Calcul scientific report
Subspace iteration methods

RICARD Mattéo
BAURIAUD Laura

Département Sciences du Numérique - Première année
2022-2023

Contents

List of Figures

1 Introduction

The subject of this document is subspace iteration methods. The goal is to compare different methods and implements new ones.

2 Partie 1

2.1 Limitations of the power method

2.1.1 Question 1

Running time for a 200x200 matrix				
Name of the algorithm	1	2	3	4
eig	1.094e-01	6.250e+02	0.000e+00	7.818e-02
power method	5.250e+00	6.250e02	2.188e-01	5.156e+00
power method v2	1.141e+00	4.668e-02	1.562e-02	1.078e+00

Table 1: Running time for the 4 types of matrix for each algorithm

If we compare the different methods, we can see that the performances of the algorithms depend of the type of matrix we choose.
Furthermore the power method is less efficient than the eig method except for the type 2 of matrix.

2.1.2 Question 2

Algorithm 1 Vector power method

```
Input : Matrix $A \in \mathbb{R}^{n \times n}$
Output : $(\lambda_1, v_1)$
    $v \in \mathbb{R}^n$ given
    $\beta = v^T A v$
    $y = Av$
repeat
    $v = \frac{y}{\|y\|}$
    $y = Av$
    $\beta_{\text{old}} = \beta$
    $\beta = v^T y$
until $|\beta - \beta_{\text{old}}| / |\beta_{\text{old}}| < \epsilon$
    $\lambda_1 = \beta$ and $v_1 = v$
```

The solution to reduce the complexity of the algorithm is to calculate y after v. Thus we don't have to calculate y a second time when we want to obtain β. We obtain a result twice faster than with power method version 1.

2.1.3 Question 3

The main drawback of the deflated method in terms of computing time is the matrix x vector products in the loop.

2.2 Extending the power method to compute dominant eigensoace vectors

2.2.1 Question 4

The basic version of Algorithm 1 computes an eigenvector associated to the largest eigenvalue of a symmetric matrix A.
If we apply this algorithm to a set of m vectors we will obtain a matrix and all the column will approximately be the same associated to the largest eigenvalue (or the 2nd largest sometimes). That's why we need to add the deflation process to compute all the eigenpairs and not just the biggest one.

2.2.2 Question 5

We are looking to a small number of vectors so it's not a problem to compute the whole spectral decomposition. The dimension is mXm (a small one).

2.2.3 Question 6

We have complete the subspace iter v0 and complete the table.

2.2.4 Question 7

Algorithm 2 Subspace iteration method v1 with Raleigh-Ritz projection

```
Input: Symmetric matrix $A \in \mathbb{R}^{n\times n}$ , tolerance $\epsilon$, $MaxIter$ (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 $\Lambda_{out}$.

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 ← orthonormalisation of the columns of Y
    $Rayleigh - Ritzprojection$ 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 $k > MaxIter$)
```

Now, we will identify the step of the Algorithm in the code :

- l 48-49 : to generate an initial set of m orthonormal vectors V
- l 54 : for k = k+1
- l 56 : to compute Y = $A \cdot V$
- l 58 : the orthonormalisation
- l 61 : $Rayleigh - Ritzprojection$ applied on matrix A and orthonormal vectors V
- l 70-109 : *Convergence analysis step*
- l 115 : the condition to go out the loop

2.3 Toward an efficient method

2.3.1 Question 8

Now we will considere the cost of each operation :

- the product $A \times A$: n-1 additions and n product for each element of A^2 so to compute A^2, it costs $(2n-1)n^2$ flops
- A^p : it costs $(p-1)(2n-1)n^2$ flops
- $A \cdot V$ (a product *matrix* $\times$ *vector*) : it costs $(2n-1)n$ flops

To conclude, we need $(p-1)(2n-1)n^2+(2n-1)n$ flops to compute $A^p.V$. We have a complexity of $\mathcal{O}(2 \cdot p \cdot n^3)$.

However, if we first compute $A \cdot V$, it costs $(2n-1)n$ flops and the result is a vector which has the same dimension as V. Then if we compute $A \cdot (A \cdot V)$ it only costs $(2n-1)n$ more flops. So we can compute $A^p.V$ by doing $A.(A.(A.(...A.(A.V)...)))$ and it will only costs $p(2n-1)n$ flops, a complexity of $\mathcal{O}(2 \cdot p \cdot n^2)$.

2.3.2 Question 9

We have complete the matlab file *subspace_iter_v2*. We are doing p products each time so we are accelerating the algorithm.

2.4 Question 10

First, by increasing P, we are decreasing the time of processing the algorithm.
However, if we increase P too much, the algorithm never ends. We can explain it by the fact we find all the eigenvalues in the same time so the algorithm doesn't find the time to proceed the orthonormalisation. Thus we obtain all the eigenvectors associated to the same eigenvalue, the same problem as before (question 4).

We test the algorithm with a matrix 200 x 200 - type 4.

Behaviour of *subspace_iter_v2*					
P	1	5	10	50	1000
Time	3.500e-01	1.1e-01	8.000e-2	5.000e-02	$\emptyset$
Number of iteration	265	53	27	6	$\emptyset$

Table 2: Running time subspace _iter _v2 for different value of p

2.4.1 Question 11

The more the quality of the eigenpairs *qv* is important (an eigenvalue with an important module), the more the difference has to be small to be inferior to ϵ.
On the contrary if the eigenpairs *qv* is small, the easier the convergence criteria will be completed so the vectors won't be enough accurate.

2.4.2 Question 12

It will be better because we stop to compute the eigenpairs when they had converged. So it will reduce the number of iterations.

2.4.3 Question 13

We have complete the matlab file *subspace_iter_v3* to implement the algorithm.

2.5 Numerical experiments

2.5.1 Question 14

The eigenvalues distribution depends of the type of the matrix (figure **??**) :

- type 1 : eigenvalues D(i) = i
- type 2 : eigenvalues D(i) = random(1/cond, 1), cond = 1e10
- type 3 : eigenvalues D(i) = cond**(-(i-1)/(n-1)), cond = 1e5
- type 4 : eigenvalues D(i) = 1 - ((i-1)/(n-1))*(1 - 1/cond), cond = 1e2

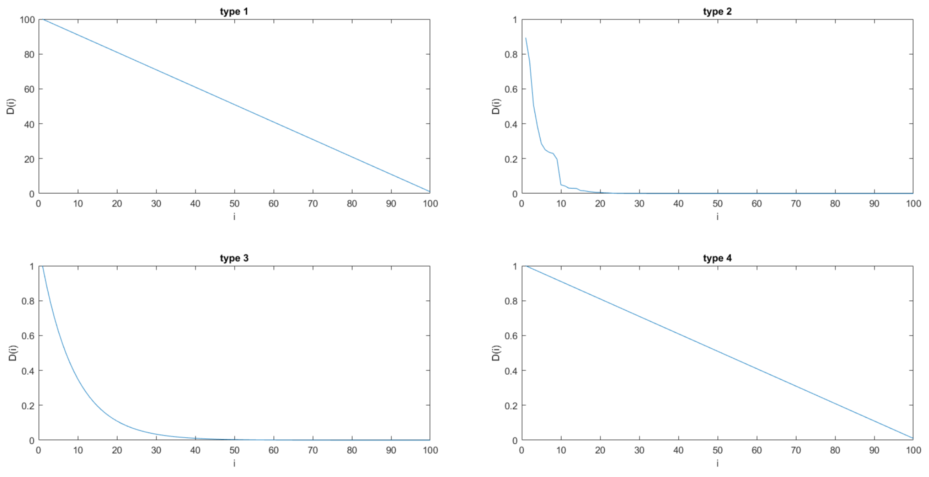

Figure 1: The distribution of the eigenvalues for the different types.

2.6 Question 15

Running time for a 200x200 matrix				
Name of the algorithm	1	2	3	4
eig	0.000e+00	0.000e+00	0.000e+00	0.000e+00
power method	2.031e+00	3.125e-02	6.250e-02	1.953e+00
power method v2	1.141e+00	4.668e-02	1.562e-02	1.078e+00
V0	5.988e+01	1.406e+00	5.688e+00	6.188e+01
V1	6.250e-01	0.000e+00	0.000e+00	6.250e-01
V2 (P=10)	0.000e+00	0.000e+00	0.000e+00	1.562e-01
V3 (P=10)	0.000e+00	0.000e+00	0.000e+00	1.562e-01
V2 (P=50)	0.000e+00	∅	∅	1.562e-01
V3 (P=50)	0.000e+00	∅	∅	1.406e-01

Table 3: Running time for the 4 types of matrix for each algorithm with 200x200 matrix

Running time for a 500x500 matrix				
Name of the algorithm	1	2	3	4
eig	0.000e+00	1.094e-01	1.406e-01	1.250e-01
power method	∅	1.062e+00	1.375e+00	∅
power method v2	∅	5.781e-01	8.125e-01	∅
V0	6.059e+02	1.730e+01	6.917e+01	5.962e+02
V1	3.703e+00	1.250e-01	2.500e-01	4.250e+00
V2 (P=10)	1.000e+00	1.250e-01	1.406e-01	8.906e-01
V3 (P=10)	7.656e-01	0.000e+00	0.000e+00	7.656e-01
V2 (P=50)	5.938e-01	∅	∅	6.250e-01
V3 (P=50)	5.000e-01	∅	∅	6.250e-01

Table 4: Running time for the 4 types of matrix for each algorithm with 500x500 matrix

3 Partie 2

3.1 Question 1

The sizes are :

- Σ_k = (k,k)
- U_k = (q,k)
- V_k = (p,k)

3.2 Question 2

Running time for reconstruction of the image (a matrix of 1187x1920)							
Name of the algorithm	eig	power method	power method v2	V0	V1	V2 (P=3)	V3 (P=3)
Time	5.622e-01	0	0	9.329e+01	8.286e+01	2.212e+01	3.191e+01

Table 5: Running time for the reconstruction of the image with the different algorithms

We achieve to reconstruct the image with all the algorithms except for V0 but we can see that the running time is very different in function of the methods. The main problem here is the size of the matrix which increases a lot the running time.
We can add that to complete the V2 and V3 methods we have to choose a small p otherwise the algorithms never end.
To conclude, the eig method is still the best method to reconstruct an image.

To reconstruct a colored image we just have to do the same process for each component (rgb) of the image. So it takes 3 times longer to compute.

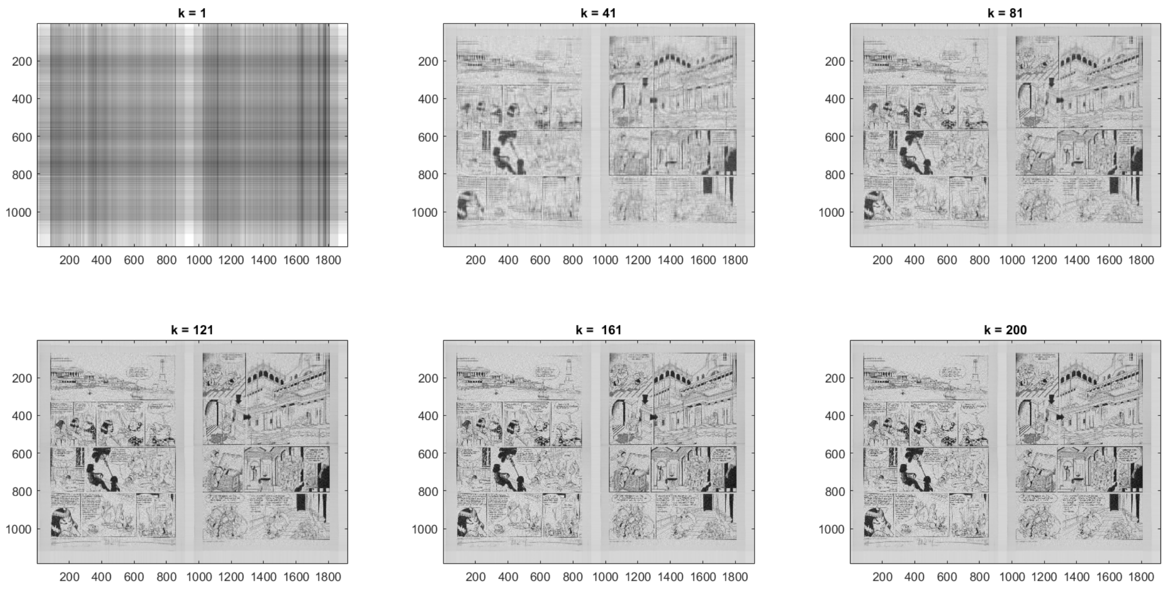

Figure 2: Reconstruction image 1.

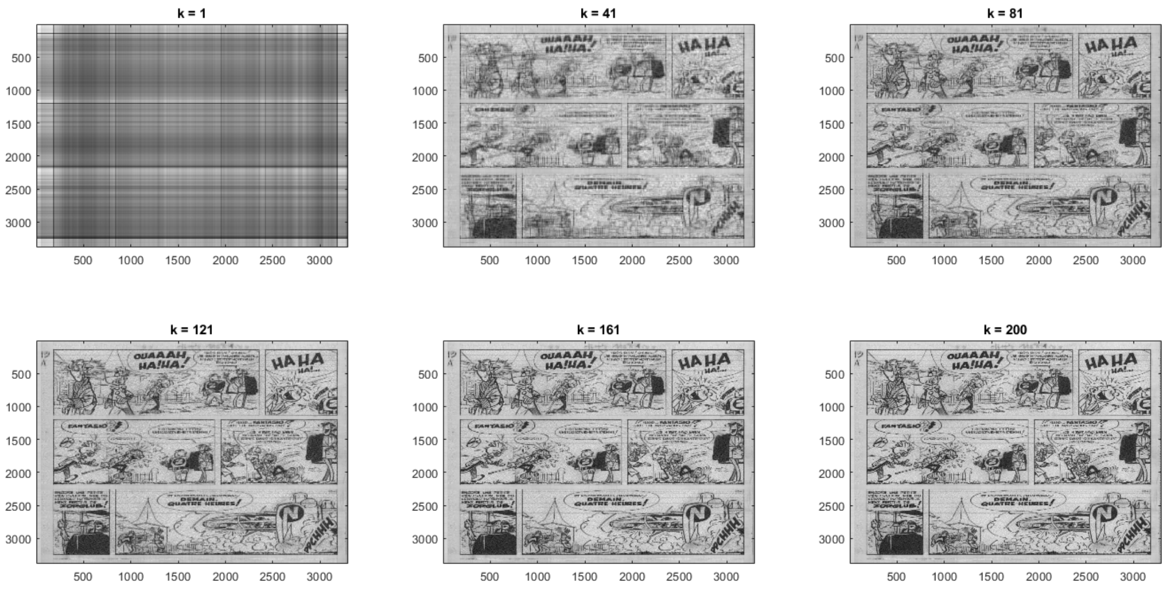

Figure 3: Reconstruction image 2.

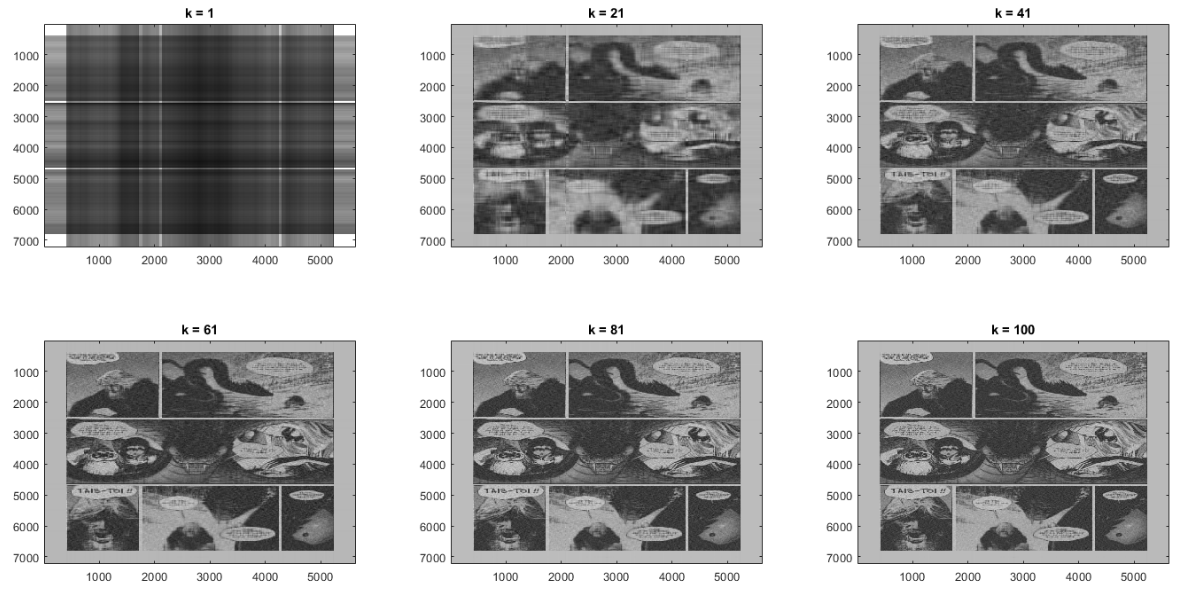

Figure 4: Reconstruction image 3.

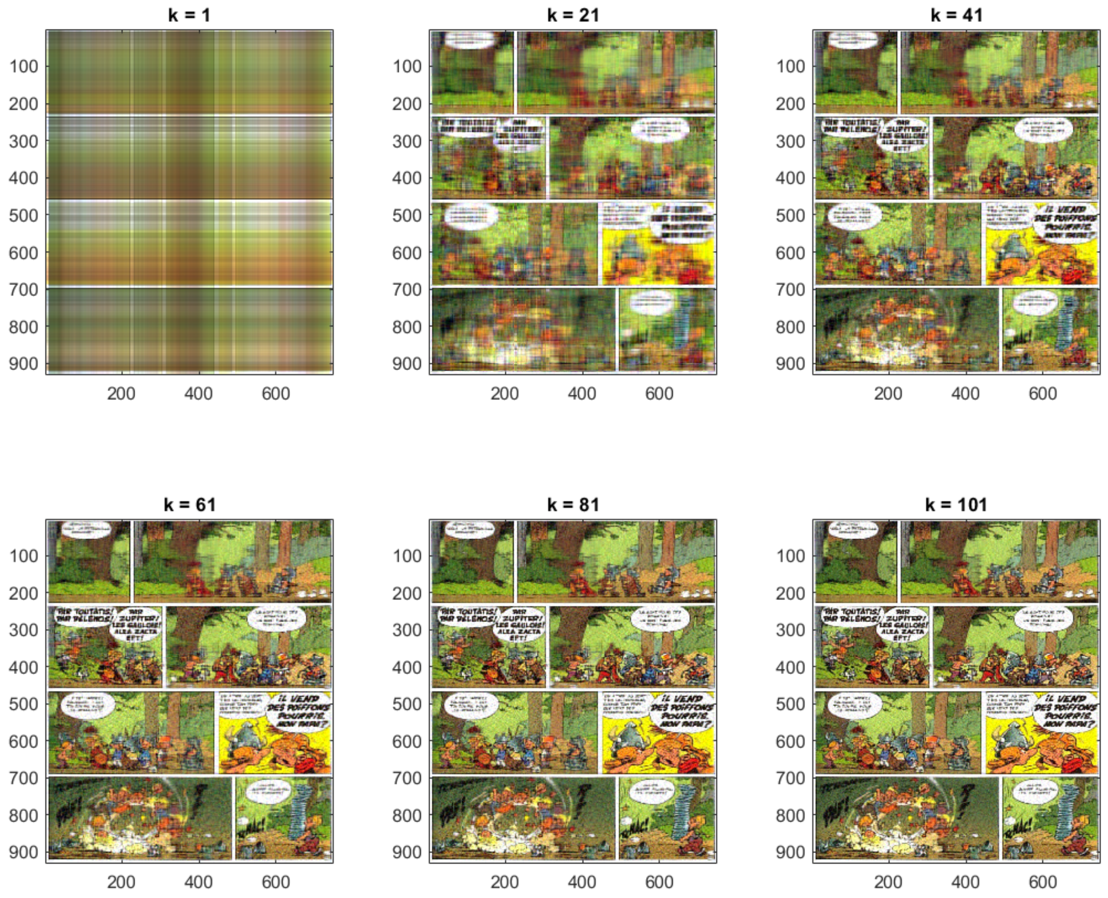

Figure 5: Reconstruction image 1 in color.