Calcul Scientifique

Projet de Calcul Scientifique

Élèves :

THEVENET Louis

SABLAYROLLES Guillaume

Décembre 2023

1.

1.1.

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.

1.2.

```
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`.

1.3.

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.

1.4.

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.

1.5.

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.

1.6.

1.7.

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 **(line 52)** $\text{PercentReached} > \text{PercentTrace} \vee n_{\text{ev}} = m \vee k > \text{MaxIter}$
    $k \leftarrow k + 1$
    Compute Y such that $Y = A \cdot V$
    $V \leftarrow$ orthonormalisation of the columns of Y
    Rayleigh-Ritz projection **(line 60)** applied on matrix A and orthonormal vectors V
    Convergence analysis step **(line 70)** : save eigenpairs that converged and update *PercentReached*

1.8.

We can precompute the product A^p

1.9.

1.10.

Matrix dimension	Matrix type	Flops for `subspace_iter0`	Flops for `subspace_iter1`	Flops for `subspace_iter2`	p (A^p)
200×200	Type 1	2309	263	132	2
200×200	Type 1	2309	263	88	3
200×200	Type 1	2309	263	53	5
200×200	Type 1	2309	263	27	10

Table 2: Before precomputing

C'EST FAUX

Matrix dimension	Matrix type	Flops for `subspace_iter0`	Flops for `subspace_iter1`	Flops for `subspace_iter2`	p (A^p)
200×200	Type 1	2309	263	132	2
200×200	Type 1	2309	263	88	3
200×200	Type 1	2309	263	53	5
200×200	Type 1	2309	263	27	10

Table 3: After precomputing

When increasing the value of p to compute A^p in `subspace_iter2`, the number of flops to compute the results is : Flops(`iter2`) $\simeq \frac{\text{Flops}(\texttt{iter2})}{p}$.

1.11.

The accuracy differs because eigenpairs are computed from the columns of new matrix V. However, the firsts columns are recalculated at each step, that will lead to have different approximate size for the new eigenpairs computed. By recomputing them, the quality reduces and diverge from the approximate size of the first one.

1.12.

By freezing the converged columns, the algorithm will not have to recalculate them everytime. Which means that the accuracy for the eigenpairs will be more equal. The first and last will have the same approximate size.

2.

2.1.

Σ_k is of size (k, k)

U_k is of size (q, k)

V_k is of size (p, k)

2.2.

MANQUE DES COMMENTAIRES

eps	10^{-8}
maxit	10000
search_space	400
percentage	0.995
puiss	1

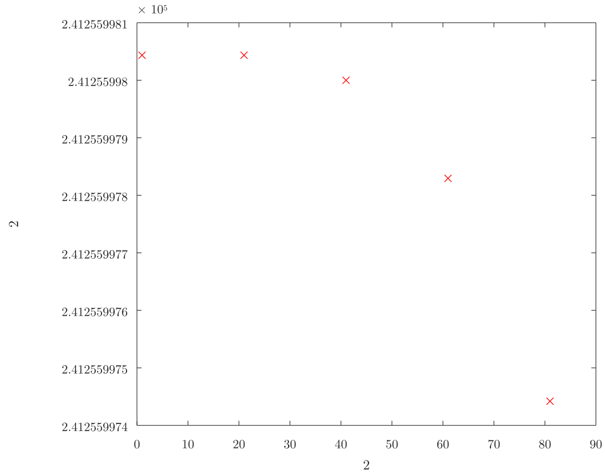

Figure 1: eig method

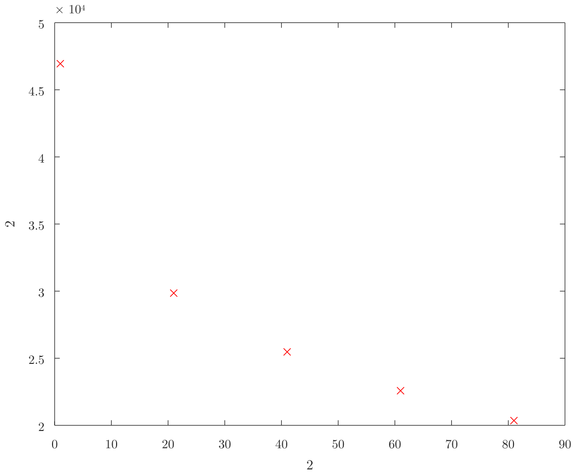

Figure 2: power method

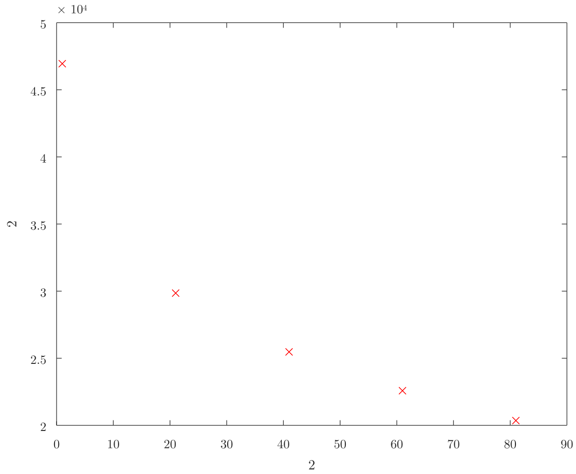

Figure 3: subspace_iter0 method

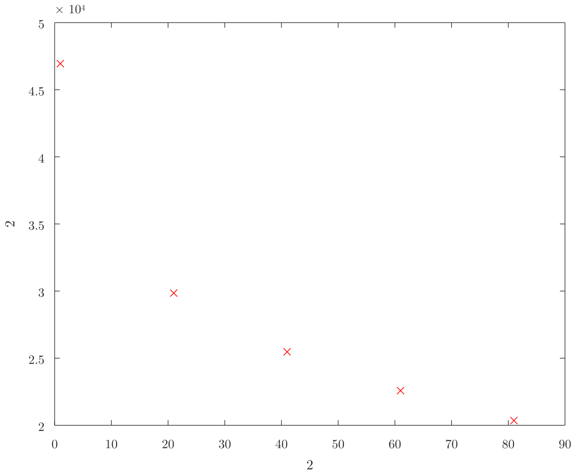

Figure 4: subspace_iter1 method

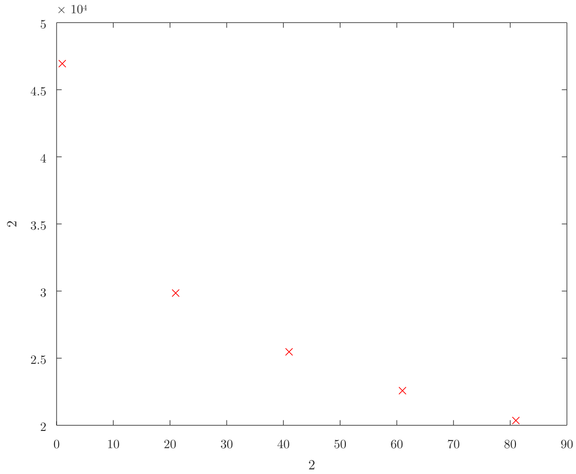

Figure 5: subspace_iter2 method

eps	10^{-8}
maxit	7500
search_space	600
percentage	0.995
puiss	2

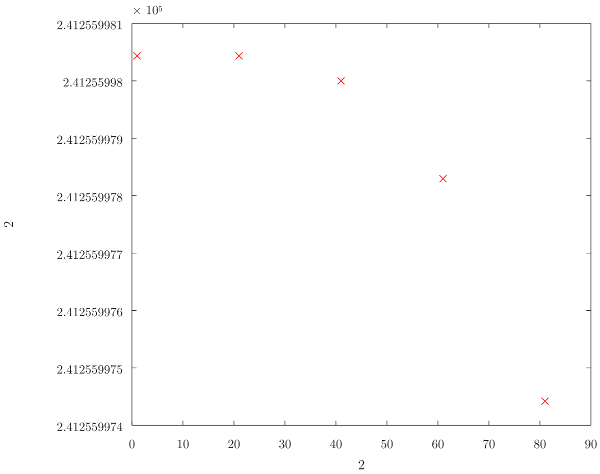

Figure 6: eig method

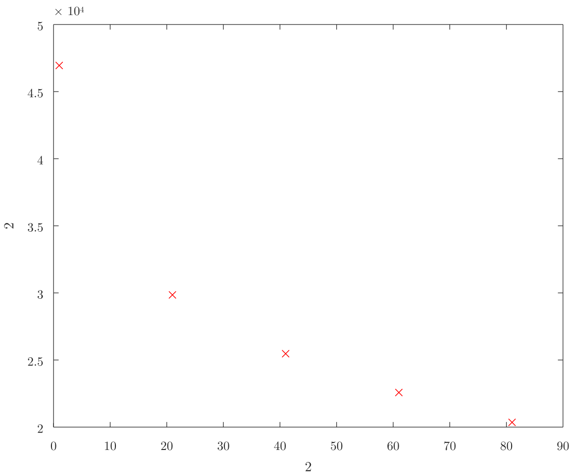

Figure 7: power method

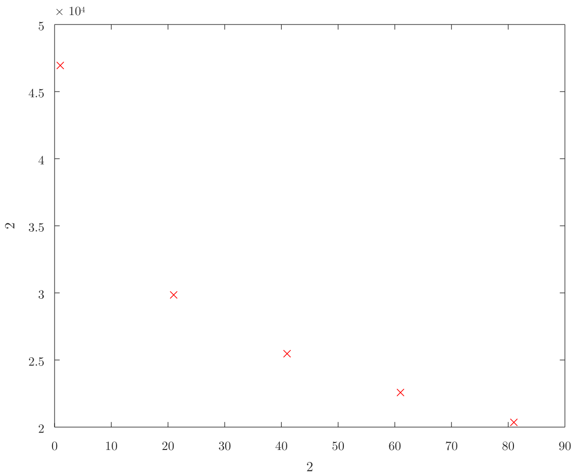

Figure 8: subspace_iter0 method

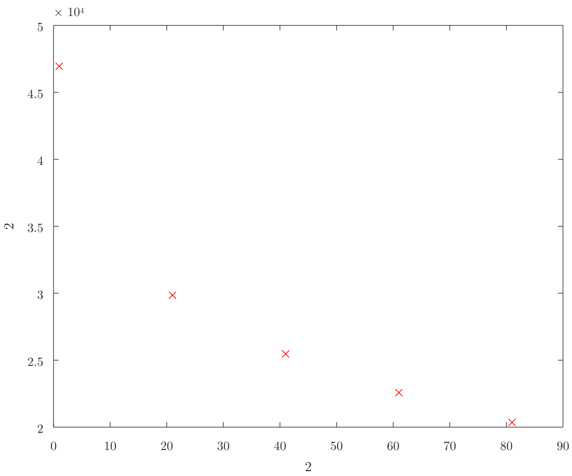

Figure 9: subspace_iter1 method

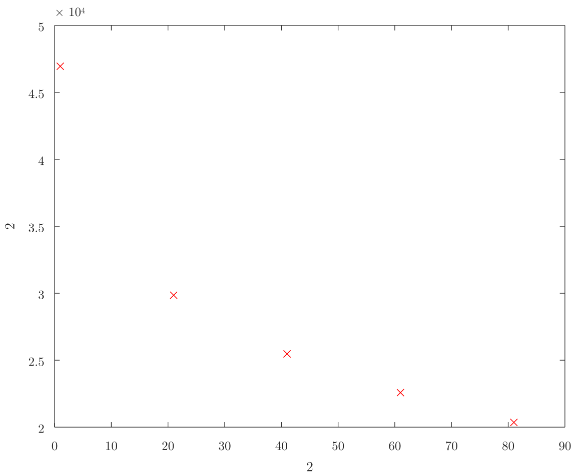

Figure 10: subspace_iter2 method

eps	10^{-8}
maxit	3000
search_space	500
percentage	0.995
puiss	1

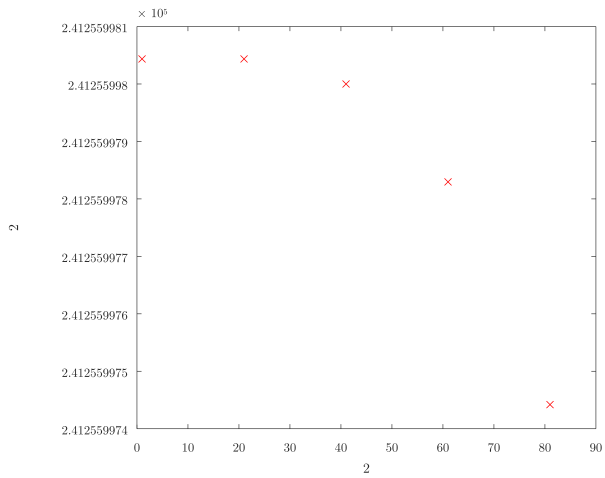

Figure 11: eig method

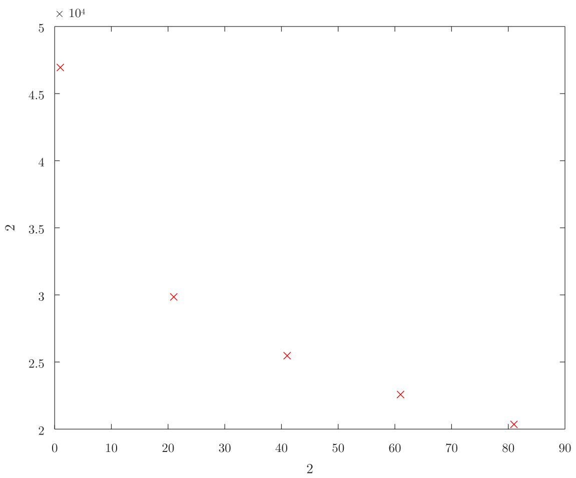

Figure 12: power method

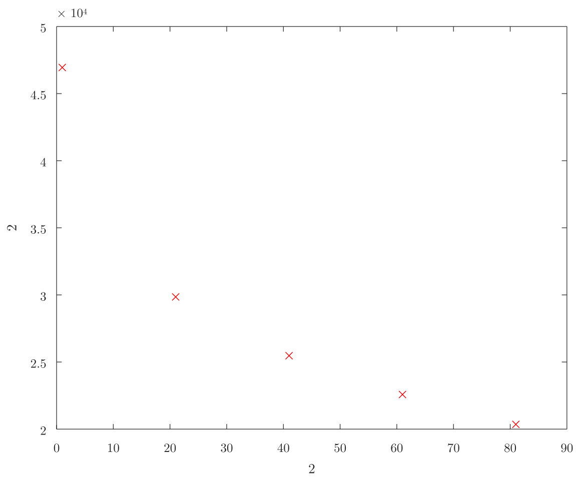

Figure 13: subspace_iter0 method

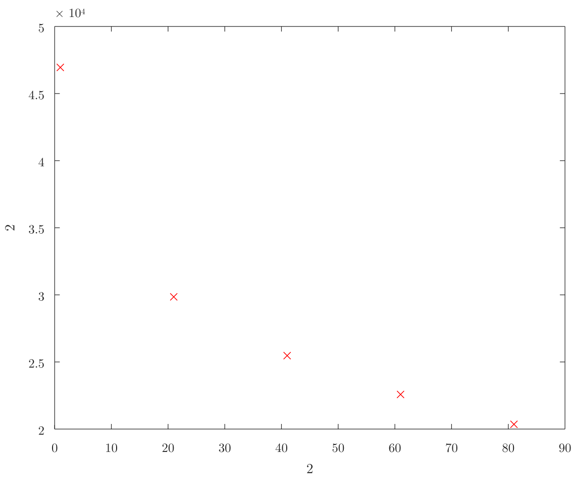

Figure 14: subspace_iter1 method

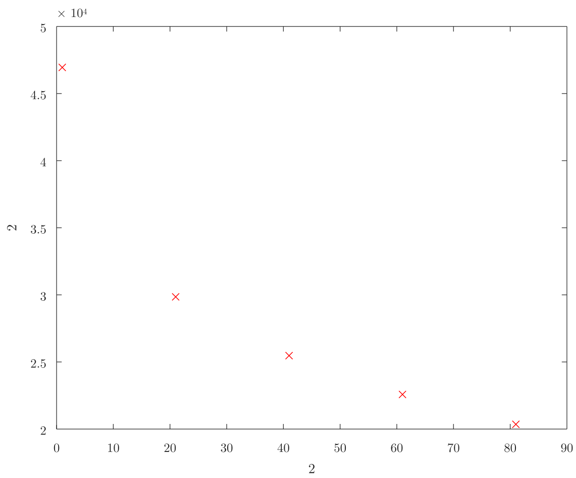

Figure 15: subspace_iter2 method