## Département d'informatique

Master 1: M1 ISC

Master Ingénierie des Systèmes Complexes

# RAPPORT D'APPLICATION DES MÉTHODES REDÉMARRÉES ET PONDÉRÉES DE FOM ET GMRES SUR DIVERS SYSTÈMES LINÉAIRES SUR LA BASE DU PROCESSUS D'ORTHONORMALISATION D'ARNOLDI


**Réalisé par :**

- Salim ISSA
- Abdelouahed AIT RAHOU
- Fabienne JANVIER

**Enseignant**:

- Mr Mohammed HEYOUNI


**Module :** Calcul Numérique et Formel

**Année Universitaire : 2024-2025**

# Table des matières

Introduction....3
Analyse des exemples de la fiche de TD....4
Matrice de l’Exemple 1:....4
Matrice de l’exemple 2....6
Matrices spécifiques....8
Matrice Saad....8
Matrice bcsstk16....10
Matrice bcsstm22....11
Matrice fs_541_2....22
Matrice bfw782a....23
Matrice memplus....23
Matrice de Poisson....24
Observations globales....27
Conclusion....27
Références....27

# Introduction

Les systèmes linéaires servent de nos jours non seulement à modéliser mais aussi à résoudre numériquement bon nombre de problèmes dans un nombre incalculable de domaines. Pour y parvenir, diverses méthodes ont été proposées par de brillants chercheurs qu'on classe généralement en deux catégories : les méthodes directes (qui trouvent la solution exacte en un nombre fini d'opérations) et les méthodes itératives (qui calculent une solution approchée du système selon la marge d'erreur autorisée). Ces dernières regroupent beaucoup de techniques dont par exemple, les méthodes de Jacobi, de Gauss-Seidel, du gradient conjugué, bi-conjugué et bien d'autres ; toutes se valent compte tenu du type de système qu'on a à résoudre c'est à dire le type de la matrice principale du système. Dans la suite, nous nous intéresserons aux méthodes itératives GMRES (Generalized Minimal RESidual : elle approche la solution par un vecteur d'un sous-espace de Krylov qu'on trouve par la méthode itérative d'Arnoldi) et FOM (Full Orthogonalization Method : basée sur la méthode d'Arnoldi). Elles sont de loin les plus utilisées actuellement pour la résolution des systèmes linéaires. Nous construirons des matrices ayant des propriétés particulières ou nous en récupérerons sur des sites de référence comme MatrixMarket pour tester sur des systèmes linéaires générés à partir de ces matrices, les différentes méthodes étudiées à savoir rFOM (FOM redémarrée), rGMRES (GMRES redémarrée), rwFOM (FOM redémarrée et pondérée), rwGMRES (GMRES redémarrée et pondérée). On observera alors les résultats obtenus pour enfin discuter des performances de ces différents algorithmes. Sauf pour des cas particuliers, la tolérance est fixée à 10-10

# Analyse des exemples de la fiche de TD

## Matrice de l'Exemple 1:

Il s'agit d'une matrice triangulaire supérieure dont les valeurs dépendent de deux paramètres : alpha et beta. Ci-dessous figurent les résultats obtenus pour divers paramètres de test

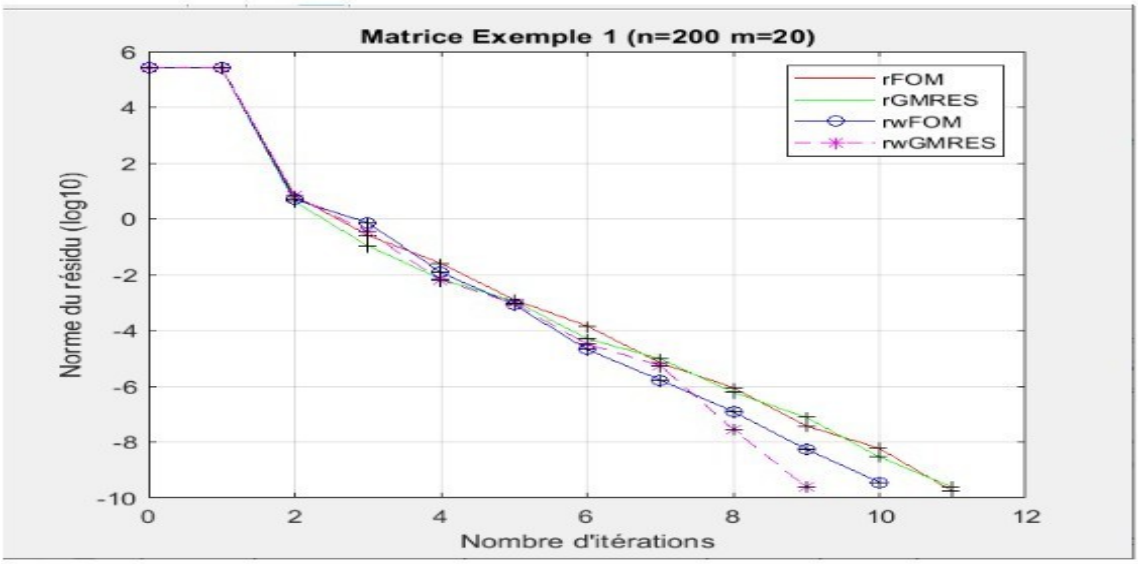



Figure 1: Matrice Exemple 1

alpha = 1.000000 beta = 0.900000

Matrice Exemple 1

| | rFOM | rwFOM | rGMRES | rwGMRES | |
|---|---|---|---|---|---|
| cycle | 11 | 10 | 11 | 9 | |
| résidu | 1.712095e-10 | 3.610203e-10 | 2.373441e-10 | 2.427691e-10 | |
| erreur | 1.800616e-12 | 1.171112e-11 | 5.095687e-12 | 8.104222e-12 | |
| t_CPU | 1.854820e-02 | 2.905560e-02 | 1.341010e-02 | 1.905780e-02 | |

Table 1: Matrice Exemple 1

- m=50,n=500,beta=0.9,alpha=0

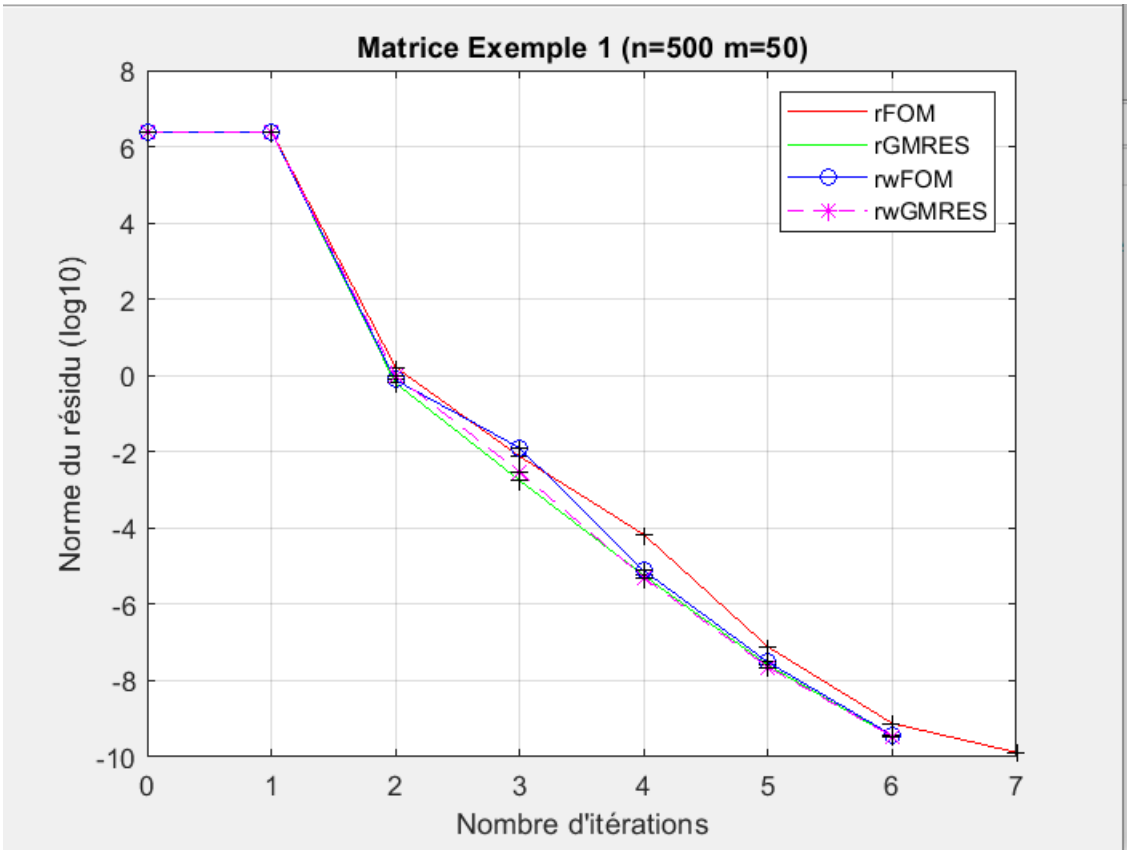



Figure 2: Matrice Exemple 1

```
alpha = 0.000000    beta = 0.900000

Matrice Exemple 1
          rFOM          rwFOM          rGMRES          rwGMRES
cycle        7              6               6                6
résidu       1.317531e-10   3.674358e-10    3.421521e-10     3.455810e-10
erreur       1.094428e-13   2.759200e-13    3.398565e-13     5.021266e-13
t_CPU        4.893390e-02   1.409406e-01    3.305510e-02     1.192847e-01
```

Table 2 : Matrice Exemple 1 avec n=500

## Matrice de l'exemple 2

C'est une matrice tridiagonale ayant respectivement -1 et 1 sur les diagonales inférieures et supérieures puis un paramètre epsilon sur la diagonale principale.

- Stockage en mode 'sparse'
  - m=20,n=40,epsilon= 0.1,redémarrage=2000

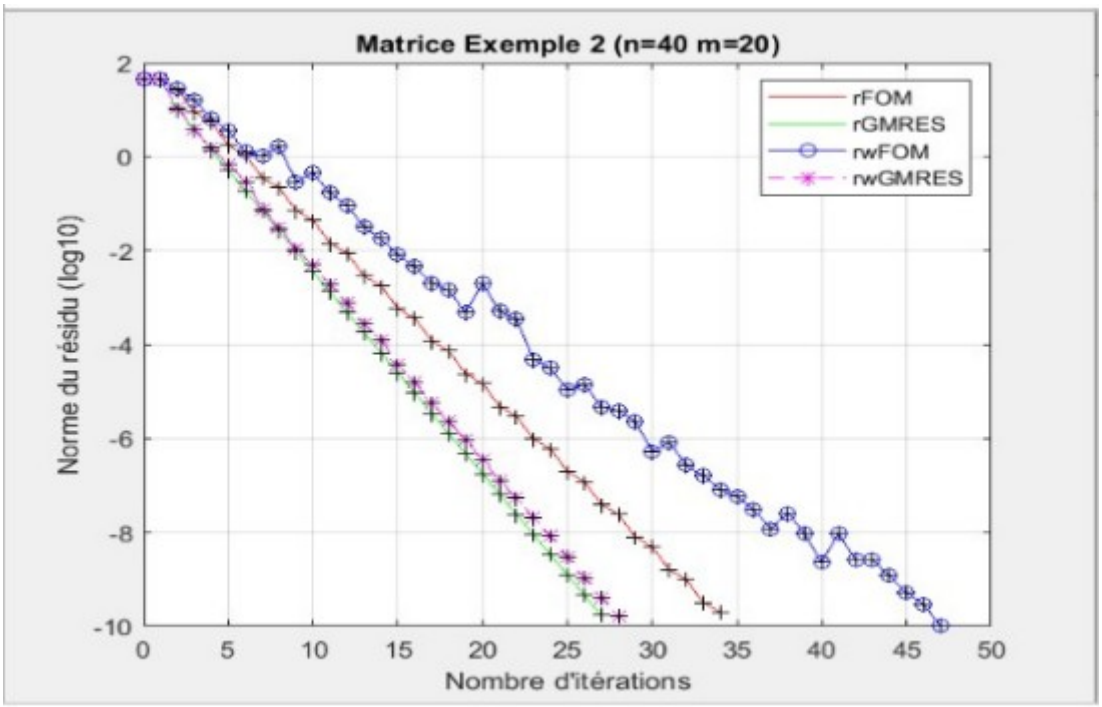



Figure 3 : Matrice Exemple2 avec n=40

Matrice Exemple 2

| | rFOM | rwFOM | rGMRES | rwGMRES |
|---|---|---|---|---|
| cycle | 34 | 47 | 27 | 28 |
| résidu | 1.996676e-10 | 1.001918e-10 | 1.692234e-10 | 1.614270e-10 |
| erreur | 2.375887e-10 | 5.987061e-11 | 4.130442e-10 | 3.296287e-10 |
| t_CPU | 1.742610e-02 | 3.541410e-02 | 8.004700e-03 | 1.790000e-02 |

Table 3 : Matrice Exemple2 avec n=40

- Stockage en mode 'full'
  - m=50,n=100,epsilon=0.5,redémarrage=1500

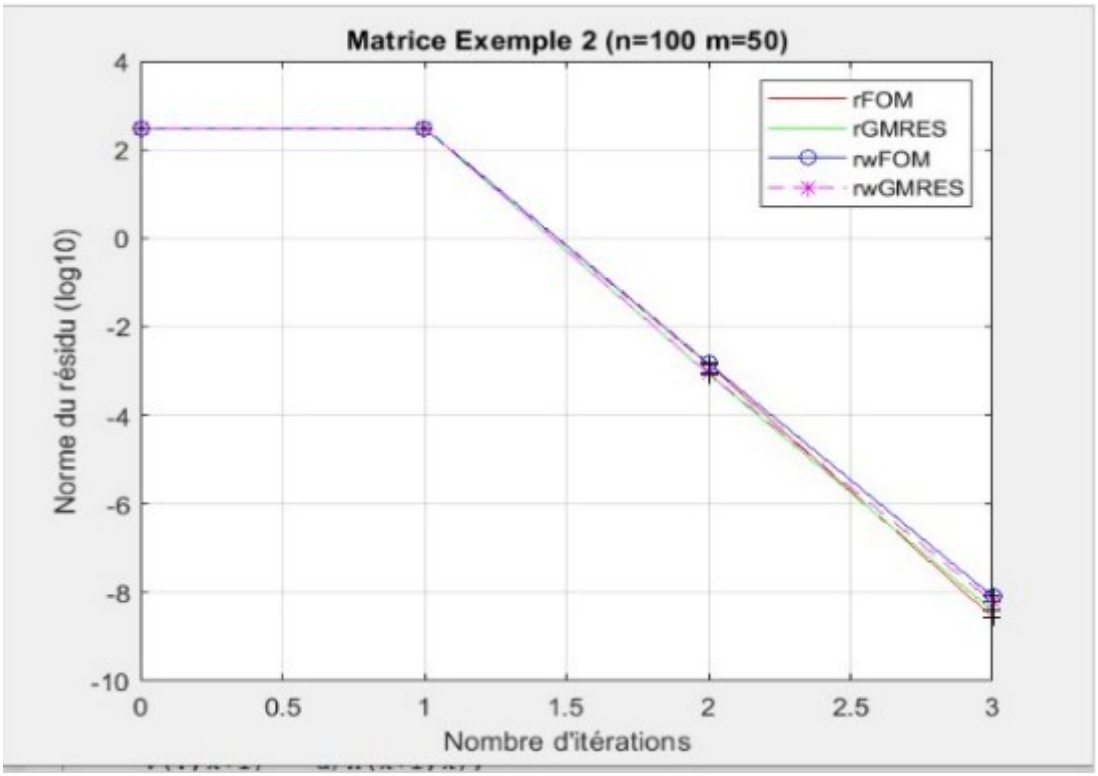



Figure 4 : Matrice Exemple 2 avec n=100

Matrice Exemple 2

| | rFOM | rwFOM | rGMRES | rwGMRES |
|---|---|---|---|---|
| cycle | 3 | 3 | 3 | 3 |
| résidu | 2.733854e-09 | 8.189053e-09 | 3.837369e-09 | 6.203634e-09 |
| erreur | 1.757522e-14 | 3.371570e-14 | 3.334017e-14 | 3.645929e-14 |
| t_CPU | 1.322200e-02 | 2.471920e-02 | 7.876800e-03 | 1.696930e-02 |

Table 4 : Matrice Exemple2 avec n=100

# Matrices spécifiques

## Matrice Saad

Les matrices associées à Saad sont souvent utilisées pour tester et évaluer les méthodes itératives, comme GMRES, CG (Conjugate Gradient), ou d'autres méthodes de sousespaces de Krylov. Ces méthodes sont essentielles pour résoudre efficacement des systèmes d'équations linéaires de grande taille. Elle est creuse, c'est-à-dire qu'elle contient un grand nombre d'éléments nuls. Les matrices creuses sont courantes dans les problèmes de calcul scientifique et d'ingénierie, où elles représentent des systèmes avec des interactions locales. Les matrices utilisées dans les recherches de Saad ont des propriétés spécifiques qui les rendent intéressantes pour tester la performance des algorithmes, comme des spectres de valeurs propres particuliers ou des structures de bande.

- Stockage en mode ‘sparse’
    - m=10,p=20,q=50,delta=0.5,redémarrage=1500

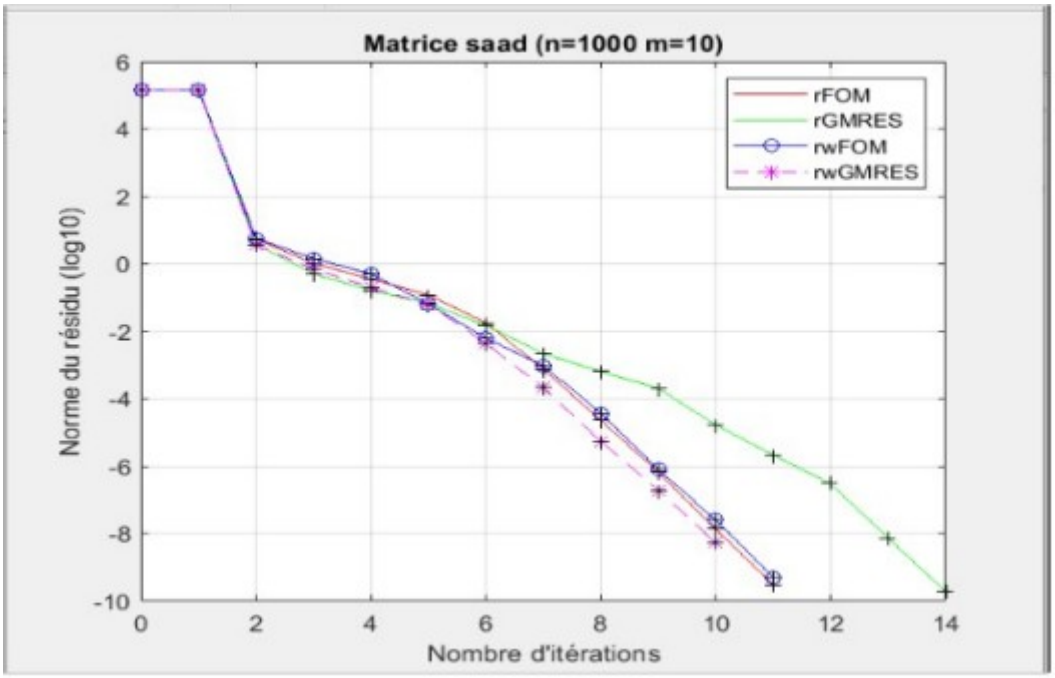



Figure 5 : Matrice saad avec n= 1000

Matrice saad

| | rFOM | rwFOM | rGMRES | rwGMRES | |
|---|---|---|---|---|---|
| cycle | 11 | 11 | 14 | 10 | |
| résidu | 3.057764e-10 | | 5.052531e-10 | 1.881608e-10 | 5.296179e-09 |
| erreur | 1.033056e-11 | | 2.010343e-11 | 1.916014e-11 | 1.427324e-10 |
| t_CPU | 2.644090e-02 | | 4.141380e-02 | 1.285740e-02 | 2.813360e-02 |

Table 5 :Matrice saad avec n=1000

- m=20,p=50,q=50,delta=10,redémarrage=1000

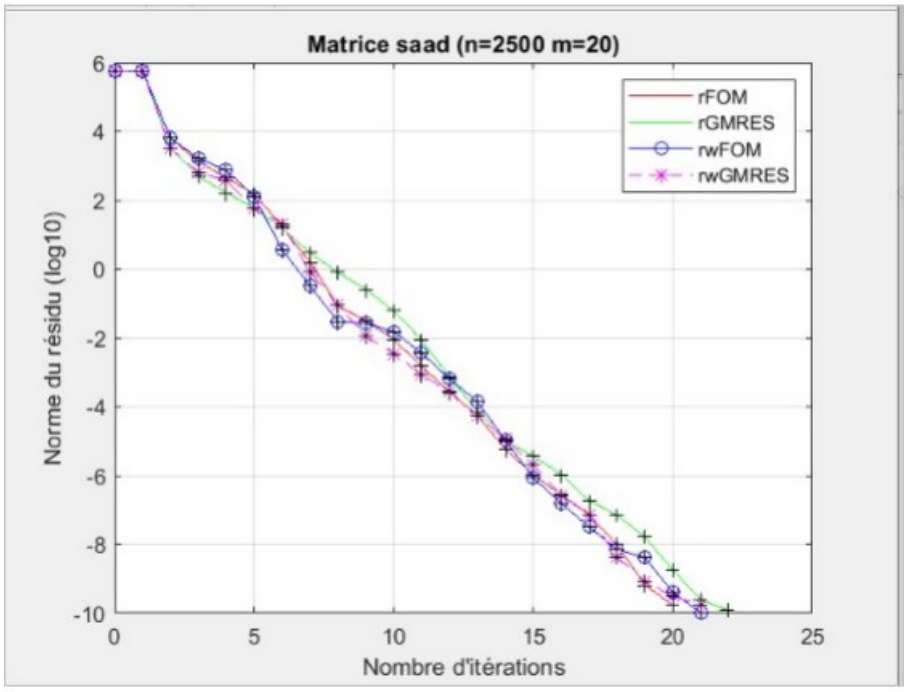


Figure 6 :Matrice saad avec n=2500

| Matrice saad | | | | |
|---|---|---|---|---|
| | rFOM | rwFOM | rGMRES | rwGMRES | |
| cycle | 20 | 21 | 22 | 21 | |
| résidu | 1.716314e-10 | | 1.018848e-10 | 1.206254e-10 | 1.725795e-10 |
| erreur | 8.596619e-12 | | 9.058022e-12 | 1.330455e-11 | 1.811188e-11 |
| t_CPU | 6.159230e-02 | | 2.634594e-01 | 5.947500e-02 | 2.057440e-01 |

Table 6 : Matrice saad avec n=2500

***À partir de la matrice bcsstk16 et jusqu'au numéro XI, nous testons des matrices issus de 'MatrixMarket'.***

## Matrice bcsstk16

Elle est d'ordre 4884, creuse à 98,78 % et provient d'un ensemble de matrices d'ingénierie structurelle de l'unité militaire des États-Unis. Pour m = 50, on voit qu'aucune des méthodes ne converge même après 2000 itérations.

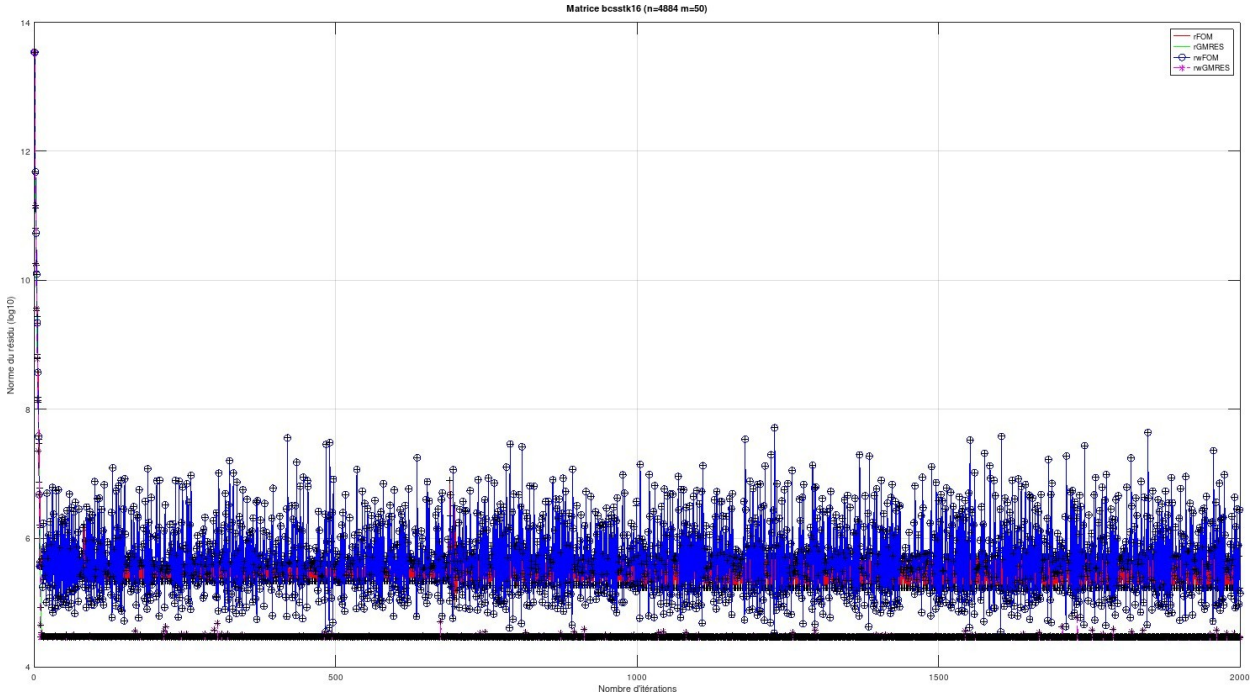


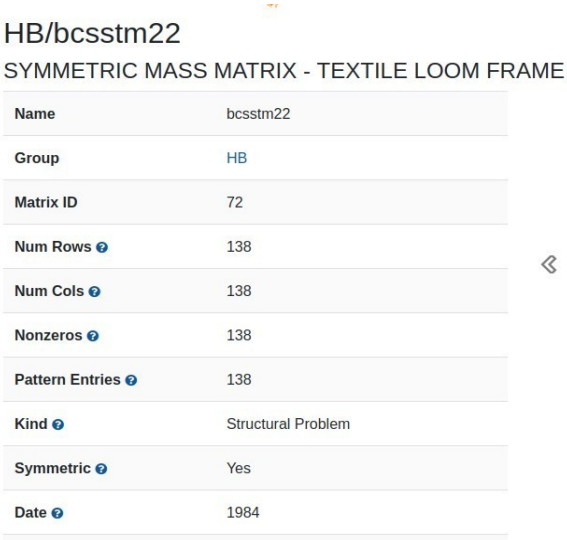

Figure 7 : Matrice bcsstk16

| Matrice bcsstk16 | | | | |
|---|---|---|---|---|
| | rFOM | rwFOM | rGMRES | rwGMRES |
| cycle | 2000 | 2000 | 2000 | 2000 |
| résidu | 1.680004e+05 | 1.398061e+05 | 2.928061e+04 | 2.864823e+04 |
| erreur | 2.927437e+04 | 2.821159e+04 | 2.928051e+04 | 2.864565e+04 |
| t_CPU | 2.934738e+02 | 3.540592e+02 | 3.026333e+02 | 3.592878e+02 |

Table 7: Matrice  bcsstk16

## Matrice bcsstm22

La matrice bcsstm22 est une matrice de dimension 138 x 138 appartenant au groupe HB, provenant des ensembles indépendants et des générateurs du Matrix Market, avec 138 entiers non nuls



HB/bcsstm22

SYMMETRIC MASS MATRIX - TEXTILE LOOM FRAME

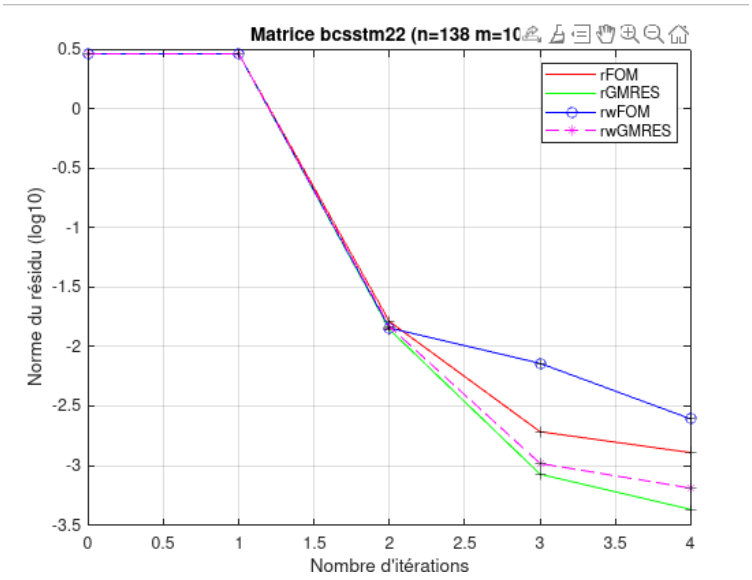
| | |
|---|---|
| Name | bcsstm22 |
| Group | HB |
| Matrix ID | 72 |
| Num Rows | 138 |
| Num Cols | 138 |
| Nonzeros | 138 |
| Pattern Entries | 138 |
| Kind | Structural Problem |
| Symmetric | Yes |
| Date | 1984 |

Figure 7 : Matrice bcsstm22

Interprétation des résultats pour la matrice bcsstm22 :

- Itérations : Toutes les méthodes ont convergé en 4 itérations, indiquant une convergence rapide pour cette matrice.
- RésiduFinal : Les résidus finaux sont comparables entre les méthodes, avec rgmres ayant le plus bas et rwfom ayant le plus élevé. Cependant, les différences sont petites.
- Erreur : Les erreurs calculées montrent des différences légères entre les méthodes. rwfom a l'erreur la plus élevée, tandis que rgmres a la plus basse.
- TempsCPU : Les temps CPU sont comparables entre les méthodes, avec rgmres et rwgmres étant légèrement plus rapides que rfom et rwfom.

Pour illustrer cet exemple, la figure et le tableau ci-dessous présentent les résultats obtenus pour m=10 avec un nombre de redémarrages égal à 4.




| Méthode | rfom | rgmres | rwfom | rwgmres |
|---|---|---|---|---|
| Itérations | 4 | 4 | 4 | 4 |
| RésiduFinal | 1.28e-03 | 4.27e-04 | 2.46e-03 | 6.45e-04 |
| Erreur | 1.29e+01 | 1.38e+01 | 7.79e+00 | 1.30e+01 |
| TempsCPU | 4.22e-02 | 4.21e-03 | 1.38e-02 | 7.58e-03 |

## Matrice e40r5000

La matrice e40r5000 est une matrice de dimension 17281 x 17281 appartenant au groupe HB, issue des ensembles indépendants et des générateurs du Matrix Market, avec 553956 entiers non nuls.

**Size**

17281 x 17281, 553956 entries

**Type**

real unsymmetric

**Nonzeros**

| total | diagonal | below diagonal | above diagonal | A-A' |
|---|---|---|---|---|
| 553562 | 12482 | 270540 | 270540 | 374402 |

**Column Data**

Average nonzeros per column : 32
Standard deviation : 16

| | index | nonzeros |
|---|---|---|
| longest | 453 | 62 |
| shortest | 9 | 8 |

**Row Data**

Average nonzeros per row : 32
Standard deviation : 16

| | index | nonzeros |
|---|---|---|
| longest | 453 | 62 |
| shortest | 9 | 8 |

**Bandwidths**

| lower | 452 | upper | 452 |
|---|---|---|---|
| average \|i-j\| | 2e+02 | std.dev. | 2.1e+02 |

**Profile Storage**

| | min | max | ave. | std.dev. |
|---|---|---|---|---|
| lower bandwidth | 0 | 451 | 4.3e+02 | 69 |
| upper bandwidth | 0 | 451 | 4.3e+02 | 69 |

Symmetric skyline storage requirement: 14968347

Interprétation des résultats pour la matrice e40r5000 :

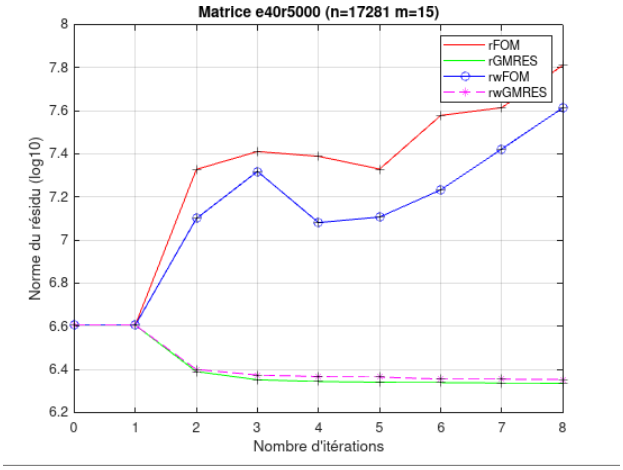
- Itérations : Toutes les méthodes ont convergé en 8 itérations, indiquant une convergence rapide pour cette matrice.

- RésiduFinal : Les résidus finaux varient entre les méthodes. rgmres a le résidu final le plus bas, suivi de près par rwgmres. rwfom a le résidu final le plus élevé.

- Erreur : Les erreurs calculées montrent des différences significatives entre les méthodes. rgmres a la plus basse erreur, suivie par rwgmres et rfom, tandis que rwfom a l'erreur la plus élevée.

- TempsCPU : Les temps CPU montrent que rwgmres est la méthode la plus rapide, suivie de près par rwfom. Les méthodes rgmres et rfom nécessitent plus
  de temps CPU

Pour illustrer cet exemple, la figure 6.1 et le tableau 6 présentent les résultats obtenus pour m=15 avec un nombre de démarrages égal à 8.




| Méthode | rfom | rgmres | rwfom | rwgmres |
|---|---|---|---|---|
| Itérations | 8 | 8 | 8 | 8 |
| RésiduFinal | 6.47e+07 | 2.16e+06 | 4.10e+07 | 2.26e+06 |
| Erreur | 1.04e+07 | 1.37e+06 | 9.93e+06 | 1.38e+06 |
| TempsCPU | 1.81e-01 | 2.13e-01 | 3.22e-01 | 3.07e-01 |

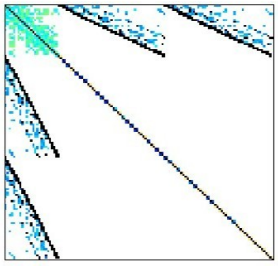
## Matrice add20

La matrice add20 est une matrice de dimension 2395 x 2395 provenant du groupe Hamm, issue des ensembles indépendants et des générateurs du Matrix Market, avec 17319 entiers non nuls. Le nombre conditionnel estimé est de 1,76E+4.

Hamm/add20

20-bit adder, from Steve Hamm (Motorola) hamm@austoto.sps.mot.com

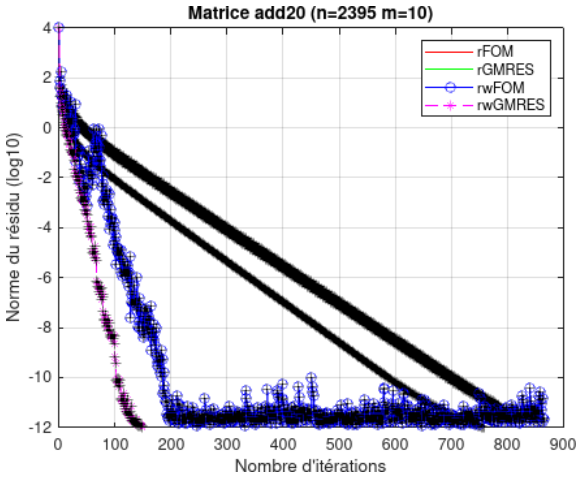
| | |
|---|---|
| Name | add20 |
| Group | Hamm |
| Matrix ID | 539 |
| Num Rows | 2,395 |
| Num Cols | 2,395 |
| Nonzeros | 13,151 |
| Pattern Entries | 17,319 |
| Kind | Circuit Simulation Problem |
| Symmetric | No |
| Date | 1991 |



Interprétation des résultats pour la matrice add20 :

- Itérations : La méthode rwgmres a convergé en seulement 149 itérations, tandis que les autres méthodes ont nécessité un nombre plus élevé d'itérations, en particulier rgmres avec 759 itérations.
- RésiduFinal : Les résidus finaux sont extrêmement faibles pour toutes les méthodes, tous de l'ordre de 1e-12.
- Erreur : Les erreurs calculées sont également très faibles pour toutes les méthodes, avec rwfom ayant la plus haute erreur, mais toujours très basse.
- TempsCPU : rwgmres est la méthode la plus rapide en termes de temps CPU, suivie de près par rgmres. rwfom et rfom nécessitent plus de temps CPU.

Pour illustrer cet exemple, la figure 7.1 et le tableau 7 présentent les résultats obtenus pour m=10 avec un nombre de démarrages égal à 865.




| Méthode | rfom | rgmres | rwfom | rwgmres |
|---|---|---|---|---|
| Itérations | 865 | 759 | 865 | 149 |
| RésiduFinal | 1.93e-12 | 1.04e-12 | 2.21e-12 | 1.12e-12 |
| Erreur | 2.03e-09 | 1.53e-08 | 1.54e-09 | 1.06e-08 |
| TempsCPU | 4.80e-01 | 4.27e-01 | 1.85e+00 | 3.31e-01 |

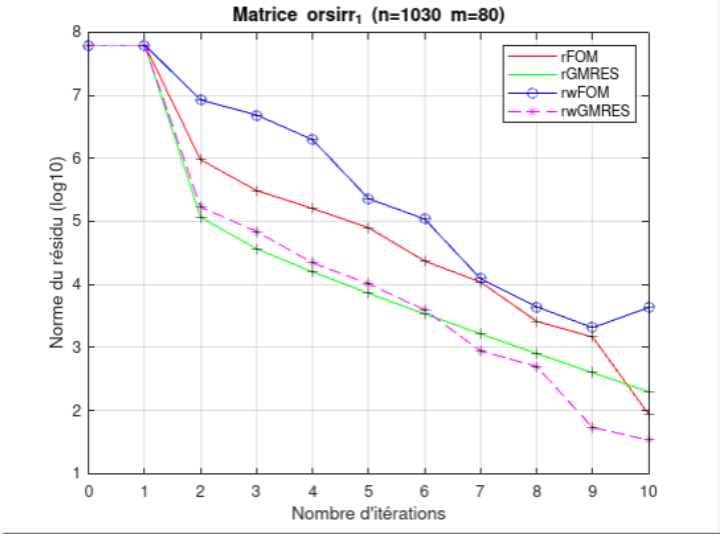
## Matrice $orsirr_1$

La matrice orsirr_1 est une matrice de dimension 1030 x 1030 issue d'une simulation de réservoir d'huile en 3D, provenant du package OILGEN de la collection Harwell Boeing, avec 6858 entiers non nuls. Le nombre conditionnel estimé est de 1,76E+4

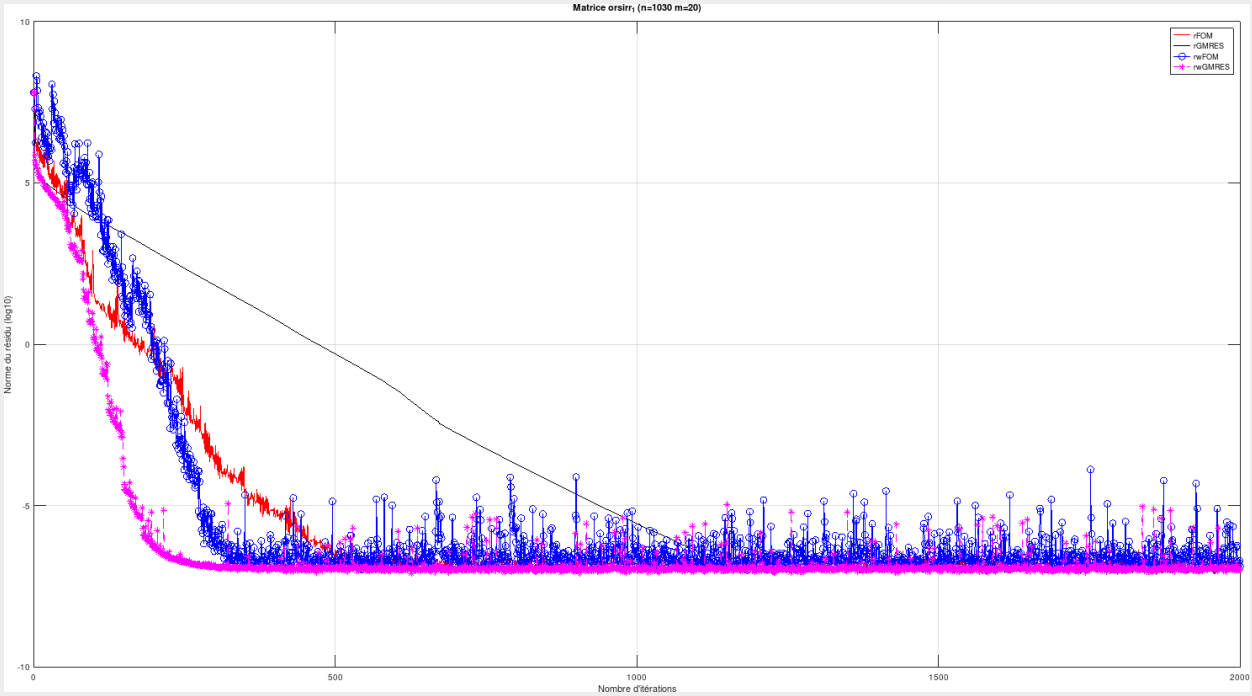
Interprétation des résultats pour la matrice orsirr_1 :

- Itérations : Toutes les méthodes ont convergé en 10 itérations, indiquant une convergence réussie pour cette matrice.
- RésiduFinal : Les résidus finaux varient entre les méthodes. rwgmres a le résidu final le plus bas, suivi de près par rgmres. rwfom a le résidu final le plus élevé, mais reste relativement bas.
- Erreur : Les erreurs calculées montrent des différences significatives entre les méthodes. rwgmres a la plus basse erreur, suivie par rgmres. Les méthodes rfom et rwfom ont des erreurs plus élevées.
- TempsCPU : Les temps CPU montrent que rgmres et rfom sont les méthodes les plus rapides, suivies de près par rwgmres. rwfom nécessite un peu plus de temps CPU.

Pour illustrer cet exemple, la figure 8.1 et le tableau 8 présentent les résultats obtenus pour m=80 avec un nombre de démarrages égal à 10 .

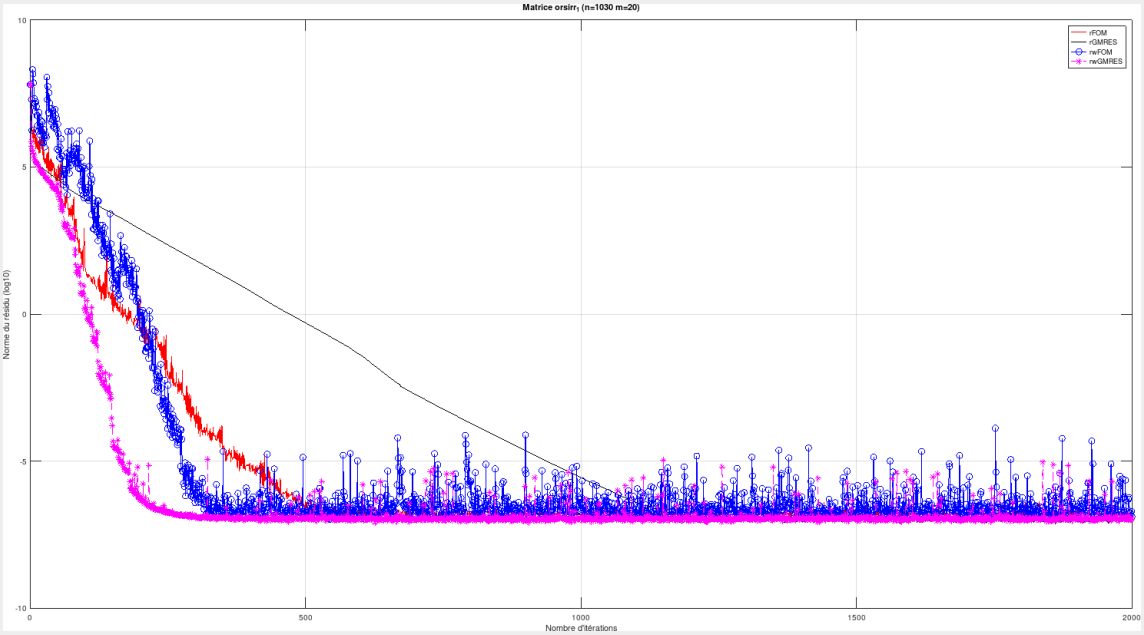



| Méthode | rfom | rgmres | rwfom | rwgmres |
|---|---|---|---|---|
| Itérations | 10 | 10 | 10 | 10 |
| RésiduFinal | 8.75e+01 | 1.98e+02 | 4.30e+03 | 3.40e+01 |
| Erreur | 4.27e-01 | 1.10e+01 | 7.89e+00 | 6.72e-01 |
| TempsCPU | 8.83e-02 | 8.00e-02 | 6.28e-01 | 5.79e-01 |



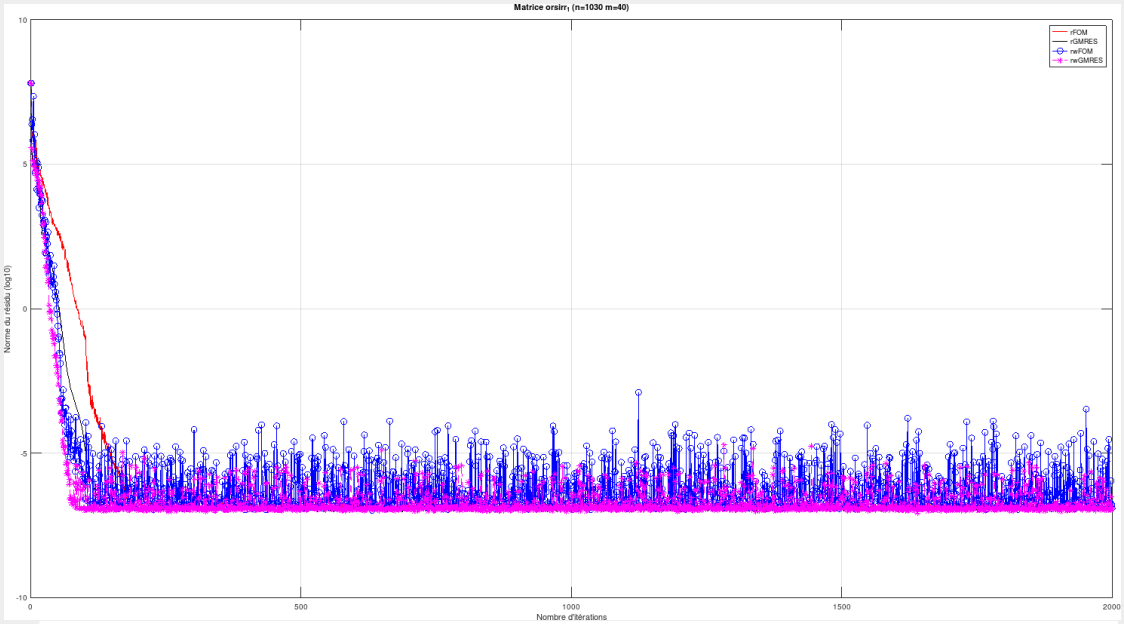

| Matrice orsirr_1 | | | | |
|---|---|---|---|---|
| | rFOM | rwFOM | rGMRES | rwGMRES |
| cycle | 2000 | 2000 | 2000 | 2000 |
| résidu | 6.535440e-02 | 2.508094e-07 | 7.324484e+04 | 1.149089e-02 |
| erreur | 1.903064e-03 | 5.499859e-10 | 8.663166e+03 | 1.657740e-03 |
| t_CPU | 3.449029e+00 | 4.784147e+00 | 3.525329e+00 | 4.934392e+00 |

## Test 1




| Matrice orsirr_1 | | | | |
|---|---|---|---|---|
| | rFOM | rwFOM | rGMRES | rwGMRES |
| cycle | 2000 | 2000 | 2000 | 2000 |
| résidu | 1.608753e-07 | 1.307902e-07 | 9.749879e-08 | 1.084787e-07 |
| erreur | 4.476451e-10 | 5.318118e-10 | 5.112542e-10 | 5.268859e-10 |
| t_CPU | 9.939905e+00 | 1.625783e+01 | 1.132506e+01 | 1.786109e+01 |

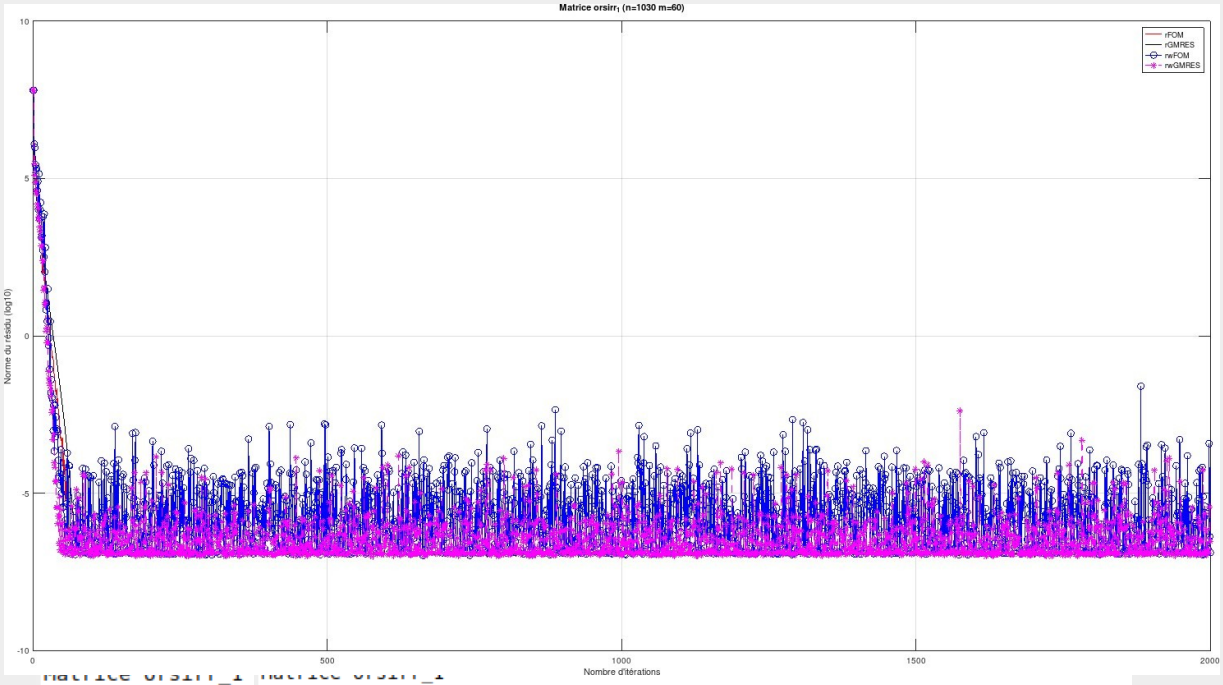
## Test 2




Matrice orsirr_1

| | rFOM | rwFOM | rGMRES | rwGMRES |
|---|---|---|---|---|
| cycle | 2000 | 2000 | 2000 | 2000 |
| résidu | 1.928858e-07 | 5.986260e-07 | 1.049833e-07 | 1.446901e-07 |
| erreur | 4.792150e-10 | 7.455794e-10 | 4.340006e-10 | 5.173805e-10 |
| t_CPU | 2.062084e+01 | 3.504810e+01 | 2.408586e+01 | 3.619751e+01 |

## Test 3

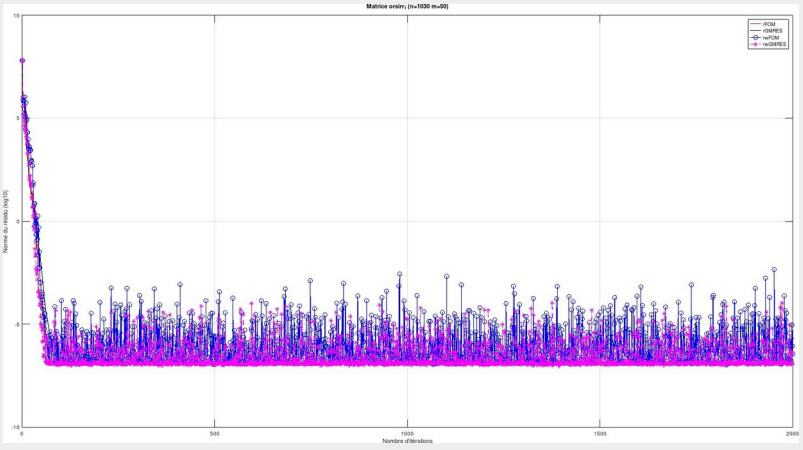



| | rFOM | rwFOM | rGMRES | rwGMRES |
|---|---|---|---|---|
| cycle | 2000 | 2000 | 2000 | 2000 |
| résidu | 1.266557e-07 | 3.641994e-07 | 1.124523e-07 | 1.116101e-07 |
| erreur | 5.592456e-10 | 6.385678e-10 | 5.551494e-10 | 6.696769e-10 |
| t_CPU | 5.562311e+01 | 8.707191e+01 | 6.239367e+01 | 9.281090e+01 |

## Test 4

Matrice orsirr_1

| | rFOM | rwFOM | rGMRES | rwGMRES |
|---|---|---|---|---|
| cycle | 2000 | 2000 | 2000 | 2000 |
| résidu | 1.153795e-07 | 1.311586e-07 | 1.221807e-07 | 2.981663e-07 |
| erreur | 6.430460e-10 | 7.056562e-10 | 6.149332e-10 | 6.483035e-10 |
| t_CPU | 8.190701e+01 | 1.314301e+02 | 8.952923e+01 | 1.391887e+02 |



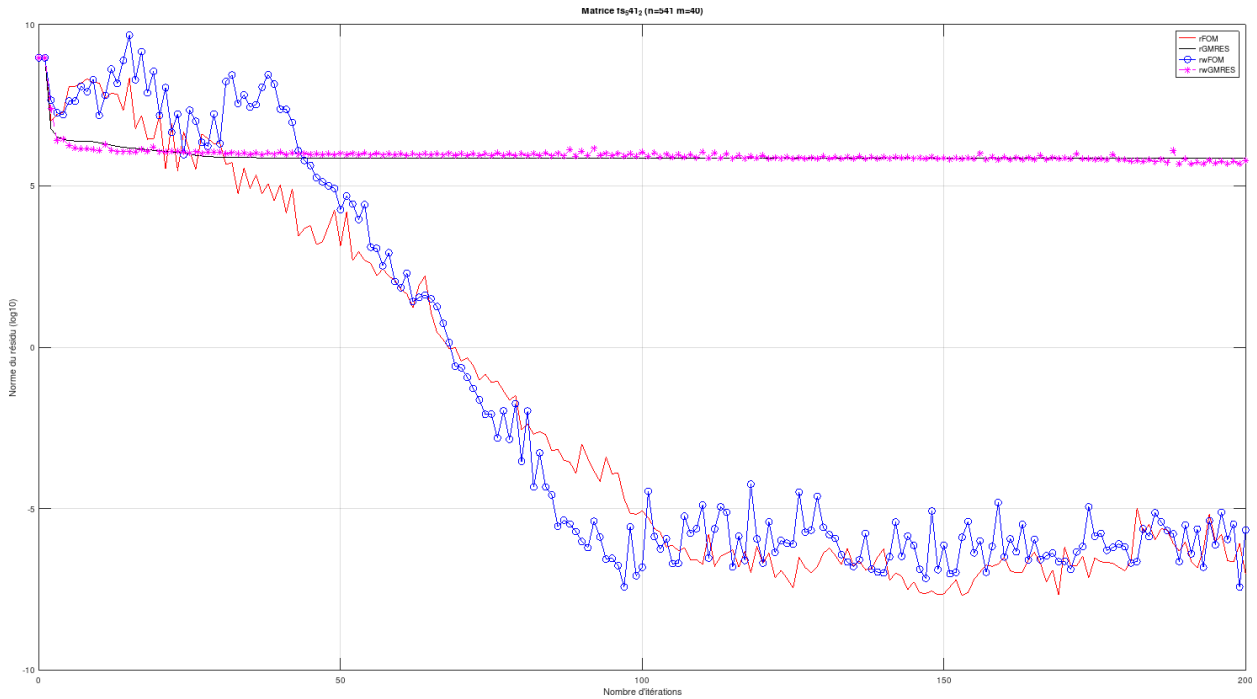
## Test 5

- **Les résultats ne sont pas évidents au départ mais lorsque m augmente, on voit clairement que les méthodes pondérées convergent plus vite même si un problème se pose : la stagnation de la norme du résidu au delà de $10^{-5}$.**

## Matrice fs_541_2

C'est une matrice carrée d'ordre 541 issue du laboratoire Harwell en Angleterre concernant un système d'équations différentielles ordinaires pour un problème de pollution atmosphérique et creuse à 98,54 %.

Comme dans l'article, nous prenons m = 40 et une tolérance de $10^{-10}$. Et on voit également que les méthodes FOM convergent contrairement aux GMRES qui stagnent.



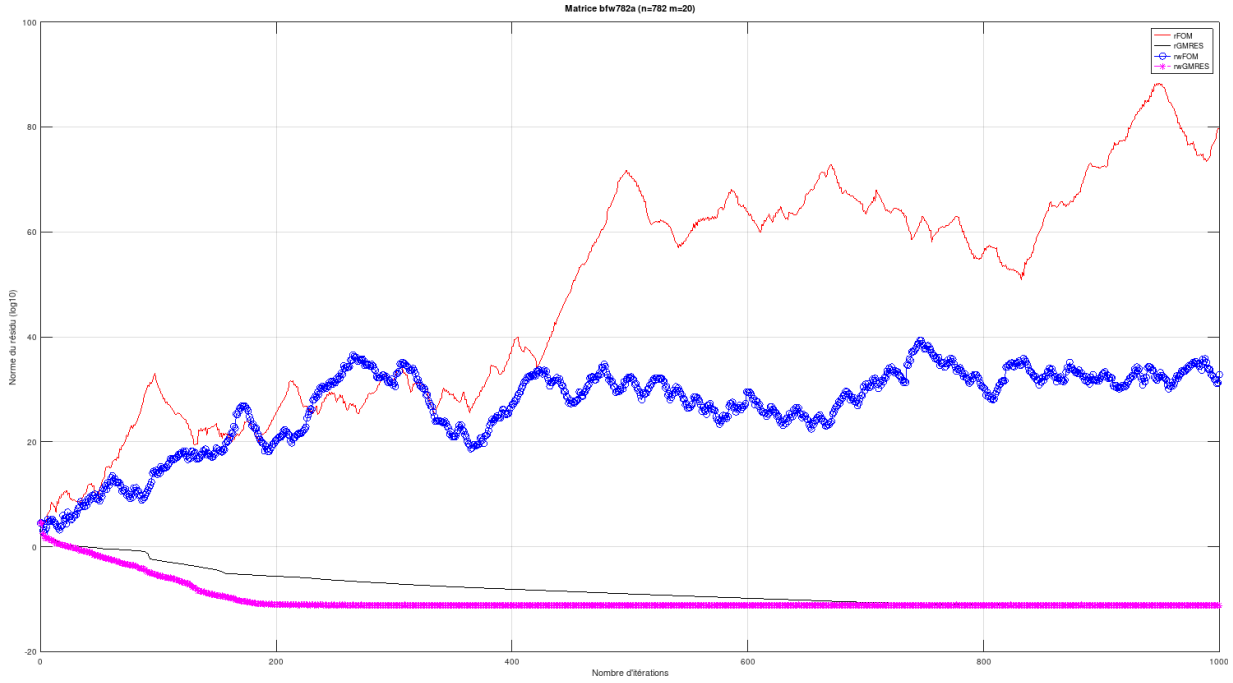

Matrice fs_541_2

| | rFOM | rwFOM | rGMRES | rwGMRES |
|---|---|---|---|---|
| cycle | 200 | 200 | 200 | 200 |
| résidu | 9.491789e-08 | 2.175402e-06 | 7.144124e+05 | 6.069345e+05 |
| erreur | 3.419289e-08 | 4.986411e-08 | 7.166360e+06 | 6.992124e+06 |
| t_CPU | 2.906851e+00 | 3.889679e+00 | 3.098452e+00 | 4.128634e+00 |

## Matrice bfw782a

Elle est d'ordre 782, creuse à 98,77 % et provenant d'un guide d'onde diélectrique.

Les méthodes GMRES convergent cette fois plutôt bien contrairement aux méthodes FOM qui divergent totalement.

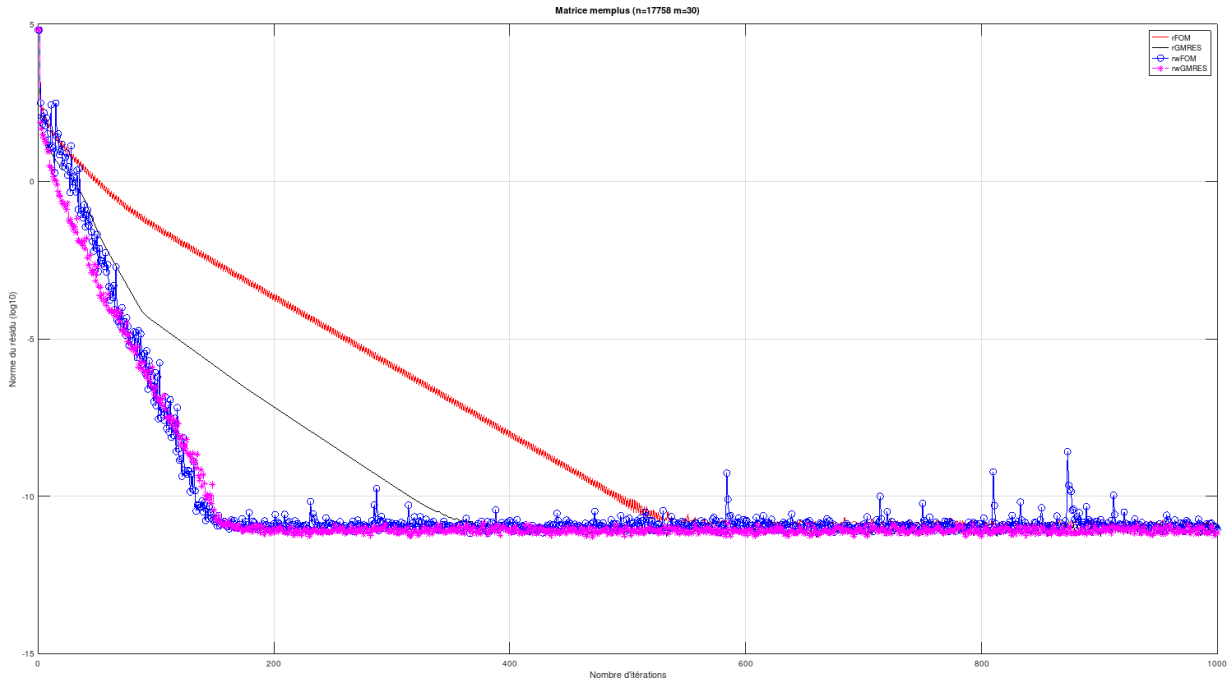



| Matrice bfw782a | | | | |
|---|---|---|---|---|
| | rFOM | rwFOM | rGMRES | rwGMRES |
| cycle | 1000 | 1000 | 1000 | 1000 |
| résidu | 3.288494e+79 | 6.657813e+32 | 7.111848e-12 | 6.629870e-12 |
| erreur | 1.175974e+80 | 1.029021e+34 | 1.966130e-10 | 1.062426e-10 |
| t_CPU | 4.821933e+00 | 7.121377e+00 | 5.201983e+00 | 7.595654e+00 |

## Matrice memplus

Cette matrice est très volumineuse (d'ordre 17758) et creuse à 98,84 %. Il s'agit d'un circuit mémoire pour des modèles de composants d'ordinateur. On peut donc s'attendre à un temps de calcul assez élevé.

Le résultat est que les méthodes pondérées (WFOM et WGMRES) convergent plus vite que leurs équivalents habituels ce qui prouve l'efficacité de ces méthodes pour des systèmes linéaires de très grande taille.




Matrice memplus

| | rFOM | rwFOM | rGMRES | rwGMRES |
|---|---|---|---|---|
| cycle | 1000 | 1000 | 1000 | 1000 |
| résidu | 1.234015e-11 | 9.089998e-12 | 6.206692e-12 | 6.844182e-12 |
| erreur | 5.815929e-08 | 5.842085e-08 | 5.950113e-08 | 5.746054e-08 |
| t_CPU | 6.786573e+01 | 1.299002e+02 | 7.255944e+01 | 1.350319e+02 |

## Matrice de Poisson

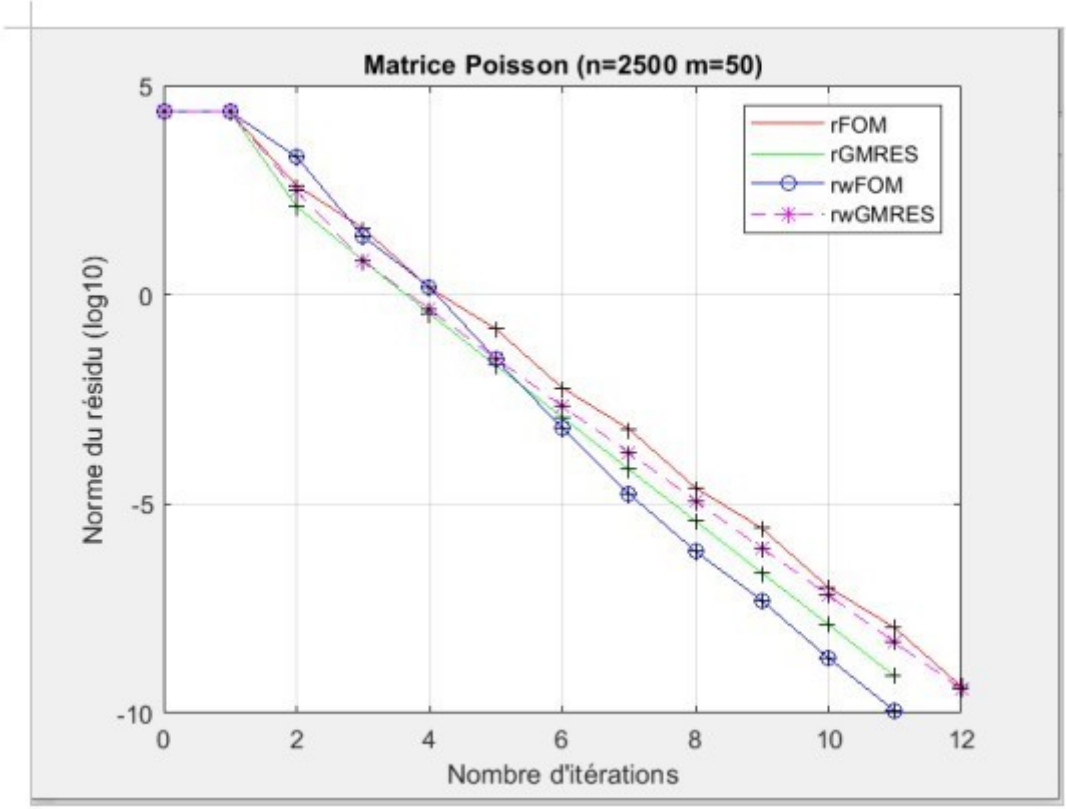
La création de la matrice Poisson implique la discrétisation de l'équation de Poisson, souvent par des méthodes de différences finies, d'éléments finis ou de volumes finis. Cela consiste à transformer l'équation continue en un système d'équations linéaires représenté sous forme matricielle.
La matrice Poisson résultant de la discrétisation est typiquement creuse, reflétant la localité des interactions dans l'équation de Poisson. Par exemple, dans une grille 2D avec des différences finies, chaque ligne de la matrice représente une équation pour un point de la grille, et les non-zéros représentent les interactions avec les points voisins.
La matrice Poisson est souvent symétrique et définie positive, surtout dans les cas où l'équation de Poisson est discrétisée à l'aide de différences finies ou d'éléments finis standard. Cela permet l'utilisation de méthodes efficaces de résolution, telles que les méthodes de gradient conjugué.

Pour résoudre un système linéaire utilisant la matrice Poisson, on utilise souvent des méthodes itératives, en particulier lorsque la taille de la matrice est grande, comme c'est couramment le cas dans les simulations numériques. Ces méthodes incluent les

méthodes de gradient conjugué, GMRES, et d'autres algorithmes adaptés aux matrices creuses et symétriques.

- Stockage en mode ‘sparse’

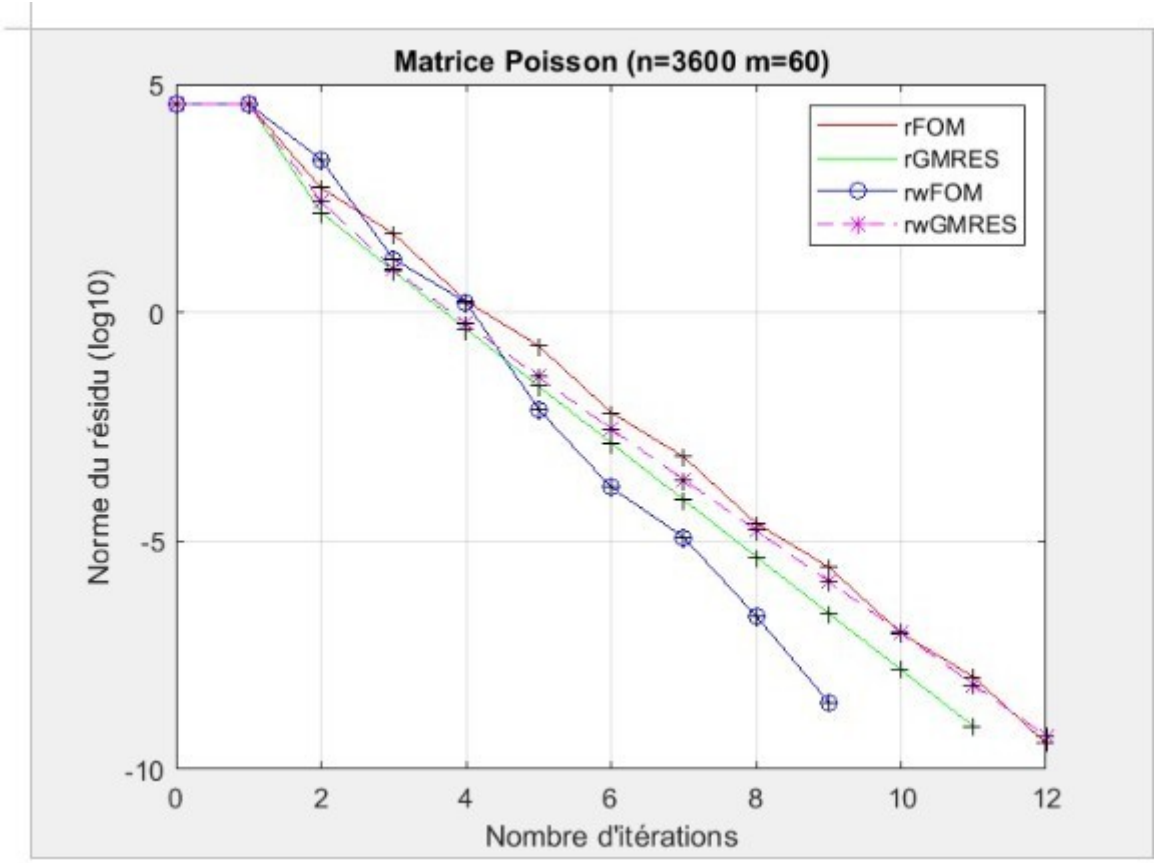
1. m=50,n=2500,redémarrage=700




| Matrice Poisson | | | | |
|---|---|---|---|---|
| | rFOM | rwFOM | rGMRES | rwGMRES |
| cycle | 12 | 11 | 11 | 12 |
| résidu | 4.324854e-10 | 1.131567e-10 | 7.648762e-10 | 3.829496e-10 |
| erreur | 3.206353e-10 | 9.920138e-11 | 4.959005e-09 | 2.121316e-09 |
| t_CPU | 1.227738e-01 | 6.364730e-01 | 7.804510e-02 | 5.860889e-01 |

- Stockage en mode ‘full’

  2. m=60,n=3600,redémarrage=800




| Matrice Poisson | | | | |
|---|---|---|---|---|
| | rFOM | rwFOM | rGMRES | rwGMRES |
| cycle | 12 | 9 | 11 | 12 |
| résidu | 3.686518e-10 | 2.952998e-09 | 8.622467e-10 | 5.328799e-10 |
| erreur | 4.003918e-10 | 8.643375e-10 | 7.879586e-09 | 4.285570e-09 |
| t_CPU | 3.220517e+00 | 3.160044e+00 | 2.966779e+00 | 4.164343e+00 |

## Observations globales

1 - Les lignes des méthodes où nous cherchons à déterminer dm (pour rFOM et rGMRES) ont parfois posé problème lors de certains tests car le calcul de l'inverse n'est pas toujours évident. Il en est résulté des délais de traitement trop élevés. C'est la raison pour laquelle nous avons préféré utiliser la décomposition QR dans les méthodes WFOM et WGMRES pour la détermination de ym. Et même dans ce cas, un calcul d'inverse est nécessaire pour résoudre le système $Q^tA = R$ , ce qui nous ramène au même problème.

2 – Le problème s'est encore posé lors de la détermination de la matrice de l'exemple 1 pour certaines valeurs (calcul de l'inverse de S).

3 - Le mode de stockage de la matrice (full ou sparse) influe grandement sur la vitesse de traitement des algorithmes. C'est presque imperceptible pour de petites matrices mais pour d'autres, volumineuses comme memplus, la différence en temps CPU est énorme.

## Conclusion

Les méthodes FOM et GMRES sont très efficaces pour la résolution de systèmes linéaires mais seulement pour des cas particuliers. Leurs versions pondérées sont encore meilleures car globalement la convergence est plus rapide et le temps de traitement, moindre. Surtout dans le cas de la méthode WGMRES qui, nous l'avons constaté, a presque toujours convergé plus rapidement que la méthode WFOM et ce, même avec une matrice très volumineuse. D'où sa popularité dans tous ces domaines scientifiques où elle est employée pour résoudre des problèmes.

## Références


1 http://exo7.emath.fr/
2 https://fr.wikipedia.org/wiki/Syst%C3%A8me_d%27%C3%A9quations_lin %C3%A9aires
3 https://fr.wikipedia.org/wiki/GMRES
4 https://www.sciencedirect.com/science/article/pii/S0898122108006160
5 https://math.nist.gov/MatrixMarket/
6 Azedine Essai, Weighted FOM and GMRES for solving nonsymmetric linear systems, Numerical Algorithms (Université des Sciences et Technologies de Lille, 1998)