TP-Projet 2 : Subspace Iteration Methods

Reminders and introduction

Previously, we have seen that for reducing the dimension by the mean of Principal Component Analysis (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 terms of performance. Then we will study three more efficient algorithms, based on an object called *Rayleigh quotient*.

1 Limitations 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.
$v \in \mathbb{R}^n$ given
$\beta = v^T \cdot A \cdot v$
repeat
  $y = A \cdot v$
  $v = y / \|y\|$
  $\beta_{old} = \beta$
  $\beta = v^T \cdot A \cdot v$
until $|\beta - \beta_{old}| / |\beta_{old}| < \varepsilon$
$\lambda_1 = \beta$ and $v_1 = v$

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 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 `power_v11.f90`.

TODO :
Question 1: Compare the running time of the subroutine `deflated_power_method` to compute a few eigenpairs with the running time of the Lapack subroutine `dsyev`[a]. Comment.
Question 2: What do you think to be the main drawback of the deflated power method in terms of computing time?

[a] all you need to know about this subroutine is given in the file "Manipulation_Tableaux_en_Fortran"

Our objective is to extend the power method to compute a block of dominant eigenpairs.

2 Extending the power method to compute dominant eigenspace vectors

2.1 subspace_iter_v0: a basic method to compute a dominant eigenspace

The *basic version of the method* to compute an invariant subspace associated to the largest eigenvalues of a symmetric matrix A is described in Algorithm 2. This subspace is also called dominant eigenspace.

Given a set of m orthonormal vectors V, the Algorithm 2 computes the eigenvectors associated with the m largest (in module) eigenvalues.

Algorithm 2 Subspace iteration method v0 : the 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 orthonormal vectors $V \in \mathbb{R}^{n \times m}$; $k = 0$
repeat
  $k = k + 1$
  $Y = A \cdot V$
  $H = V^T \cdot Y$ $\{H = V^T \cdot A \cdot V\}$
  Compute $acc = \|A \cdot V - V \cdot H\| / \|A\|$
  $V \longleftarrow$ orthonormalisation of the columns of Y
until ($k > MaxIter$ **or** $acc \leq \varepsilon$)
Compute the spectral decomposition $X \cdot \Lambda_{out} \cdot X^T = H$, where the eigenvalues of H ($diag(\Lambda_{out})$) are arranged in descending order of magnitude.
Compute the corresponding eigenspace $V_{out} = V \cdot X$

This algorithm is very close to Algorithm 1:

- The process is mainly based on iterative products between the matrix A and the columns of V,
- Inside the loop, the matrix H plays the same role than the scalar β in Algorithm 1.

There are however some differences:

- An orthonormalisation of the columns of Y is realised at each iteration,
- The relationship between both stopping criteria is not trivial,
- At the end of the loop, the columns of V are not the eigenvectors of A. Other operations have to be done after convergence to obtain the actual dominant eigenspace of A.

2.1.1 Orthonormalisation

The first question you may ask is: "why do not simply apply Algorithm 1 on m initial vectors (matrix V), instead of just one (vector v)?"

Actually, if one extends Algorithm 1 to iterate on such a matrix, then it will not tend to the expect result, i.e. V does not converge towards a matrix whose columns contain m different eigenvectors of A.

TODO :
Question 3: Towards which matrix does V converge in such an algorithm ? Modify the Matlab script `puissance_iteree.m` from the TP-Projet 1 to check your conjecture.

To avoid this phenomenon, an orthonormalisation of the columns of Y is realised at the end of each iteration in the Algorithm 2. The new set V is the result of this orthonormalisation.

2.1.2 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 must be tested for convergence and not only one vector.

The current stopping criterion in Algorithm 1 relies on the stagnation of the computed eigenvalue (it tests the fact that the computed eigenvalue no longer changes "much"). This choice has been done because it is computationally cheap. But this does not take into account the invariance of the eigenvector which is numerically more meaningful (see Optimisation Lectures).

One should notice that, in Algorithm 1, we have reached the convergence once $v = x_1$ and $\beta = \lambda_1$, and therefore $A \cdot v = \beta \cdot v$. Then, a more meaningful stopping criterion for Algorithm 1 is to compute the relative invariance of the eigenvector that can be estimated as $\|A \cdot v - \beta \cdot v\|/\|A\|$. The **Frobenius norm** could then be used to compute the norm of a matrix.

In Algorithm 2, we assume we have converged so that $AV = VH$. Thus, in our context, a possible measure of the convergence could be: $\|AV - VH\|/\|A\|$. As before, the **Frobenius norm** can be used to compute the norm of both numerator and denominator.

2.1.3 Rayleigh quotient

Once the convergence is reached in Algorithm 2, V **does not contain eigenvectors of** A. But some spectral information about A can be extracted from H. Indeed, the loop in Algorithm 2 has converged once $A \cdot V = V \cdot H$. Then some eigenpairs of A can easily be obtained from eigenpairs of H: if (λ, x) is an eigenpair for H, then $(\lambda, y = V \cdot x)$ is an eigenpair for A[1]. For a symmetric matrix A and a matrix V whose columns are orthonormal, such a matrix $H = V^T \cdot A \cdot V$ is called a **Rayleigh quotient**. It will play a crucial role in the two last algorithms. For complements, see [1].

TODO :
Question 4: We are looking at variants of the power method in order to avoid computing the whole spectral decomposition of the matrix A. But in Algorithm 2, a computation of the whole spectral decomposition of the matrix H is performed. Explain why it is not a problem by investigating the dimensions of H.

TODO :
Question 5: In the file `iter_v0.f90`, fill in the subroutine `subspace_iter_v0` to obtain Algorithm 2.

[1] $A \cdot y = A \cdot V \cdot x = V \cdot H \cdot x = V \cdot \lambda x = \lambda V \cdot x = \lambda y$.

2.2 subspace_iter_v1: improved version making use of Raleigh-Ritz projection

Several modifications are needed to make the basic subspace iteration an efficient code.

2.2.1 First improvements

In our Principal Component Analysis 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. Let's call n_{ev} the number of the dominant eigenvalues needed.

Because this number n_{ev} is not known in advance, we chose to operate on a subspace whose dimension m is larger than n_{ev}. Note that if we reach the given size m of the subspace V without obtaining the percentage of the trace we have to stop: the method should be called again with a higher m.

Moreover to be able to stop the algorithm when the expected percentage is reached, an adaptation of the spectral decomposition of the Rayleigh quotient is used **inside** the subspace iteration.

This adaptation is called the Rayleigh-Ritz projection procedure; an algorithmic description is given in Algorithm 3, for a symmetric positive definite matrix A, as the variance/covariance matrix.

Algorithm 3 Raleigh-Ritz projection

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

The algorithm of the subspace_iter_v1 is then:

Algorithm 4 Subspace iteration method v1 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 of A
Output: n_{ev} dominant eigenvectors V_{out} and the corresponding eigenvalues Λ_{out}.

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

2.2.2 Convergence analysis step

A convergence analysis step is performed immediately after the Rayleigh-Ritz Projection step; its goal is to determine which eigenvectors have converged at the current iteration k.

We consider that the eigenvector j, stored in the j_{th} column of V has converged when

$$\|r_j\| = \|A \cdot V_j - \Lambda_j \cdot V_j\| / \|A\| \leq \varepsilon.$$

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 $j = 1, 2, \ldots$:

- we consider that we do not converge at this iteration with the first one to fail the test,
- it is also not useful to test again the vectors that converged at the previous iteration.

TODO :
Question 6: In the file `iter_v1.f90` or `subspace_iter_v1.m`, identify all the steps of Algorithm 4.

3 subspace_iter_v2: toward an efficient solver

Two ways of improving the efficiency of the solver are proposed. Our aim is to build an algorithm that combines both the block approach and the deflation method in order to speed-up the convergence of the solver.

3.1 Block approach

Orthonormalisation is performed at each iteration and is quite costly. One simple way to accelerate the approach is to perform p products at each iteration (replace $V = A \cdot V$ (first step of the iteration) by $V = A^p \cdot V$). Note that this very simple acceleration method is applicable to all versions of the algorithm.

TODO :
Question 7: Modify the subprogram `subspace_iter_v2` in file `iter_v2.f90` to implement this acceleration (note that the initial code of this subprogram is the v1 version of the method, the only difference is the input p).

3.2 Deflation method

Because the columns of V converge in order, we can freeze the converged columns of V. This freezing results in significant savings in the matrix-vector ($V = A \cdot V$), the orthonormalisation and Rayleigh-Ritz Projection steps.

Specifically, suppose the first nbc[2] columns of V have converged, and partition $V = [V_c, V_{nc}]$ where V_c has nbc columns and V_{nc} has $m - nbc$ columns[3]. Then, we can form the matrix $[V_c, A \cdot V_{nc}]$, which is the same

[2] **n**um**b**er of vectors that have **c**onverged
[3] V_c, Vectors that have **c**onverged, V_{nc} Vectors that have **n**ot **c**onverged

as if we multiply V by A. However, we still need to orthogonalise V_{nc} with respect to the frozen vectors V_c by first orthogonalising V_{nc} against V_c and then against itself.

Finally, the Rayleigh-Ritz Projection step can also be limited to the columns V_{nc} of V.

TODO :
Question 8: Running the program *main* with option *disp = 2*, displays the accuracy of a vector when it converges and also at the end of the computation to check if everything is ok.
Explain why, with subspace_iter_v1 method, this accuracy differs for some of the vectors.
Question 9: Try to anticipate what will occur with the subspace_iter_v2 method
Question 10: Modify the subprogram `subspace_iter_v2` to implement this deflation[a].

[a] once may want to first develop the Matlab version; it is possible, starting from the code Matlab of v1

4 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.txt` file in the sources.

To run your program, you have to set up several parameters (see also the `README.txt` file):

1. the solver (the LAPACK DSYEV, the deflated power method, subspace_iter_v0, subspace_iter_v1 and subspace_iter_v2 versions);
2. the size n of the matrix A whose eigenpairs have to be computed;
3. the type of matrix to be generated: the program can generate 4 types of matrices. Each type has different spectral properties and will therefore yield a different convergence behaviour for the subspace iteration variants;
4. m, the maximum number of eigenpairs (deflated power method) or the dimension of the invariant subspace (subspace iteration methods);
5. the percentage of the trace needed (only useful for the deflated power iteration and subspace_iter_v1 and subspace_iter_v2 variants), to be expressed as a value between 0.0 and 1.0);

TODO : Question 11: Observe the behaviour of the subprogram you wrote in Section 3.1 when increasing p. Explain your results.

TODO : Question 12: What are the differences between the 4 types of matrices ? Create some figures that show the eigenvalue distribution of these different types. The spectrum of a tested matrix is returned by the `main` program (cf the `README.txt` to know how). Feel free to create these figures with Matlab, OpenOffice Calc, ...

TODO : Question 13: Compare the performances of the algorithms you have implemented and those provided (including DSYEV) for different types and sizes of matrices.

5 Bibliography

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