Calcul Scientifique

Projet de Calcul Scientifique

Élèves :

THEVENET Louis

SABLAYROLLES Guillaume

Décembre 2023

1.

Matrix dimension	Matrix type	Exec. time for `eig` (s)	Exec. time for `power_v11`, (s)
200×200	Type 1	9.000e-02	1.510e+00
400×400	Type 1	4.000e-02	1.831e+01
600×600	Type 1	6.000e-02	6.021e+01
200×200	Type 2	3.000e-02	3.000e-02
400×400	Type 2	4.000e-02	4.000e-02
600×600	Type 2	7.000e-02	1.700e-01
200×200	Type 3	1.000e-02	5.000e-02
400×400	Type 3	3.000e-02	5.200e-01
600×600	Type 3	7.000e-02	1.270e+00
200×200	Type 4	2.000e-02	1.670e+00
400×400	Type 4	3.000e-02	2.094e+01
600×600	Type 4	6.000e-02	5.456e+01

Table 1: Execution time for different sizes and types of matrices

We can see that the `power_v11` algorithm is generally slower than the `eigen` function especially for the type 2 and 4 matrices.

2.

```
nb_it = 1;
norme = norm(beta*v - z, 2)/norm(beta,2);

while(norme > eps && nb_it < maxit)
    beta_old = beta;
    v = z/norm(z, 2);
    z = A*v;
    beta = (v'*z)/(v'*v);
    norme = abs(beta-beta_old)/abs(beta_old);
    nb_it = nb_it + 1;
end
```

Listing 1: Inner loop of the new algorithm

Matrix dimension	Matrix type	Exec. time for `power_v11`, (s)	Exec. time for `power_v12`, (s)
200×200	Type 1	1.960e+00	3.200e-01
400×400	Type 1	1.888e+01	2.660e+00
600×600	Type 1	5.031e+01	7.070e+00
200×200	Type 2	1.000e-02	1.000e-02
400×400	Type 2	7.000e-02	1.000e-02
600×600	Type 2	1.800e-01	4.000e-02
200×200	Type 3	3.000e-02	1.000e-02
400×400	Type 3	6.100e-01	1.100e-01
600×600	Type 3	1.270e+00	2.600e-01
200×200	Type 4	1.530e+00	2.900e-01

400 × 400	Type 4	2.113e+01	3.060e+00
600 × 600	Type 4	5.914+e01	6.480e+00

We can see that the `power_v12` algorithm is globally faster than the `power_v11`.

3.

The main drawback of the deflated power method is the numerous matrix-vector products required to compute the eigenvectors as well as the fact that each iteration compute only one eigenvalue which can be slow if a lot of eigenvalues are desired.

4.

If we apply Algorithm 1 to m vectors, there is no reason for the columns of V to converge to a base. Each vector will converge toward a different projection of the dominant eigenvalue.

5.

In Algorithm 2, the matrix H is a smaller matrix, with dimension $n \times m$, therefore, even for larger matrices A, computing the spectral decomposition of H will not be computionally expensive.

6.

7.

function Subspace iter v1 (Raleigh-Ritz projection)
  Input : $A \in \mathbb{R}^{n\times n}, \varepsilon, \text{MaxIter}, \text{PercentTrace}$
  Output : n_{ev} dominant eigenvectors V_{out} and the corresponding eigenvalues Λ_{out}
  Generate an initial set of m orthonormal vectors $V \in \mathbb{R}^{n\times m}$; $k = 0$; $\text{PercentReached} = 0$
  repeat until $\text{PercentReached} > \text{PercentTrace} \vee n_{\text{ev}} = m \vee k > \text{MaxIter}$
    $k \leftarrow k + 1$
    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 converged and update *PercentReached*

V_k is of size (p, k)

7.1.

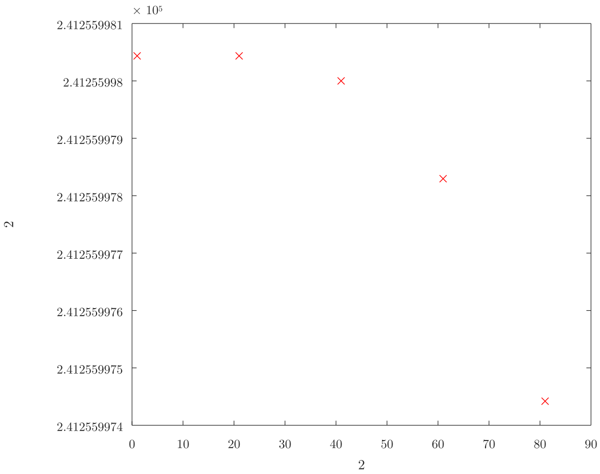

Figure 1: eig method

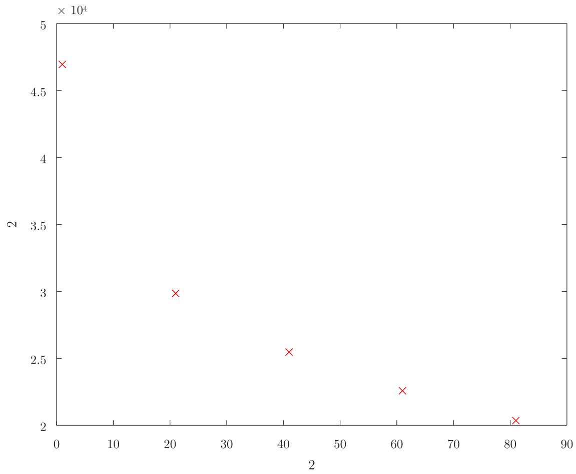

Figure 2: power method

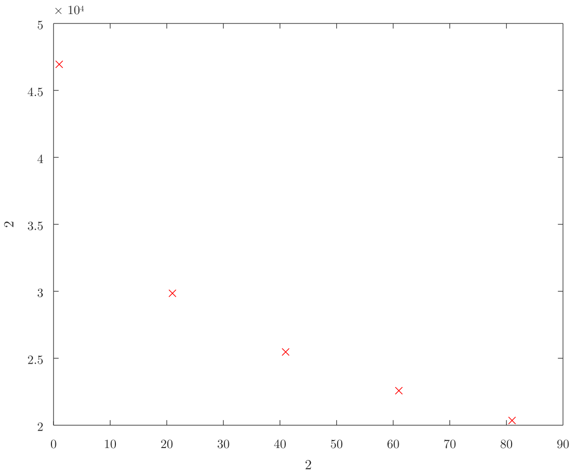

Figure 3: subspace_iter0 method

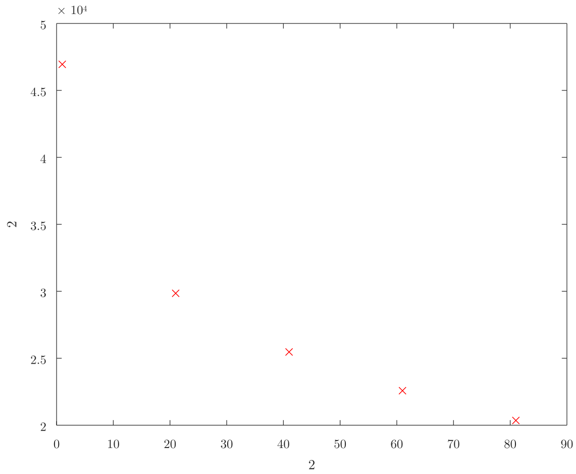

Figure 4: subspace_iter1 method

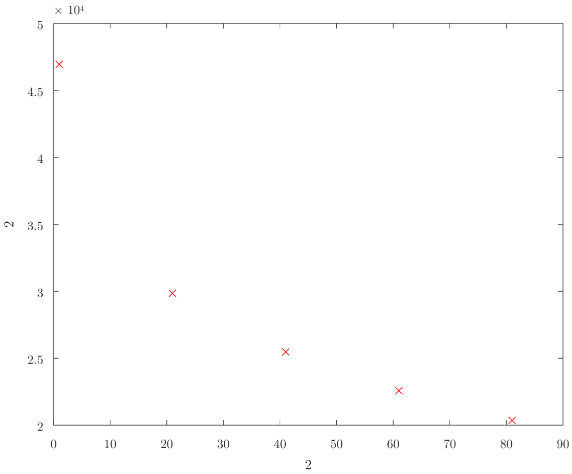

Figure 5: subspace_iter2 method