TME n°3 - Introduction à l'optimisation

Victor Piriou - M1 IMA

Exercice 6 - Fonction de Rosenbrock et méthode de Newton

1.

```
% x1 et x2 sont des variables symboliques
syms x1 x2

f = 100*(x2 - x1^2)^2 + (1 - x1)^2;

H = hessian(f, [x1, x2]);
G = gradient(f, [x1, x2]);
```

2.

x^* est un minimum de f si sa matrice Hessienne est définie-positive. Une matrice est définie-positive si chacune de ses valeurs propres est positive.

```
x1_star = 1;
x2_star = 1;

x1 = x1_star;
x2 = x2_star;

eigenvalues = eig(double(subs(H)));
nb_eigenvalues = size(eigenvalues,1);

positive = true;

for i = 1:nb_eigenvalues
   if(eigenvalues(i) < 0)
        positive = false;
        break;
   end
end

if(positive)
    disp('x* est bien un minimum local de f');
else
    disp('x* n est pas un minimum local de f');
end
```

3.

```
x_0 = [-1; -2];
x_star = [1; 1];

x = []; % liste des abscisses des points du chemin parcouru
y = []; % liste des ordonn es des points du chemin parcouru
errors = []; % liste des normes de l'erreur

x_i = x_0;
```

```
for i=1:5
    x1 = x_i(1); % abscisse
    x2 = x_i(2); % ordonn e

    H_x = double(subs(H)); % Hessienne
    G_x = double(subs(G)); % Gradient

    L = chol(H_x); % d composition de Choleski : H_x = LL'

    s_i = L\(L'\(-G_x)); % on r soud LL' s_i = -G_x

    x_i = x_i + s_i; % calcul de l'it ration

    % on sauvegarde le chemin parcouru
    x = [x, x1];
    y = [y, x2];

    % on sauvegarde l'erreur
    errors = [errors, norm(x_i - x_star)];
end

% courbes de niveau + it r s
ezcontour(f, [-1.5;2;-3;3]);
hold on;
plot(x, y, '-*');

% courbe de l'erreur en fct du nb d'it ration
figure();
plot(errors);
```

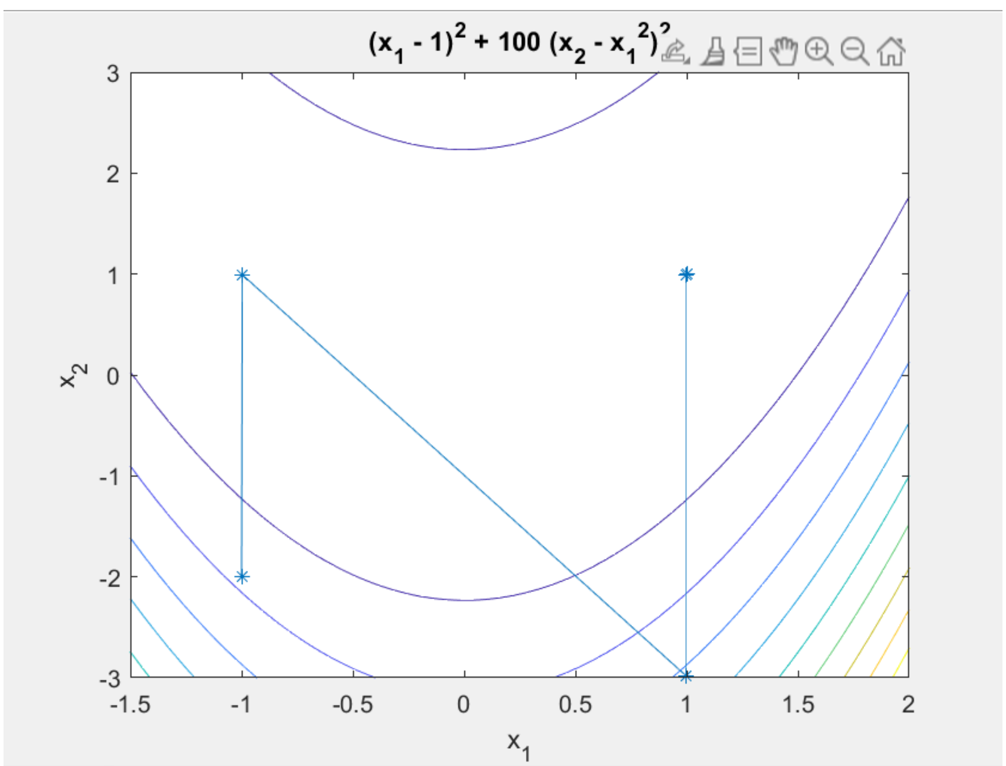


Figure 1: Affichage des itérés sur le graphe des lignes de niveau de la fonction f

4.

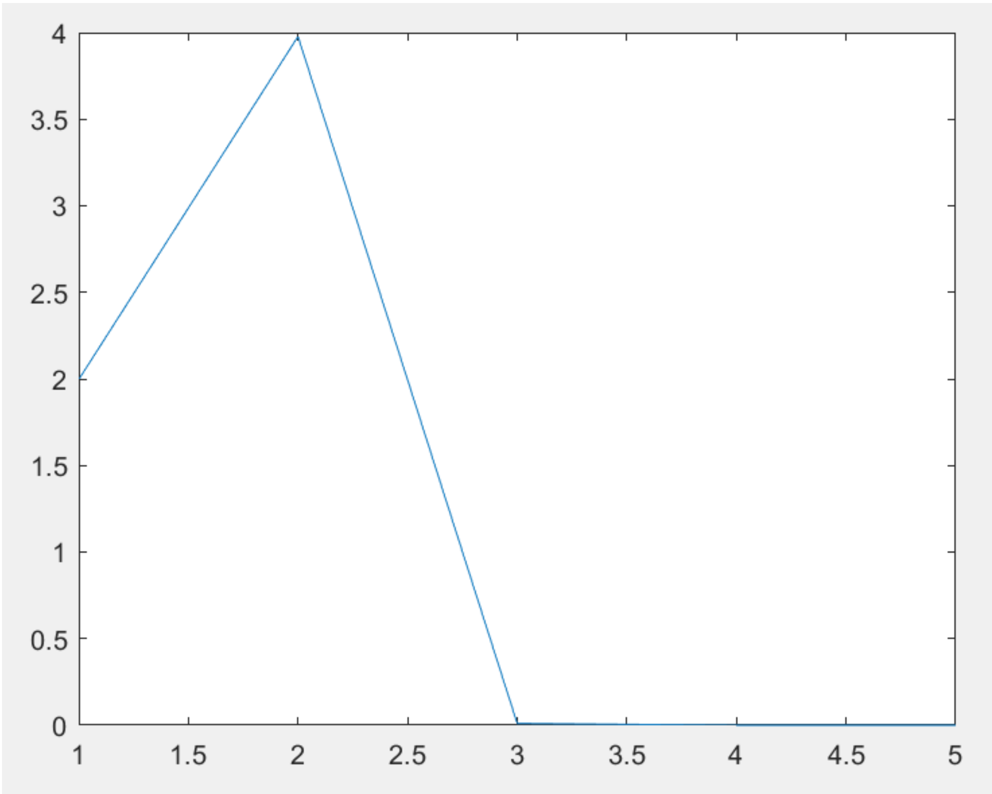
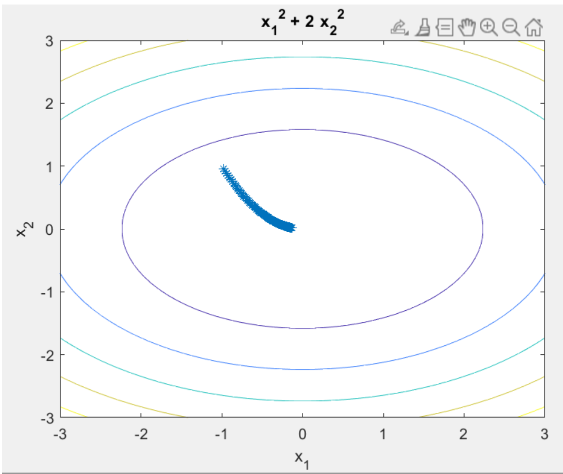


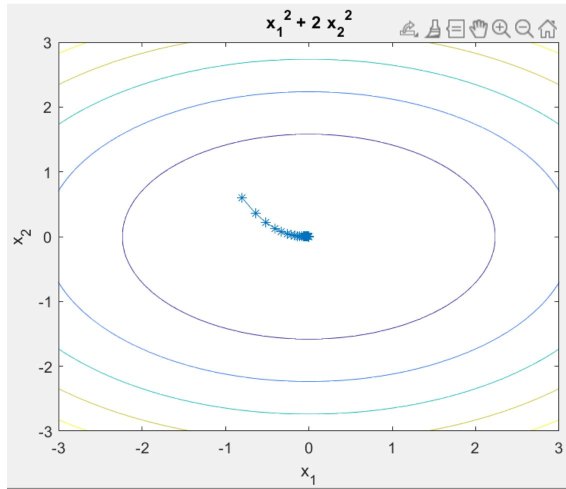
Figure 2: Affichage de la courbe de la norme de l'erreur en fonction des itérations

Exercice 7 - Méthode de gradient à pas optimal et méthode de Wolfe

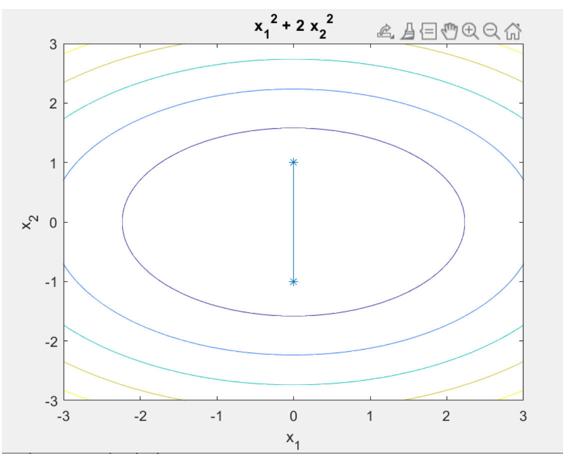
1.



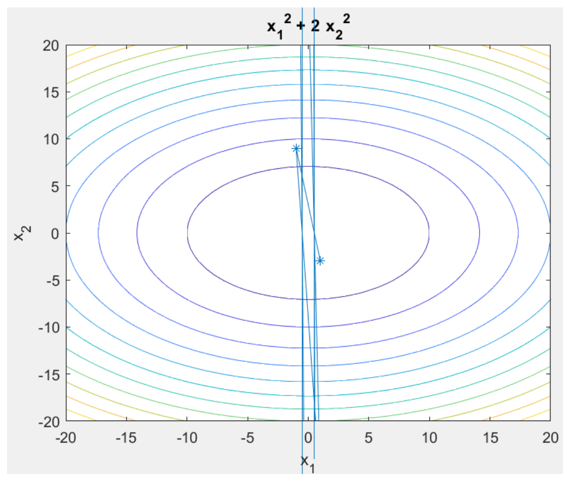
$\alpha = 0.001$, ça converge très lentement



$\alpha = 0.01$, ça converge assez rapidement



$\alpha = 0.5$, alternance entre deux points



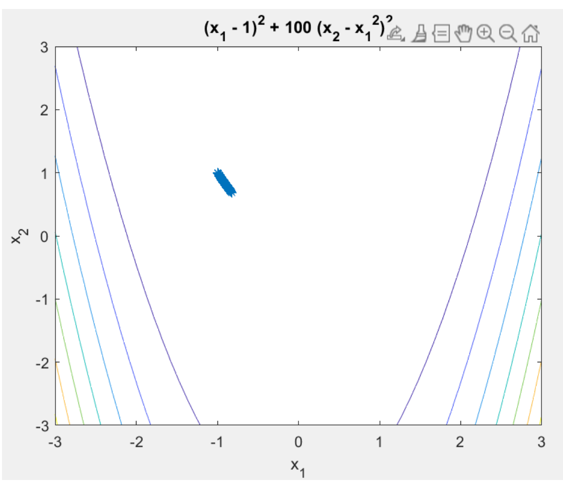
$\alpha = 1$, ça diverge

Tests avec plusieurs pas de descente

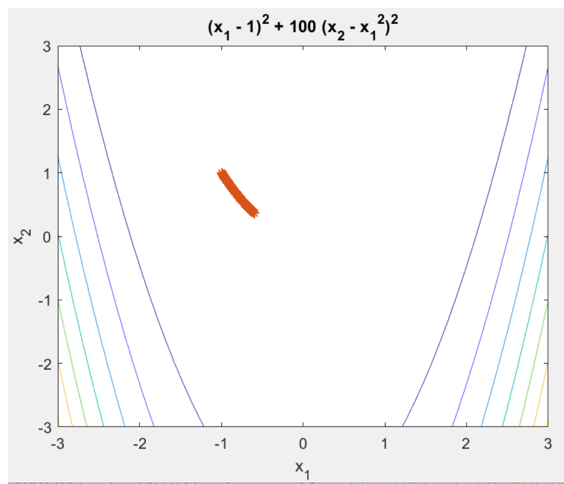
```
syms x1 x2; % variables symboliques

f = x1^2 + 2*x2^2; % fonction   minimiser
nb_iter = 100;
x_0 = [-1;1]; % point initial
alpha = 1; % pas

[x_star, path] = gd(gradient(f),nb_iter,[-1;1],alpha); %descente de gradient
ezcontour(f, [-20;20;-20;20]); % lignes de niveau

hold on;
plot(path(1,:), path(2,:), '-*'); % on trace le chemin parcouru

```

```
function [x_i, path] = gd(df,nb_iter,x_0,alpha)

    x_i = x_0; % point initial
    path = [];

    for i = 1:nb_iter

        x1 = x_i(1); % abscisse
        x2 = x_i(2); % ordonn e

        G = double(subs(df)); % calcul du Gradient
        x_i = x_i - alpha*G;  % calcul de l'it ration

        path = [path, x_i]; % chemin parcouru
    end

end
```

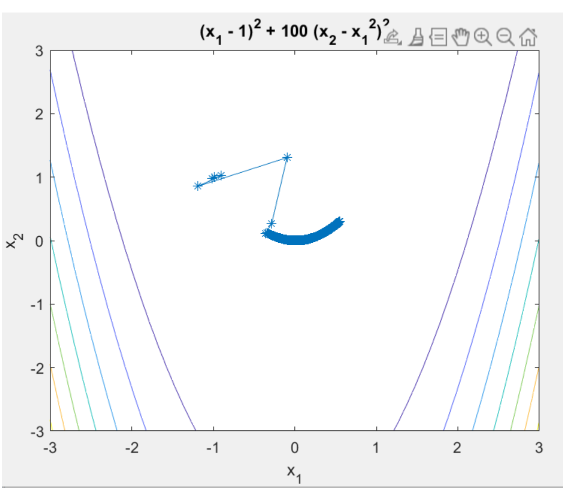
2.



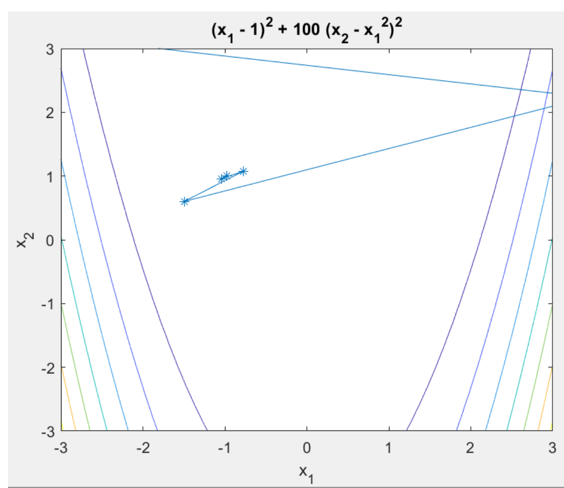
$\alpha = 0.001$, ça converge très lentement



$\alpha = 0.002$, ça converge très lentement



$\alpha = 0.004$, ça converge très lentement

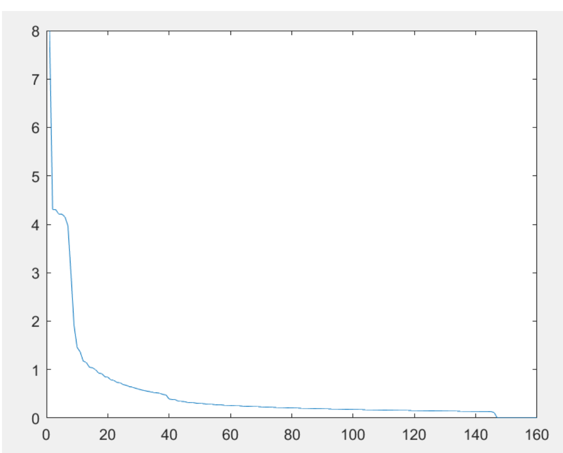


$\alpha = 0.005$, ça diverge

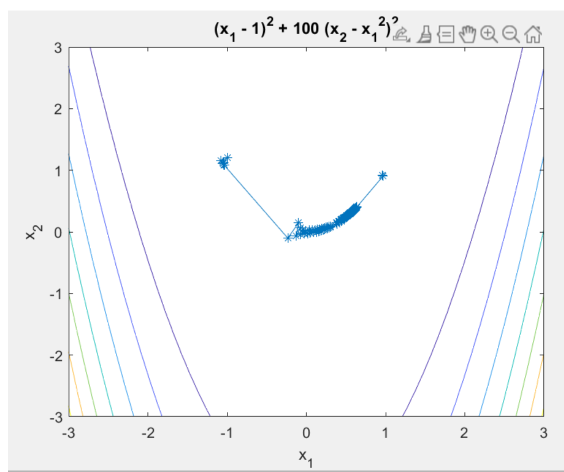
Tests avec plusieurs pas de descente

```
syms x1 x2; % variables symboliques

rosenbrock = 100*(x2 - x1^2)^2 + (1 - x1)^2; % fonction    minimiser
nb_iter = 200;
x_0 = [-1;1.2]; % point initial
alpha = 0.004; % pas

[x_star, path] = gd(gradient(rosenbrock),nb_iter,[-1;1],alpha); %descente de gradient
ezcontour(rosenbrock, [-3;3;-3;3]); % lignes de niveau

hold on;
plot(path(1,:), path(2,:), '-*'); % on trace le chemin parcouru

function [x_i, path] = gd(df,nb_iter,x_0,alpha)

    x_i = x_0; % point initial
    path = [];

    for i = 1:nb_iter

        x1 = x_i(1); % abscisse
        x2 = x_i(2); % ordonn e

        G = double(subs(df)); % calcul du Gradient
        x_i = x_i - alpha*G;  % calcul de l'it ration

        path = [path, x_i]; % chemin parcouru
    end

end
```

3.



évaluation en fonction du nombre d'itération



itérés sur les lignes de niveau

On atteint le minimum en 147 itérations

```
syms x1 x2;

rosenbrock = 100*(x2 - x1^2)^2 + (1 - x1)^2;
f = x1^2 + 2*x2^2;

[xstar, path, iter, f_value] = gd_opt(rosenbrock, gradient(rosenbrock), [-1;1.2], 160);

ezcontour(rosenbrock, [-3;3;-3;3]);
hold on;
plot(path(1, :), path(2, :), '-*');
```

```

figure();
plot(iter,f_value);

function [x_i, path, iter, f_value] = gd_opt(f, df, x_0, nb_iter)
    x_i = x_0;
    path = [];
    iter = [];
    f_value = [];

    for i = 1:nb_iter
        iter = [iter, i];

        x1 = x_i(1);
        x2 = x_i(2);

        d = -double(subs(df));

        tg = 0; td = Inf; t = 1; m1 = 0.1; m2 = 0.9;

        g0 = double(subs(f));

        f_value = [f_value, g0];

        gp0 = double(d.*subs(df));

        while abs(tg-td)>1e-3
            path = [path, x_i];

            x = x_i + t*d;
            x1 = x(1);
            x2 = x(2);

            gt = double(subs(f));
            gpt = double(subs(df).*d);

            if(gt <= (g0 + m1*t*gp0))
                if (gpt >= m2*gp0)
                    break;
                else
                    tg = t;

                    if(td == Inf)
                        t = 10*tg;
                    else
                        t = (td + tg)/2;
                    end
                end
            else
                td = t;

                if(td == Inf)
                    t = 10*tg;
                else
                    t = (td + tg)/2;
                end
            end
        end
        x_i = x_i + t*d;
    end
end
```