Subspace Iteration Methods
Application to Image Compression

Reminders and introduction

During the first semester, you have seen, in the *Analyse de Données* course, that, for reducing the dimension by the mean of Principal Component Analysis (PCA), you do not need the whole spectral decomposition of the symmetric variance/covariance matrix. Indeed you only need the leading eigenpairs which provide enough information about the data.

In the first *Calcul Scientifique* CTD session, we have introduced the **power method** to compute the leading eigenpair of a matrix. Coupled with a deflation process, the following eigenpairs can be computed.

In this project, we will see that this specific algorithm is not efficient in terms of performance. Then we will present a more efficient method called **subspace iteration method**, based on a mathematical object called *Rayleigh quotient*.

We will study **four variants of this method**.

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 eigenvector 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 the eigenpairs we need, for instance, to reach a certain percentage of the trace of A.

A Matlab version of the **basic** *power method with deflation* is provided in the file `power_v11.m` with a different computation of the stopping criteria (see 2.1.2).

TODO :
Question 1: Using the file `test_v11.m`, compare the running time of the function `power_v11` to compute a few eigenpairs with the running time of the function `eig` of Matlab that compute all the eigenpairs for different sizes and types of matrices.

You will present these results in a table to answer this question in your report. Comment.

Question 2: Looking closely at the proposed algorithm, two matrix × vector products, $A.v$, are identified inside the loop.
By rearranging the operations, propose an algorithm that will have only one matrix × vector product in the loop[a].

Implement this algorithm in Matlab (`power_v12.m`) and check that it is twice as fast as the proposed algorithm.

You can present the results in the same table or in a second table.
You will write this new algorithm in your report.

Question 3: What do you think to be the main drawback of the deflated power method in terms of computing time?

New Matlab files for this first work:

1. `power_v12.m`, that implements the new algorithm
2. `test_v11v12.m`, a copy of `test_v11.m` with the call to the new algorithm

[a] You can draw inspiration from the version v0 of the subspace iteration algorithm of section 2.1

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 $A \in \mathbb{R}^{n \times n}$, number of required eigenpairs m, 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
  $k = k + 1$
  $Y = A \cdot V$
  $H = V^T \cdot A \cdot V$ {ou $H = V^T \cdot Y$}
  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. Additionnal 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.

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

TODO :
Question 4: Towards which matrix V does the power method algorithm converge if we apply it to a set of m vectors?

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 $\|A \cdot v - \beta \cdot v\|/|\beta|\|v\|$ which is equivalent to $\|A \cdot v - \beta \cdot v\|/|\beta|$ as the norm of v is one[1].

In Algorithm 2, we assume we have converged so that $AV \approx VH$. Thus, in our context, a possible measure of the convergence could be: $\|AV - VH\|/\|A\|$. 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 $A \cdot V \approx V \cdot H$. 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[2]. 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 last algorithms. For complements, see [1].

TODO :
Question 5: 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 to compute the whole spectral decomposition of H by investigating its dimensions.

TODO :
Question 6: In the file `subspace_iter_v0.m`, fill in the function to obtain Algorithm 2.

Matlab files for this step:

1. `subspace_iter_v0.m`
2. `test_v0v1.m`, that tests this version and the next one (`v1`)

[1] it is this criteria that is implemented in the provided file, *power_v11.m*
[2] $A \cdot y = A \cdot V \cdot x = V \cdot H \cdot x = V \cdot \lambda x = \lambda V \cdot x = \lambda y$.

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 7: In the file `subspace_iter_v1.m`, identify all the steps of Algorithm 4

for the report, copy the algorithm[a] and indicate the line of the code corresponding to each step

[a] for those you want to use LaTeX, we give the file algo_v1.tex with the LaTeX code of the algorithm

3 subspace_iter_v2 and subspace_iter_v3: 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 (subspace_iter_v2)

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 8: what is the cost in term of flops of the computation of A^p, then $A^p \cdot V$? How organize differently this computation to reduce this cost?
Question 9: Modify the file `subspace_iter_v2.m` 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).
Question 10: Observe the behaviour of this approach when increasing p. **Explain your results in your report.**

Matlab files for this step:

1. `subspace_iter_v2.m`
2. `test_v0v1v2.m`, that tests the 3 versions, and for `v2`, different p

3.2 Deflation method (subspace_iter_v3)

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[3] 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[4]. Then, we can form the matrix $[V_c, A \cdot V_{nc}]$, which is the same 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 11:
One result of the functions that compute the eigenpairs is the quality of each eigenpairs (`qv`) that is used as a convergence criteria, $\|A \cdot v - \beta \cdot v\|/|\beta|$
Explain why, with subspace_iter_v1 method, this accuracy differs for some of the vectors (same for v2 version).
Question 12: Try to anticipate what will occur, in term of accuracy of the eigenpairs, with the subspace_iter_v3 method
Question 13: Copy the file `subspace_iter_v2.m` into a file `subspace_iter_v3.m` to implement this deflation.

Matlab files for this step:

1. `subspace_iter_v3.m`
2. `test_v0v1v2v3.m`, that tests the 4 versions

4 TO DO: Numerical experiments

TODO : Question 14: What are the differences between the 4 types of matrices ? Create some figures that show the eigenvalue distribution of these different types (the matrix and its spectrum are stored in a file when you create the matrix).

TODO : Question 15: Compare the performances of the algorithms you have implemented and those provided (including `eig`) for different types and sizes of matrices.

5 Application to Image Compression: Next session

In one month, during the second session of the Project, we will use those algorithms to perform image compression.

You **have to work** between these two sessions and arrive at the second session with mostly all the algorithms developed.

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

6 Bibliography

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