TP1 - Analyse Numérique

Constanza Corentin

```
f(3, 1, 1)
```

```
ans = 3×1
     1
     1
     1
```

```
f(3, 2, 1)
```

```
ans = 6×1
     1
     1
     1
     1
     1
     1
```

Question 1 :

```
full(Laplace1d(5))
```

```
ans = 5×5
    72   -36     0     0     0
   -36    72   -36     0     0
     0   -36    72   -36     0
     0     0   -36    72   -36
     0     0     0   -36    72
```

```
full(Laplace2d(3,4))
```

```
ans = 12×12
   82.0000  -16.0000         0  -25.0000         0         0         0         0         0         0
  -16.0000   82.0000  -16.0000         0  -25.0000         0         0         0         0         0
         0  -16.0000   82.0000         0         0  -25.0000         0         0         0         0
  -25.0000         0         0   82.0000  -16.0000         0  -25.0000         0         0         0
         0  -25.0000         0  -16.0000   82.0000  -16.0000         0  -25.0000         0         0
         0         0  -25.0000         0  -16.0000   82.0000         0         0  -25.0000         0
         0         0         0  -25.0000         0         0   82.0000  -16.0000         0  -25.0000
         0         0         0         0  -25.0000         0  -16.0000   82.0000  -16.0000         0
         0         0         0         0         0  -25.0000         0  -16.0000   82.0000         0
         0         0         0         0         0         0  -25.0000         0         0   82.0000
```

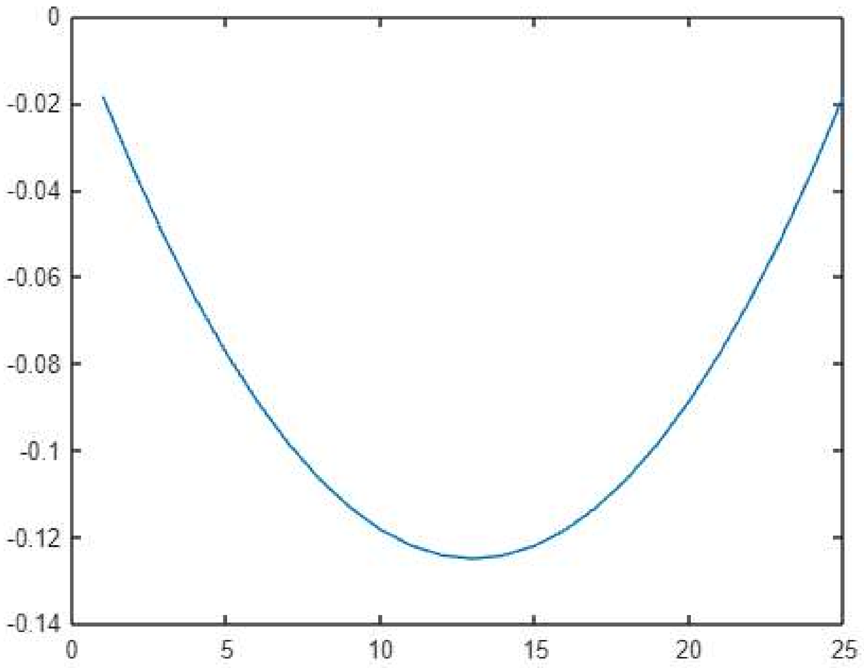
```
full(Laplace3d(2,3,4))
```

```
ans = 24×24
  100.0000   -9.0000  -16.0000         0         0         0  -25.0000         0         0         0
   -9.0000  100.0000         0  -16.0000         0         0         0  -25.0000         0         0
```

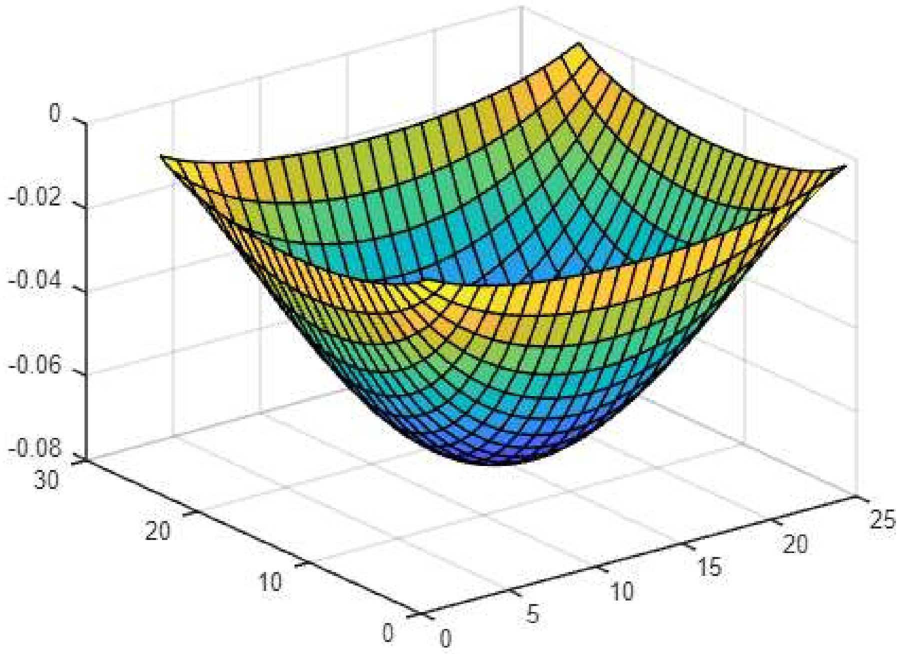
```
  -16.0000         0  100.0000   -9.0000  -16.0000         0         0         0  -25.0000         0
         0  -16.0000   -9.0000  100.0000         0  -16.0000         0         0         0  -25.0000
         0         0  -16.0000         0  100.0000   -9.0000         0         0         0         0
         0         0         0  -16.0000   -9.0000  100.0000         0         0         0         0
  -25.0000         0         0         0         0         0  100.0000   -9.0000  -16.0000         0
         0  -25.0000         0         0         0         0   -9.0000  100.0000         0  -16.0000
         0         0  -25.0000         0         0         0  -16.0000         0  100.0000   -9.0000
```

Question 2 :

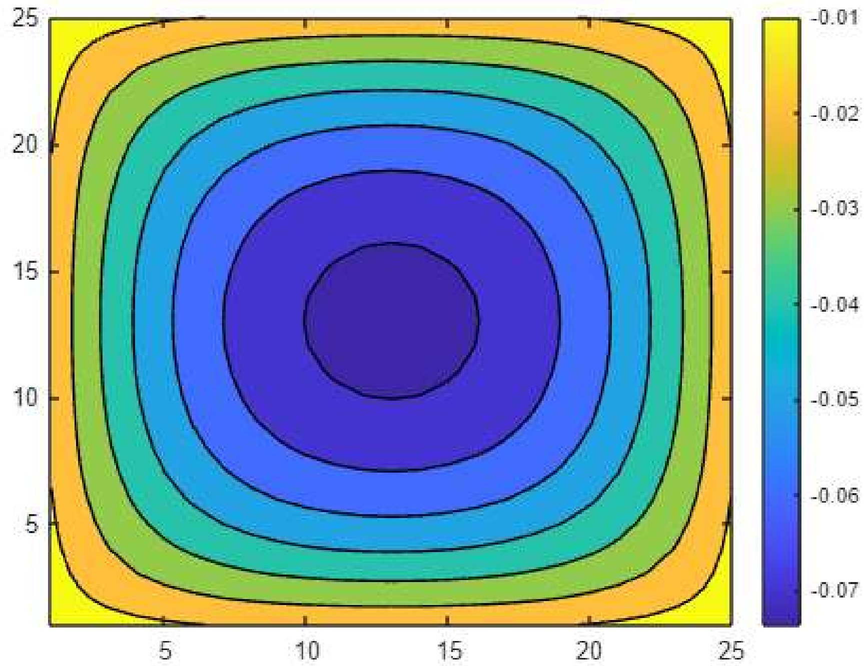
```
n = 25;
m = 25;
p = 25;
%en 1D
A = Laplace1d(n);
u = -A\f(n, 1, 1);
plot(u)
```



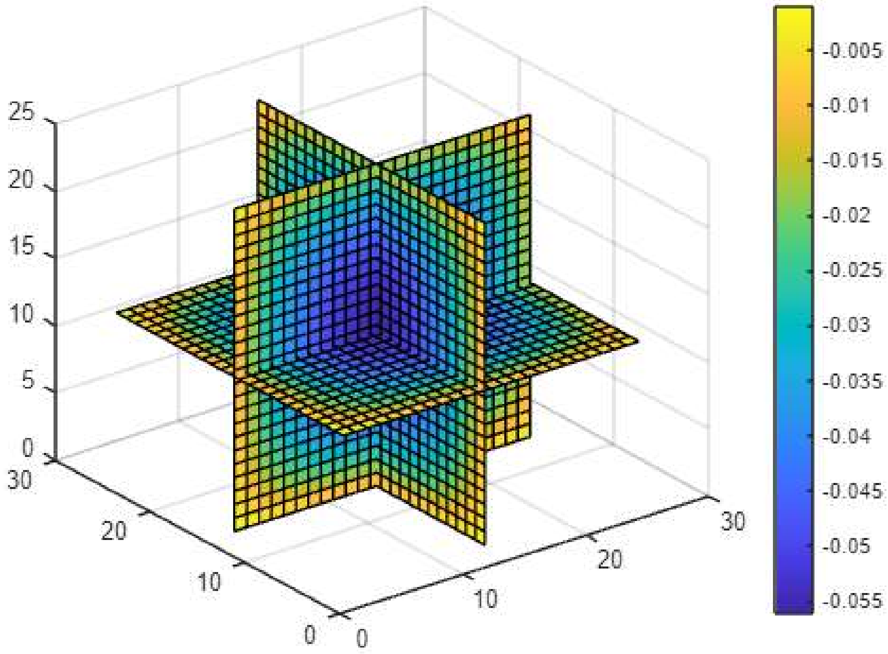

```
%en 2D
A = Laplace2d(n, m);
u = -A\f(n, m, 1);
matU = reshape(u, n, m);
surf(matU)
```




```
contourf(matU)
colorbar()
```

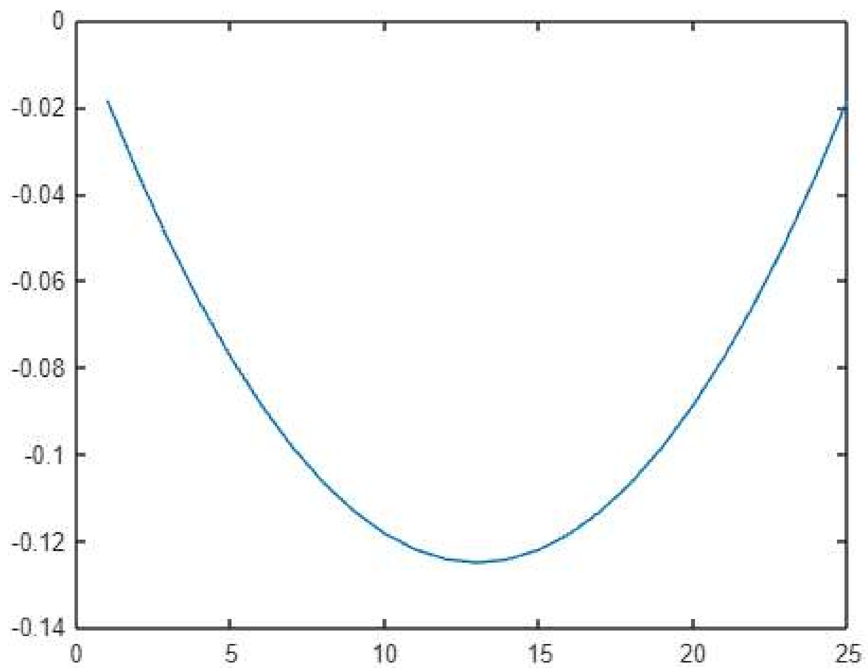



```
%en 3D
A = Laplace3d(n, m, p);
u = -A\f(n, m, p);
xslice = [12.5];
yslice = [12.5];
zslice = [12.5];
matU = reshape(u, n, m, p);
slice(matU, xslice, yslice, zslice)
colorbar()
```



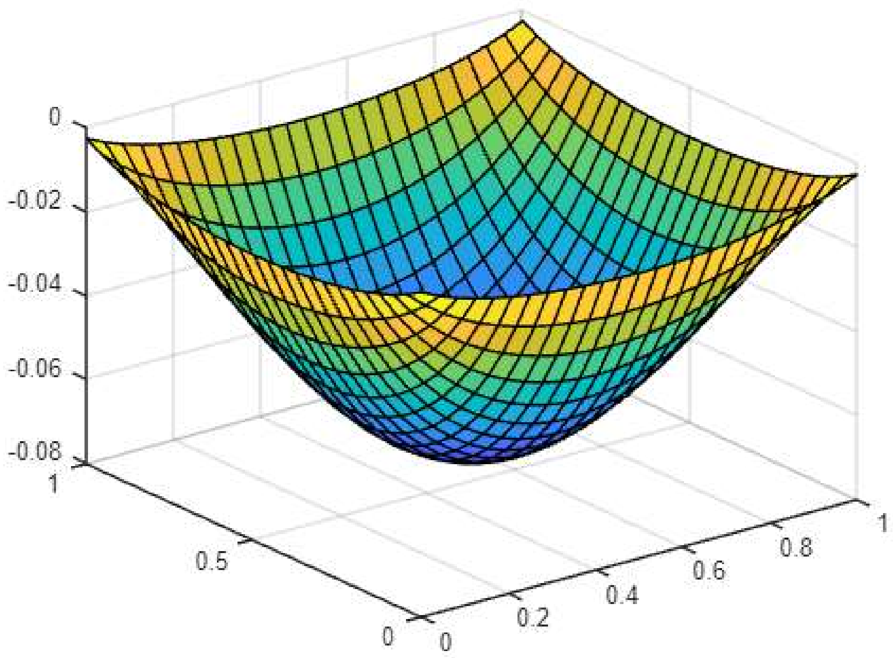
Question 3 :

```
rng('default')

% en 1d
A = -Laplace1d(n);
b = f(n, 1, 1);
x0 = rand(n, 1);


[u_1D, res_1D, iter_1D] = GCP(A, x0, b, @Precond_ID);
plot(u_1D)
```

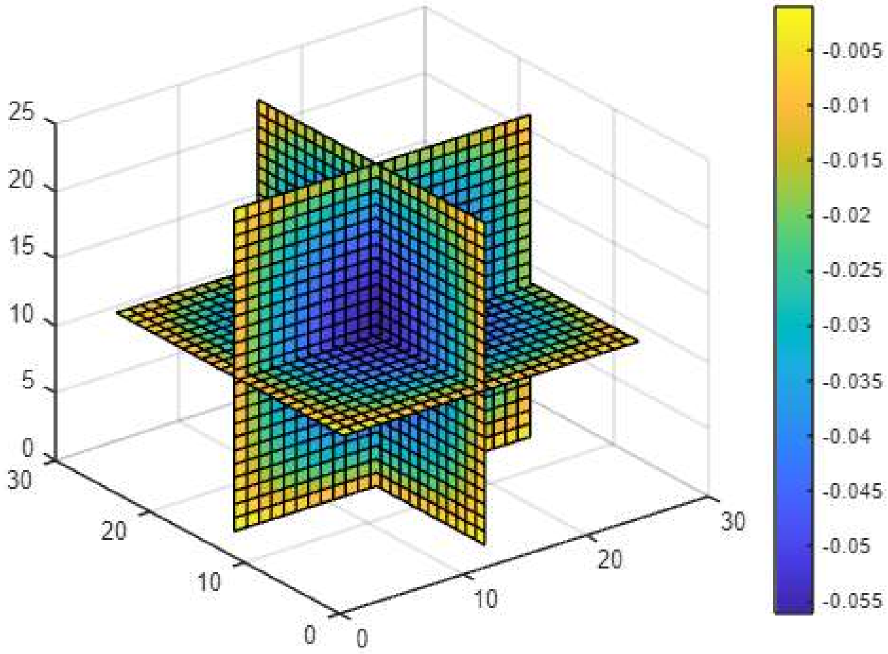



```
% en 2d
```

```
A = -Laplace2d(n, m);
b = f(n, m, 1);
x0 = rand(n * m, 1);

[u_2D, res_2D, iter_2D] = GCP(A, x0, b, @Precond_ID);
surf(linspace(0, 1, n), linspace(0, 1, m), ...
     reshape(u_2D, n, m))
```




```
% en 3d
A = -Laplace3d(n, m, p);
b = f(n, m, p);
x0 = rand(n * m * p, 1);

[u_3D, res_3D, iter_3D] = GCP(A, x0, b, @Precond_ID);
u_3D = reshape(u, n, m, p);
slice(u_3D, [12.5], [12.5], [12.5])
colorbar()
```

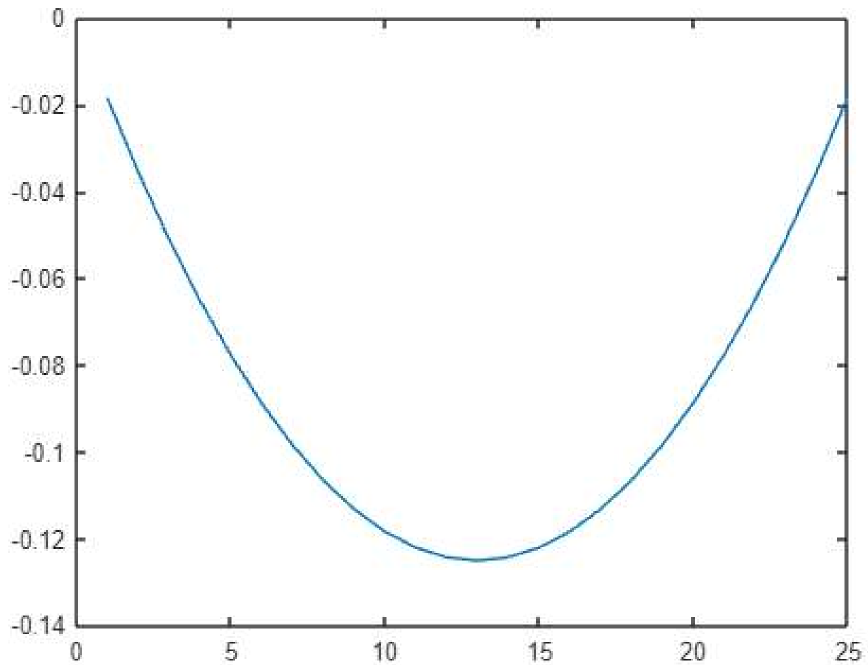


Question 4 :

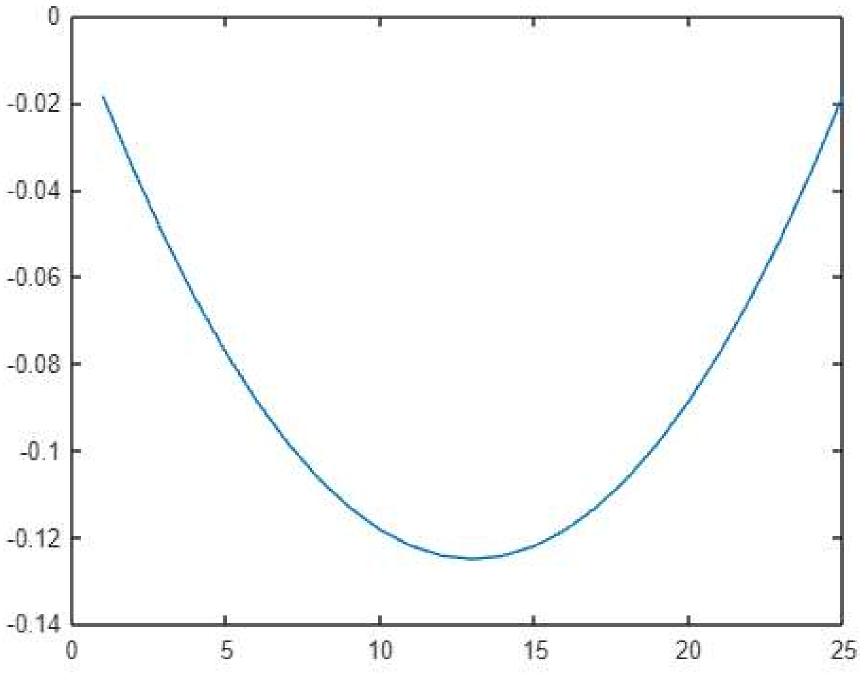
```
max_iter = 50
```

```
max_iter = 50
```

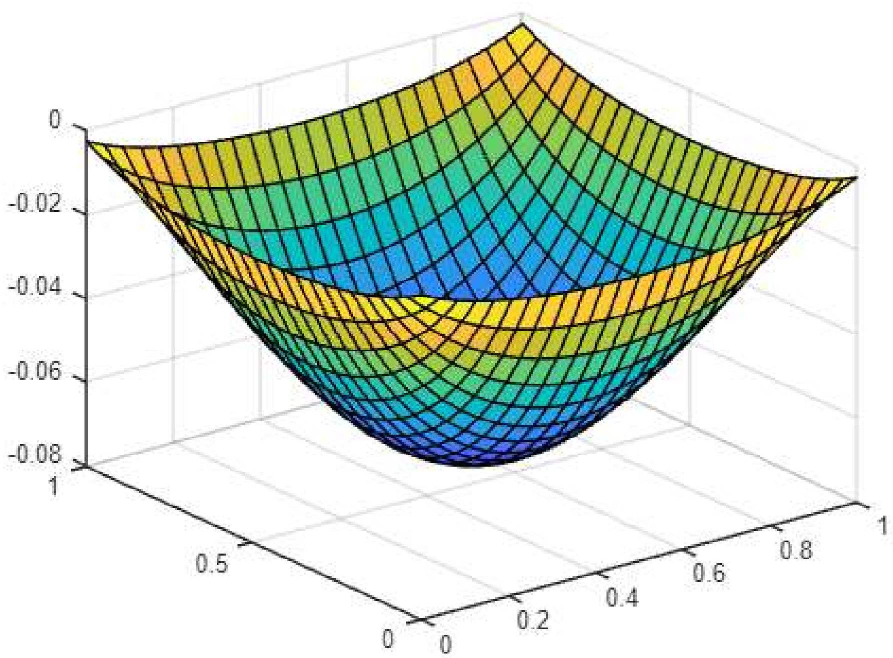
```
% en 1d
A = -Laplace1d(25);
b = f(25, 1, 1);
x0 = rand(25, 1);

[u_ILU_1d, r_total_ILU_1d, iter_ILU_1d] = GCP(A, x0, b, @Precond_ILU);
[u_SSOR_1d, r_total_SSOR_1d, iter_SSOR_1d] = GCP(A, x0, b, @Precond_SSOR);
plot(u_SSOR_1d)
```



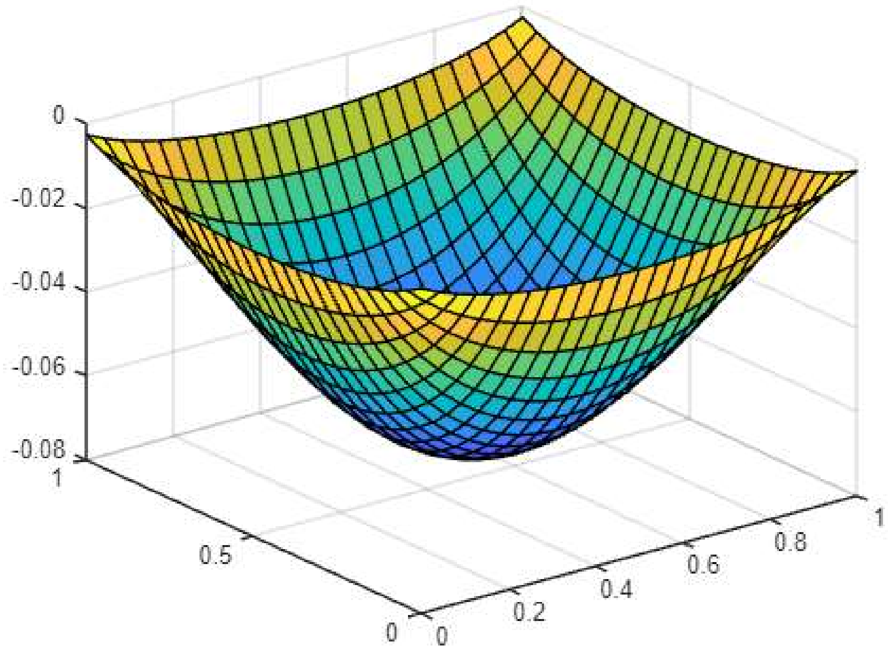

```
plot(u_ILU_1d)
```



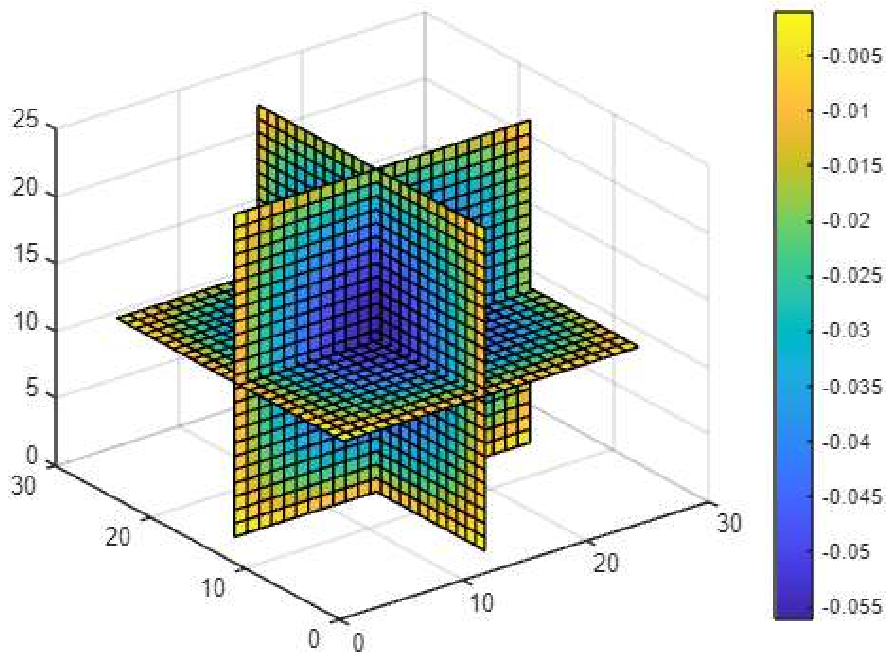

```
% en 2d
A = -Laplace2d(25, 25);
b = f(25, 25, 1);
x0 = rand(25 * 25, 1);

[u_ILU_2d, r_total_ILU_2d, iter_ILU_2d] = GCP(A, x0, b, @Precond_ILU);
[u_SSOR_2d, r_total_SSOR_2d, iter_SSOR_2d] = GCP(A, x0, b, @Precond_SSOR);
surf(linspace(0, 1, 25), linspace(0, 1, 25), ...
     reshape(real(u_SSOR_2d), 25, 25))
```



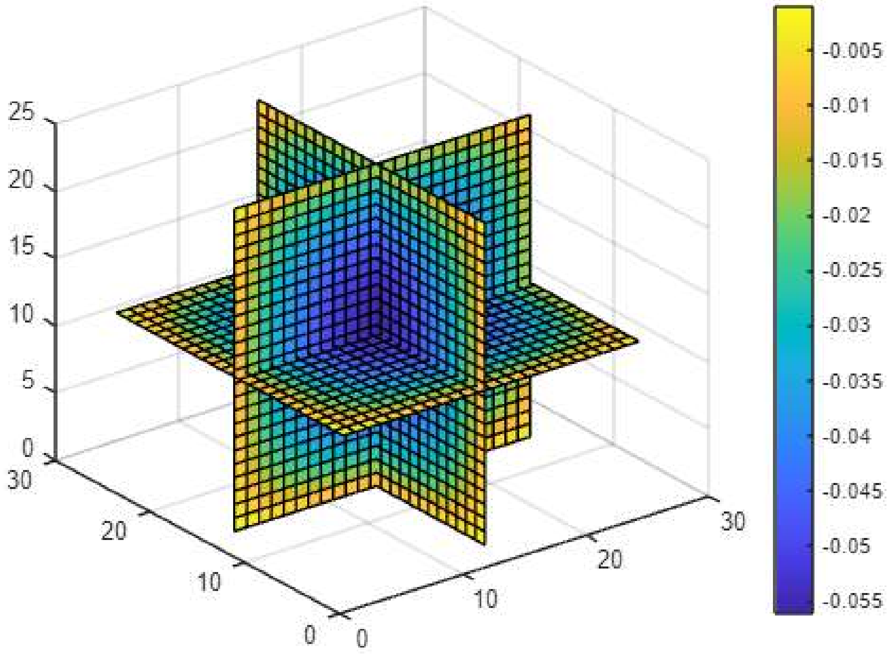

```
surf(linspace(0, 1, 25), linspace(0, 1, 25), ...
    reshape(real(u_ILU_2d), 25, 25))
```




```
% en 3d
A = -Laplace3d(25, 25, 25);
b = f(25, 25, 25);
x0 = rand(25 * 25 * 25, 1);

[u_ILU_3d, r_total_ILU_3d, iter_ILU_3d] = GCP(A, x0, b, @Precond_ILU);
[u_SSOR_3d, r_total_SSOR_3d, iter_SSOR_3d] = GCP(A, x0, b,@Precond_SSOR);
u_SSOR_mat = reshape(real(u_SSOR_3d), 25, 25, 25);
slice(u_SSOR_mat, [12.5], [12.5], [12.5])
colorbar()
```

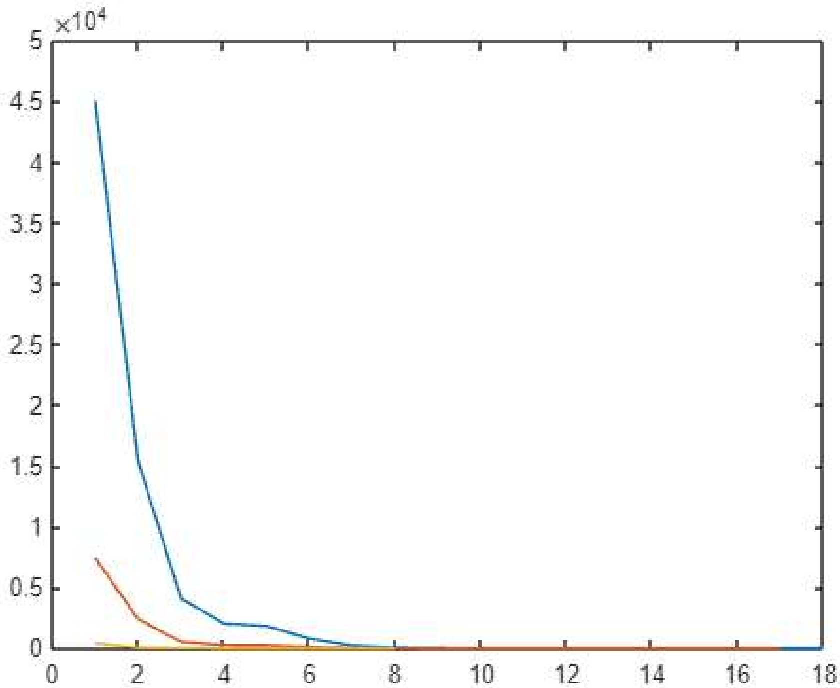



```
u_ILU_mat = reshape(real(u_ILU_3d), 25, 25, 25);
slice(u_ILU_mat, [12.5], [12.5], [12.5])
colorbar()
```

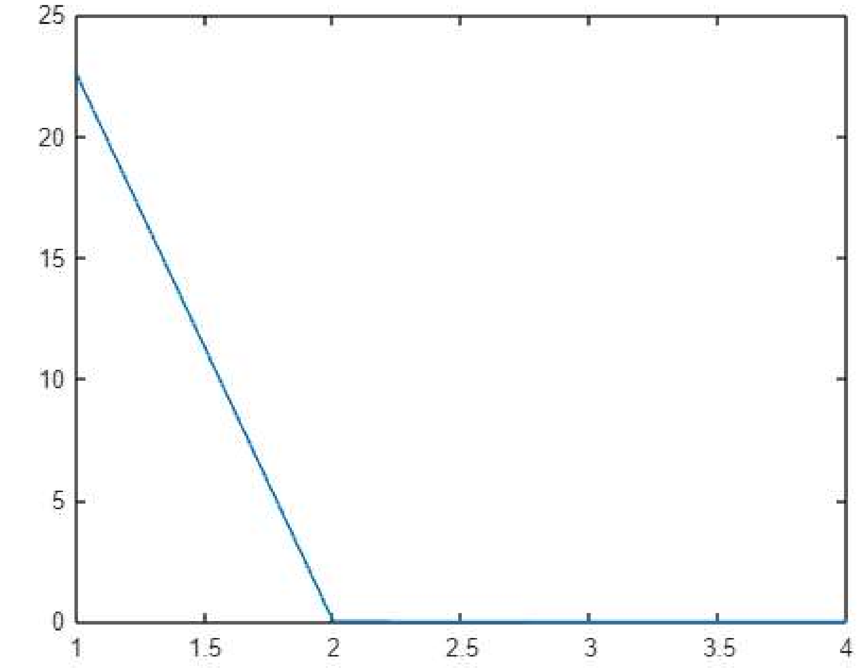


Question 5 :

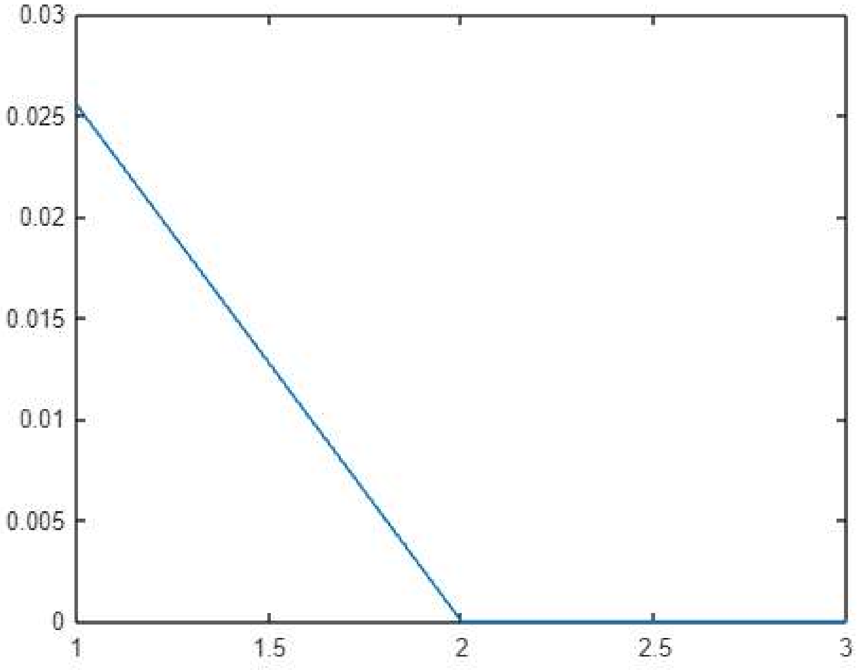
```
plot(vecnorm(r_total_SSOR_3d))
hold on
plot(vecnorm(r_total_SSOR_2d))
plot(vecnorm(r_total_SSOR_1d))
hold off
```



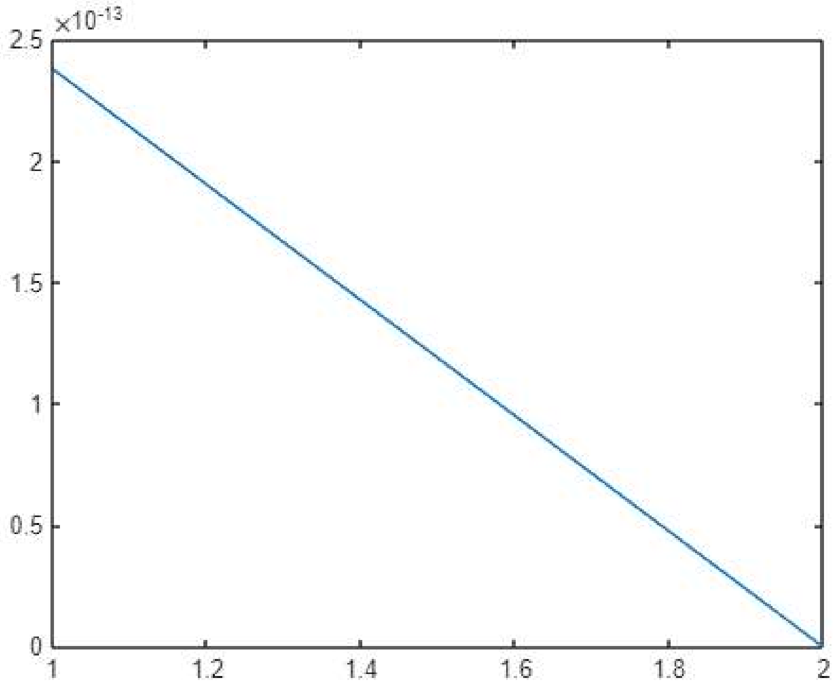

```
plot(vecnorm(r_total_ILU_3d))
```



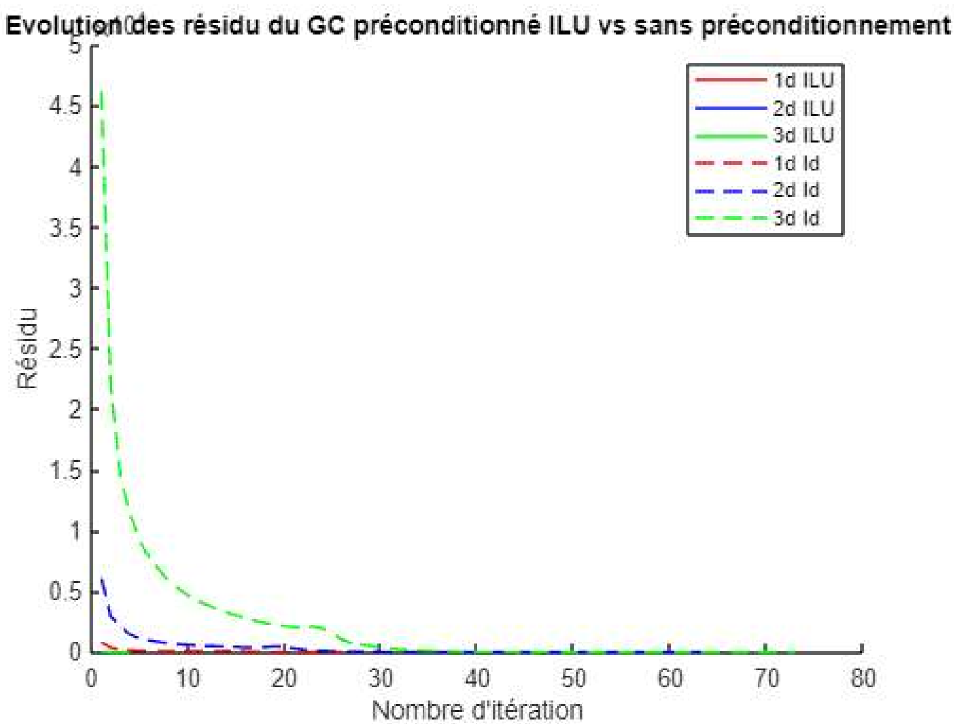

```
plot(vecnorm(r_total_ILU_2d))
```



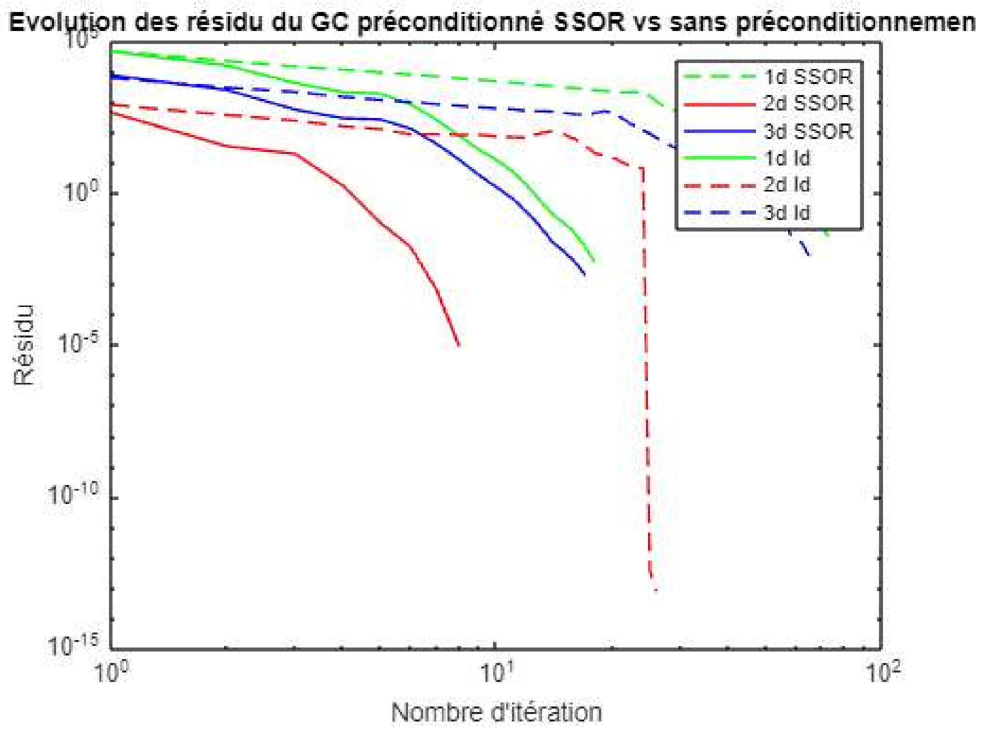

```
plot(vecnorm(r_total_ILU_1d))
```




```
figure
xlabel("Nombre d'itération")
ylabel("Résidu")
title("Evolution des résidu du GC préconditionné ILU vs sans préconditionnement")
hold on
loglog(1:iter_ILU_1d,vecnorm(r_total_ILU_1d),color="red")
loglog(1:iter_ILU_2d,vecnorm(r_total_ILU_2d),color="blue")
loglog(1:iter_ILU_3d,vecnorm(r_total_ILU_3d),color="green")
loglog(1:iter_1D,vecnorm(res_1D),color="red", LineStyle='--')
loglog(1:iter_2D,vecnorm(res_2D),color="blue", LineStyle='--')
loglog(1:iter_3D,vecnorm(res_3D),color="green", LineStyle='--')
hold off
legend(["1d ILU";"2d ILU"; "3d ILU";"1d Id";"2d Id"; "3d Id"])
```



```
xlabel("Nombre d'itération")
ylabel("Résidu")
title("Evolution des résidu du GC préconditionné SSOR vs sans préconditionnement")
hold on
loglog(1:iter_SSOR_1d,vecnorm(r_total_SSOR_1d),color="red")
loglog(1:iter_SSOR_2d,vecnorm(r_total_SSOR_2d),color="blue")
loglog(1:iter_SSOR_3d,vecnorm(r_total_SSOR_3d),color="green")
loglog(1:iter_1D,vecnorm(res_1D),color="red", LineStyle='--')
loglog(1:iter_2D,vecnorm(res_2D),color="blue", LineStyle='--')
loglog(1:iter_3D,vecnorm(res_3D),color="green", LineStyle='--')
hold off
legend(["1d SSOR";"2d SSOR"; "3d SSOR";"1d Id";"2d Id"; "3d Id"])
```



Question 6 :

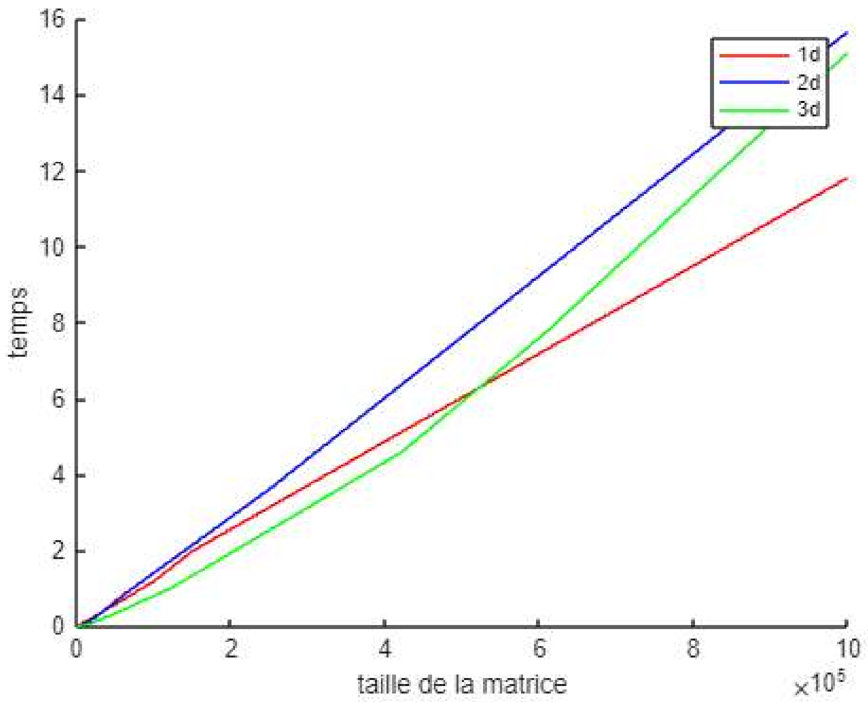
L'objectif de cette question est de tracer les temps de résolution obtenus pour les différentes matriceset méthodes en fonction des tailles de matrice.

- On commence par tracer les courbes de temps de résolution pour le gradient conjugué sans préconditionnement :

```
%%%%%%%%%%%%%%%%%%%%%%% 1D %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N_1D = [1,5,10,50,100,150,1000, 1500, 10000, 15000, 100000, 150000, 1000000];
tps_1D = zeros(length(N_1D),1);
for i = 1:length(N_1D)
    A = -Laplace1d( N_1D(i) );
    b = f(N_1D(i), 1, 1);
    x0 = rand(N_1D(i),1);

    tic
    [u_1D, res_1D, iter_1D] = GCP(A, x0, b, @Precond_ID);
    tps_1D(i) = toc;
end
%%%%%%%%%%%%%%%%%%%%%%% 2D %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N_2D = [1,5,10,25,50,100,150,250,500,1000];
tps_2D = zeros(length(N_2D),1);
for i = 1:length(N_2D)
    A = -Laplace2d(N_2D(i),N_2D(i));
```

```
    b = f(N_2D(i), N_2D(i), 1);
    x0 = rand(N_2D(i)*N_2D(i),1);
    tic
    [u_2D, res_2D, iter_2D] = GCP(A, x0, b, @Precond_ID);
    tps_2D(i) = toc;
end
%%%%%%%%%%%%%%%%%%%%%%% 3D %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N_3D = [1,5,10,25,35,50,75,85,100];
tps_3D = zeros(length(N_3D),1);
for i = 1:length(N_3D)
    A = -Laplace3d(N_3D(i),N_3D(i),N_3D(i));
    b = f(N_3D(i), N_3D(i), N_3D(i));
    x0 = rand(N_3D(i)*N_3D(i)*N_3D(i),1);
    tic
    [u_3D, res_3D, iter_3D] = GCP(A, x0, b, @Precond_ID);
    tps_3D(i) = toc;
end
```

```
figure
xlabel("taille de la matrice")
ylabel("temps")
title("")
hold on
loglog(N_1D,tps_1D, Color="red")
loglog(N_2D.^2,tps_2D, Color="blue")
loglog(N_3D.^3,tps_3D, Color="green")
legend(["1d";"2d";"3d"])
hold off
```

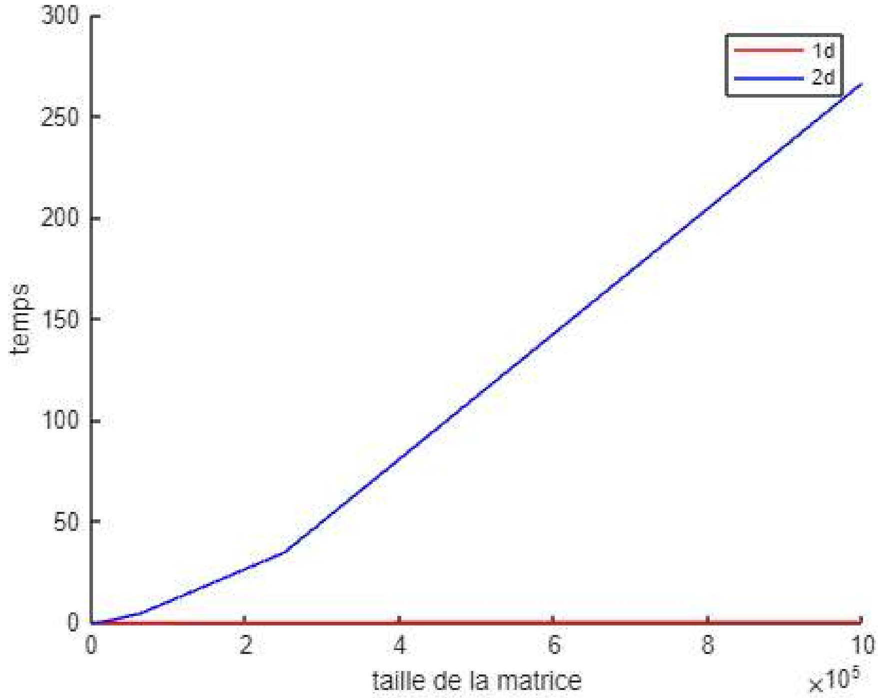


- De même pour le préconditionnement ILU :

```
%%%%%%%%%%%%%%%%%%%%%%% 1D ILU %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N_1D = [1,5,10,50,100,150,1000, 1500, 10000, 15000, 100000, 150000, 1000000];
tps_1D_ILU = zeros(length(N_1D),1);
for i = 1:length(N_1D)
    A = -Laplace1d( N_1D(i) );
    b = f(N_1D(i), 1, 1);
    x0 = rand(N_1D(i),1);
```

```
    tic
    [u_1D, res_1D, iter_1D] = GCP(A, x0, b, @Precond_ILU);
    tps_1D_ILU(i) = toc;
end
%%%%%%%%%%%%%%%%%%%%%%% 2D ILU %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N_2D = [1,5,10,25,50,100,150,250,500,1000];
tps_2D_ILU = zeros(length(N_2D),1);
for i = 1:length(N_2D)
    A = -Laplace2d(N_2D(i),N_2D(i));
    b = f(N_2D(i), N_2D(i), 1);
    x0 = rand(N_2D(i)*N_2D(i),1);
    tic
    [u_2D, res_2D, iter_2D] = GCP(A, x0, b, @Precond_ILU);
    tps_2D_ILU(i) = toc;
end
% %%%%%%%%%%%%%%%%%%%%%%% 3D ILU %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% N_3D = [1,5,10,25,35,50,75,85,100];
% tps_3D_ILU = zeros(length(N_3D),1);
% for i = 1:length(N_3D)
%     A = -Laplace3d(N_3D(i),N_3D(i),N_3D(i));
%     b = f(N_3D(i), N_3D(i), N_3D(i));
%     x0 = rand(N_3D(i)*N_3D(i)*N_3D(i),1);
%     tic
%     [u_3D, res_3D, iter_3D] = GCP(A, x0, b, @Precond_ILU);
%     tps_3D_ILU(i) = toc;
% end
```

```
figure
xlabel("taille de la matrice")
ylabel("temps")
title("")
hold on
loglog(N_1D,tps_1D_ILU, Color="red")
loglog(N_2D.^2,tps_2D_ILU, Color="blue")
%loglog(N_3D.^3,tps_3D_ILU, Color="green")
%legend(["1d";"2d";"3d"])
legend(["1d";"2d"])
hold off
```



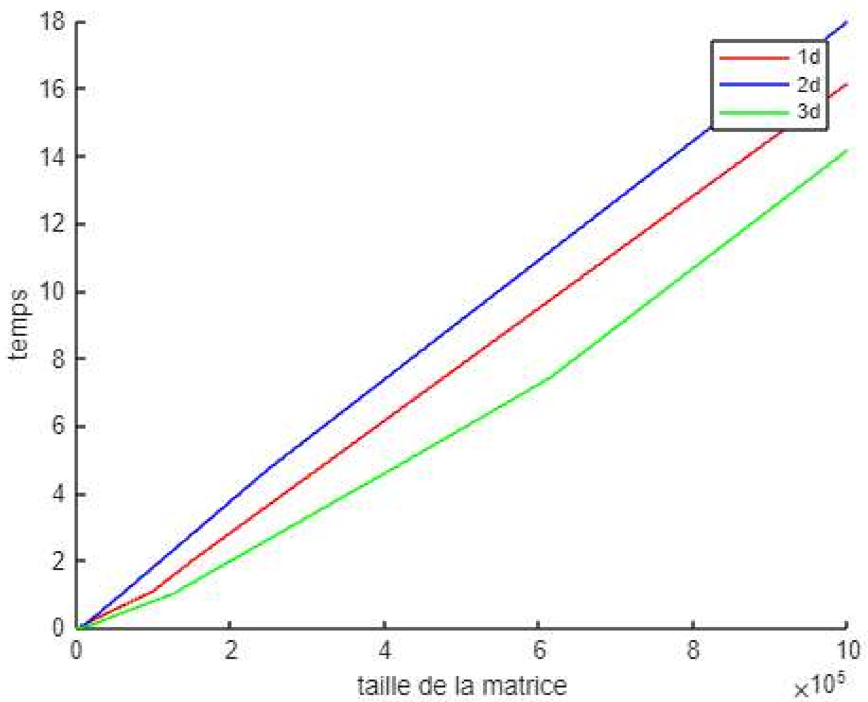
Comme on peut le voir, la méthodes ILU semble très performante en 1d mais pas en 2d et 3d (pour le 3d les calcule n'ont même pas réussi à se terminer.

- De même pour le preconditionnement SSOR :

```
%%%%%%%%%%%%%%%%%%%%%%% 1D SSOR %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N_1D = [1,5,10,50,100,150,1000, 1500, 10000, 15000, 100000, 150000, 1000000];
tps_1D_SSOR = zeros(length(N_1D),1);
for i = 1:length(N_1D)
    A = -Laplace1d( N_1D(i) );
    b = f(N_1D(i), 1, 1);
    x0 = rand(N_1D(i),1);

    tic
    [u_1D, res_1D, iter_1D] = GCP(A, x0, b, @Precond_ID);
    tps_1D_SSOR(i) = toc;
end
%%%%%%%%%%%%%%%%%%%%%%% 2D SSOR %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N_2D = [1,5,10,25,50,100,150,250,500,1000];
tps_2D_SSOR = zeros(length(N_2D),1);
for i = 1:length(N_2D)
    A = -Laplace2d(N_2D(i),N_2D(i));
    b = f(N_2D(i), N_2D(i), 1);
    x0 = rand(N_2D(i)*N_2D(i),1);
    tic
    [u_2D, res_2D, iter_2D] = GCP(A, x0, b, @Precond_ID);
    tps_2D_SSOR(i) = toc;
end
%%%%%%%%%%%%%%%%%%%%%%% 3D SSOR %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N_3D = [1,5,10,25,35,50,75,85,100];
tps_3D_SSOR = zeros(length(N_3D),1);
for i = 1:length(N_3D)
    A = -Laplace3d(N_3D(i),N_3D(i),N_3D(i));
    b = f(N_3D(i), N_3D(i), N_3D(i));
    x0 = rand(N_3D(i)*N_3D(i)*N_3D(i),1);
    tic
    [u_3D, res_3D, iter_3D] = GCP(A, x0, b, @Precond_ID);
```

```
    tps_3D_SSOR(i) = toc;
end
```

```
figure
xlabel("taille de la matrice")
ylabel("temps")
title("")
hold on
loglog(N_1D,tps_1D_SSOR, Color="red")
loglog(N_2D.^2,tps_2D_SSOR, Color="blue")
loglog(N_3D.^3,tps_3D_SSOR, Color="green")
legend(["1d";"2d";"3d"])
hold off
```

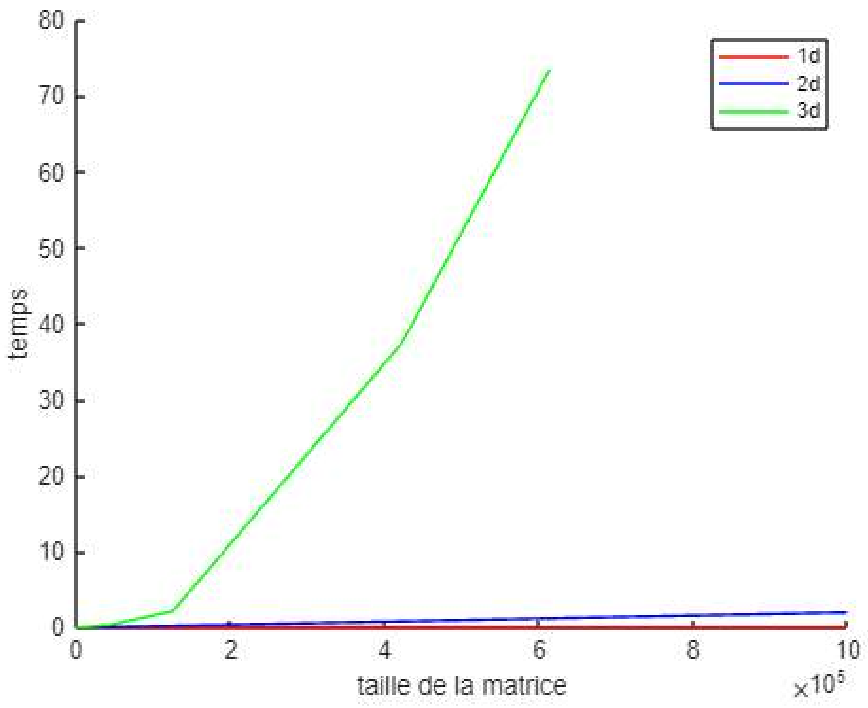


- Enfin regardons la méthode umfpack (" \ ") de Matlab :

```
%%%%%%%%%%%%%%%%%%%%%%% 1D \ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N_1D = [1,5,10,50,100,150,1000, 1500, 10000, 15000, 100000, 150000, 1000000];
tps_1D_um = zeros(length(N_1D),1);
for i = 1:length(N_1D)
    A = -Laplace1d( N_1D(i) );
    b = f(N_1D(i), 1, 1);
    %x0 = rand(N_1D(i),1);

    tic
    u=A\b;
    tps_1D_um(i) = toc;
end
%%%%%%%%%%%%%%%%%%%%%%% 2D \ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N_2D = [1,5,10,25,50,100,150,250,500,1000];
tps_2D_um = zeros(length(N_2D),1);
for i = 1:length(N_2D)
    A = -Laplace2d(N_2D(i),N_2D(i));
    b = f(N_2D(i), N_2D(i), 1);
    %x0 = rand(N_2D(i)*N_2D(i),1);
    tic
    u=A\b;
    tps_2D_um(i) = toc;
```

```
end
%%%%%%%%%%%%%%%%%%%%%%% 3D \ %%%%%%%%%%%%%%%%%%%%%%%%%%%%
N_3D = [1,5,10,25,35,50,75,85,100];
tps_3D_um = zeros(length(N_3D),1);
for i = 1:length(N_3D)
    A = -Laplace3d(N_3D(i),N_3D(i),N_3D(i));
    b = f(N_3D(i), N_3D(i), N_3D(i));
    %x0 = rand(N_3D(i)*N_3D(i)*N_3D(i),1);
    tic
    u=A\b;
    tps_3D_um(i) = toc;
end
```

```
figure
xlabel("taille de la matrice")
ylabel("temps")
title("")
hold on
loglog(N_1D,tps_1D_um, Color="red")
loglog(N_2D.^2,tps_2D_um, Color="blue")
loglog(N_3D(1:length(N_3D)-1).^3,tps_3D_um(1:length(N_3D)-1), Color="green")
legend(["1d";"2d";"3d"])
hold off
```

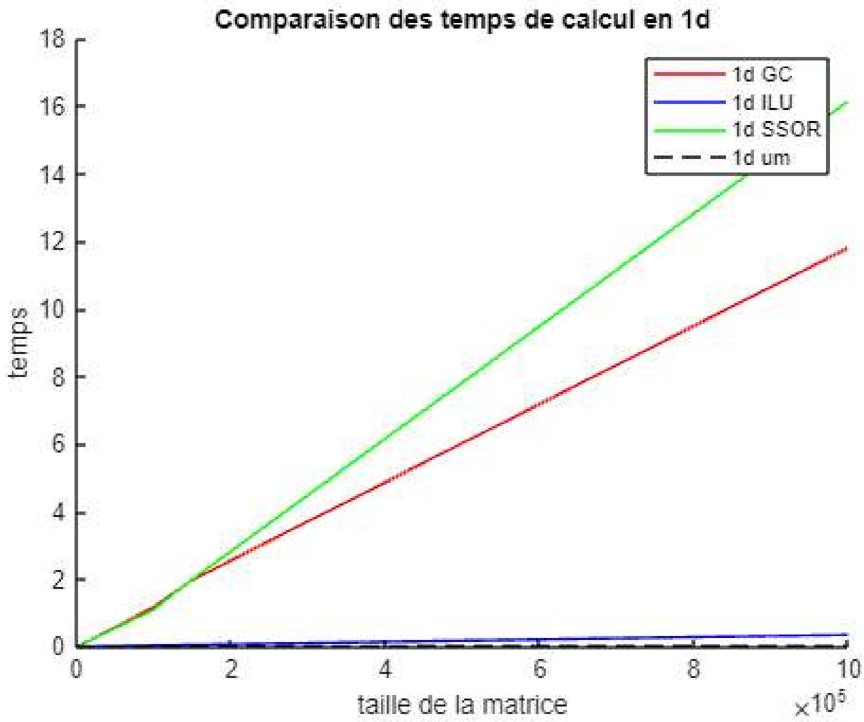


Nous pouvons voir que la méthodes umfpack est très performante pour en 1d et 2d, beaucoup moins en 3d.

Question 8 : Synthèse

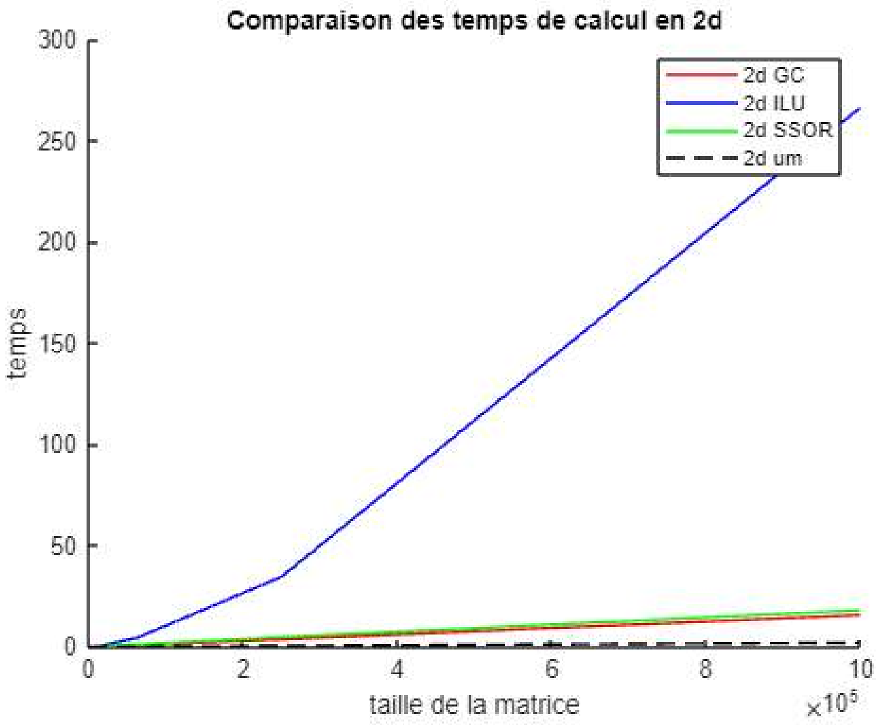
```
figure
xlabel("taille de la matrice")
ylabel("temps")
title("Comparaison des temps de calcul en 1d")
hold on
loglog(N_1D,tps_1D, Color="red")
loglog(N_1D,tps_1D_ILU, Color="blue")
loglog(N_1D,tps_1D_SSOR, Color="green")
loglog(N_1D,tps_1D_um, Color="black", LineStyle='--')
```

```
legend(["1d GC";"1d ILU";"1d SSOR";"1d um"])
hold off
```



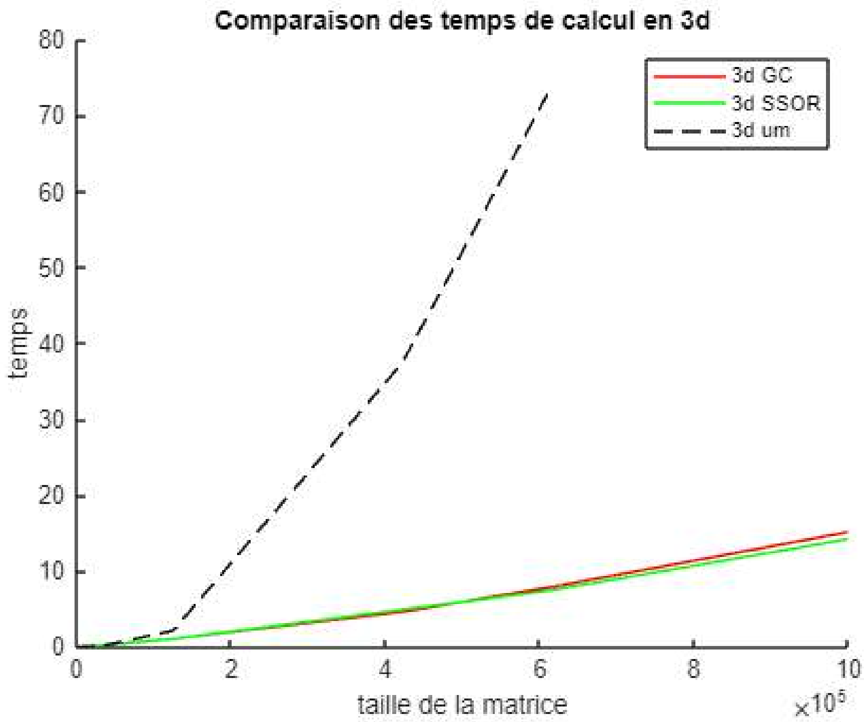
On peut voir qu'en 1d la methode umfpack est la meilleurs, suivit de près par la méthodes ILU.

```
figure
xlabel("taille de la matrice")
ylabel("temps")
title("Comparaison des temps de calcul en 2d")
hold on
loglog(N_2D.^2,tps_2D, Color="red")
loglog(N_2D.^2,tps_2D_ILU, Color="blue")
loglog(N_2D.^2,tps_2D_SSOR, Color="green")
loglog(N_2D.^2,tps_2D_um, Color="black", LineStyle='--')

legend(["2d GC";"2d ILU";"2d SSOR";"2d um"])
hold off
```



En 2d la meilleurs méthodes est toujours umfpack, suivit d'assez loins par le GC sans préconditionnement puis le GC préconditionné SSOR.

```
figure
xlabel("taille de la matrice")
ylabel("temps")
title("Comparaison des temps de calcul en 3d")
hold on

loglog(N_3D.^3,tps_3D, Color="red")
%loglog(N_3D.^3,tps_3D_ILU, Color="blue")
loglog(N_3D.^3,tps_3D_SSOR, Color="green")
loglog(N_3D(1:length(N_3D)-1).^3,tps_3D_um(1:length(N_3D)-1), Color="black", LineStyle="--")

legend(["3d GC";"3d SSOR";"3d um"])
hold off
```



En 3d cependant, la méthode SSOR est la plus performante suivit du GC puis très loin derrière de umpack. (ILU n'est pas afficher ici car il mettait beaucoup trop de temps à calculer déja en 2d et n'arrivait pas à finir le calcul en 3d. Cependant, il converge en très peu d'itération comparer au GC classic et à SSOR cf. Annexes).

Conclusion :

La méthodes umfpack est donc à privilégier pour des calcul en 1d et 2d. Cependant lorque l'on est en 3d et sur de grande matrice (de l'ordre 10^5) il vaut mieux utiliser le GC préconditionné SSOR.

Annexes :

```
fprintf("nombre d'itération du GC en 1D : %i" , iter_1D)
```

```
nombre d'itération du GC en 1D : 26
```

```
fprintf("nombre d'itération du GC en 2D : %i" , iter_2D)
```

```
nombre d'itération du GC en 2D : 65
```

```
fprintf("nombre d'itération du GC en 3D : %i" , iter_3D)
```

```
nombre d'itération du GC en 3D : 73
```

```
fprintf("nombre d'itération du GC ILU en 1D : %i" , iter_ILU_1d)
```

```
nombre d'itération du GC ILU en 1D : 2
```

```
fprintf("nombre d'itération du GC ILU en 2D : %i" , iter_ILU_2d)
```

```
nombre d'itération du GC ILU en 2D : 3
```

```
fprintf("nombre d'itération du GC ILU en 3D : %i" , iter_ILU_3d)
```

```
nombre d'itération du GC ILU en 3D : 4
```

```
fprintf("nombre d'itération du GC SSOR en 1D : %i" , iter_SSOR_1d)
```

```
nombre d'itération du GC SSOR en 1D : 8
```

```
fprintf("nombre d'itération du GC SSOR en 2D : %i" , iter_SSOR_2d)
```

```
nombre d'itération du GC SSOR en 2D : 17
```

```
fprintf("nombre d'itération du GC SSOR en 3D : %i" , iter_SSOR_3d)
```

```
nombre d'itération du GC SSOR en 3D : 18
```

Fonction :

```
function y = f(n, m, p)
    y = ones(n * m * p, 1);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [M] = Laplace1d(n)
    h = 1/(n+1);
    e = ones(n, 1);
    M = -(1/h^2)*spdiags([e -2*e e], -1:1, n, n);
end

function [M] = Laplace2d(n1, n2)
    M = kron(speye(n2), Laplace1d(n1)) + kron(Laplace1d(n2), speye(n1));
end

function [M] = Laplace3d(n1, n2, n3)
    M = kron(kron(speye(n3), speye(n2)), Laplace1d(n1)) ...
    + kron(kron(speye(n3), Laplace1d(n2)), speye(n1)) ...
    + kron(kron(Laplace1d(n3), speye(n2)), speye(n1));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [sol,res ,iter] = GC(A, x, b)
    tol = 10e-6;
    err = tol + 1;

    r = b - A*x;
    d = r;
    res = [];
    iter = 0;
    while err > tol
        xsave = x;

        a = dot(r,d) / dot(A*d,d);
        x = x + a * d;
        r = r - a*A*d;
        res = [res, r];
        beta = -dot(r,A*d) / dot(A*d,d);
        d = r + beta * d;

        err = norm(xsave - x) / norm(xsave);
        iter=iter +1;
    end

    sol = x;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [sol, res, iter] = GCP(A, x, b, Precond)
    tol = 10e-6;
    err = tol + 1;
    nb_iter = 0;
    r = b - A * x;
    z = Precond(A, r, true);
```

```
    d = z;

    res = [];
    iter = 0;
    while (err > tol)&&(nb_iter<300)
        xsave = x;
        rsave = r;
        zsave = z;

        alpha = dot(r, d) / dot(A * d, d);
        x = x + alpha * d;
        r = r - alpha * A * d;

        z = Precond(A, r, false);
        res = [res, r];
        beta = dot(r, z) / dot(rsave, zsave);
        d = z + beta * d;

        err = norm(xsave - x) / norm(xsave);
        nb_iter= nb_iter + 1;
        iter=iter +1;
    end
    sol = x;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Les fonctions suivante se trouvait dans des fichiers/scripts séparer lors
% de la compilation du notebook il se peut qu'il y faille les remettres
% dans des scripts séparer si lors d'une recompilation il y a des problèmes
% recontrés.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function z = Precond_SSOR(A, r, premier_passage)
    global L_SSOR U_SSOR D

    if premier_passage
        % On ne veut calculer la décomposition ILU qu'une fois : lors du premier passage.
        [L_SSOR, U_SSOR, D] = init_ssor(A);
    end
    z = U_SSOR \ (D * (L_SSOR \ r));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [L, U, D] = init_ssor(A)
    D = diag(diag(A));
    L = tril(A, -1);
    U = triu(A, 1);
    w_opt = 1.7; % recommendation du prof.

    L = (D + w_opt * L);
    U = (D + w_opt * U);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function z = Precond_ILU(A, r, premier_passage)
    global L_ILU U_ILU

    if premier_passage
        % On ne veut calculer la décomposition ILU qu'une fois : lors du premier passage.
        [L_ILU, U_ILU] = init_ilu(A);
    end
    z = U_ILU \ (L_ILU \ r);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [L, U] = init_ilu(A)
    [L, U] = ilu(A, struct('type', 'ilutp', 'droptol', 1e-6));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = Precond_ID(~,r,~)
    y=r;
end
```