```
function lab2
 partie1();
    partie2();
    partie3();
    partie4();
    partie5();
end

% Partie 1 - A)
function distance = calcule_distance(pt1, pt2)
 % On utilise la version matlab potentiellement plus rapide?
 % distance = sqrt( (pt2(1) - pt1(1))^2 + (pt2(2) - pt1(2))^2);
 distance = pdist2(pt1, pt2);
end

% Partie 1 - B)
function initialise_variables_globales()

 % (Ré-)initialisation de l'environnement MATLAB avant l'exécution.
 clear;
 clc;

    % Précision
    global precision;
    global x_capteur_maximum;
    global y_capteur_maximum;
    x_capteur_maximum = 7000;
    y_capteur_maximum = 500;
    precision = 100;

 % Amplitude des signaux
 global amplitude_initiale_source1;
    global amplitude_initiale_source2;
    amplitude_initiale_source1 = 1;
 amplitude_initiale_source2 = 1;

 % Distance séparant les éléments
    global distance_x_source1_source2;
    global distance_x_source2_capteur;
 distance_x_source1_source2 = 700;
 distance_x_source2_capteur = 5000;

 % Hauteur des éléments
    global hauteur_sources;
    global hauteur_capteur;
 hauteur_sources = 500;
 hauteur_capteur = 300;

 % Fréquence en hertz
 frequence_signal = 150000;

 % Période du signal
```

```
 periode_signal = 1 / frequence_signal;

 % Vitesse de la lumière en m/s
 vitesse_lumiere = 299792458;

 % Longueur d'onde
 global longueur_onde;
    longueur_onde = vitesse_lumiere / frequence_signal;

 % On place la base de la source1 à  l'origine (x=0).
 % On calcule ensuite les différentes composantes en X.
 x_source1 = 0;
 x_source2 = x_source1 + distance_x_source1_source2;
 x_capteur = x_source2 + distance_x_source2_capteur;

 % On initialise les coordonnées initiales des sources/du capteur.
 global xy_source1;
    global xy_source2;
    global xy_capteur;
    xy_source1 = [x_source1, hauteur_sources];
    xy_source2 = [x_source2, hauteur_sources];
    xy_capteur = [x_capteur, hauteur_capteur];

 % Paramètres alpha et beta
 global alpha;
    global beta;
    alpha = 0.0001;
 beta = 0;
end

% Partie 1 - C)
function [distance_source1_capteur, distance_source2_capteur] =
 calcule_distances_sources(xy_capteur)

 global xy_source1;
 global xy_source2;

    % On calcule la distance au capteur pour chacune des sources.
 distance_source1_capteur = calcule_distance(xy_source1, xy_capteur);
 distance_source2_capteur = calcule_distance(xy_source2, xy_capteur);
end

% Partie 1 - D)
function [amplitude, phase] = calcule_propagation(distance,
 amplitude_initiale)

    global longueur_onde;
    global alpha;
    global beta;

    % On calcule l'amplitude et la phase en utilisant les fonctions
    % présentées dans le document du laboratoire.
    amplitude = amplitude_initiale * exp( -alpha * distance - beta );
    phase = ( mod(distance, longueur_onde) * 2 * pi ) / longueur_onde;
```

```
end

% Partie 1 - E)
function [amplitude_totale] = calcule_somme_signaux(amplitudes,
 phases)

    % L'amplitude totale est la somme des signaux.
    somme = 0;
    for i = 1:length(amplitudes)
        phi = phases(i);
        A = amplitudes(i);
        somme = somme + A * cos(phi);
    end
    amplitude_totale = abs(somme);
end

function partie1()

    global amplitude_initiale_source1;
    global amplitude_initiale_source2;
    global xy_capteur;

 fprintf('Partie 1\n');
    initialise_variables_globales();
    [dst1, dst2] = calcule_distances_sources(xy_capteur);
    fprintf('La distance entre la source 1 et le capteur est de %g.
\n', dst1);
 fprintf('La distance entre la source 2 et le capteur est de %g.\n',
 dst2);
    [A1, phi1] = calcule_propagation(dst1,
 amplitude_initiale_source1);
    [A2, phi2] = calcule_propagation(dst2,
 amplitude_initiale_source2);
    amplitude = calcule_somme_signaux([A1, A2], [phi1, phi2]);
    fprintf('L''amplitude du signal reçue par le capteur dest de %g.
\n', amplitude);
end

% Partie 2 - A)
function [amplitude_totale] =
 calcule_amplitude_en_fonction_capteur_x(x)

    global amplitude_initiale_source1;
    global amplitude_initiale_source2;
    global hauteur_capteur;

    nouveau_xy_capteur = [x, hauteur_capteur];
    [dst1, dst2] = calcule_distances_sources(nouveau_xy_capteur);
    [A1, phi1] = calcule_propagation(dst1,
 amplitude_initiale_source1);
    [A2, phi2] = calcule_propagation(dst2,
 amplitude_initiale_source2);
    amplitude_totale = calcule_somme_signaux([A1, A2], [phi1, phi2]);
end
```

```
% Partie 2 - B)
function [Xs] = genere_xs_reflexions()

    global x_capteur_maximum;
    global precision;

    start = 0;
    stop = x_capteur_maximum;
    Xs = linspace(start, stop, precision);
end

% Partie 2 - C)
function [Xs, amplitudes] =
 trace_graphique_amplitude_en_fonction_capteur_x()

    global x_capteur_maximum;
    global precision;

    % On créer un ensemble de point sur linéairement espacé sur
    % l'intervalle sélectionnée, selon la précision.
    start = 0;
    stop = x_capteur_maximum;
    Xs = linspace(start, stop, precision);
    amplitudes = zeros(1, length(Xs));

    for i = 1:length(Xs)
        x = Xs(i);
        amplitudes(i) = calcule_amplitude_en_fonction_capteur_x(x);
    end

    % On créer une nouvelle figure pour tracer la fonction et la
 dérivée.
    figure
    subplot(2,1,1);
    plot(Xs, amplitudes);
    title('Amplitude du signal en fonction de la position du
 capteur');
    xlabel('Position x du capteur');
    ylabel('Amplitude du signal');

    % On calcule la différentiation pour chaque point calculé.
    subplot(2,1,2);
    DYs = diff(amplitudes);
    DXs = diff(Xs);
    DYDXs = zeros(1, length(DYs));
    for i = 1:length(DYs)
        DYDXs(i) = DYs(i)/DXs(i);
    end
    % On doit utiliser de nouveaux X car les valeurs sont décaler de
 1.
    Xs_diff = linspace(start, stop, length(DYs));

    subplot(2,1,2);
```

```
    plot(Xs_diff, DYDXs);
    title('Dérivé de la fonction amplitude en fonction de la position
 du capteur');
    xlabel('Position x du capteur');
    ylabel('Taux de variation');
end

% Partie 2 - D)
function [amplitude, position] = optimise_position_du_capteur(Xs,
 amplitudes)

    % On aide un peu le départ des itérations en trouvant le meilleur
 point
    % pré-calculé.
    [maximum, index] = max(amplitudes);
    fun = @calcule_amplitude_en_fonction_capteur_x;
    winner = point_fixe(fun, Xs(index), 0.0001);
    fprintf('Winner: %g\n', winner);
    amplitude = maximum;
    position = Xs(index);

end

function [racine] = point_fixe(fonction, p0, tolerance)
    maximum_iterations = 10;
    p(1) = p0;
    tolerance = 1e-05;

    for i = 1:maximum_iterations
        p(i+1) = fonction(p(i));

        if abs(p(i+1) - p(i)) < tolerance
            racine = p(i+1);
            return
        end
    end
    racine = p(i+1);
end

function partie2()

    fprintf('Partie 2\n');
    [Xs, amplitudes] =
 trace_graphique_amplitude_en_fonction_capteur_x();
    [amplitude, position] = optimise_position_du_capteur(Xs,
 amplitudes);
    fprintf('La meilleure position de x est %g donnant une amplitude
 de %g.\n', position, amplitude);
end

% Partie 3 - A)
function [distances] = distances_trajectoires(xy_source, xy_capteur,
 Xs_sol)
```

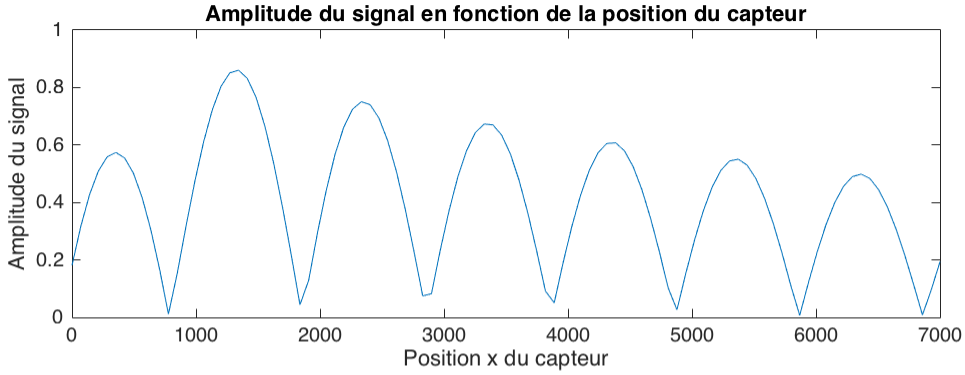
```
    distances = zeros(1, length(Xs_sol));
    for i = 1:length(Xs_sol)
        xy_reflexion = [Xs_sol(i), 0];
        distances(i) = distance_trajectoire(xy_source, xy_capteur,
 xy_reflexion);
    end
end

function [distance] = distance_trajectoire(xy_source, xy_capteur,
 xy_reflexion)
    distance = calcule_distance(xy_source, xy_reflexion) +
 calcule_distance(xy_reflexion, xy_capteur);
end

% Partie 3 - B)
function [distance, position] = optimise_trajectoires(Xs, distances)
    [minimum, index] = min(distances);
    distance = minimum;
    fun = @calcule_amplitude_en_fonction_capteur_x;
    point_fixe(fun, Xs(index), 0.0001);
    position = Xs(index);
end

function [distance] = calcule_distance_avec_reflexion(xy_source,
 xy_capteur)
    Xs_reflexion = genere_xs_reflexions();
    distances = distances_trajectoires(xy_source, xy_capteur,
 Xs_reflexion);
    [distance, position] = optimise_trajectoires(Xs_reflexion,
 distances);
end

% Partie 3 - C)
function [distances, amplitudes, phases] =
 calcule_amplitudes_et_phases(xy_capteur)

    global xy_source1;
    global xy_source2;
    global amplitude_initiale_source1;
    global amplitude_initiale_source2;
    global beta;

    beta = 0;
    distance1 = calcule_distance(xy_source1, xy_capteur);
    [a1, p1] = calcule_propagation(distance1,
 amplitude_initiale_source1);
    beta = 0.6;
    distance1r = calcule_distance_avec_reflexion(xy_source1,
 xy_capteur);
    [a1r, p1r] = calcule_propagation(distance1r,
 amplitude_initiale_source1);
    beta = 0;
    distance2 = calcule_distance(xy_source2, xy_capteur);
```

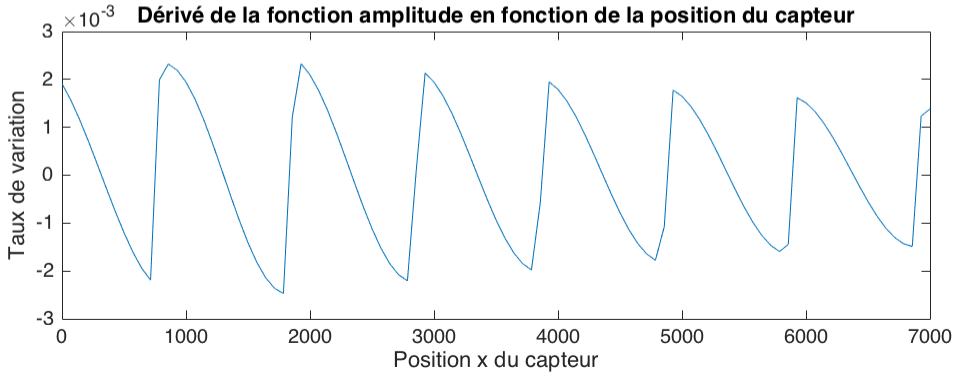
```
    [a2, p2] = calcule_propagation(distance2,
 amplitude_initiale_source2);
    beta = 0.6;
    distance2r= calcule_distance_avec_reflexion(xy_source2,
 xy_capteur);
    [a2r, p2r] = calcule_propagation(distance2r,
 amplitude_initiale_source2);

    distances = [distance1, distance1r, distance2, distance2r];
    amplitudes = [a1, a1r, a2, a2r];
    phases = [p1, p1r, p2, p2r];

end

function [amplitude_totale] =
 calcule_somme_signaux_avec_reflexion(xy_capteur)

    [distances, amplitudes, phases] =
 calcule_amplitudes_et_phases(xy_capteur);
    amplitude_totale = calcule_somme_signaux(amplitudes, phases);
end

function partie3()

    global xy_capteur;

    fprintf('Partie 3\n');
    amplitude_totale =
 calcule_somme_signaux_avec_reflexion(xy_capteur);
    fprintf('L''amplitude reçue par le capteur est de %g avec les
 réflexions sur le sol.\n', amplitude_totale);

end

% Partie 4 - A)
function [amplitude_totale] =
 calcule_amplitude_avec_reflexion_en_fonction_capteur_x(x)

    global hauteur_capteur;

    nouveau_xy_capteur = [x, hauteur_capteur];
    amplitude_totale =
 calcule_somme_signaux_avec_reflexion(nouveau_xy_capteur);
end

% Partie 4 - B)
% Voir optimise_position_du_capteur

% Partie 4 - C)
function [Xs, amplitudes] =
 trace_graphique_amplitude_avec_reflexion_en_fonction_capteur_x()

    global x_capteur_maximum;
    global precision;
```

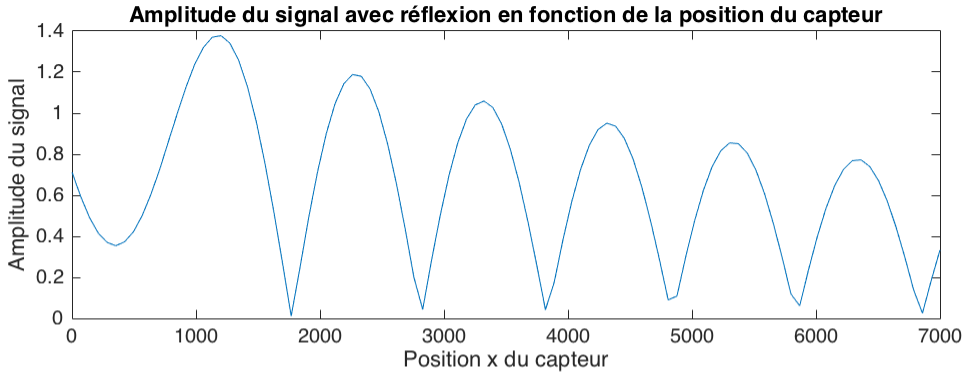
```
    % On créer un ensemble de point sur linéairement espacé sur
    % l'intervalle sélectionnée, selon la précision.
    start = 0;
    stop = x_capteur_maximum;
    Xs = linspace(start, stop, precision);
    amplitudes = zeros(1, length(Xs));

    for i = 1:length(Xs)
        x = Xs(i);
        amplitudes(i) =
 calcule_amplitude_avec_reflexion_en_fonction_capteur_x(x);
    end

    % On créer une nouvelle figure pour tracer la fonction et la
 dérivée.
    figure
    subplot(2,1,1);
    plot(Xs, amplitudes);
    title('Amplitude du signal avec réflexion en fonction de la
 position du capteur');
    xlabel('Position x du capteur');
    ylabel('Amplitude du signal');

    % On calcule la différentiation pour chaque point calculé.
    subplot(2,1,2);
    DYs = diff(amplitudes);
    DXs = diff(Xs);
    DYDXs = zeros(1, length(DYs));
    for i = 1:length(DYs)
        DYDXs(i) = DYs(i)/DXs(i);
    end
    % On doit utiliser de nouveaux X car les valeurs sont décaler de
 1.
    Xs_diff = linspace(start, stop, length(DYs));

    subplot(2,1,2);
    plot(Xs_diff, DYDXs);
    title('Dérivé de la fonction amplitude du signal avec réflexion en
 fonction de la position du capteur');
    xlabel('Position x du capteur');
    ylabel('Taux de variation');
end

function partie4()

    fprintf('Partie 4\n');
    [Xs, amplitudes] =
 trace_graphique_amplitude_avec_reflexion_en_fonction_capteur_x();
    [amplitude, position] = optimise_position_du_capteur(Xs,
 amplitudes);
    fprintf('La meilleure position de x est %g pour une amplitude de
 %g\n', position, amplitude);
end
```

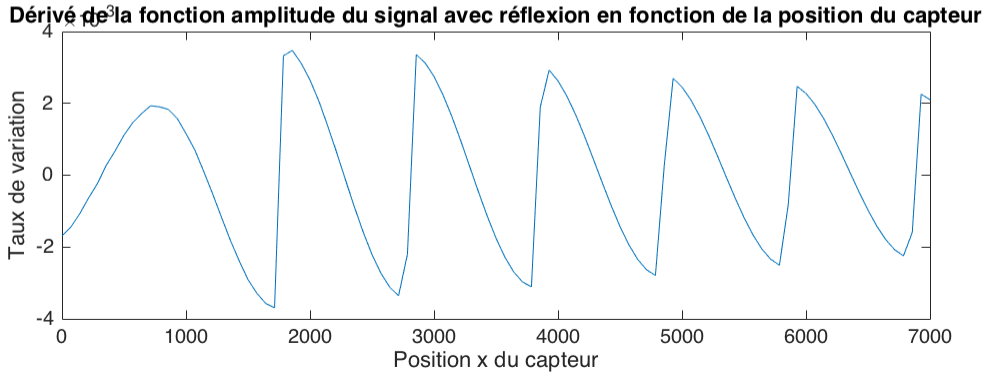
```
% Partie 5 - A)
function [Xs, Ys, Zs] = calcule_des_distances()

    global x_capteur_maximum;
    global y_capteur_maximum;
    global precision;

    Xs = linspace(0, x_capteur_maximum, precision);
    Ys = linspace(0, y_capteur_maximum, precision);

    for x = 1:length(Xs)
        for y = 1:length(Ys)
            xy_capteur = [Xs(x), Ys(y)];
            valeurs(x,y) =
 calcule_somme_signaux_avec_reflexion(xy_capteur);
        end
    end
    Zs = valeurs;
end

function partie5()

    fprintf('Partie 5\n');
    [Xs, Ys, Zs] = calcule_des_distances();
    figure
    surf(Xs, Ys, Zs);
end
```

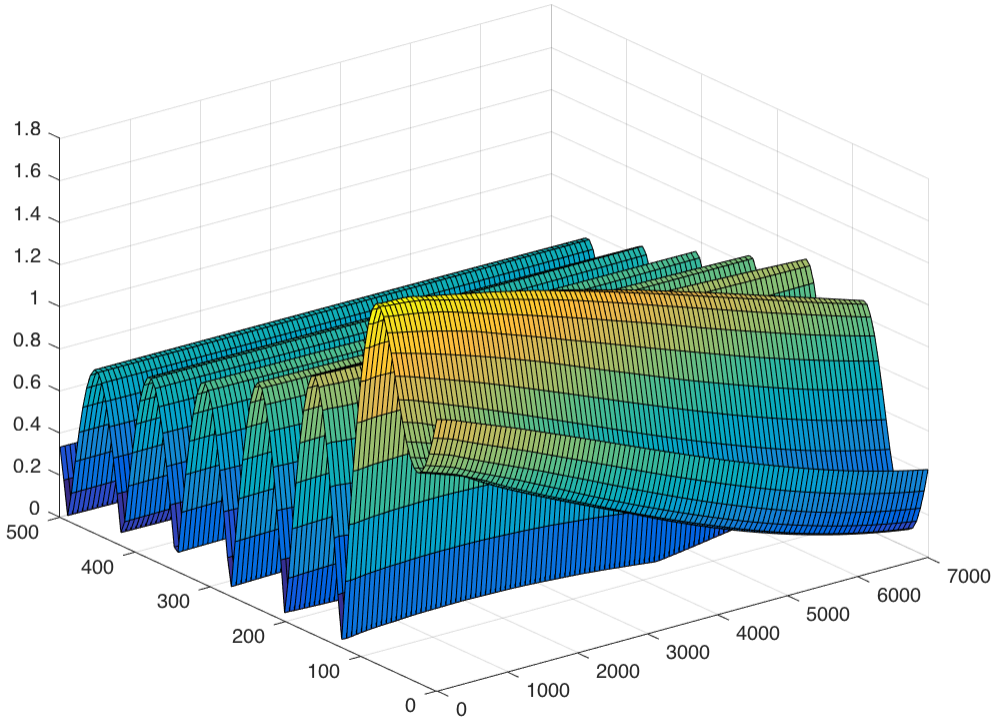
```
Partie 1
La distance entre la source 1 et le capteur est de 5703.51.
La distance entre la source 2 et le capteur est de 5004.
L'amplitude du signal reçue par le capteur dest de 0.26321.
Partie 2
Winner: 0.181502
La meilleure position de x est 1343.43 donnant une amplitude de
 0.860353.
Partie 3
L'amplitude reçue par le capteur est de 0.361756 avec les réflexions
 sur le sol.
Partie 4
Winner: 0.099518
La meilleure position de x est 1202.02 pour une amplitude de 1.37769
Partie 5
```


Amplitude du signal en fonction de la position du capteur
Amplitude du signal
Position x du capteur
0
0.2
0.4
0.6
0.8
1
0
1000
2000
3000
4000
5000
6000
7000


Dérivé de la fonction amplitude en fonction de la position du capteur
×10-3
Taux de variation
Position x du capteur
-3
-2
-1
0
1
2
3
0
1000
2000
3000
4000
5000
6000
7000


Amplitude du signal avec réflexion en fonction de la position du capteur
Amplitude du signal
Position x du capteur
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
1000
2000
3000
4000
5000
6000
7000


Dérivé de la fonction amplitude du signal avec réflexion en fonction de la position du capteur
Taux de variation
Position x du capteur
-4
-2
0
2
4
0
1000
2000
3000
4000
5000
6000
7000



Published with MATLAB® R2015b