$ and generate for $k = 1, 2, \ldots$ unit vectors ,$\{\mathbf{x}_k\}$ as follows:

$$\begin{aligned} (i) &\quad \mathbf{y}_k = \mathbf{A}\mathbf{x}_{k-1} \\ (ii) &\quad \mathbf{x}_k = \mathbf{y}_k/\|\mathbf{y}_k\|. \end{aligned} \tag{1}$$

Example

- $$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad \mathbf{z}_0 = [1.0, 1.0]^T, \quad \mathbf{x}_0 = [0.707, 0.707]^T$$
- $\mathbf{x}_1 = [0.39, 0.92]$, $\mathbf{x}_2 = [0.4175, 0.9087]$, $\mathbf{x}_3 = [0.4159, 0.9094], \ldots$
- converges to an eigenvector of $\mathbf{A}$

Convergence

- The way $\{\mathbf{x}_k\}$ converges to an eigenvector will depend on the sign of λ_1. Suppose $\mathbf{x}_k = K\mathbf{v}_1$ for some $K \neq 0$.
- Then $\mathbf{A}\mathbf{x}_k = \lambda_1 \mathbf{x}_k$ and $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k / \|\mathbf{A}\mathbf{x}_k\| = \frac{\lambda_1}{|\lambda_1|}\mathbf{x}_k$.
- Thus $\mathbf{x}_{k+1} = -\mathbf{x}_k$ if $\lambda_1 < 0$.

eigenvalue

- Suppose we know an approximate eigenvector of $\mathbf{A} \in \mathbb{C}^{n,n}$. How should we estimate the corresponding eigenvalue?
- If $(\lambda, \mathbf{u})$ is an exact eigenpair then $\mathbf{A}\mathbf{u} - \lambda\mathbf{u} = \mathbf{0}$.
- If $\mathbf{u}$ is an approximate eigenvector we can minimize the function $\rho : \mathbb{C} \to \mathbb{R}$ given by

$$\rho(\mu) := \|\mathbf{A}\mathbf{u} - \mu\mathbf{u}\|_2.$$

Theorem

ρ is minimized when $\mu = \nu := \frac{\mathbf{u}^H\mathbf{A}\mathbf{u}}{\mathbf{u}^H\mathbf{u}}$ is the Rayleigh quotient of $\mathbf{u}$.

Power with Rayleigh

```
function [l,x,it]=powerit(A,z,K,tol)
af=norm(A,'fro'); x=z/norm(z);
for k=1:K
    y=A*x; l=x'*y;
    if norm(y-l*x)/af < tol
      it=k; x=y/norm(y); return
    end
    x=y/norm(y);
end
it=K+1;
```

The shifted power method

- A variant of the power method is the shifted power method.
- In this method we choose a number s and apply the power method to the matrix $\mathbf{A} - s\mathbf{I}$.
- The number s is called a shift since it shifts an eigenvalue λ of $\mathbf{A}$ to $\lambda - s$ of $\mathbf{A} - s\mathbf{I}$.
- Sometimes the convergence can be faster if the shift is chosen intelligently.

Example

- $$\mathbf{A}_1 := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad \mathbf{A}_2 := \begin{bmatrix} 1.7 & -0.4 \\ 0.15 & 2.2 \end{bmatrix}, \text{ and } \mathbf{A}_3 = \begin{bmatrix} 1 & 2 \\ -3 & 4 \end{bmatrix}.$$
- Start with a random vector and tol=10^{-6}.
- Get convergence in 7 iterations for $\mathbf{A}_1$, 174 iterations for $\mathbf{A}_2$ and no convergence for $\mathbf{A}_3$.
- $\mathbf{A}_3$ has two complex eigenvalues so assumption i) is not satisfied
- Rate of convergence depends on $r = |\lambda_2/\lambda_1|$. Faster convergence for smaller r.
- We have $r \approx 0.07$ for $\mathbf{A}_1$ and $r = 0.95$ for $\mathbf{A}_2$.

The inverse power method

- We apply the power method to the inverse matrix $(\mathbf{A} - s\mathbf{I})^{-1}$, where s is a shift.
- If $\mathbf{A}$ has eigenvalues $\lambda_1, \dots, \lambda_n$ in no particular order then $(\mathbf{A} - s\mathbf{I})^{-1}$ has eigenvalues

$$\mu_1(s) = (\lambda_1 - s)^{-1}, \mu_2(s) = (\lambda_2 - s)^{-1}, \dots, \mu_n(s) = (\lambda_n - s)^{-1}.$$

- Suppose λ_1 is a simple eigenvalue of $\mathbf{A}$.
- Then $\lim_{s\to\lambda_1} |\mu_1(s)| = \infty$, while $\lim_{s\to\lambda_1} \mu_j(s) = (\lambda_j - \lambda_1)^{-1} < \infty$ for $j = 2, \dots, n$.
- Hence, by choosing s sufficiently close to λ_1 the inverse power method will converge to that eigenvalue.

For the inverse power method (1) is replaced by.

$$\begin{array}{ll} (i) & (\mathbf{A} - s\mathbf{I})\mathbf{y}_k = \mathbf{x}_{k-1} \\ (ii) & \mathbf{x}_k = \mathbf{y}_k / \|\mathbf{y}_k\|. \end{array} \tag{2}$$

Note that we solve the linear system rather than computing the inverse matrix. Normally the PLU-factorization of $\mathbf{A} - s\mathbf{I}$ is pre-computed in order to speed up the iteration.

Rayleigh quotient iteration

We can combine inverse power with Rayleigh quotient calculation.

$$
\begin{aligned}
(i) \quad & (\mathbf{A} - s_{k-1}\mathbf{I})\mathbf{y}_k = \mathbf{x}_{k-1}, \\
(ii) \quad & \mathbf{x}_k = \mathbf{y}_k / \|\mathbf{y}_k\|, \\
(iii) \quad & \mathbf{s}_k = \mathbf{x}_k^H \mathbf{A}\mathbf{x}_k, \\
(iv) \quad & \mathbf{r}_k = \mathbf{A}\mathbf{x}_k - s_k \mathbf{x}_k.
\end{aligned}
$$

- We can avoid the calculation of $\mathbf{A}\mathbf{x}_k$ in (iii) and (iv).

Example

- $\mathbf{A}_1 := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$.
- $\lambda = (5 - \sqrt{33})/2 \approx -0.37$
- Start with $\mathbf{x} = [1, 1]^T$

k	1	2	3	4	5
$\|\mathbf{r}\|_2$	1.0e+000	7.7e-002	1.6e-004	8.2e-010	2.0e-020
$\|s_k - \lambda\|$	3.7e-001	-1.2e-002	-2.9e-005	-1.4e-010	-2.2e-016

Table: Quadratic convergence of Rayleigh quotient iteration

Problem with singularity?

- The linear system in *i*) becomes closer and closer to singular as s_k converges to the eigenvalue.
- Thus the system becomes more and more ill-conditioned and we can expect large errors in the computed $\mathbf{y}_k$.
- This is indeed true, but we are lucky.
- Most of the error occurs in the direction of the eigenvector and this error disappears when we normalize $\mathbf{y}_k$ in *ii*).
- Miraculously, the normalized eigenvector will be quite accurate.

Discussion

- Since the shift changes from iteration to iteration the computation of $\mathbf{y}$ will require $O(n^3)$ flops for a full matrix.
- For such a matrix it might pay to reduce it to a upper Hessenberg form or tridiagonal form before starting the iteration.
- However, if we have a good approximation to an eigenpair then only a few iterations are necessary to obtain close to machine accuracy.
- If Rayleigh quotient iteration converges the convergence will be quadratic and sometimes even cubic.

The QR-algorithm

- An iterative method to compute all eigenvalues and eigenvectors of a matrix $\mathbf{A} \in \mathbb{C}^{n,n}$.
- The matrix is reduced to triangular form by a sequence of unitary similarity transformations computed from the QR-factorization of $\mathbf{A}$.
- Recall that for a square matric the QR-factorization and the QR-decomposition are the same.
- If $\mathbf{A} = \mathbf{Q}\mathbf{R}$ is a QR-factorization then $\mathbf{Q} \in \mathbb{C}^{n,n}$ is unitary, $\mathbf{Q}^H\mathbf{Q} = \mathbf{I}$ and $\mathbf{R} \in \mathbb{C}^{n,n}$ is upper triangular.

Basic QR

$$\boxed{\begin{array}{l} \mathbf{A}_1 = \mathbf{A} \\ \text{for } k = 1, 2, \ldots \\ \quad \mathbf{Q}_k \mathbf{R}_k = \mathbf{A}_k \qquad (\text{QR-factorization of } \mathbf{A}_k) \\ \quad \mathbf{A}_{k+1} = \mathbf{R}_k \mathbf{Q}_k. \\ \text{end} \end{array}} \tag{3}$$

Example

- $\mathbf{A}_1 = \mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$
- $\mathbf{A}_1 = \mathbf{Q}_1\mathbf{R}_1 = (\frac{1}{\sqrt{5}} \begin{bmatrix} -2 & -1 \\ -1 & 2 \end{bmatrix}) * (\frac{1}{\sqrt{5}} \begin{bmatrix} -5 & -4 \\ 0 & 3 \end{bmatrix})$
- $\mathbf{A}_2 = \mathbf{R}_1\mathbf{Q}_1 = \frac{1}{5} \begin{bmatrix} -5 & -4 \\ 0 & 3 \end{bmatrix} * \begin{bmatrix} -2 & -1 \\ -1 & 2 \end{bmatrix} * = \begin{bmatrix} 2.8 & -0.6 \\ -0.6 & 1.2 \end{bmatrix}.$
- $\mathbf{A}_3 = \begin{bmatrix} 2.997 & -0.074 \\ -0.074 & 1.0027 \end{bmatrix}, \quad \mathbf{A}_9 = \begin{bmatrix} 3.0000 & -0.0001 \\ -0.0001 & 1.0000 \end{bmatrix}$
- $\mathbf{A}_9$ is almost diagonal and contain the eigenvalues $\lambda_1 = 3$ and $\lambda_2 = 1$ on the diagonal.

Example 2

$$\mathbf{A}_1 = \mathbf{A} = \begin{bmatrix} 0.9501 & 0.8913 & 0.8214 & 0.9218 \\ 0.2311 & 0.7621 & 0.4447 & 0.7382 \\ 0.6068 & 0.4565 & 0.6154 & 0.1763 \\ 0.4860 & 0.0185 & 0.7919 & 0.4057 \end{bmatrix}$$

we obtain

$$\mathbf{A}_{14} = \left[\begin{array}{r|rr|r} 2.323 & 0.047223 & -0.39232 & -0.65056 \\ \hline -2.1e-10 & 0.13029 & 0.36125 & 0.15946 \\ -4.1e-10 & -0.58622 & 0.052576 & -0.25774 \\ \hline 1.2e-14 & 3.3e-05 & -1.1e-05 & 0.22746 \end{array}\right].$$

$\mathbf{A}_{14}$ is close to quasi-triangular.

Example 2

$$\mathbf{A}_{14} = \left[\begin{array}{r|rr|r} 2.323 & 0.047223 & -0.39232 & -0.65056 \\ \hline -2.1e-10 & 0.13029 & 0.36125 & 0.15946 \\ -4.1e-10 & -0.58622 & 0.052576 & -0.25774 \\ \hline 1.2e-14 & 3.3e-05 & -1.1e-05 & 0.22746 \end{array}\right].$$

- The 1×1 blocks give us two real eigenvalues $\lambda_1 \approx 2.323$ and $\lambda_4 \approx 0.2275$.
- The middle 2×2 block has complex eigenvalues resulting in $\lambda_2 \approx 0.0914 + 0.4586i$ and $\lambda_3 \approx 0.0914 - 0.4586i$.
- From Gerschgorin's circle theorem it follows that the approximations to the real eigenvalues are quite accurate.
- We would also expect the complex eigenvalues to have small absolute errors.

Why QR works

- Since $\mathbf{Q}_k^H \mathbf{A}_k = \mathbf{R}_k$ we obtain

$$\mathbf{A}_{k+1} = \mathbf{R}_k \mathbf{Q}_k = \mathbf{Q}_k^H \mathbf{A}_k \mathbf{Q}_k = \tilde{\mathbf{Q}}_k^H \mathbf{A} \tilde{\mathbf{Q}}_k, \text{ where } \tilde{\mathbf{Q}}_k = \mathbf{Q}_1 \cdots \mathbf{Q}_k. \tag{4}$$

- Thus $\mathbf{A}_{k+1}$ is similar to $\mathbf{A}_k$ and hence to $\mathbf{A}$.
- It combines both the power method and the Rayleigh quotient iteration.
- If $\mathbf{A} \in \mathbb{R}^{n,n}$ has real eigenvalues, then under fairly general conditions, the sequence $\{\mathbf{A}_k\}$ converges to an upper triangular matrix, the Schur form.
- If $\mathbf{A}$ is real, but with some complex eigenvalues, then the convergence will be to the quasi-triangular Schur form

The Shifted QR-algorithms

Like in the inverse power method it is possible to speed up the convergence by introducing shifts. The explicitly shifted QR-algorithm works as follows:

$$
\begin{aligned}
&\mathbf{A}_1 = \mathbf{A} \\
&\text{for } k = 1, 2, \ldots \\
&\quad \text{Choose a shift } s_k \\
&\quad \mathbf{Q}_k \mathbf{R}_k = \mathbf{A}_k - s_k \mathbf{I} \qquad (\text{QR-factorization of } \mathbf{A}_k - s\mathbf{I}) \\
&\quad \mathbf{A}_{k+1} = \mathbf{R}_k \mathbf{Q}_k + s_k \mathbf{I}. \\
&\text{end}
\end{aligned}
\tag{5}
$$

Remarks

1. $\mathbf{A}_{k+1} = \mathbf{Q}_k^H \mathbf{A}_k \mathbf{Q}_k$ is unitary similar to $\mathbf{A}_k$.
2. Before applying this algorithm we reduce $\mathbf{A}$ to upper Hessenberg form
3. If $\mathbf{A}$ is upper Hessenberg then all matrices $\{\mathbf{A}_k\}_{k\geq 1}$ will be upper Hessenberg.
4. Why? because $\mathbf{A}_{k+1} = \mathbf{R}_k \mathbf{A}_k \mathbf{R}_k^{-1}$. A product of two upper triangular matrices and an upper Hessenberg matrix is upper Hessenberg.
5. Givens rotations is used to compute the QR-factorization of $\mathbf{A}_k - s_k \mathbf{I}$.

More remarks

6. If $a_{i+1,i} = 0$ we can split the the eigenvalue problem into two smaller problems. We do this splitting if one $a_{i+1,i}$ becomes sufficiently small.
7. $s_k := \mathbf{e}_n^T \mathbf{A}_k \mathbf{e}_n$ is called the Rayleigh quotient shift
8. The Wilkinson shift is the eigenvalue of the lower right 2×2 corner closest to the n, n entry of $\mathbf{A}_k$.
9. Convergence is at least quadratic for both kinds of shifts.
10. In the implicitly shifted QR-algorithm we combine two shifts at a time. Can find both real and complex eigenvalues without using complex arithmetic.

More remarks

11. To find the eigenvectors we first find the eigenvectors of the triangular or quasi-triangular matrix. We then compute the eigenvalues of the upper Hessenberg matrix and finally the eigenvectors of $\mathbf{A}$.
12. Normally we find all eigenvalues in $O(n^3)$ flops. This is because we need $O(n^3)$ flops to find the upper Hessenberg form, each iterations requires $O(n^2)$ flops if $\mathbf{A}_k$ is upper Hessenberg and $O(n)$ flops if $\mathbf{A}_k$ is tridiagonal, and we normally need O(n) iterations.