Rapport de laboratoire 4

MTH8211

Téo Dumoutier

```
using Pkg
Pkg.activate("labo4_env")
Pkg.add("Krylov")
Pkg.add("SuiteSparseMatrixCollection")
Pkg.add("MatrixMarket")
Pkg.add("PrettyTables")
Pkg.add("Plots")

using LinearAlgebra, SparseArrays
using MatrixMarket
using Krylov, SuiteSparseMatrixCollection
using PrettyTables
using Plots
```

Contexte

Dans ce laboratoire, on demande de comparer des méthodes basées sur le processus de Golub-Kahan et de Lanczos pour résoudre des problèmes aux moindres carrés et de moindre norme. On se réfèrera aux carnets Jupyter vus en laboratoire pour les extraits de code pertinents.

Questions

Sur base du code vu en laboratoire, écrire une fonction qui prend en entrée un nom de matrice `matrix_name` et renvoie la matrice `A` et le membre de droite `b` correspondants.

```
function get_mm(matrix_name)
  ssmc = ssmc_db()
  pb = ssmc_matrices(ssmc, "", matrix_name)
  fetch_ssmc(pb, format="MM")
  pb_path = fetch_ssmc(pb, format="MM")
  path_mtx = pb_path[1]
  A = MatrixMarket.mmread(joinpath(path_mtx, matrix_name * ".mtx"));
  b = MatrixMarket.mmread(joinpath(path_mtx, matrix_name * "_b.mtx"))[:];

  n, m = size(A)
  K = [sparse(1.0I,n,n) A; A' spzeros(m,m)]
  rhs = [b; zeros(m)]
  return A, b, K, rhs
end
```

Utiliser cette fonction pour télécharger les problèmes aux moindres carrés `illc1033`, `illc1850`, `well1033`,

`well1850` de la SuiteSparse Matrix Collection, y compris les membres de droite. Attention, traiter les problèmes en séquence et ne pas tous les mémoriser simultanément.

Problèmes aux moindres carrés

Méthodes basées sur Golub-Kahan

Résoudre ces problèmes avec LSQR et LSMR en utilisant les tolérances par défaut et comparer les résultats dans un tableau à l'aide de PrettyTables.jl (voici un exemple). Votre tableau doit montrer

- le nombre d'itérations ;
- la norme du résidu aux moindres carrés initial et final ;
- la norme du résidu d'optimalité du problème aux moindres carrés initial et final ;
- le statut final.

Tracer sur un graphe l'évolution du résidu aux moindres carrés, du résidu d'optimalité du problème aux moindres carrés, et de l'erreur en norme euclidienne.

```
solve_QR(A, b) = qr(A) \ b
```

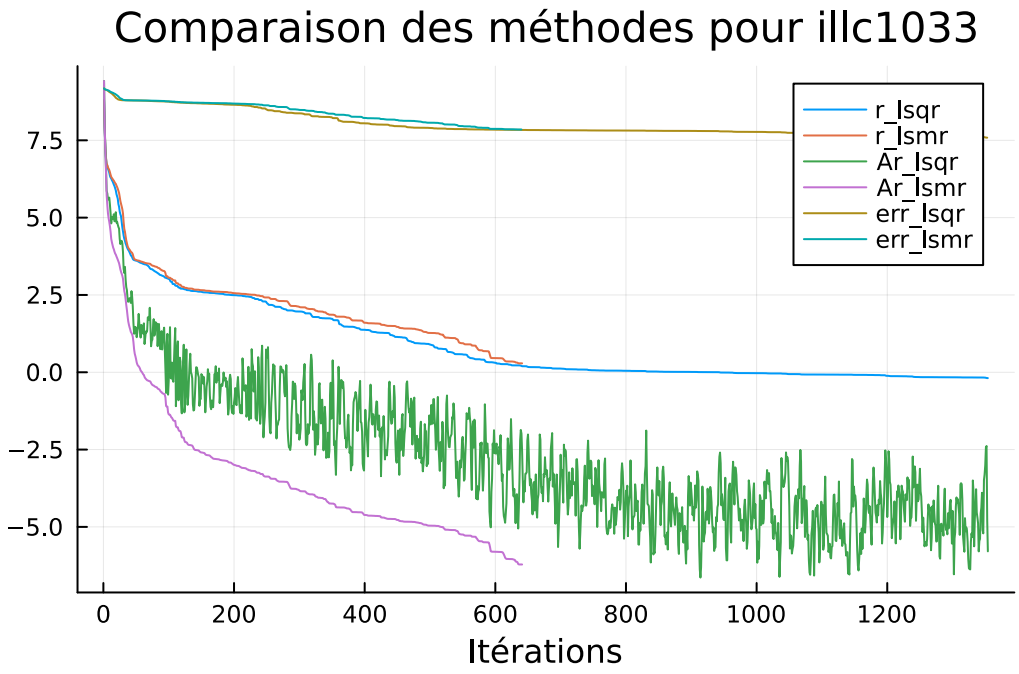
```
x_qr = [1]

list_error_lsqr = []
list_error_lsmr = []
function custom_callback_lsqr(workspace::KrylovWorkspace,
 ↪  list_error_lsqr=list_error_lsqr)
  push!(list_error_lsqr, norm(workspace.x - x_qr))
  return false
end
function custom_callback_lsmr(workspace::KrylovWorkspace,
 ↪  list_error_lsmr=list_error_lsmr)
  push!(list_error_lsmr, norm(workspace.x - x_qr))
  return false
end

list_names = ["illc1033", "illc1850", "well1033", "well1850"]
for i in 1:4
  name = list_names[i]
  A, b, K, rhs = get_mm(name)

  data = Array{Any}(undef, 6, 3)
  headers = [name, "Lsqr", "Lsmr"]
  data[1, 1] = "Itérations"
  data[2, 1] = "Norme résidu initial"
  data[3, 1] = "Norme résidu final"
  data[4, 1] = "Norme résidu optimalité initial"
  data[5, 1] = "Norme résidu optimalité final"
  data[6, 1] = "Statut final"

  x_qr = solve_QR(A, b)
  list_error_lsqr = []
  list_error_lsmr = []

  (x_lsqr, stats_lsqr) = lsqr(A, b, history=true, callback=custom_callback_lsqr)
  (x_lsmr, stats_lsmr) = lsmr(A, b, history=true, callback=custom_callback_lsmr)

  data[1, 2] = stats_lsqr.niter
```

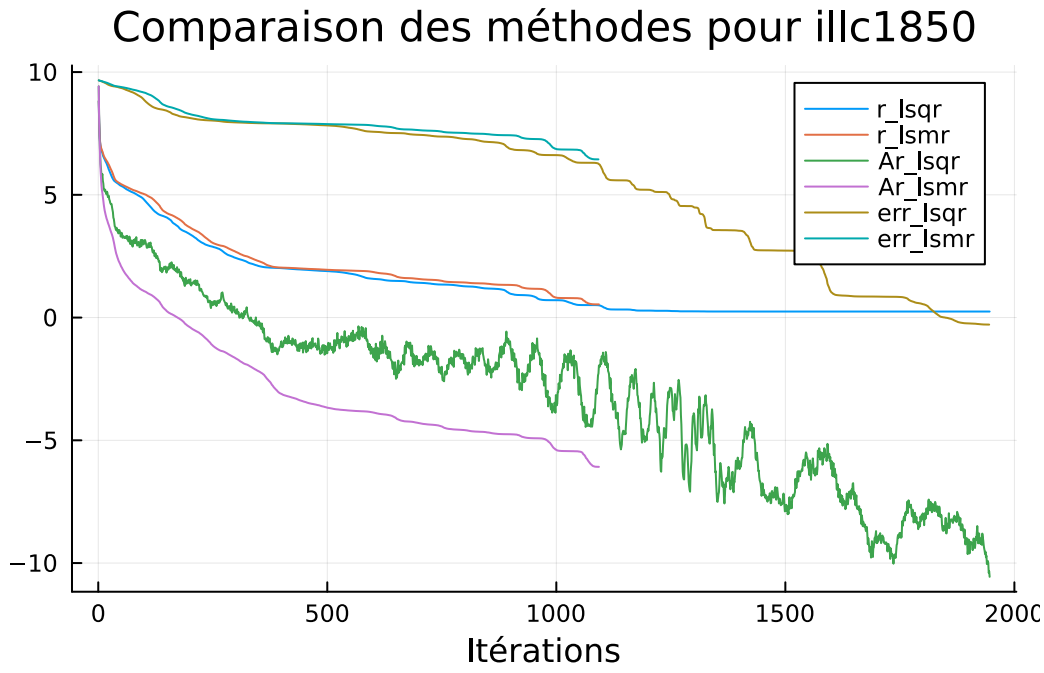
```
  data[1, 3] = stats_lsmr.niter
  data[2, 2] = stats_lsqr.residuals[1]
  data[2, 3] = stats_lsmr.residuals[1]
  data[3, 2] = stats_lsqr.residuals[end]
  data[3, 3] = stats_lsmr.residuals[end]
  data[4, 2] = stats_lsqr.Aresiduals[1]
  data[4, 3] = stats_lsmr.Aresiduals[1]
  data[5, 2] = stats_lsqr.Aresiduals[end]
  data[5, 3] = stats_lsmr.Aresiduals[end]
  data[6, 2] = stats_lsqr.solved
  data[6, 3] = stats_lsmr.solved

  pretty_table(data, header=headers)

  p = plot(log.(stats_lsqr.residuals), label="r_lsqr", legend=:topright)
  plot!(p, log.(stats_lsmr.residuals), label="r_lsmr")
  plot!(p, log.(stats_lsqr.Aresiduals), label="Ar_lsqr")
  plot!(p, log.(stats_lsmr.Aresiduals), label="Ar_lsmr")
  plot!(p, log.(list_error_lsqr), label="err_lsqr")
  plot!(p, log.(list_error_lsmr), label="err_lsmr")

  xlabel!("Itérations")
  title!("Comparaison des méthodes pour " * name)

  display(p)
end
```

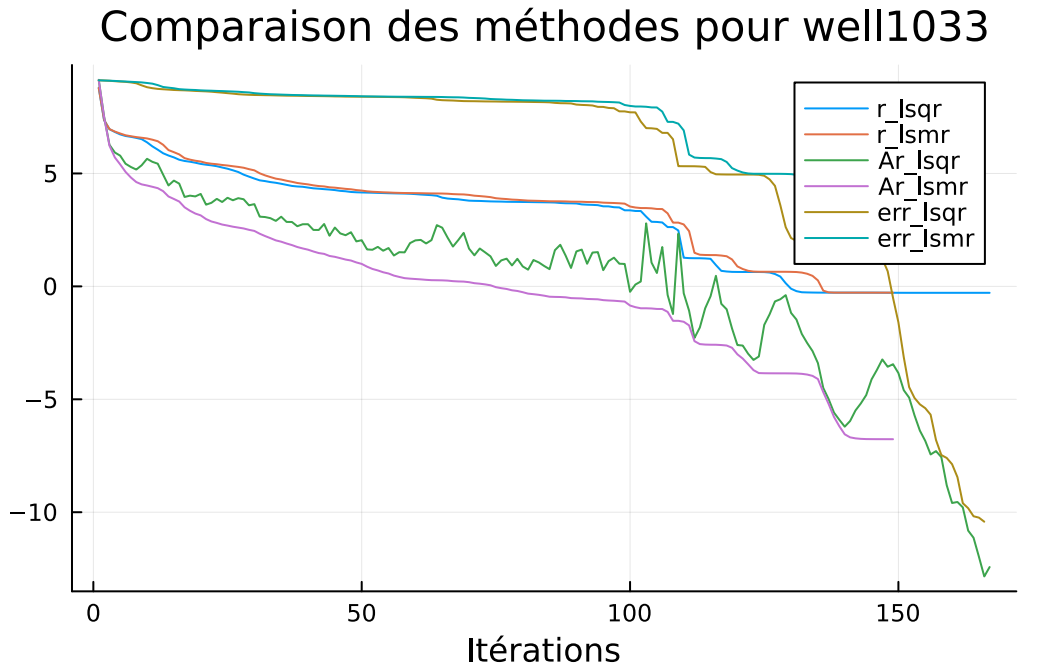
```
                        illc1033        Lsqr        Lsmr

                      Itérations        1353         640
           Norme résidu initial      6597.79     6597.79
             Norme résidu final     0.828491     1.33479
 Norme résidu optimalité initial      12317.4     12317.4
   Norme résidu optimalité final   0.00305537   0.00199859
                    Statut final        false        true


                        illc1850       Lsqr        Lsmr

                      Itérations        1945        1092
           Norme résidu initial     6784.94     6784.94
             Norme résidu final     1.27814     1.71139
 Norme résidu optimalité initial     12319.3     12319.3
   Norme résidu optimalité final   2.5898e-5   0.00228747
                    Statut final        true        true


                        well1033        Lsqr        Lsmr

                      Itérations         166         148
           Norme résidu initial      6597.79     6597.79
             Norme résidu final     0.752158    0.757757
 Norme résidu optimalité initial      9363.74     9363.74
```

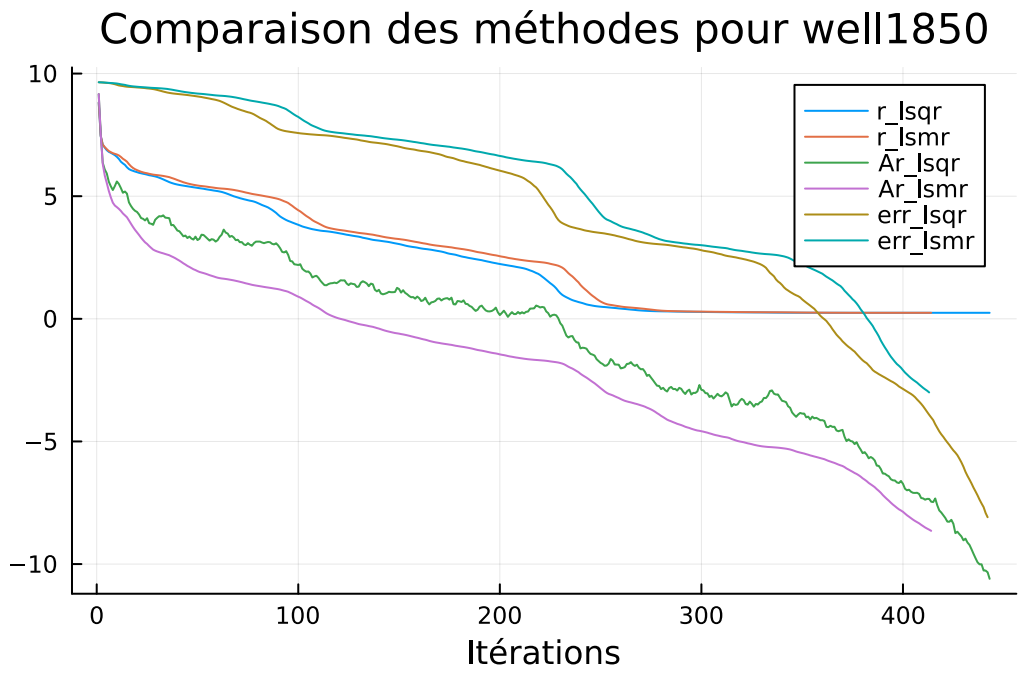
Norme résidu optimalité final	4.01326e-6	0.00115101
Statut final	true	true

well1850	Lsqr	Lsmr
Itérations	442	413
Norme résidu initial	6784.94	6784.94
Norme résidu final	1.27814	1.27814
Norme résidu optimalité initial	9567.43	9567.43
Norme résidu optimalité final	2.4862e-5	0.000175869
Statut final	true	true




Comparaison des méthodes pour illc1850
r_lsqr
r_lsmr
Ar_lsqr
Ar_lsmr
err_lsqr
err_lsmr
10
5
0
−5
−10
0
500
1000
1500
2000
Itérations


Comparaison des méthodes pour well1033
r_lsqr
r_lsmr
Ar_lsqr
Ar_lsmr
err_lsqr
err_lsmr
5
0
−5
−10
0
50
100
150
Itérations



Commenter ces résultats sur base de ce qui a été vu en classe :

On remarque que le résidu et l'erreur de la méthode LSQR sont toujours plus petits que pour la méthode LSMR. Par contre, ceci vient au prix d'un résidu d'optimalité qui n'est pas monotone pour LSQR, alors qu'il l'est pour LSMR. Ceci est attendu puisque LSQR est basée sur la minimisation du résidu à chaque itération, alors que LSMR est basée sur la minimisation du résidu d'optimalité, et que l'erreur est intimement liée au résidu. Qui plus est, le résidu, le résidu d'optimalité et l'erreur sont tous décroissants en moyenne, ce qui est logique étant donné que l'on s'approche de plus en plus de la solution.

Outre cela, on remarque que LSMR a tendance à s'arrêter plus tôt que LSQR, et même que LSQR considère que le problème n'est pas résolu dans le cas 'illc1033'. Ceci provient de la façon dont le seuil d'arrêt lié aux tolérances par défaut est géré dans chacune de ces méthodes, et s'explique peut-être par le fait que si le résidu d'optimalité est un critère important pour cela, alors puisque le résidu d'optimalité décroît en moyenne plus fortement dans le cas de LSMR, qui se concentre pour le minimiser de façon monotone, alors il est logique que LSMR ait plus de facilité à s'arrêter avec les tolérances par défaut.

Méthodes basées sur Lanczos

Répéter la question précédente en appliquant MINRES, MINRES-QLP, MINARES et SYMMLQ sur le système de point de selle et produire un tableau semblable.

Tracer sur un graphe l'évolution du résidu d'optimalité du problème aux moindres carrés et du résidu du système de point de selle.

```
n = 1
A = [1]

list_r_symmlq = []
list_r_minres = []
list_r_minres_qlp = []
list_r_minares = []
list_Ar_symmlq = []
list_Ar_minres = []
list_Ar_minres_qlp = []
```

```
list_Ar_minares = []
function custom_callback_symmlq(workspace::KrylovWorkspace, list_r_symmlq=list_r_symmlq,
 ↪  list_Ar_symmlq=list_Ar_symmlq)
  r = workspace.x[1:n]
  push!(list_r_symmlq, norm(r))
  push!(list_Ar_symmlq, norm(A'*r))
  return false
end
function custom_callback_minres(workspace::KrylovWorkspace, list_r_minres=list_r_minres,
 ↪  list_Ar_minres=list_Ar_minres)
  r = workspace.x[1:n]
  push!(list_r_minres, norm(r))
  push!(list_Ar_minres, norm(A'*r))
  return false
end
function custom_callback_minres_qlp(workspace::KrylovWorkspace,
 ↪  list_r_minres_qlp=list_r_minres_qlp, list_Ar_minres_qlp=list_Ar_minres_qlp)
  r = workspace.x[1:n]
  push!(list_r_minres_qlp, norm(r))
  push!(list_Ar_minres_qlp, norm(A'*r))
  return false
end
function custom_callback_minares(workspace::KrylovWorkspace,
 ↪  list_r_minares=list_r_minares, list_Ar_minares=list_Ar_minares)
  r = workspace.x[1:n]
  push!(list_r_minares, norm(r))
  push!(list_Ar_minares, norm(A'*r))
  return false
end

list_names = ["illc1033", "illc1850", "well1033", "well1850"]
for i in 1:4
  name = list_names[i]
  A, b, K, rhs = get_mm(name)

  data = Array{Any}(undef, 6, 5)
  headers = ["\'illc1033\'", "Symmlq", "Minres", "Minres_QLP", "Minares"]
  data[1, 1] = "Itérations"
  data[2, 1] = "Norme résidu initial"
  data[3, 1] = "Norme résidu final"
  data[4, 1] = "Norme résidu optimalité initial"
  data[5, 1] = "Norme résidu optimalité final"
  data[6, 1] = "Statut final"

  n, m = size(A)
  list_r_symmlq = []
  list_r_minres = []
  list_r_minres_qlp = []
  list_r_minares = []
  list_Ar_symmlq = []
  list_Ar_minres = []
  list_Ar_minres_qlp = []
  list_Ar_minares = []
```

```
  (sol_symmlq, stats_symmlq) = symmlq(K, rhs, history=true,
 ↪  callback=custom_callback_symmlq)
  (sol_minres, stats_minres) = minres(K, rhs, history=true,
 ↪  callback=custom_callback_minres)
  (sol_minres_qlp, stats_minres_qlp) = minres_qlp(K, rhs, history=true,
 ↪  callback=custom_callback_minres_qlp)
  (sol_minares, stats_minares) = minares(K, rhs, history=true,
 ↪  callback=custom_callback_minares)

  data[1, 2] = stats_symmlq.niter
  data[1, 3] = stats_minres.niter
  data[1, 4] = stats_minres_qlp.niter
  data[1, 5] = stats_minares.niter
  data[2, 2] = list_r_symmlq[1]
  data[2, 3] = list_r_minres[1]
  data[2, 4] = list_r_minres_qlp[1]
  data[2, 5] = list_r_minares[1]
  data[3, 2] = list_r_symmlq[end]
  data[3, 3] = list_r_minres[end]
  data[3, 4] = list_r_minres_qlp[end]
  data[3, 5] = list_r_minares[end]
  data[4, 2] = list_Ar_symmlq[1]
  data[4, 3] = list_Ar_minres[1]
  data[4, 4] = list_Ar_minres_qlp[1]
  data[4, 5] = list_Ar_minares[1]
  data[5, 2] = list_Ar_symmlq[end]
  data[5, 3] = list_Ar_minres[end]
  data[5, 4] = list_Ar_minres_qlp[end]
  data[5, 5] = list_Ar_minares[end]
  data[6, 2] = stats_symmlq.solved
  data[6, 3] = stats_minres.solved
  data[6, 4] = stats_minres_qlp.solved
  data[6, 5] = stats_minares.solved

  pretty_table(data, header=headers)

  p = plot(log.(list_Ar_symmlq), label="Ar_symmlq", legend=:topright)
  plot!(p, log.(list_Ar_minres), label="Ar_minres")
  plot!(p, log.(list_Ar_minres_qlp), label="Ar_minres_qlp")
  plot!(p, log.(list_Ar_minares), label="Ar_minares")

  plot!(p, log.(stats_symmlq.residuals), label="r_sadle_symmlq")
  plot!(p, log.(stats_minres.residuals), label="r_sadle_minres")
  plot!(p, log.(stats_minres_qlp.residuals), label="r_sadle_minres_qlp")
  plot!(p, log.(stats_minares.residuals), label="r_sadle_minares")

  xlabel!("Itérations")
  title!("Comparaison des méthodes pour " * name)

  display(p)
end
```

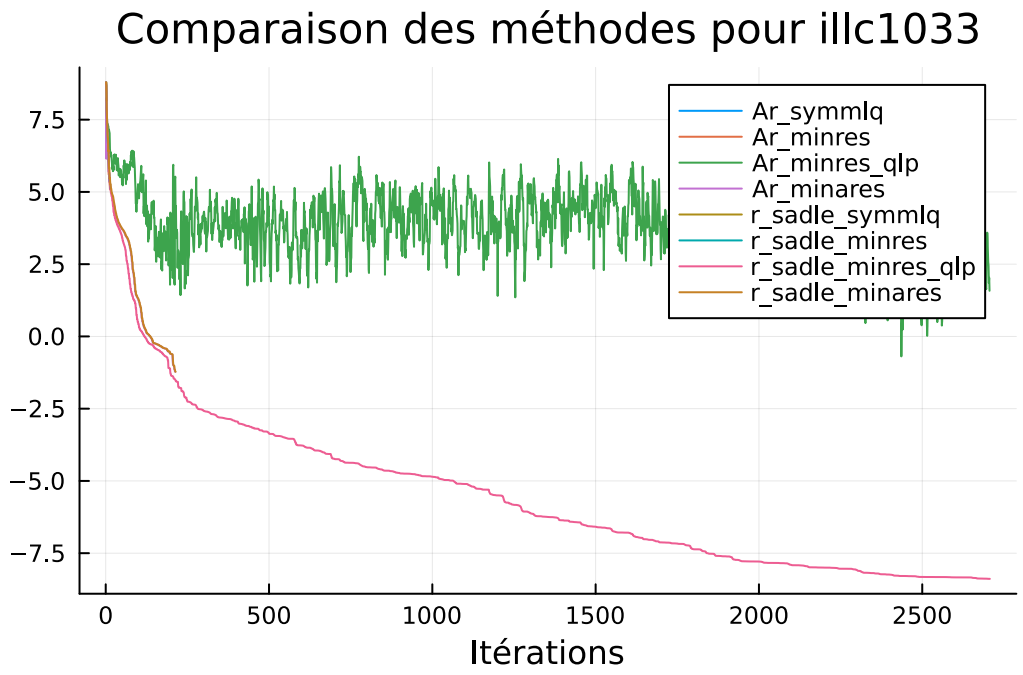
```
'illc1033'    Symmlq    Minres   Minres_QLP    Minares
```

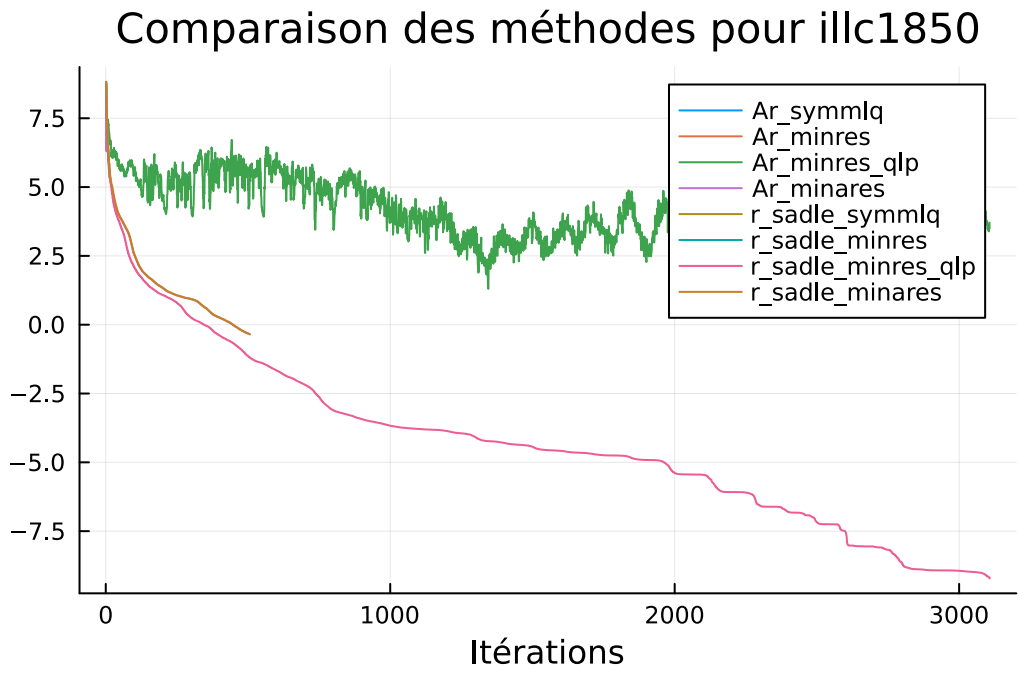
Itérations	2	14	2706	213
Norme résidu initial	1470.98	1470.98	0.0	2041.31
Norme résidu final	704.846	711.557	6.5496	22.7977
Norme résidu optimalité initial	2746.17	2746.17	0.0	3810.92
Norme résidu optimalité final	771.953	160.909	4.84678	0.294303
Statut final	true	true	false	true

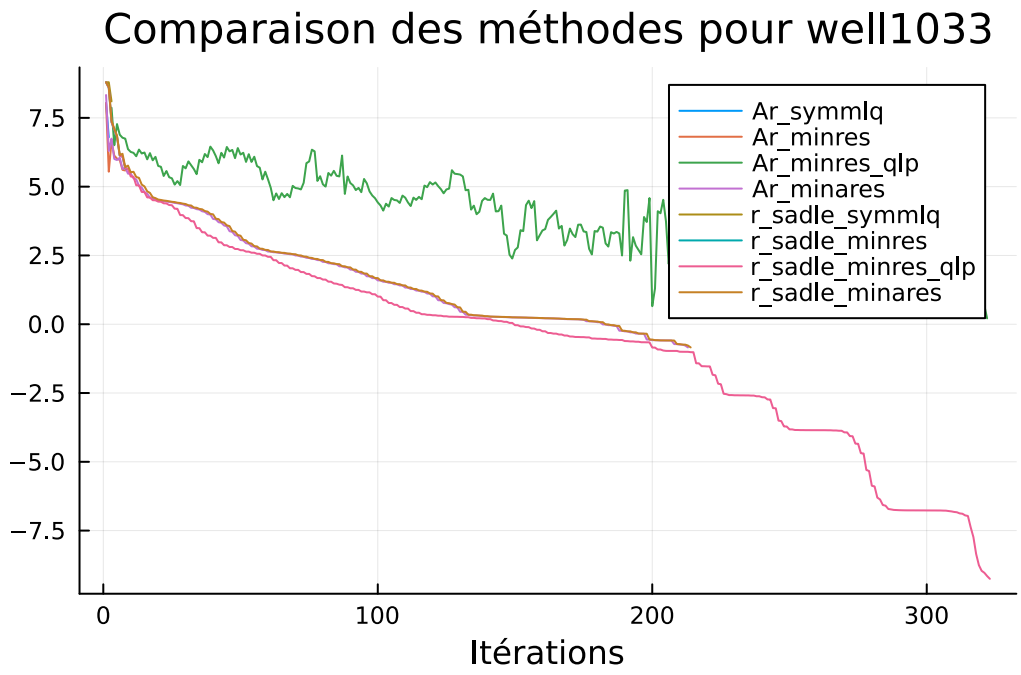
'illc1033'	Symmlq	Minres	Minres_QLP	Minares
Itérations	2	14	3107	506
Norme résidu initial	1579.1	1579.1	0.0	2146.42
Norme résidu final	809.492	896.887	41.2288	48.0435
Norme résidu optimalité initial	2867.15	2867.15	0.0	3897.22
Norme résidu optimalité final	853.271	205.711	40.7434	0.705561
Statut final	true	true	true	true

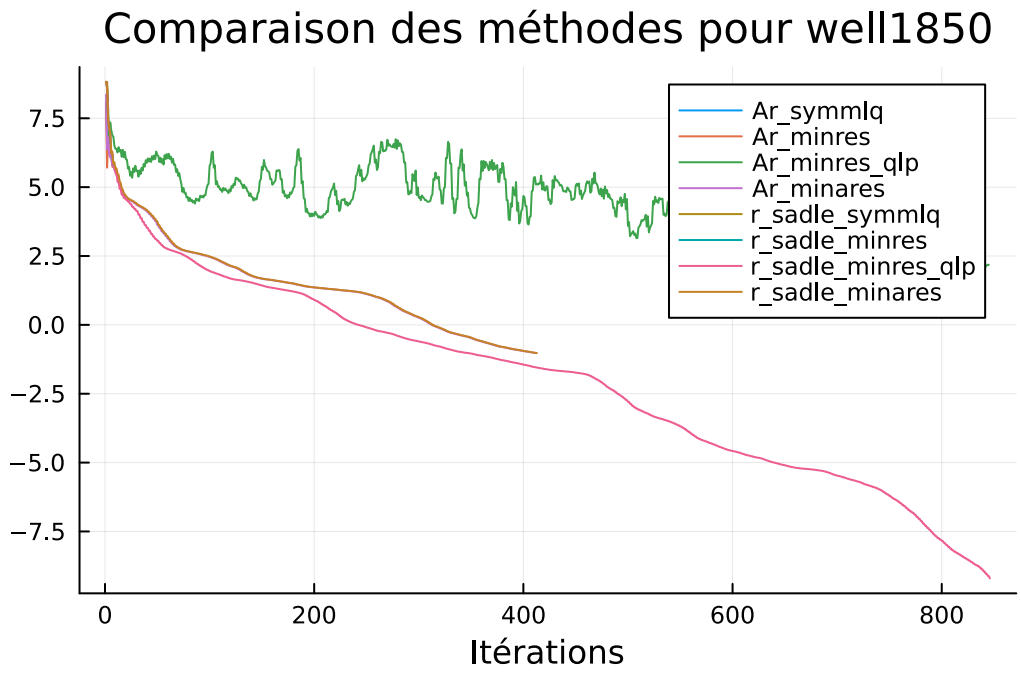
'illc1033'	Symmlq	Minres	Minres_QLP	Minares
Itérations	2	8	322	213
Norme résidu initial	2188.91	2188.91	0.0	2924.97
Norme résidu final	757.247	839.253	1.77062	36.5409
Norme résidu optimalité initial	3106.55	3106.55	0.0	4151.19
Norme résidu optimalité final	892.813	267.189	1.23682	0.430412
Statut final	true	true	true	true

'illc1033'	Symmlq	Minres	Minres_QLP	Minares
Itérations	2	8	845	412
Norme résidu initial	2270.45	2270.45	0.0	3017.27
Norme résidu final	839.225	975.763	8.93109	20.7819
Norme résidu optimalité initial	3201.55	3201.55	0.0	4254.64
Norme résidu optimalité final	927.606	305.601	8.85836	0.359389
Statut final	true	true	true	true


Comparaison des méthodes pour illc1033
Ar_symmlq
Ar_minres
Ar_minres_qlp
Ar_minares
r_sadle_symmlq
r_sadle_minres
r_sadle_minres_qlp
r_sadle_minares
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
0
500
1000
1500
2000
2500
Itérations


Comparaison des méthodes pour illc1850
Ar_symmlq
Ar_minres
Ar_minres_qlp
Ar_minares
r_sadle_symmlq
r_sadle_minres
r_sadle_minres_qlp
r_sadle_minares
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
0
1000
2000
3000
Itérations





Commenter ces résultats sur base de ce qui a été vu en classe :

D'abord, on remarque que SYMMLQ et MINRES s'arrêtent bien trop rapidement, avec des normes de résidu et de résidu d'optimalité qui sont assez élevées. Il semble donc que ces méthodes ne fonctionnent pas très bien avec les tolérances par défaut sur ces problèmes. Il faudrait probablement les resserer afin d'observer de meilleurs résultats. Dans ce cas-ci, il est difficile d'établir d'autres conclusions que cela par rapport à ces deux méthodes étant donné le peu d'itérations générées.

Ensuite, pour MINRES_QLP on remarque que le résidu du système de point de selle est monotone, alors que le résidu d'optimalité ne l'est pas, contrairement à MINARES pour laquelle ces deux quantités sont monotones. Ceci est bien ce qui est attendu, d'où la raison d'être de MINARES qui est justement conçue

pour garder le résidu d'optimalité monotone en plus du résidu. Par contre, le résidu du système de point de selle de MINRES_QLP est toujours inférieur à celui de MINARES, ce qui est logique puisque MINRES_QLP est basé sur la minimisation de ce dernier sans avoir la contrainte de chercher à rendre le résidu d'optimalité monotone comme le fait MINARES. On note que toutes ces quantités sont décroissantes en moyenne, ce qui est logique puisque l'on s'approche de plus en plus de la solution.

Par ailleurs, pour MINARES on remarque que la courbe du résidu du système de point de selle et que celle du résidu d'optimalité sont très proches l'une de l'autre, les rendant presque indistinguables entre elles, bien qu'en agrandissant on s'aperçoit que le résidu d'optimalité est toujours inférieur au résidu du système de point de selle. Ceci est logique, car le résidu du système de point de selle est un vecteur qui contient la différence entre le résidu du problème aux moindres carrés et sa définition $(r - (b - Ax))$ ainsi que le résidu d'optimalité du problème aux moindres carrés $(A'x)$. Sa norme est donc nécessairement supérieure ou égale à celle du résidu d'optimalité, et visiblement puisque le résidu produit par MINARES concorde fortement avec sa définition $(r - (b - Ax) \sim 0)$, cela ne laisse que $A'x$ comme composante significative du résidu du système de point de selle, ce qui le rend ainsi pratiquement égal au résidu d'optimalité.

Finalement, on remarque que MINARES a tendance à s'arrêter plus tôt que MINRES_QLP, qui considère même que le problème n'est pas résolu dans le cas 'illc1033'. Ce dernier cas semble être particulièrement difficile à traiter, d'où son nom signifiant "mal conditionné". Cette tendance d'arrêt peut s'expliquer de manière similaire à précédemment. En effet, le résidu d'optimalité décroît plus fortement dans le cas de MINARES, qui s'assure de le garder monotone, et si le critère d'arrêt se base sur celui-ci, alors il est logique que MINARES ait plus de facilité à s'arrêter que MINRES_QLP.

Problèmes de moindre norme

En transposant chacune des matrices `A` de la question précédente, former un système sous-déterminé consistant et résoudre le problème de moindre norme avec CRAIG, LSQR et LSMR en utilisant les tolérances par défaut. Comparer les résultats dans un tableau à l'aide de PrettyTables.jl. Votre tableau doit montrer

- le nombre d'itérations ;
- la norme de la solution initiale et finale ;
- la norme de l'erreur intiale et finale ;
- le statuts final.

Les itérés générés par ces trois méthodes vérifient-ils $Ax_k = b$ à chaque itération ? Expliquer.

Tracer sur un graphe l'évolution de la norme de la solution et de l'erreur en norme euclidienne.

```
function create_system_moindre_norme(A)
  A = A'
  n, m = size(A)
  b = A[:, 1]
  K = [sparse(1.0I,m,m) A'; A spzeros(n,n)]
  rhs = [zeros(m); b]
  return A, b, K, rhs
end
```

```
solve_LQ(A, b) = lq(Matrix(A)) \ b
```

```
A = [1]
b = [1]
x_lq = [1]

list_error_craig = []
list_error_lsqr = []
list_error_lsmr = []
list_norm_craig = []
```

```
list_norm_lsqr = []
list_norm_lsmr = []
list_r_craig = []
list_r_lsqr = []
list_r_lsmr = []
function custom_callback_craig(workspace::KrylovWorkspace, list_r_craig=list_r_craig,
 ↪  list_norm_craig=list_norm_craig, list_error_craig=list_error_craig)
  x = workspace.x
  push!(list_error_craig, norm(x - x_lq))
  push!(list_norm_craig, norm(x))
  push!(list_r_craig, norm(A*x - b))
  return false
end
function custom_callback_lsqr(workspace::KrylovWorkspace, list_r_lsqr=list_r_lsqr,
 ↪  list_norm_lsqr=list_norm_lsqr, list_error_lsqr=list_error_lsqr)
  x = workspace.x
  push!(list_error_lsqr, norm(x - x_lq))
  push!(list_norm_lsqr, norm(x))
  push!(list_r_lsqr, norm(A*x - b))
  return false
end
function custom_callback_lsmr(workspace::KrylovWorkspace, list_r_lsmr=list_r_lsmr,
 ↪  list_norm_lsmr=list_norm_lsmr, list_error_lsmr=list_error_lsmr)
  x = workspace.x
  push!(list_error_lsmr, norm(x - x_lq))
  push!(list_norm_lsmr, norm(x))
  push!(list_r_lsmr, norm(A*x - b))
  return false
end

list_names = ["illc1033", "illc1850", "well1033", "well1850"]
for i in 1:4
  name = list_names[i]
  A, b, K, rhs = get_mm(name)
  A, b, K, rhs = create_system_moindre_norme(A)

  data = Array{Any}(undef, 6, 4)
  headers = ["\'illc1033\' (transposé)", "Craig", "Lsqr", "Lsmr"]
  data[1, 1] = "Itérations"
  data[2, 1] = "Norme solution initiale"
  data[3, 1] = "Norme solution finale"
  data[4, 1] = "Norme erreur initiale"
  data[5, 1] = "Norme erreur finale"
  data[6, 1] = "Statut final"

  x_lq = solve_LQ(A, b)
  list_error_craig = []
  list_error_lsqr = []
  list_error_lsmr = []
  list_norm_craig = []
  list_norm_lsqr = []
  list_norm_lsmr = []
  list_r_craig = []
  list_r_lsqr = []
```

```
  list_r_lsmr = []

  (x_craig, y_craig, stats_craig) = craig(A, b, history=true,
↪   callback=custom_callback_craig)
  (x_lsqr, stats_lsqr) = lsqr(A, b, history=true, callback=custom_callback_lsqr)
  (x_lsmr, stats_lsmr) = lsmr(A, b, history=true, callback=custom_callback_lsmr)

  data[1, 2] = stats_craig.niter
  data[1, 3] = stats_lsqr.niter
  data[1, 4] = stats_lsmr.niter
  data[2, 2] = list_norm_craig[1]
  data[2, 3] = list_norm_lsqr[1]
  data[2, 4] = list_norm_lsmr[1]
  data[3, 2] = list_norm_craig[end]
  data[3, 3] = list_norm_lsqr[end]
  data[3, 4] = list_norm_lsmr[end]
  data[4, 2] = list_error_craig[1]
  data[4, 3] = list_error_lsqr[1]
  data[4, 4] = list_error_lsmr[1]
  data[5, 2] = list_error_craig[end]
  data[5, 3] = list_error_lsqr[end]
  data[5, 4] = list_error_lsmr[end]
  data[6, 2] = stats_craig.solved
  data[6, 3] = stats_lsqr.solved
  data[6, 4] = stats_lsmr.solved

  pretty_table(data, header=headers)

  p = plot(log.(list_norm_craig), label="norm_craig", legend=:topright)
  plot!(p, log.(list_norm_lsqr), label="norm_lsqr")
  plot!(p, log.(list_norm_lsmr), label="norm_lsmr")

  plot!(p, log.(list_error_craig), label="error_craig")
  plot!(p, log.(list_error_lsqr), label="error_lsqr")
  plot!(p, log.(list_error_lsmr), label="error_lsmr")

  plot!(p, log.(list_r_craig), label="r_craig")
  plot!(p, log.(list_r_lsqr), label="r_lsqr")
  plot!(p, log.(list_r_lsmr), label="r_lsmr")

  xlabel!("Itérations")
  title!("Comparaison des méthodes pour " * name)

  display(p)
end
```

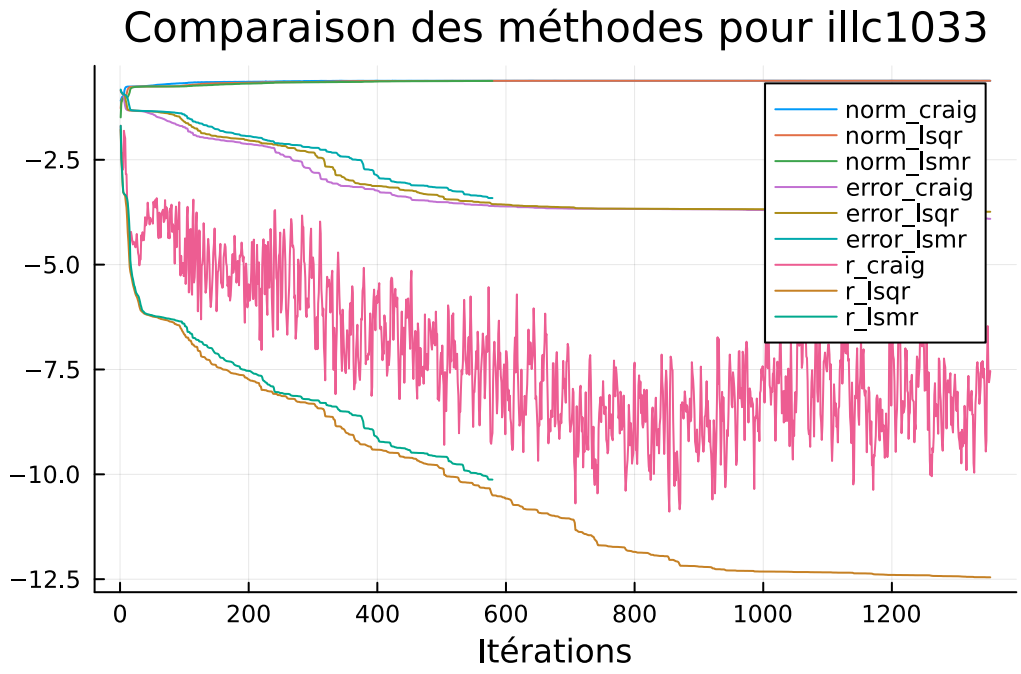
'illc1033' (transposé)	Craig	Lsqr	Lsmr
Itérations	1353	1353	579
Norme solution initiale	0.334557	0.280034	0.225318
Norme solution finale	0.540001	0.539952	0.537671
Norme erreur initiale	0.424592	0.428079	0.43842
Norme erreur finale	0.020081	0.023773	0.032943

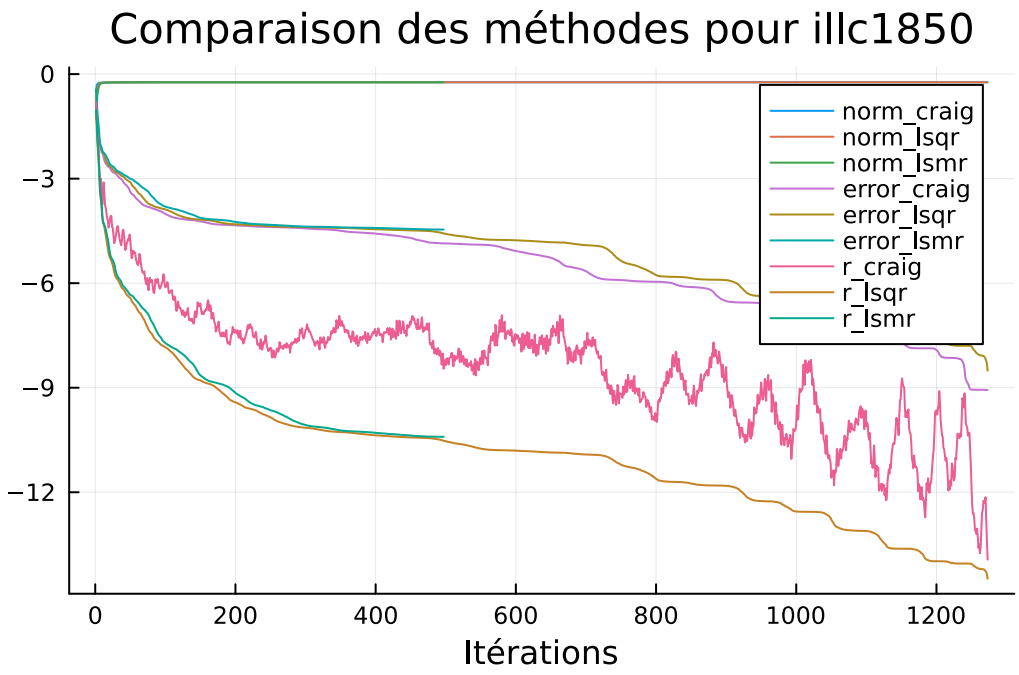
Statut final	false	false	true

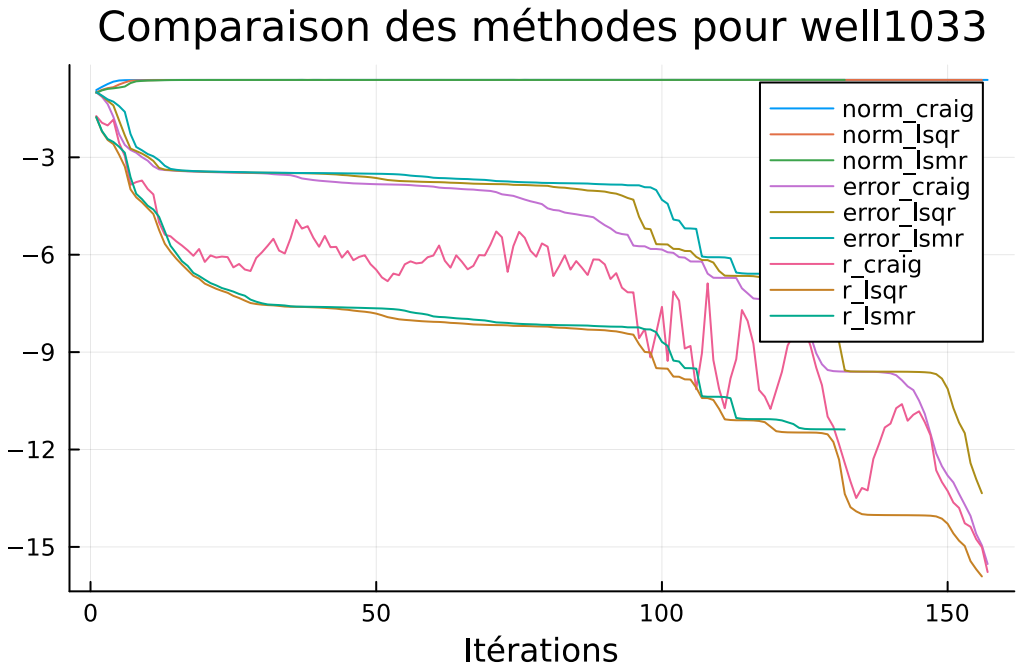
'illc1033' (transposé)	Craig	Lsqr	Lsmr
Itérations	1273	1273	497
Norme solution initiale	0.516752	0.32215	0.280215
Norme solution finale	0.791849	0.791849	0.791741
Norme erreur initiale	0.599993	0.630763	0.644936
Norme erreur finale	0.000115484	0.000201301	0.0115509
Statut final	true	true	true

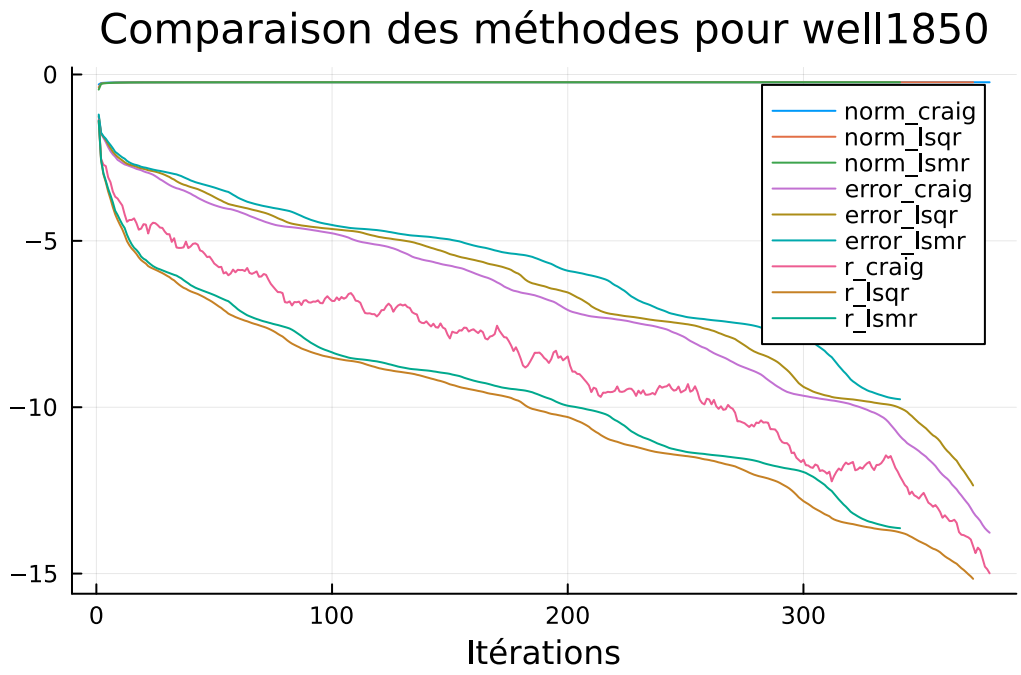
'illc1033' (transposé)	Craig	Lsqr	Lsmr
Itérations	157	156	132
Norme solution initiale	0.394786	0.368067	0.360432
Norme solution finale	0.540562	0.540562	0.540554
Norme erreur initiale	0.369258	0.370223	0.370852
Norme erreur finale	1.80549e-7	1.5972e-6	0.0010257
Statut final	true	true	true

'illc1033' (transposé)	Craig	Lsqr	Lsmr
Itérations	379	372	341
Norme solution initiale	0.74024	0.680227	0.634726
Norme solution finale	0.791849	0.791849	0.791849
Norme erreur initiale	0.281191	0.287524	0.300336
Norme erreur finale	1.04433e-6	4.31909e-6	5.76583e-5
Statut final	true	true	true




Comparaison des méthodes pour illc1850
0
−3
−6
−9
−12
0
200
400
600
800
1000
1200
Itérations
norm_craig
norm_lsqr
norm_lsmr
error_craig
error_lsqr
error_lsmr
r_craig
r_lsqr
r_lsmr


Comparaison des méthodes pour well1033
−3
−6
−9
−12
−15
0
50
100
150
Itérations
norm_craig
norm_lsqr
norm_lsmr
error_craig
error_lsqr
error_lsmr
r_craig
r_lsqr
r_lsmr



Commenter ces résultats sur base de ce qui a été vu en classe :

Tout d'abord, la norme de la solution est croissante et monotone pour CRAIG, LSQR et LSMR. C'est bien ce qui est attendu, car on part d'une solution nulle pour la raffiner au fur et à mesure que l'on construit les espaces de Krylov au cours des itérations, ce qui l'accroît de plus en plus avec pour limite la norme de la solution exacte.

Le résidu a été tracé afin de constater que les itérés ne vérifient pas exactement $Ax_k = b$ au cours des itérations, autrement dit que le résidu $r = b - Ax_k$ n'est pas nul. En effet, on commence avec une solution nulle et ainsi un résidu $r = b$, pour la raffiner et ainsi le diminuer au fur et à mesure que l'on améliore la solution. Les itérés tentent donc de vérifier $Ax_k = b$ du mieux qu'ils peuvent selon les choix des différentes méthodes au cours des itérations, sans que cela ne soit une contrainte absolue. De ce fait, le résidu et l'erreur sont décroissants en moyenne pour ces trois méthodes, ce qui est logique puisque l'on s'approche de plus en plus de la solution. Pour la même raison que précédemment, le résidu et l'erreur de LSQR sont toujours inférieurs à ceux de LSMR, et ils sont monotones dans les deux cas, ce qui est attendu. Par contre, pour CRAIG le résidu n'est pas monotone, car la condition de Galerkin utilisée dans le cadre de cette méthode est directement liée au fait de se concentrer pour minimiser l'erreur plutôt que le résidu ou le résidu d'optimalité, ce pourquoi on remarque que les erreurs de LSQR et de LSMR sont toujours supérieures à celle de CRAIG.

Finalement, on remarque que LSMR a tendance à s'arrêter avant CRAIG et LSQR, qui semblent s'arrêter presque en même temps. Encore une fois, ceci peut s'expliquer par le fait que seul LSMR se consacre à la minimisation du résidu d'optimalité de façon monotone, ce qui l'aide à décroître de façon plus importante, et ainsi donne à LSMR une plus grande facilité à s'arrêter par rapport aux autres méthodes (en supposant que le critère d'arrêt lui accorde effectivement une grande importance). On note que le cas 'illc1033' semble encore une fois difficile à traiter pour ces autres méthodes, qui le considèrent non-résolu contrairement à LSMR.

Méthodes basées sur Lanczos

Répéter la question précédente en appliquant MINRES, MINRES-QLP, MINARES et SYMMLQ sur le système de point de selle et produire un tableau semblable.

Tracer sur un graphe l'évolution du résidu d'optimalité du problème aux moindres carrés et du résidu du système de point de selle.

```
A = [1]
b = [1]
x_lq = [1]
m = 1

list_norm_symmlq = []
list_norm_minres = []
list_norm_minres_qlp = []
list_norm_minares = []
list_error_symmlq = []
list_error_minres = []
list_error_minres_qlp = []
list_error_minares = []
list_Ar_symmlq = []
list_Ar_minres = []
list_Ar_minres_qlp = []
list_Ar_minares = []
function custom_callback_symmlq(workspace::KrylovWorkspace,
 ↪  list_error_symmlq=list_error_symmlq, list_norm_symmlq=list_norm_symmlq,
 ↪  list_Ar_symmlq=list_Ar_symmlq)
  x = workspace.x[1:m]
  r = A*x - b
  push!(list_error_symmlq, norm(x - x_lq))
  push!(list_norm_symmlq, norm(x))
  push!(list_Ar_symmlq, norm(A'*r))
  return false
end
function custom_callback_minres(workspace::KrylovWorkspace,
 ↪  list_error_minres=list_error_minres, list_norm_minres=list_norm_minres,
 ↪  list_Ar_minres=list_Ar_minres)
  x = workspace.x[1:m]
  r = A*x - b
  push!(list_error_minres, norm(x - x_lq))
  push!(list_norm_minres, norm(x))
  push!(list_Ar_minres, norm(A'*r))
  return false
end
function custom_callback_minres_qlp(workspace::KrylovWorkspace,
 ↪  list_error_minres_qlp=list_error_minres_qlp,
 ↪  list_norm_minres_qlp=list_norm_minres_qlp, list_Ar_minres_qlp=list_Ar_minres_qlp)
  x = workspace.x[1:m]
  r = A*x - b
  push!(list_error_minres_qlp, norm(x - x_lq))
  push!(list_norm_minres_qlp, norm(x))
  push!(list_Ar_minres_qlp, norm(A'*r))
  return false
end
function custom_callback_minares(workspace::KrylovWorkspace,
 ↪  list_error_minares=list_error_minares, list_norm_minares=list_norm_minares,
 ↪  list_Ar_minares=list_Ar_minares)
  x = workspace.x[1:m]
  r = A*x - b
  push!(list_error_minares, norm(x - x_lq))
  push!(list_norm_minares, norm(x))
```

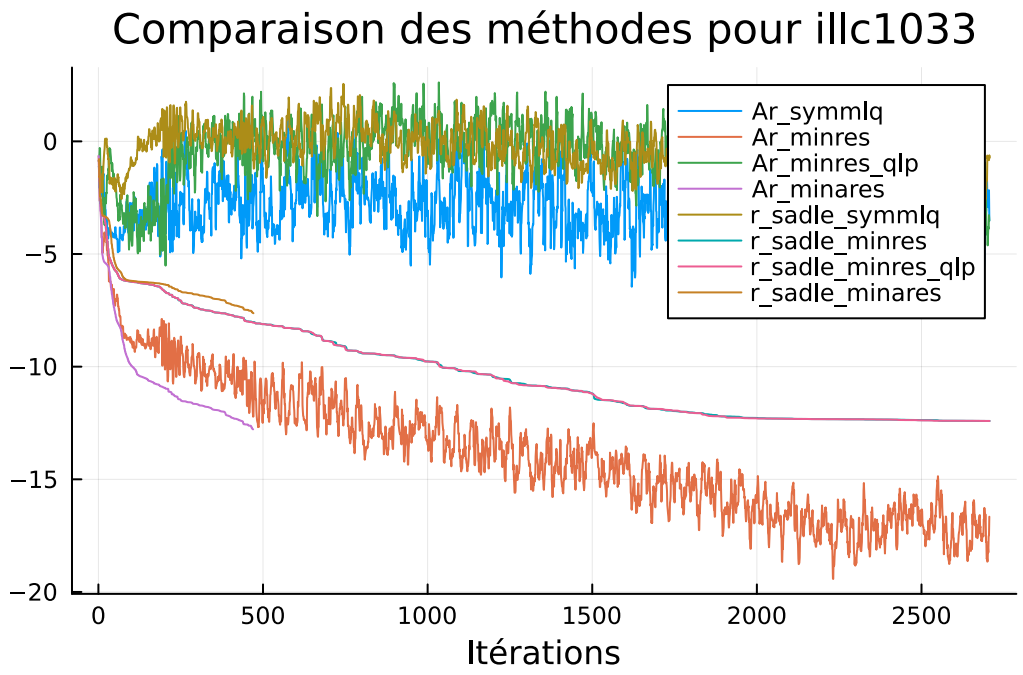
```
  push!(list_Ar_minares, norm(A'*r))
  return false
end

list_names = ["illc1033", "illc1850", "well1033", "well1850"]
for i in 1:4
  name = list_names[i]
  A, b, K, rhs = get_mm(name)
  A, b, K, rhs = create_system_moindre_norme(A)

  data = Array{Any}(undef, 6, 5)
  headers = ["\'illc1033\' (transposé)", "Symmlq", "Minres", "Minres_QLP", "Minares"]
  data[1, 1] = "Itérations"
  data[2, 1] = "Norme solution initiale"
  data[3, 1] = "Norme solution finale"
  data[4, 1] = "Norme erreur initiale"
  data[5, 1] = "Norme erreur finale"
  data[6, 1] = "Statut final"

  n, m = size(A)
  list_norm_symmlq = []
  list_norm_minres = []
  list_norm_minres_qlp = []
  list_norm_minares = []
  list_error_symmlq = []
  list_error_minres = []
  list_error_minres_qlp = []
  list_error_minares = []
  list_Ar_symmlq = []
  list_Ar_minres = []
  list_Ar_minres_qlp = []
  list_Ar_minares = []

  x_lq = solve_LQ(A, b)
  (sol_symmlq, stats_symmlq) = symmlq(K, rhs, history=true,
↪  callback=custom_callback_symmlq)
  (sol_minres, stats_minres) = minres(K, rhs, history=true,
↪  callback=custom_callback_minres)
  (sol_minres_qlp, stats_minres_qlp) = minres_qlp(K, rhs, history=true,
↪  callback=custom_callback_minres_qlp)
  (sol_minares, stats_minares) = minares(K, rhs, history=true,
↪  callback=custom_callback_minares)

  data[1, 2] = stats_symmlq.niter
  data[1, 3] = stats_minres.niter
  data[1, 4] = stats_minres_qlp.niter
  data[1, 5] = stats_minares.niter
  data[2, 2] = list_norm_symmlq[1]
  data[2, 3] = list_norm_minres[1]
  data[2, 4] = list_norm_minres_qlp[1]
  data[2, 5] = list_norm_minares[1]
  data[3, 2] = list_norm_symmlq[end]
  data[3, 3] = list_norm_minres[end]
  data[3, 4] = list_norm_minres_qlp[end]
```

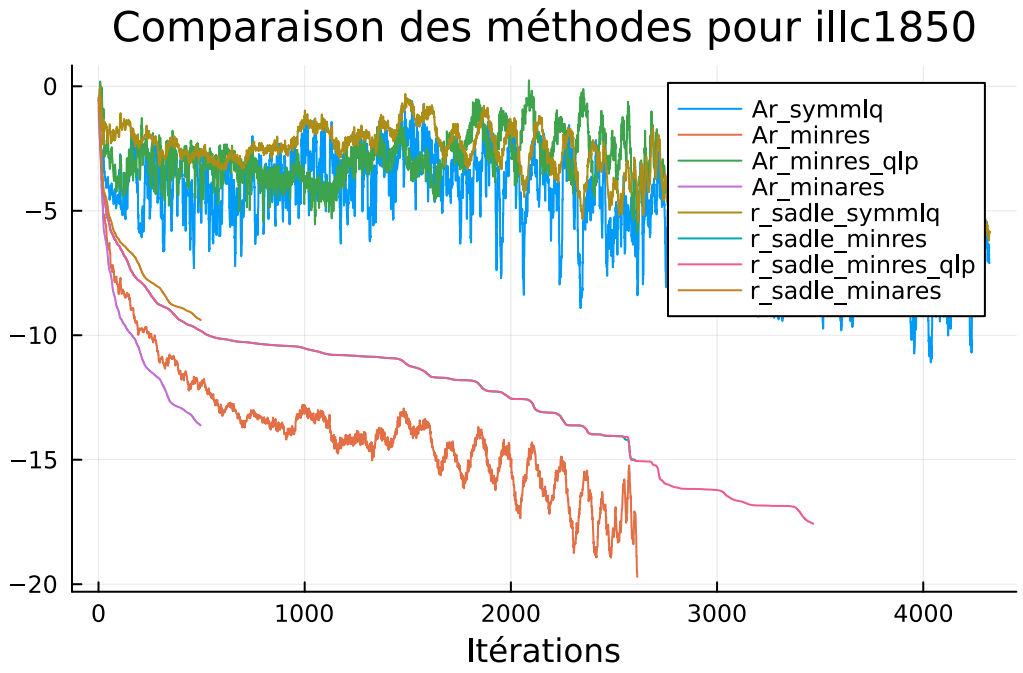
```
  data[3, 5] = list_norm_minares[end]
  data[4, 2] = list_error_symmlq[1]
  data[4, 3] = list_error_minres[1]
  data[4, 4] = list_error_minres_qlp[1]
  data[4, 5] = list_error_minares[1]
  data[5, 2] = list_error_symmlq[end]
  data[5, 3] = list_error_minres[end]
  data[5, 4] = list_error_minres_qlp[end]
  data[5, 5] = list_error_minares[end]
  data[6, 2] = stats_symmlq.solved
  data[6, 3] = stats_minres.solved
  data[6, 4] = stats_minres_qlp.solved
  data[6, 5] = stats_minares.solved

  pretty_table(data, header=headers)

  p = plot(log.(list_Ar_symmlq), label="Ar_symmlq", legend=:topright)
  plot!(p, log.(list_Ar_minres), label="Ar_minres")
  plot!(p, log.(list_Ar_minres_qlp), label="Ar_minres_qlp")
  plot!(p, log.(list_Ar_minares), label="Ar_minares")

  plot!(p, log.(stats_symmlq.residuals), label="r_sadle_symmlq")
  plot!(p, log.(stats_minres.residuals), label="r_sadle_minres")
  plot!(p, log.(stats_minres_qlp.residuals), label="r_sadle_minres_qlp")
  plot!(p, log.(stats_minares.residuals), label="r_sadle_minares")

  xlabel!("Itérations")
  title!("Comparaison des méthodes pour " * name)

  display(p)
end
```

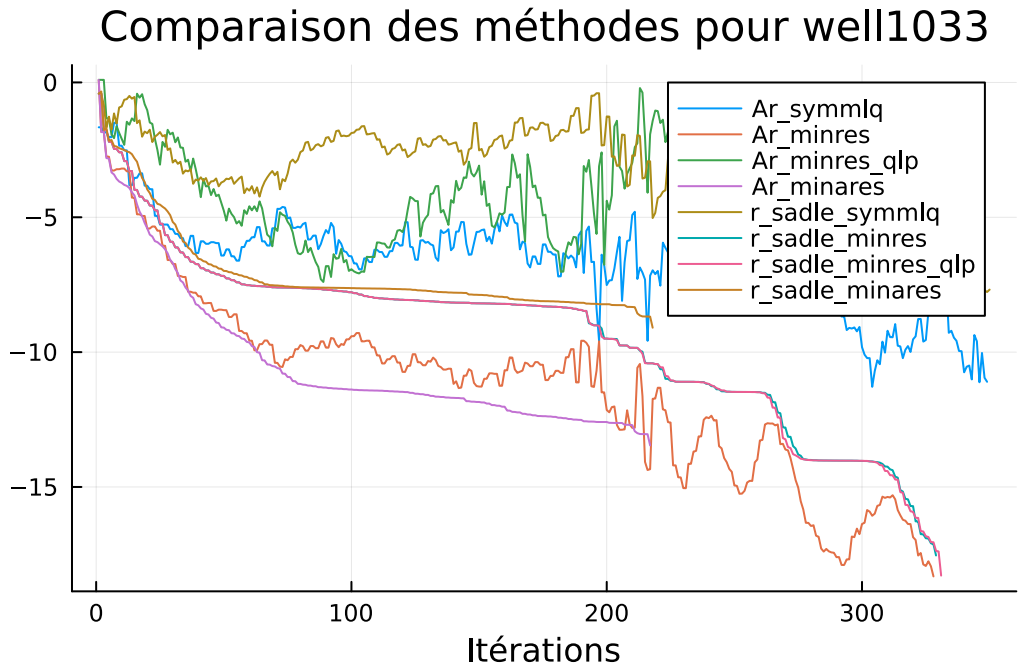
```
 'illc1033' (transposé)     Symmlq     Minres   Minres_QLP    Minares

             Itérations       2706       2706         2706       470
 Norme solution initiale   0.334557        0.0          0.0        0.0
   Norme solution finale   0.546083   0.539945     0.539088   0.505256
   Norme erreur initiale   0.424592   0.540562     0.540562   0.540562
     Norme erreur finale  0.0408896  0.0243614    0.0402706   0.139928
            Statut final      false      false        false       true


 'illc1033' (transposé)       Symmlq      Minres   Minres_QLP     Minares

             Itérations         4321        2614         3466         495
 Norme solution initiale     0.516752         0.0          0.0         0.0
   Norme solution finale     0.791848    0.791849     0.791898    0.791658
   Norme erreur initiale     0.599993    0.791849     0.791849    0.791849
     Norme erreur finale  0.000234493  0.00012675   0.00345132   0.0137568
            Statut final         true        true         true        true


 'illc1033' (transposé)      Symmlq      Minres   Minres_QLP     Minares
```

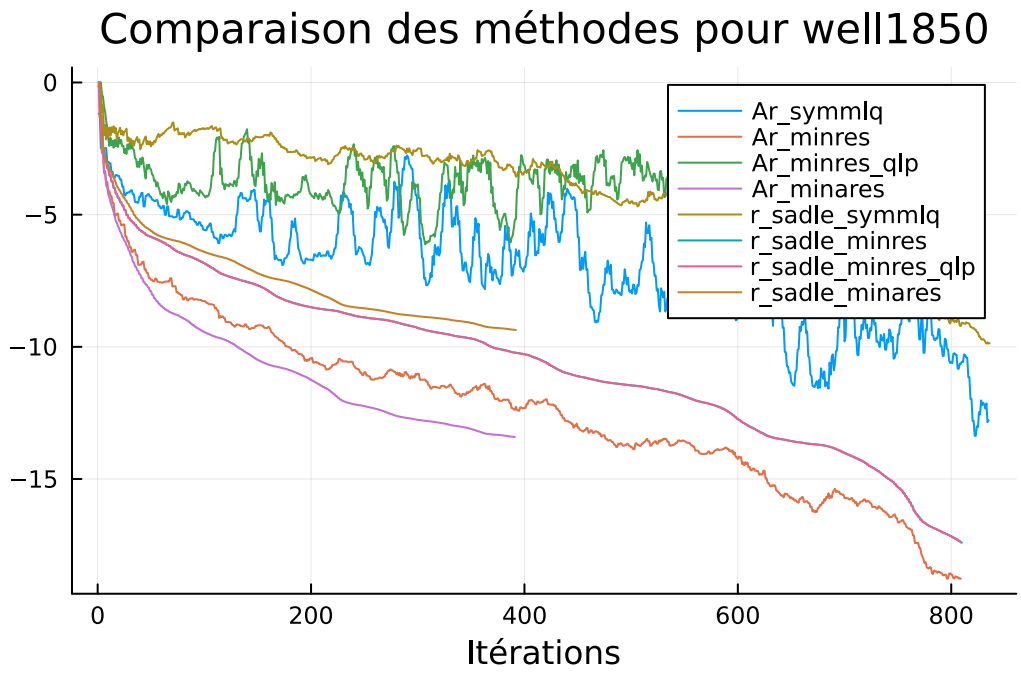
Itérations	349	328	330	217
Norme solution initiale	0.394786	0.0	0.0	0.0
Norme solution finale	0.540559	0.540562	0.540699	0.540159
Norme erreur initiale	0.369258	0.540562	0.540562	0.540562
Norme erreur finale	1.51197e-5	1.02201e-7	0.00165177	0.00903715
Statut final	true	true	true	true

'illc1033' (transposé)	Symmlq	Minres	Minres_QLP	Minares
Itérations	835	809	808	391
Norme solution initiale	0.74024	0.0	0.0	0.0
Norme solution finale	0.791849	0.791849	0.79187	0.791775
Norme erreur initiale	0.281191	0.791849	0.791849	0.791849
Norme erreur finale	1.19824e-6	4.29613e-7	0.00330549	0.00506133
Statut final	true	true	true	true




Comparaison des méthodes pour illc1850
0
−5
−10
−15
−20
0
1000
2000
3000
4000
Itérations
Ar_symmlq
Ar_minres
Ar_minres_qlp
Ar_minares
r_sadle_symmlq
r_sadle_minres
r_sadle_minres_qlp
r_sadle_minares


Comparaison des méthodes pour well1033
0
−5
−10
−15
0
100
200
300
Itérations
Ar_symmlq
Ar_minres
Ar_minres_qlp
Ar_minares
r_sadle_symmlq
r_sadle_minres
r_sadle_minres_qlp
r_sadle_minares



Commenter ces résultats sur base de ce qui a été vu en classe :

Cette fois-ci, les quatres méthodes s'arrêtent dans un nombre significatif d'itérations, ce qui suggèrent qu'elles fonctionnent correctement pour les cas à l'étude avec les tolérances par défaut.

On remarque que le résidu du système de point de selle et que le résidu d'optimalité du problème aux moindres carrés sont décroissants en moyenne, ce qui est logique puisque l'on s'approche de plus en plus de la solution. Les résidus du système de point de selle sont monotones pour MINARES, MINRES et MINRES_QLP, ce qui est attendu. Par contre, seul MINARES possède un résidu d'optimalité monotone, ce qui est également attendu puisque MINARES est la seule méthode se concentrant pour le minimiser de façon monotone, contrairement à MINRES et MINRES_QLP qui se consacrent entièrement à la minimisation du résidu. Ainsi, comme précédemment, ceci vient au prix d'avoir un résidu du système de point de selle toujours supérieur à celui de MINRES et de MINRES_QLP, qui semblent avoir exactement le même résidu. Cela vient probablement du fait que MINRES et MINRES_QLP sont en réalité pratiquement la même méthode, outre le fait que MINRES_QLP effectue s'il y a lieu des manipulations supplémentaires liées à une factorisation LQ pour s'assurer de trouver la solution de norme minimale lorsque le système est sous-déterminé. D'ailleurs, peut-être sont-ce ces manipulations supplémentaires qui ont permis à MINRES_QLP de fonctionner correctement dans le cas des problèmes aux moindres carrés présentés plus tôt, par opposition à MINRES. De plus, on remarque que le résidu du système de point de selle et que le résidu d'optimalité ne sont pas monotones dans le cas de SYMMLQ, ce qui est attendu, car cette méthode est intimement liée à CG et se consacre à la minimisation de l'erreur pour des matrices qui n'ont pas besoin d'être définies positives plutôt qu'à la minimisation du résidu ou du résidu d'optimalité.

Qui plus est, pour MINARES on remarque que les courbes du résidu du système de point de selle et du résidu d'optimalité ne sont plus proches l'une de l'autre comme c'était le cas auparavant. Ceci est dû à la structure du système de point de selle, dont le résidu est maintenant composé de la différence entre les multiplicateurs de Lagrange et leur définition $(x - A'y)$ ainsi que du résidu du problème de moindre norme $(Ax - b)$. Le résidu d'optimalité n'apparaît donc plus ici.

Finalement, on remarque que MINARES a tendance à s'arrêter plus tôt que toutes les autres méthodes, qui semblent s'arrêter environ en même temps. Encore une fois, ceci peut s'expliquer par le fait que seul MINARES s'assure de minimiser le résidu d'optimalité de façon monotone, ce qui le fait décroître de façon plus importante et permet ainsi à MINARES de s'arrêter plus tôt (en supposant que le critère d'arrêt accorde une grande importance au résidu d'optimalité). On note que le cas 'illc1033' semble une fois de plus difficile

à traiter pour les autres méthodes, qui le considèrent non-résolu contrairement à MINARES.

Notes

Les résidus d'optimalité des problèmes aux moindres carrés ou des problèmes de moindre norme mentionnés dans les différents cas ci-dessus ne sont techniquement pas les mêmes que les résidus d'optimalité des systèmes de point de selle. Or, il semble y avoir une connection forte entre ces résidus d'optimalité, car les tendances des méthodes concernant le résidu d'optimalité du système de point de selle semblent concorder avec le résidu du problème aux moindres carrés ou du problème de moindre norme selon le cas.