TP-Projet 2 : Une autre méthode de recherche des couples propres : la méthode de projection Raleigh-Ritz

Contents

1 Numerical issues and extension of the power method **2**
1.1 The Schur decomposition . . . 2
1.2 Rayleigh quotient and Rayleigh-Ritz projection method . . . 3
1.3 Extending the power method to computing a dominant eigenspace . . . 4
1.3.1 First remark . . . 4
1.3.2 Stopping criterion . . . 5
1.3.3 Efficiency and controlled accuracy . . . 5

2 Extending the power method to compute dominant eigenspace vectors **5**
2.1 Subspace_V0: basic version . . . 5
2.2 Subspace_V1: improved version making use of Raleigh-Ritz projection . . . 6

3 TO DO : Numerical experiments **7**

4 Bibliography **8**

Reminders and introduction

Previously, we have seen that for reducing the dimension by the mean of PCA we do not need the whole spectral decomposition of the symmetric variance/covariance matrix. Indeed we only need the leading eigenpairs which provide enough information about the data.

We have implemented the power method, which was introduced in the Calcul Scientifique lectures, to compute the leading eigenpair. As it has been presented during the same lectures, it is possible to add a deflation process to this algorithm, in order to also compute the following eigenpairs.

In this part of the project, we will see that this specific algorithm is not efficient in the sense of running time. Then we will study two more efficient algorithms, based on an object called *Rayleigh quotient*.

1 Numerical issues and extension of the power method

The basic *power method*, which was introduced in the Calcul Scientifique lectures, is recalled in Algorithm 1; it can be used to determine the eigenpair associated to the largest (in module) eigenvalue.

Algorithm 1 Vector power method

Input: Matrix $A \in \mathbb{R}^{n \times n}$
Output: (λ_1, v_1) eigenpair associated to the largest (in module) eigenvalue.
$x_0 \in \mathbb{R}^n$ given and $p = 0$
$\beta_p = {x_p}^T \cdot A \cdot x_p$
repeat
  $y_{p+1} = A \cdot x_p$
  $x_{p+1} = y_{p+1} / \|y_{p+1}\|$
  $\beta_{p+1} = {x_{p+1}}^T \cdot A \cdot x_{p+1}$
  $p = p + 1$
until $|\beta_{p+1} - \beta_p| / |\beta_p| < \varepsilon$
$\lambda_1 = \beta_{p+1}$ and $v_1 = x_{p+1}$

By adding the deflation process, we are able to compute all the eigenpairs we need to reach a certain percentage of the trace of A. But sadly, this algorithm is not efficient at all in terms of computing time.

A version of the power method with deflation is provided in the file `eigenpair_computation.f90`. It consists in a subroutine called `deflated_power_method`.

Question 1 : Compare the running times of the subroutine `deflated_power_method` to compute a few eigenpairs and of the lapack subroutine `dsyev` – all you need to know about this subroutine is given in the file "compléments.pdf". Comment.

Question 2 : What do you think to be the main drawback of the deflated power method in terms of computing time ?

Our objectiv is to extend the basic power method to compute blocks of eigenpairs and more precisely to **compute an invariant subspace associated to the largest eigenvalues.**

1.1 The Schur decomposition

As said in the introduction, the algorithms on which we will focus make a great use of the *Rayleigh quotient*.

But before introducing this notion, we need to do some recalls about the Schur decomposition (introduced in the "Calcul Scientifique" lectures). Indeed, *Rayleigh quotient* and *Rayleigh-Ritz projection* are largely built on the two fundamental theorems below.

Theorem 1 (General Schur decomposition)**.**

Let $A \in \mathbb{R}^{n\times n}$, there exists an orthogonal matrix U such that

$$U^T \cdot A \cdot U = T,$$

where T is upper triangular. By appropriate choice of U, the eigenvalues of A, which are the diagonal elements of T, may be made to appear in any order.

For the proof see [1] (Chapter 1, page 13).

The decomposition $U^T \cdot A \cdot U = T$, is called a *Schur decomposition* of A. Columns of U are called *Schur vectors*. This decomposition is not unique since it depends on the order of the eigenvalues in T and because of potential multiple eigenvalues.

Furthermore, if A is symmetric $T^T = \left(U^T \cdot A \cdot U\right)^T = U^T \cdot A^T \cdot U^{T^T} = U^T \cdot A \cdot U = T$, T is thus both upper and lower triangular and hence is diagonal.

$$\Lambda = U^T \cdot A \cdot U \;\text{ with }\; \Lambda = diag(\lambda_1, \cdots, \lambda_n)$$

Therefore $A \cdot U = U \cdot \Lambda$ so that, if u_i denotes the i-th column of U then (λ_i, u_i) is an eigenpair of A. We summarize all this in the following theorem.

Theorem 2 (Spectral decomposition of a symmetric matrix)**.**

If A symmetric, it thus has an orthogonal basis of eigenvectors and there exists a decomposition called spectral decomposition:

$$U^T \cdot A \cdot U \;=\; \Lambda \;\text{ , where } U \text{ is orthogonal, and } \Lambda = diag(\lambda_1, \cdots, \lambda_n)$$

This second theorem ensures that our variance/covariance matrix can be diagonalized in an orthonormal basis.

1.2 Rayleigh quotient and Rayleigh-Ritz projection method

We finally describe in this section the most commonly used method for extracting an approximate eigenspace from a larger subspace: *the Rayleigh-Ritz projection method*, which makes deep use of the notion of *Rayleigh quotient*.

To keep the introduction of Rayleigh-Ritz method simple we first show – as in [1] (Chapter 4.1) – how to find exact eigenpairs; we extend this to computing exact eigenspaces and will then suggest (without giving the proof) a procedure to compute approximate eigenpairs.

Theorem 3 (Rayleigh quotient)**.**

Let $\mathcal{U}$ be a subspace and let the columns of the matrix U be an orthogonal basis for $\mathcal{U}$ (U^T is the left inverse of U, $U^T U = I$). We define the Rayleigh quotient to be

$$H = U^T \cdot A \cdot U \;.$$

If $\mathcal{X} \subset \mathcal{U}$ is an eigenspace of A then there is an eigenpair (λ, w) of H such that $(\lambda, U \cdot w)$ is an eigenpair of A.

Proof. Note that no assumption is made on the size of the subspace $\mathcal{U}$. Let (λ, x) be an eigenpair of A corresponding to $\mathcal{X}$ and let $x = U \cdot w$. By definition, (λ, x) being an eigenpair of A, $A \cdot x = \lambda x$, and $A \cdot U \cdot w = \lambda U \cdot w$. Thus,

$$H \cdot w = U^T \cdot A \cdot U \cdot w = U^T \cdot U \cdot \lambda w = \lambda w \;,$$

so that (λ, w) is an eigenpair of H. □

With Theorem 3, we have shown that if $\mathcal{U}$ contains an invariant subspace, an exact eigenpair of A can be obtained by looking at eigenpairs of the Rayleigh quotient H.

Question 3 : If A is of dimension n, and the subspace $\mathcal{U}$ is of range p, what is the dimension of H ? Hence, what should be the relation between n and p to make the Rayleigh quotient interesting to obtain eigenpairs of A, and why ?

Furthermore (and we assume it by continuity), we can expect that when $\mathcal{U}$ contains an approximate eigenspace $\widetilde{\mathcal{X}}$ of A there would be an eigenpair $(\widetilde{L}, \widetilde{w})$ of H such that $(\widetilde{L}, U \cdot \widetilde{w})$ is an approximate eigenpair of A.

This suggests that to approximate an eigenpair of A we can perform the following steps that define the main steps :

1. Find U an orthogonal basis of a subspace $\mathcal{U}$.
2. Form the Rayleigh quotient $H = U^T \cdot A \cdot U$.
3. Compute (M, w) an eigenpair of H.
4. Then $(M, U \cdot w)$ is an approximate eigenpair of A.

But the most important thing is that one can also extend the previous procedure to compute an approximation of **an eigenspace of dimension** m – i.e. an approximation of a block of m eigenpairs. This is illustrated below in the case of a symmetric matrix A (this is often referred to as the *Rayleigh-Ritz projection method*):

Given U an orthogonal basis of a subspace $\mathcal{U}$,

1. Form the Rayleigh quotient $H = U^T \cdot A \cdot U$.
2. H is symmetric : compute the spectral decomposition of H : $X^T \cdot H \cdot X = \Lambda$
3. X is thus an eigenspace of H corresponding to eigenvalues in $diag(\Lambda)$ so that $U \cdot X$ is an approximate eigenspace of A corresponding to eigenvalues in $diag(\Lambda)$.

In the following, we are going to present two extensions of the power method, to compute simultaneously blocks of eigenpairs. The both extensions use the Rayleigh quotient, but not in the same way.

1.3 Extending the power method to computing a dominant eigenspace

Now, we have all the formal mathematical ingredients we need to build our two new algorithms. But before presenting them, here are pieces of advice and remarks you have to keep in mind in order to extend the power method to compute blocks of eigenpairs.

1.3.1 First remark

The first question you may ask is : why do not simply apply Algorithm 1 on m initial vectors (matrix X_p), instead of just one (vector x_p) ? Actually, if one extends Algorithm 1 to iterate on such a matrix, then it will not tend to the expect result, i.e. X_p does not converge towards a matrix whose columns contain m different eigenvectors of A.

Question 4 : Towards which matrix does X_p converge in such an algorithm ? Modify the Matlab script `puissance_iteree.m` from the TP-Projet 1 to check your conjecture.

1.3.2 Stopping criterion

Another difficulty to adapt Algorithm 1 in order to compute blocks of eigenpairs at once is the stopping criterion, because a set of eigenpairs and not only one vector must be tested for convergence.

The current stopping criterion in Algorithm 1 relies on the stability of the computed eigenvalue (it tests the fact that the computed eigenvalue no longer changes "much"). This does not take into account the invariance of the eigenvector which is numerically more meaningful.

One should first notice that, in Algorithm 1, at convergence $x_p = v_1$ holds and therefore $A \cdot x_p = \lambda_1 \cdot x_p$. By analogy with the backward error introduced during the lectures for the solution of linear systems ($A \cdot x = b$ leading to backward error $\frac{\|A \cdot x - b\|}{\|A\|\|x\| + \|b\|}$) we suggest to replace the stopping criterion in Algorithm 1 by a backward error on the invariance of the eigenvectors that can be estimated as $\|A \cdot x_p - \beta_{p+1} \cdot x_p\| / |\beta_p + 1|$ where the Frobenious norm could used to compute the norm of a matrix.

Furthermore, when a complete set of eigenpairs is expected to converge then the previous criterion should be generalized to a matrix X_p instead of a vector x_p.

1.3.3 Efficiency and controlled accuracy

At each iteration p, columns of X_p are expected to be an approximation of eigenvectors of A. Hence, X_p is expected to be an approximate eigenspace of A.

The Rayleigh quotient H computed on an estimate of the eigenspace U ($H = U^T \cdot A \cdot U$) is a small square symmetric matrix which holds the eigenspace information and can thus be used for computing the invariance of the eigenspace and thus a global stopping criterion. This point is used to build the first extension of the power method "Subspace_V0" given in Algorithm 2.

Furthermore, since H is small, computing the spectral decomposition of H is cheap, and the Rayleigh-Ritz projection method can be applied to improve the stopping criterion, to accelerate the method and to obtain the dominant eigenpairs of A. In other words, instead of waiting the global convergence towards an invariant subspace, another possibility is to check after each iteration if some columns in X_p have converged already towards some eigenvectors of A. This point will be used in the "Subspace_V1" extension of the power method, whose pseudo-code is provided in Algorithm 4.

2 Extending the power method to compute dominant eigenspace vectors

As mentioned before, a direct extension of the power method where we iterate simultaneously on m initial vectors V, instead of just one, will not provide in general the expected result. Indeed, it is a little bit more complicated to create an extension of the power method to compute blocks of eignepairs, which works efficiency.

In this section we are going to present two extensions of the power method.

2.1 Subspace_V0: basic version

The *first basic version of the method* to compute an invariant subspace associated to the largest eigenvalues is described in Algorithm 2 and makes great use of *Rayleigh quotient and spectral decomposition.*

Given a set of m linearly independent vectors Y, the Algorithm 2 computes the eigenvectors associated with the m largest (in module) eigenvalues in $\mathcal{R}(Y)$ – $\mathcal{R}(Y)$ being the subspace generated by the m columns of Y.

Algorithm 2 Dominant eigenspace method: basic version

Input: Symmetric matrix $A \in \mathbb{R}^{n\times n}$, tolerance ε and $MaxtIter$ (max nb of iterations)
Output: m dominant eigenvectors V_{out} and the corresponding eigenvalues Λ_{out}.

Generate a set of m linearly independent vectors $V \in \mathbb{R}^{n\times m}$; $niter = 0$
repeat
  $V \longleftarrow$ orthonormalization of the columns of Y
  Compute Y such that $Y = A \cdot V$
  Form the Rayleigh quotient $H = V^T \cdot A \cdot V$
  $niter = niter + 1$
until ($niter > MaxIter$ **or** Invariance of the subspace V at precision ε : $\|AV - VH\|/\|A\| \leq \varepsilon$)
Compute the *spectral decomposition of the Rayleigh quotient* H from which the m *dominant eigenvalues* Λ_{out} *and corresponding eigenspace* V_{out} *can be deduced.*

Question 5 : Why do Algorithms 2 and 4 proceed with an "orthonormalization of the columns of Y" ?

In Algorithm 2, our stopping criterion is built on the invariance of the subspace V. To define the invariance of the subspace, we can extend the notion of backward error for the solution of linear systems to this eigenspace method, as it is stated in the Section 1.3.2.

To simplify our discussion, assume that we have converged so that $AV = V\Lambda_{out}$ with $\Lambda_{out} = \text{diag}(\lambda_1, \ldots, \lambda_m)$. At convergence we thus have $VH = VV^TAV = VV^TV\Lambda_{out} = V\Lambda_{out}$. Thus, in our context, a possible measure of the backward error could be : $\|AV - VH\|/\|A\|$.

Question 6 : In the file `eigenpair_computation.f90`, fill in the subroutine `subspace_iter_v0` to obtain Algorithm 2.

2.2 Subspace_V1: improved version making use of Raleigh-Ritz projection

Several modifications are needed to make the simple subspace iteration an efficient and practically applicable code.

First we may chose to operate on a subspace whose dimension m is larger than the number of the dominant eigenvalues (n_{ev}) needed. Moreover, in our application, it is more likely that the user asks to compute the smallest eigenspace such that the sum of the associated dominant eigenvalues is larger than a given percentage of the trace of the matrix A, than a given number of eigenpairs. The Rayleigh-Ritz projection procedure can then be used to get the approximate eigenspace and stop when the expected percentage is reached.

The Rayleigh-Ritz projection procedure is combined with subspace iteration method to improve the convergence. It consists in rearranging the orthogonal matrix V so that its columns reflect the fast convergence of the dominant subspaces.

The algorithmic description of this procedure is given below, for a symmetric matrix A. We assume that the matrix is symmetric positive definite, define the Raleigh-Ritz projection algorithm and then introduce the improved version of our algorithm.

Algorithm 3 Raleigh-Ritz projection

Input: Matrix $A \in \mathbb{R}^{nxn}$ and an orthonormal set of vectors V.
Output: The approximate eigenvectors V and the corresponding eigenvalues Λ.
Compute the Rayleigh quotient $H = V^T A V$.
Compute the spectral decomposition $H = X \Lambda X^T$, where the eigenvalues of H ($diag(\Lambda)$) are arranged in descending order of magnitude.
Compute $V = V X$.

Algorithm 4 Dominant eigenspace method with Raleigh-Ritz projection

Input: Symmetric matrix $A \in \mathbb{R}^{n\times n}$, tolerance ε, $MaxIter$ (max nb of iterations) and $PercentTrace$ the target percentage of the trace A
Output: m dominant eigenvectors V_{out} and the corresponding eigenvalues Λ_{out}.

Generate an initial set of m orthonormal vectors $V \in \mathbb{R}^{n\times m}$; $niter = 0$; $PercentReached = 0$
repeat
  $V \longleftarrow$ orthonormalization of the columns of Y
  Compute Y such that $Y = A \cdot V$
  Rayleigh-Ritz projection applied on orthonormal vectors V and matrix A
  Convergence: save eigenpairs that have converged and update $PercentReached$
  $niter = niter + 1$
until ($PercentReached > PercentTrace$ or $niter > MaxIter$)

Convergence is tested immediately after a Rayleigh-Ritz Projection step. We want to test which approximate eigenvector has converged; we will test in order the columns associated to the largest entries in Λ and will stop as soon as one eigenvector has not converged.

Note that if the j_{th} column of V has converged then $\|r_j\| = \|A \cdot V_j - \Lambda_j \cdot V_j\| \leq \gamma\,\varepsilon$, where ε is a convergence criterion and γ is a scaling factor. A natural choice of γ is an estimate of some norm of A.

Convergence theory says the eigenvectors corresponding to the largest eigenvalues will converge more swiftly than those corresponding to smaller eigenvalues. For this reason, we should test convergence of the eigenvectors in the order $j = 1, 2, \ldots$ and stop with the first one to fail the test.

Question 7 : Explain in your own words – by taking great care to be clear – the differences between Algorithms 2 and 4, and their consequences. Highlight especially the different uses of the Rayleigh quotient.

Question 8 : In the file `eigenpair_computation.f90`, write the subroutine `subspace_iter_v1` to obtain Algorithm 4.

3 TO DO : Numerical experiments

The Fortran code can be compiled by typing the `make` command (a "Makefile" is provided). This will generate an executable file named "main", which executes the driver program. Everything you need to know to run your `main` program is in the `readme` file in the sources.

To run your program, you have to set up several parameters :

1. the method variant (the Lapack subroutine `dsyev`, the deflated power method, Algorithm 2 or Algorithm 4);
2. the size n of the A matrix whose eigenpairs have to be computed;
3. the type of matrix to be generated: the driver program can generate 4 types of matrices. Each type has different spectral properties and will therefore yield a different convergence behavior for the subspace iteration variants;
4. the dimension m of the invariant subspace;
5. the percentage of the trace needed (only for Algorithm 4, to be expressed as a value between 0 and 1);

We ask you to play with these parameters to compare the three algorithms and also to see their own performance according to these different parameters. You have to write a report to explain your results.

Of course, you have to play with all the parameters, but you need to take a particular care to the size and the type of the matrices.

In particular, we would like to see some figures which show the eigenvalue distribution of the different types of matrices (parameter 3), in order to explain the differences of performance of the algorithms according to the spectrum of A. The spectrum of the tested matrix can be returned by the `main` program (cf the `readme` to know how and where). Feel free to create these figures with Matlab, OpenOffice Calc, or whatever.

We are also expecting you to play with very big matrices.

4 Bibliography

[1] G. W. Stewart. *Matrix Algorithms: Volume 2, Eigensystems.* Society for Industrial and Applied Mathematics (SIAM), 2001.