Subspace Iteration Methods

Salma Elkhatri , Khadija Akkar

April 2023

SN Department -2022-2023

Contents

Introduction

Subspace Iteration Methods are numerical algorithms used to compute a small number of eigenpairs of a matrix. They construct a subspace containing the eigenvectors of interest and iteratively refine it for accurate approximations. These methods are computationally efficient and widely used in various applications such as image and signal processing, quantum mechanics, and machine learning.

1 Limitations of the power method

Question 1 :

We first begin by comparing the execution time by changing the matrice type :

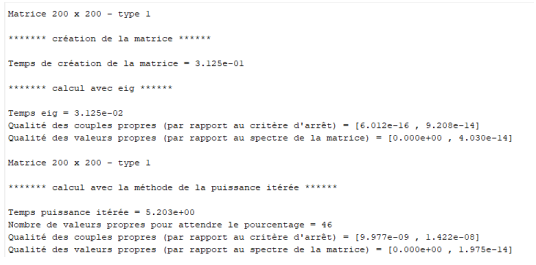

Figure 1: Matrice 200 × 200 type 1

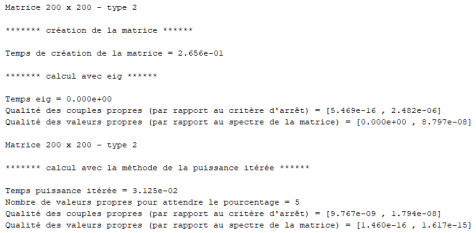

Figure 2: Matrice 200 × 200 type 2

```
Matrice 200 x 200 - type 3

******* création de la matrice ******

Temps de création de la matrice = 2.031e-01

******* calcul avec eig ******

Temps eig = 3.125e-02
Qualité des couples propres (par rapport au critère d'arrêt) = [5.880e-16 , 8.993e-11]
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 1.389e-11]

Matrice 200 x 200 - type 3

******* calcul avec la méthode de la puissance itérée ******

Temps puissance itérée = 1.562e-01
Nombre de valeurs propres pour attendre le pourcentage = 9
Qualité des couples propres (par rapport au critère d'arrêt) = [9.733e-09 , 1.487e-08]
Qualité des valeurs propres (par rapport au spectre de la matrice) = [1.176e-16 , 1.332e-15]
```

Figure 3: Matrice 200 × 200 type 3

```
Matrice 200 x 200 - type 4

******* création de la matrice ******

Temps de création de la matrice = 3.438e-01

******* calcul avec eig ******

Temps eig = 0.000e+00
Qualité des couples propres (par rapport au critère d'arrêt) = [5.979e-16 , 4.904e-14]
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 1.878e-14]

Matrice 200 x 200 - type 4

******* calcul avec la méthode de la puissance itérée ******

Temps puissance itérée = 5.297e+00
Nombre de valeurs propres pour attendre le pourcentage = 46
Qualité des couples propres (par rapport au critère d'arrêt) = [9.954e-09 , 1.422e-08]
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 1.954e-14]
```

Figure 4: Matrice 200 × 200 type 4

We observe from these various results that the eig method is rather consistent in terms of execution time across different types of matrices (in the order of hundredths of a second), and the quality of the eigenvectors is successful because it remains relatively constant for the first eigenvalue with an order of magnitude of 10^-16. However, the second eigenvalue is more variable with an order of magnitude between 10^-14 and 10^-6, especially on type 2 matrices which have the highest value.

As for the power iteration method, we can see that it never obtains all the eigenvectors. It is slower on type 4 matrices where it has an execution time of 5.09 seconds, while the other types remain in the order of magnitude of hundredths of a second. However, we can see that on type 2 and type 3 matrices, fewer than ten eigenvectors were found, compared to 46 for type 1 and type 4 matrices. Finally, the quality of the eigenvectors remains constant with values of 10^-9 and 10^-8.

In conclusion, the power iteration method is more effective on type 1 matrices, but it remains consistent in the quality of the eigenvectors provided, even if it provides few. The eig method, on the other hand, is more stable in terms of execution time to provide all the eigenvectors, but the quality of the eigenvectors is more variable than that of the power iteration method. The eig method is more effective on type 1 and type 4 matrices.
We now compare the execution time by changing the size of the matrice:
We can see that regardless of the size of the matrix, the eig method calculates

```
Matrice 10 x 10 - type 1

******* création de la matrice ******

Temps de création de la matrice = 3.125e-02

******* calcul avec eig ******

Temps eig = 0.000e+00
Qualité des couples propres (par rapport au critère d'arrêt) = [3.675e-16 , 4.939e-15]
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 3.331e-15]

Matrice 10 x 10 - type 1

******* calcul avec la méthode de la puissance itérée ******

Temps puissance itérée = 3.125e-02
Nombre de valeurs propres pour attendre le pourcentage = 3
Qualité des couples propres (par rapport au critère d'arrêt) = [9.512e-09 , 1.555e-08]
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 8.882e-16]
```

Figure 5: Matrice 10×10 type 1

```
Matrice 100 x 100 - type 1

******* création de la matrice ******

Temps de création de la matrice = 3.125e-02

******* calcul avec eig ******

Temps eig = 0.000e+00
Qualité des couples propres (par rapport au critère d'arrêt) = [6.552e-16 , 5.975e-14]
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 1.743e-14]

Matrice 100 x 100 - type 1

******* calcul avec la méthode de la puissance itérée ******

Temps puissance itérée = 2.500e-01
Nombre de valeurs propres pour attendre le pourcentage = 23
Qualité des couples propres (par rapport au critère d'arrêt) = [9.941e-09 , 1.428e-08]
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 9.948e-15]
```

Figure 6: Matrice 100×100 type 1

```
Matrice 200 x 200 - type 1

******* création de la matrice ******

Temps de création de la matrice = 3.125e-01

******* calcul avec eig ******

Temps eig = 3.125e-02
Qualité des couples propres (par rapport au critère d'arrêt) = [6.012e-16 , 9.208e-14]
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 4.030e-14]

Matrice 200 x 200 - type 1

******* calcul avec la méthode de la puissance itérée ******

Temps puissance itérée = 5.203e+00
Nombre de valeurs propres pour attendre le pourcentage = 46
Qualité des couples propres (par rapport au critère d'arrêt) = [9.977e-09 , 1.422e-08]
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 1.975e-14]
```

Figure 7: Matrice 200×200 type 1

```
Matrice 1000 x 1000 - type 1

******* création de la matrice ******

Temps de création de la matrice = 7.683e+01

******* calcul avec eig ******

Temps eig = 3.906e-01
Qualité des couples propres (par rapport au critère d'arrêt) = [6.010e-16 , 9.315e-13]
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 1.881e-13]

Matrice 1000 x 1000 - type 1

******* calcul avec la méthode de la puissance itérée ******
puissance_itérée : convergence non atteinte pour un des couples propres
Qualité des couples propres (par rapport au critère d'arrêt) = [0.000e+00 , 0.000e+00]
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 0.000e+00]
```

Figure 8: Matrice 1000×1000 type 1

eigenvectors and eigenvalues more quickly and accurately than the power iteration method. Additionally, the power iteration method is limited in matrix size as it fails to converge on a 1000 × 1000 size matrix of type 1. Thus, we can conclude that the eig method is faster than the power iteration algorithm regardless of the matrix size, but eig is not necessarily more accurate than the power iteration method.

Question 2

Algorithm 1 Vector power method

Input: Matrix $A \in R^{n \times n}$, $v \in R^n$
Output: (λ_1, v_1) eigenpair associated to the largest (in module) eigenvalue.
$z = A \cdot v$
$\beta = v^T \cdot z$
repeat
  $y = z$
  $v = \frac{y}{||y||}$
  $z = A \cdot v$
  $\beta_{old} = \beta$
  $\beta = v^T \cdot z$
until $|\beta - \beta_{old}|/|\beta_{old}| < \varepsilon$
$\lambda_1 = \beta$ and $v_1 = v$

Question 3

The method has several drawbacks. For matrices of the same size, the computation time for the eigenpairs can vary significantly for different matrix types. The computation time for a few eigenpairs is also higher than the computation time for all eigenvalues using the eig function. Moreover, for type 1 matrices, if we use a matrix size that is too large (ex 1000 x 1000), the method does not converge.

2 Extending the power method to compute dominant eigenspace vectors

2.1 subspace iter v0: a basic method to compute a dominant eigenspace

Question 4 :

the matrix V converges to the change of basis matrix that relates the eigenvectors of H (a matrix that has the same eigenvalues as A) to the eigenvectors of A.

Question 5 :

We have A $\in \mathrm{R}^{n*n}$ and V $\in \mathrm{R}^{n*m}$
consequently H $\in \mathrm{R}^{m*m}$. Because H = v^T.A.v
Hence, H is of a reasonable size, such as $m \ll n$. Therefore, calculating the spectral decomposition of H is not a problem.

2.2 subspace iter v1: improved version making use of Raleigh-Ritz projection

Question 7 :

Algorithm 2 Subspace iteration method v1 with Raleigh-Ritz projection

Input: Symmetric matrix $A \in R^{n \times n}$, tolerance ϵ, MaxIter (max nb of iterations) and PercentTrace the target percentage of the trace of A
Output: nev dominant eigenvectors V_{out} and the corresponding eigenvalues Λ_{out}
Generate an initial set of m orthonormal vectors $V \in R^{n \times m}$; $k = 0$; PercentReached = 0
repeat
$k = k + 1$
Compute Y such that $Y = A \cdot V$
$V \leftarrow$ 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 nev = m or $k >$ MaxIter)

3 subspace iter v2 and subspace iter v3: toward an efficient solver

3.1 Block approach (subspace iter v2)

Question 8 :

If we take A $\in \mathrm{R}^{n*m}$ and B $\in \mathrm{R}^{n*p}$, the cost in flops of the calculation A $\cdot$ B is O(2 $\cdot$m$\cdot$n$\cdot$ p). In our case, we have A $\in \mathrm{R}^{n*n}$ and we calculate A $\cdot$ A, so the cost of this multiplication is O($2 \cdot n^3$). Thus, to compute Ap, the cost is O(2p $\cdot n^3$) since we perform p times the multiplication A $\cdot$ A.
We know that V $\in \mathrm{R}^{n*m}$ with m ¡¡ n, so computing $A^p \cdot$ V amounts to performing a multiplication $R^{n*n} \cdot R^{n*m}$, so the cost of the last multiplication is
O($2 \cdot$ m $\cdot n^2$), and the total cost of the computation is: O(2p $\cdot n^3 + 2 \cdot$ m $\cdot n^2$) = O(2p $\cdot n^3$).
To reduce the cost of this calculation, we should perform the calculation of A^p once before entering the loop, and then only perform the multiplication with V inside the loop, which costs O($2 \cdot$ m $\cdot n^2$), which is less expensive since $m \ll n$.

Question 10 :

We notice that initially increasing the value of p allows our algorithm to converge more quickly for eigenvectors that require more iterations, but if we increase p too much, this effect tends to increase the number of iterations required to find the first eigenvectors and prevent convergence.
for p=1 : 57 iterations (last eigenvectors)
for p=2 : 31 iterations (last eigenvectors)
for p=3 : 21 iterations (last eigenvectors)
for p=4 : 16 iterations (last eigenvectors)
for p=10 : No convergence

3.2 Deflation method (subspace iter v3)

Question 11 :

The precision of the eigenvectors varies depending on their eigenvalue: the larger the eigenvalue, the easier it is to obtain a high precision for the corresponding eigenvector. This can be observed in the convergence criterion for an eigenvector, which is given by $||AV_j - \Lambda_j V_j||/||A|| < \epsilon$.

Question 12 :

The convergence of the eigenfaces should certainly be faster but less precise, as we only aim to refine the yet undetermined eigenvectors.

4 Numerical experiments

Question 14 :

The 4 types of matrices we have chosen to use for our tests have eigenvalues that are respectively incremented from 1 to n, random with uniformly distributed logarithms, and finally uniformly distributed between 0 and 1.

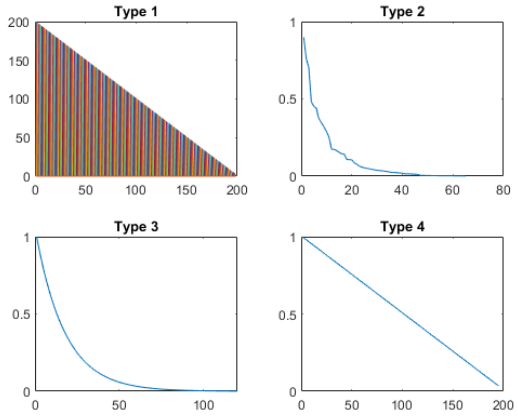

Figure 9: Eigenvalues for the different types of matrices

Question 15 :

For type 1 matrices, the subspace_iter_v2 method appears to be the best combination in terms of both time and accuracy it provides. (even though version v3 is slightly faster, v2 remains much more accurate for the last eigenpairs)
For type 2 matrices, the subspace_iter_v3 method seems to be the best combination in terms of both time and accuracy it provides (v1 is too slow).
For type 3 matrices, the subspace_iter_v1 method appears to be the best combination in terms of both time and accuracy it provides (v2 is too slow compared to the accuracy it offers).
For type 4 matrices, the subspace_iter_v2 method appears to be the best combination in terms of time and accuracy for large matrices, while the subspace_iter_v1 method is preferred for small matrices due to the time savings it provides (with version v3 not being accurate enough for the time it saves).

5 Application to image compression

Introduction:

Subspace Iteration Methods are a powerful technique used in image compression to reduce the storage requirements of digital images while maintaining

reasonable image quality. By leveraging the redundancies or patterns present in images, Subspace Iteration Methods approximate the original image with a lower-dimensional subspace, resulting in a reduced number of coefficients needed for representation. This leads to more efficient storage and transmission of images in applications with limited storage space or bandwidth.

Question 1 :

- Σ_k: Σ_k is a rectangular matrix of size q x p, where q is the number of rows in the original image I and p is the number of columns in I.
- U_k: U_k is a matrix of size q x k, where q is the number of rows in I and k is the rank of the low-rank approximation. U_k is formed of q orthonormal eigenvectors associated with the q eigenvalues of AA^T, as mentioned in the theorem.
- V_k: V_k is a matrix of size p x k, where p is the number of columns in I and k is the rank of the low-rank approximation. V_k is formed of p orthonormal eigenvectors associated with the p eigenvalues of A^TA, as mentioned in the theorem.

Question 2 :

Images reconstructed

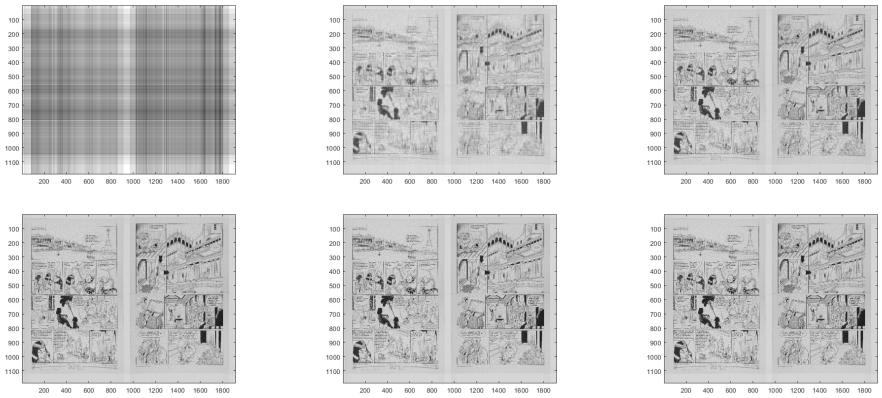

Figure 10: Images reconstructed

The difference between I and I_k

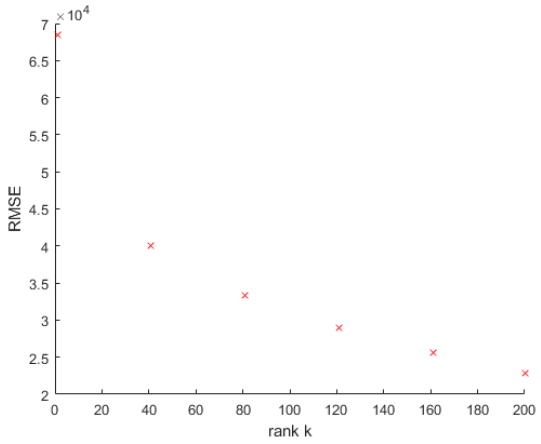

Figure 11: The difference between I and I_k

The results obtained from the experiments demonstrate the effectiveness of the subspace iteration method for image compression using Singular Value Decomposition (SVD). The best low-rank approximation theorem is proven to be a valuable technique for reducing the dimensionality of the image while preserving its essential features. The reconstructed images using different functions, such as eig, power method, and the four versions of the subspace iteration method, show varying levels of image quality in terms of visual appearance and difference calculation ($\|I - I_k\|$). The eigenvalue-based subspace iteration method (Subspace Iteration Method 4) outperforms the other methods in terms of image reconstruction, achieving the lowest difference value ($\|I - I_k\|$) among all the functions used. However, it is worth noting that the computational time for the subspace iteration method is higher compared to the other methods, which may be a trade-off for its improved performance. Further investigation could explore optimization techniques to reduce the computational time while maintaining the image quality. Overall, the results highlight the potential of the subspace iteration method with SVD for image compression, and provide valuable insights for future research in this area.