Rapport de laboratoire 2

MTH8408

Yasmine Amami

```
using Pkg
Pkg.activate("labo2_env")
using LinearAlgebra
using Printf
```

Contexte

Dans ce laboratoire, on demande d'implémenter deux méthodes itératives pour résoudre

$$\min_x \; g^T x + \tfrac{1}{2} x^T H x \tag{1}$$

où $g \in \mathbb{R}^n$ et H est une matrice $n \times n$ symétrique et définie positive.

Question 1

En cours, nous avons vu la méthode de la plus forte pente avec recherche linéaire exacte pour résoudre (1).

Dans cette question, on demande d'implémenter et de tester cette méthode sur divers objectifs quadratiques convexes.

Votre implémentation doit avoir les caractéristiques suivantes :

1. un critère d'arrêt absolu et relatif sur le gradient de l'objectif ;
2. un critère d'arrêt portant sur le nombre d'itérations (le nombre maximum d'itérations devrait dépendre du nombre de variables n du problème) ;
3. toujours démarrer de l'approximation initiale 0 ;
4. allouer un minimum en utilisant les opérations vectorisées (`.=`, `.+`, `.+=`, etc.) autant que possible ;
5. calculer *un seul* produit entre H et un vecteur par itération ;
6. n'utiliser H qu'à travers des produits avec un vecteur (ne pas accéder aux éléments de H ou indexer dans H) ;
7. ne dépendre que de `LinearAlgebra`.
8. votre fonction principale doit être documentée—reportez-vous à https://docs.julialang.org/en/v1/manual/documentation ;
9. votre fonction doit faire afficher les informations pertinentes à chaque itération sous forme de tableau comme vu en cours.

Tester votre implémentation sur les problèmes quadratiques de la section *Problèmes test* ci-dessous.

```
"""
    steepest_qp(g, H, eps_a=1.0e-5, eps_r=1.0e-5)

# Arguments
- `g`: vecteur gradient (dimension n)
```

```
- `H`: matrice symetrique positive definie (n×n)
- `eps_a`: tolerance absolue pour la norme du gradient (par defaut: 1e-5)
- `eps_r`: tolerance relative pour la norme du gradient (par defaut: 1e-5)

# Retours
- `x`: solution optimale
- `iter`: nombre d'iterations
- `converged`: true si converge, false si max iterations atteint
"""
function steepest_qp(g, H, eps_abs=1.0e-5, eps_rel=1.0e-5)
    n = length(g)
    max_iter = n
    x = zeros(n)
    history = []
    x_exact = -H \ g

    println("Iter\t\t||résidu relatif||\t||erreur relatif||")

    for k in 1:max_iter
        gradient = g + H * x
        grad_norm = norm(gradient)

        # Critère d'arrêt
        if grad_norm   eps_abs + eps_rel * norm(g)
            break
        end

        # Direction de descente
        direction = -gradient

        # Produit unique H * direction
        H_direction = H * direction

        # Recherche linéaire exacte
        alpha = dot(gradient, gradient) / dot(direction, H_direction)

        # Mise à jour
        x .+= alpha * direction

        # Évaluation de la fonction
        f_x = dot(g, x) + 0.5 * dot(x, H * x)

        # Calcul de l'erreur
        error = norm(x_exact - x)

        println(@sprintf("%d\t\t\t%.3e\t\t%.3e", k, grad_norm, error))


        # Historique
        push!(history, (iteration=k, x=copy(x), grad_norm=grad_norm))
    end
    return x, history
end
```

`Main.Notebook.steepest_qp`

Question 2

Dans cette question, on demande d'implémenter la méthode BFGS pour résoudre le problème quadratique convexe (1).

Votre implémentation doit avoir les mêmes caractéristiques qu'à la question 1.

Ici, on cherche notamment à valider le résultat disant que la méthode se termine en au plus n itérations (en arithmétique exacte) et reconstruit H, c'est-à-dire que $B_k = H$ à la convergence.

Tester votre implémentation sur les problèmes quadratiques de la section *Problèmes test* ci-dessous.

```
"""
    bfgs_qp(g, H, eps_a=1.0e-5, eps_r=1.0e-5)

# Arguments
- `g`: vecteur gradient (dimension n)
- `H`: matrice symetrique positive definie (n×n)
- `eps_a`: tolerance absolue pour la norme du gradient (par defaut: 1e-5)
- `eps_r`: tolerance relative pour la norme du gradient (par defaut: 1e-5)

# Retours
- `x`: solution optimale
- `iter`: nombre d'iterations
- `converged`: true si converge, false si max iterations atteint
- `B`: approximation  BFGS
"""
function bfgs_qp(g, H, eps_abs=1.0e-5, eps_rel=1.0e-5)
    n = length(g)
    max_iter = n
    x = zeros(n)
    B = Matrix(1.0I, n, n)  # Approximation initiale de H ¹
    history = []
    x_exact = -H \ g

    gradient_prev = g + H * x

    println("Iter\t\t||résidu relatif||\t||erreur relatif||")

    for k in 1:max_iter
        gradient = g + H * x
        grad_norm = norm(gradient)

        # Critère d'arrêt
        if grad_norm   eps_abs + eps_rel * norm(g)
            break
        end

        # Direction de descente
        direction = -B * gradient

        # Produit unique H * direction
        H_direction = H * direction
```

```
        # Recherche linéaire exacte
        alpha = -dot(gradient, direction) / dot(direction, H_direction)

        # Mise à jour de x
        step = alpha * direction
        x .+= step

        # Mise à jour du gradient
        gradient_new = g + H * x
        y = gradient_new - gradient

        # Mise à jour BFGS
        rho = 1.0 / dot(y, step)
        I_n = I(n)
        V = I_n .- rho .* step * y'
        B .= V * B * V' .+ rho .* step * step'


        f_x = dot(g, x) + 0.5 * dot(x, H * x)

        # Calcul de l'erreur
        error = norm(x_exact - x)

        println(@sprintf("%d\t\t\t%.3e\t\t%.3e", k, grad_norm, error))

        # Historique
        push!(history, (iteration=k, x=copy(x), grad_norm=grad_norm))
        gradient_prev .= gradient_new
    end
    return x, history
end
```

Main.Notebook.bfgs_qp

Résultats numériques

Problèmes test

Votre premier problème test sera généré aléatoirement avec $n = 10$.

```
n = 10
g_rand = ones(n)
H_rand = randn(n, n)
H = H_rand' * H_rand
x, history = bfgs_qp(g_rand, H)
```

```
Iter        ||résidu relatif||  ||erreur relatif||
1           3.162e+00       4.437e+02
2           4.291e+00       4.433e+02
3           5.679e+00       4.427e+02
4           4.226e+00       4.423e+02
5           2.178e+00       4.414e+02
6           2.501e+00       4.394e+02
7           2.568e+00       4.344e+02
```

```
8           5.136e+00       4.286e+02
9           6.733e+00       3.124e+02
10          7.056e+00       3.262e-11
```

```
([-110.29693511161757, -178.23548014169717, -135.97207203157046, 76.0610125588469, -103.98508964802261,
```

Utiliser ensuite les problèmes quadratiques convexes de la collection Maros et Meszaros. Vous pouvez y accéder à l'aide de l'extrait de code suivant :

```
Pkg.add("QPSReader")  # collection + outils pour lire les problèmes

using QPSReader
using Logging
using SparseArrays

function get_gH(name, reg=0)
    mm_path = fetch_mm()  # chemin vers les problèmes sur votre disque
    qpdata = with_logger(Logging.NullLogger()) do
        readqps(joinpath(mm_path, name))
    end
    n = qpdata.nvar
    g = qpdata.c
    H = Symmetric(sparse(qpdata.qrows, qpdata.qcols, qpdata.qvals, n, n) + reg * I, :L)
    return g, H
end
```

Les noms des problèmes sont listés sur https://bitbucket.org/optrove/maros-meszaros/src/master/.

Leurs dimensions sont donnés dans le tableau sur la page https://www.doc.ic.ac.uk/%7Eim/00README.QP (avec des noms qui ne correspondent pas tout à fait ; les noms corrects sont ceux de la page Bitbucket).

NB : ces problèmes ont des contraintes, mais dans ce laboratoire, on les ignore.

Choisissez 3 problèmes :

- un avec $n \approx 10$;
- un avec $n \approx 50$;
- un avec $n \approx 100$.

```
g10, H10 = get_gH("DUALC1.SIF ", 1.0e-3)
g50, H50 = get_gH("QCAPRI.SIF", 1.0e-3)
g100, H100 = get_gH("QPILOTNO.SIF", 1.0e-3)
```

```
([0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0  …  -0.566583, -0.89, -0.853296, -0.75723, -0.634515
```

Attention :

- il se peut que $g = 0$—dans ce cas, changez g en `ones(n)` ;
- il se peut que H soit seulement semi-définie positive et pas définie positive—dans ce cas, ajoutez-lui un petit multiple de l'identité via, par exemple,

```
g, H = get_gH(name, 1.0e-3)
```

Validation de la méthode de la plus forte pente

```
steepest_qp(g10, H10)
steepest_qp(g50, H50)
steepest_qp(g100, H100)
```

```
Iter        ||résidu relatif||  ||erreur relatif||
```

```
1           3.425e+06       3.417e+02
2           2.149e+06       3.416e+02
3           2.697e+06       3.416e+02
4           1.704e+06       3.416e+02
5           2.159e+06       3.416e+02
6           1.375e+06       3.416e+02
7           1.762e+06       3.416e+02
8           1.132e+06       3.416e+02
9           1.470e+06       3.416e+02
Iter        ||résidu relatif||  ||erreur relatif||
1           2.361e+00       1.865e+03
2           3.087e+00       1.864e+03
3           3.322e+00       1.864e+03
4           2.493e+00       1.864e+03
5           2.960e+00       1.864e+03
6           2.429e+00       1.864e+03
7           2.923e+00       1.864e+03
8           2.424e+00       1.864e+03
9           2.918e+00       1.864e+03
10          2.424e+00       1.864e+03
11          2.916e+00       1.864e+03
12          2.426e+00       1.864e+03
13          2.915e+00       1.863e+03
14          2.430e+00       1.863e+03
15          2.911e+00       1.863e+03
16          2.437e+00       1.863e+03
17          2.902e+00       1.863e+03
18          2.442e+00       1.863e+03
19          2.890e+00       1.863e+03
20          2.445e+00       1.863e+03
21          2.881e+00       1.863e+03
22          2.445e+00       1.863e+03
23          2.877e+00       1.863e+03
24          2.445e+00       1.863e+03
25          2.876e+00       1.863e+03
26          2.445e+00       1.862e+03
27          2.875e+00       1.862e+03
28          2.445e+00       1.862e+03
29          2.875e+00       1.862e+03
30          2.445e+00       1.862e+03
31          2.874e+00       1.862e+03
32          2.445e+00       1.862e+03
33          2.874e+00       1.862e+03
34          2.444e+00       1.862e+03
35          2.874e+00       1.862e+03
36          2.444e+00       1.862e+03
37          2.874e+00       1.862e+03
38          2.444e+00       1.862e+03
39          2.873e+00       1.862e+03
40          2.444e+00       1.861e+03
41          2.873e+00       1.861e+03
42          2.444e+00       1.861e+03
43          2.873e+00       1.861e+03
44          2.443e+00       1.861e+03
```

45	2.873e+00	1.861e+03
46	2.443e+00	1.861e+03
47	2.873e+00	1.861e+03
48	2.443e+00	1.861e+03
49	2.872e+00	1.861e+03
50	2.443e+00	1.861e+03
51	2.872e+00	1.861e+03
52	2.443e+00	1.861e+03
53	2.872e+00	1.861e+03
54	2.442e+00	1.860e+03
55	2.872e+00	1.860e+03
56	2.442e+00	1.860e+03
57	2.871e+00	1.860e+03
58	2.442e+00	1.860e+03
59	2.871e+00	1.860e+03
60	2.442e+00	1.860e+03
61	2.871e+00	1.860e+03
62	2.442e+00	1.860e+03
63	2.871e+00	1.860e+03
64	2.442e+00	1.860e+03
65	2.871e+00	1.860e+03
66	2.441e+00	1.860e+03
67	2.870e+00	1.860e+03
68	2.441e+00	1.859e+03
69	2.870e+00	1.859e+03
70	2.441e+00	1.859e+03
71	2.870e+00	1.859e+03
72	2.441e+00	1.859e+03
73	2.870e+00	1.859e+03
74	2.441e+00	1.859e+03
75	2.869e+00	1.859e+03
76	2.440e+00	1.859e+03
77	2.869e+00	1.859e+03
78	2.440e+00	1.859e+03
79	2.869e+00	1.859e+03
80	2.440e+00	1.859e+03
81	2.869e+00	1.859e+03
82	2.440e+00	1.858e+03
83	2.869e+00	1.858e+03
84	2.440e+00	1.858e+03
85	2.868e+00	1.858e+03
86	2.439e+00	1.858e+03
87	2.868e+00	1.858e+03
88	2.439e+00	1.858e+03
89	2.868e+00	1.858e+03
90	2.439e+00	1.858e+03
91	2.868e+00	1.858e+03
92	2.439e+00	1.858e+03
93	2.867e+00	1.858e+03
94	2.439e+00	1.858e+03
95	2.867e+00	1.858e+03
96	2.438e+00	1.857e+03
97	2.867e+00	1.857e+03
98	2.438e+00	1.857e+03

99	2.867e+00	1.857e+03
100	2.438e+00	1.857e+03
101	2.867e+00	1.857e+03
102	2.438e+00	1.857e+03
103	2.866e+00	1.857e+03
104	2.438e+00	1.857e+03
105	2.866e+00	1.857e+03
106	2.438e+00	1.857e+03
107	2.866e+00	1.857e+03
108	2.437e+00	1.857e+03
109	2.866e+00	1.856e+03
110	2.437e+00	1.856e+03
111	2.865e+00	1.856e+03
112	2.437e+00	1.856e+03
113	2.865e+00	1.856e+03
114	2.437e+00	1.856e+03
115	2.865e+00	1.856e+03
116	2.437e+00	1.856e+03
117	2.865e+00	1.856e+03
118	2.436e+00	1.856e+03
119	2.865e+00	1.856e+03
120	2.436e+00	1.856e+03
121	2.864e+00	1.856e+03
122	2.436e+00	1.856e+03
123	2.864e+00	1.855e+03
124	2.436e+00	1.855e+03
125	2.864e+00	1.855e+03
126	2.436e+00	1.855e+03
127	2.864e+00	1.855e+03
128	2.435e+00	1.855e+03
129	2.863e+00	1.855e+03
130	2.435e+00	1.855e+03
131	2.863e+00	1.855e+03
132	2.435e+00	1.855e+03
133	2.863e+00	1.855e+03
134	2.435e+00	1.855e+03
135	2.863e+00	1.855e+03
136	2.435e+00	1.855e+03
137	2.863e+00	1.854e+03
138	2.434e+00	1.854e+03
139	2.862e+00	1.854e+03
140	2.434e+00	1.854e+03
141	2.862e+00	1.854e+03
142	2.434e+00	1.854e+03
143	2.862e+00	1.854e+03
144	2.434e+00	1.854e+03
145	2.862e+00	1.854e+03
146	2.434e+00	1.854e+03
147	2.861e+00	1.854e+03
148	2.434e+00	1.854e+03
149	2.861e+00	1.854e+03
150	2.433e+00	1.854e+03
151	2.861e+00	1.853e+03
152	2.433e+00	1.853e+03

153	2.861e+00	1.853e+03
154	2.433e+00	1.853e+03
155	2.861e+00	1.853e+03
156	2.433e+00	1.853e+03
157	2.860e+00	1.853e+03
158	2.433e+00	1.853e+03
159	2.860e+00	1.853e+03
160	2.432e+00	1.853e+03
161	2.860e+00	1.853e+03
162	2.432e+00	1.853e+03
163	2.860e+00	1.853e+03
164	2.432e+00	1.853e+03
165	2.859e+00	1.852e+03
166	2.432e+00	1.852e+03
167	2.859e+00	1.852e+03
168	2.432e+00	1.852e+03
169	2.859e+00	1.852e+03
170	2.431e+00	1.852e+03
171	2.859e+00	1.852e+03
172	2.431e+00	1.852e+03
173	2.859e+00	1.852e+03
174	2.431e+00	1.852e+03
175	2.858e+00	1.852e+03
176	2.431e+00	1.852e+03
177	2.858e+00	1.852e+03
178	2.431e+00	1.852e+03
179	2.858e+00	1.851e+03
180	2.431e+00	1.851e+03
181	2.858e+00	1.851e+03
182	2.430e+00	1.851e+03
183	2.857e+00	1.851e+03
184	2.430e+00	1.851e+03
185	2.857e+00	1.851e+03
186	2.430e+00	1.851e+03
187	2.857e+00	1.851e+03
188	2.430e+00	1.851e+03
189	2.857e+00	1.851e+03
190	2.430e+00	1.851e+03
191	2.857e+00	1.851e+03
192	2.429e+00	1.850e+03
193	2.856e+00	1.850e+03
194	2.429e+00	1.850e+03
195	2.856e+00	1.850e+03
196	2.429e+00	1.850e+03
197	2.856e+00	1.850e+03
198	2.429e+00	1.850e+03
199	2.856e+00	1.850e+03
200	2.429e+00	1.850e+03
201	2.855e+00	1.850e+03
202	2.428e+00	1.850e+03
203	2.855e+00	1.850e+03
204	2.428e+00	1.850e+03
205	2.855e+00	1.850e+03
206	2.428e+00	1.849e+03

207	2.855e+00	1.849e+03
208	2.428e+00	1.849e+03
209	2.855e+00	1.849e+03
210	2.428e+00	1.849e+03
211	2.854e+00	1.849e+03
212	2.427e+00	1.849e+03
213	2.854e+00	1.849e+03
214	2.427e+00	1.849e+03
215	2.854e+00	1.849e+03
216	2.427e+00	1.849e+03
217	2.854e+00	1.849e+03
218	2.427e+00	1.849e+03
219	2.853e+00	1.849e+03
220	2.427e+00	1.848e+03
221	2.853e+00	1.848e+03
222	2.427e+00	1.848e+03
223	2.853e+00	1.848e+03
224	2.426e+00	1.848e+03
225	2.853e+00	1.848e+03
226	2.426e+00	1.848e+03
227	2.853e+00	1.848e+03
228	2.426e+00	1.848e+03
229	2.852e+00	1.848e+03
230	2.426e+00	1.848e+03
231	2.852e+00	1.848e+03
232	2.426e+00	1.848e+03
233	2.852e+00	1.848e+03
234	2.425e+00	1.847e+03
235	2.852e+00	1.847e+03
236	2.425e+00	1.847e+03
237	2.851e+00	1.847e+03
238	2.425e+00	1.847e+03
239	2.851e+00	1.847e+03
240	2.425e+00	1.847e+03
241	2.851e+00	1.847e+03
242	2.425e+00	1.847e+03
243	2.851e+00	1.847e+03
244	2.424e+00	1.847e+03
245	2.851e+00	1.847e+03
246	2.424e+00	1.847e+03
247	2.850e+00	1.847e+03
248	2.424e+00	1.846e+03
249	2.850e+00	1.846e+03
250	2.424e+00	1.846e+03
251	2.850e+00	1.846e+03
252	2.424e+00	1.846e+03
253	2.850e+00	1.846e+03
254	2.424e+00	1.846e+03
255	2.849e+00	1.846e+03
256	2.423e+00	1.846e+03
257	2.849e+00	1.846e+03
258	2.423e+00	1.846e+03
259	2.849e+00	1.846e+03
260	2.423e+00	1.846e+03

261	2.849e+00	1.846e+03
262	2.423e+00	1.845e+03
263	2.849e+00	1.845e+03
264	2.423e+00	1.845e+03
265	2.848e+00	1.845e+03
266	2.422e+00	1.845e+03
267	2.848e+00	1.845e+03
268	2.422e+00	1.845e+03
269	2.848e+00	1.845e+03
270	2.422e+00	1.845e+03
271	2.848e+00	1.845e+03
272	2.422e+00	1.845e+03
273	2.847e+00	1.845e+03
274	2.422e+00	1.845e+03
275	2.847e+00	1.845e+03
276	2.421e+00	1.844e+03
277	2.847e+00	1.844e+03
278	2.421e+00	1.844e+03
279	2.847e+00	1.844e+03
280	2.421e+00	1.844e+03
281	2.847e+00	1.844e+03
282	2.421e+00	1.844e+03
283	2.846e+00	1.844e+03
284	2.421e+00	1.844e+03
285	2.846e+00	1.844e+03
286	2.420e+00	1.844e+03
287	2.846e+00	1.844e+03
288	2.420e+00	1.844e+03
289	2.846e+00	1.844e+03
290	2.420e+00	1.843e+03
291	2.845e+00	1.843e+03
292	2.420e+00	1.843e+03
293	2.845e+00	1.843e+03
294	2.420e+00	1.843e+03
295	2.845e+00	1.843e+03
296	2.420e+00	1.843e+03
297	2.845e+00	1.843e+03
298	2.419e+00	1.843e+03
299	2.845e+00	1.843e+03
300	2.419e+00	1.843e+03
301	2.844e+00	1.843e+03
302	2.419e+00	1.843e+03
303	2.844e+00	1.843e+03
304	2.419e+00	1.842e+03
305	2.844e+00	1.842e+03
306	2.419e+00	1.842e+03
307	2.844e+00	1.842e+03
308	2.418e+00	1.842e+03
309	2.843e+00	1.842e+03
310	2.418e+00	1.842e+03
311	2.843e+00	1.842e+03
312	2.418e+00	1.842e+03
313	2.843e+00	1.842e+03
314	2.418e+00	1.842e+03

```
315         2.843e+00       1.842e+03
316         2.418e+00       1.842e+03
317         2.843e+00       1.842e+03
318         2.417e+00       1.841e+03
319         2.842e+00       1.841e+03
320         2.417e+00       1.841e+03
321         2.842e+00       1.841e+03
322         2.417e+00       1.841e+03
323         2.842e+00       1.841e+03
324         2.417e+00       1.841e+03
325         2.842e+00       1.841e+03
326         2.417e+00       1.841e+03
327         2.841e+00       1.841e+03
328         2.417e+00       1.841e+03
329         2.841e+00       1.841e+03
330         2.416e+00       1.841e+03
331         2.841e+00       1.841e+03
332         2.416e+00       1.840e+03
333         2.841e+00       1.840e+03
334         2.416e+00       1.840e+03
335         2.841e+00       1.840e+03
336         2.416e+00       1.840e+03
337         2.840e+00       1.840e+03
338         2.416e+00       1.840e+03
339         2.840e+00       1.840e+03
340         2.415e+00       1.840e+03
341         2.840e+00       1.840e+03
342         2.415e+00       1.840e+03
343         2.840e+00       1.840e+03
344         2.415e+00       1.840e+03
345         2.839e+00       1.840e+03
346         2.415e+00       1.839e+03
347         2.839e+00       1.839e+03
348         2.415e+00       1.839e+03
349         2.839e+00       1.839e+03
350         2.414e+00       1.839e+03
351         2.839e+00       1.839e+03
352         2.414e+00       1.839e+03
353         2.839e+00       1.839e+03
Iter        ||résidu relatif||  ||erreur relatif||
1           2.865e+00       3.623e-13
```

```
([0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0  …  566.5829999999999, 889.9999999999999, 853.295999
```

Validation de la méthode BFGS

```
bfgs_qp(g10, H10)
bfgs_qp(g50, H50)
bfgs_qp(g100, H100)
```

```
Iter        ||résidu relatif||  ||erreur relatif||
1           3.425e+06       3.417e+02
2           2.149e+06       3.416e+02
3           6.425e+05       3.415e+02
```

```
4           5.083e+05       3.413e+02
5           1.702e+05       3.411e+02
6           8.307e+04       3.410e+02
7           3.431e+04       3.401e+02
8           2.013e+04       3.395e+02
9           2.153e+03       5.013e-11
Iter        ||résidu relatif||  ||erreur relatif||
1           2.361e+00       1.865e+03
2           3.087e+00       1.860e+03
3           1.819e+01       1.738e+03
4           7.973e+01       1.284e+03
5           9.936e+01       6.575e+02
6           1.237e+02       1.214e+02
7           6.994e+01       1.028e+01
8           2.756e+01       6.092e-01
9           4.930e+00       1.665e-01
10          1.598e+00       3.344e-02
11          7.537e-01       6.675e-03
12          1.292e-01       1.784e-03
13          3.555e-02       9.672e-04
14          1.119e-02       2.402e-04
15          5.338e-03       1.335e-05
16          3.567e-04       3.592e-06
17          3.597e-05       1.441e-07
Iter        ||résidu relatif||  ||erreur relatif||
1           2.865e+00       3.623e-13
```

```
([0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0  …  566.5829999999999, 889.9999999999999, 853.2959999
```