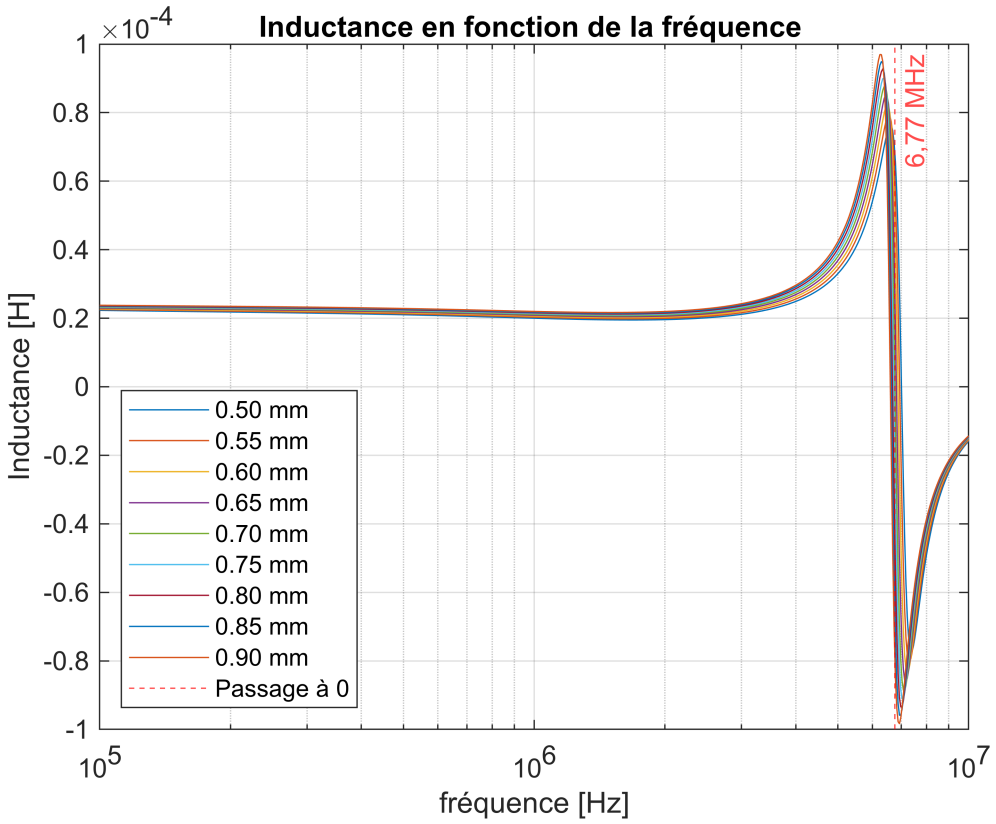
Séance 1

```
%NOTE: Changement de maquette : 09 ; montage elec 12
clear all;
close all;
clc;
% Fichiers Machine Impédances
mes_050 = readmatrix('C:\Users\romai\Bureau\HEIG-VD\S5\Capteurs\Projet\Seance1\MesuresMachine\5
mes_055 = readmatrix('C:\Users\romai\Bureau\HEIG-VD\S5\Capteurs\Projet\Seance1\MesuresMachine\5
mes_060 = readmatrix('C:\Users\romai\Bureau\HEIG-VD\S5\Capteurs\Projet\Seance1\MesuresMachine\6
mes_065 = readmatrix('C:\Users\romai\Bureau\HEIG-VD\S5\Capteurs\Projet\Seance1\MesuresMachine\6
mes_070 = readmatrix('C:\Users\romai\Bureau\HEIG-VD\S5\Capteurs\Projet\Seance1\MesuresMachine\7
mes_075 = readmatrix('C:\Users\romai\Bureau\HEIG-VD\S5\Capteurs\Projet\Seance1\MesuresMachine\7
mes_080 = readmatrix('C:\Users\romai\Bureau\HEIG-VD\S5\Capteurs\Projet\Seance1\MesuresMachine\8
mes_085 = readmatrix('C:\Users\romai\Bureau\HEIG-VD\S5\Capteurs\Projet\Seance1\MesuresMachine\8
mes_090 = readmatrix('C:\Users\romai\Bureau\HEIG-VD\S5\Capteurs\Projet\Seance1\MesuresMachine\9

%Calcul des valeurs complexes (z = R + iwL):
f = mes_050(:,1);
Z_050 = mes_050(:,3) + 2*pi*1i*f .* mes_050(:,2);
Z_055 = mes_055(:,3) + 2*pi*1i*f .* mes_055(:,2);
Z_060 = mes_060(:,3) + 2*pi*1i*f .* mes_060(:,2);
Z_065 = mes_065(:,3) + 2*pi*1i*f.* mes_065(:,2);
Z_070 = mes_070(:,3) + 2*pi*1i*f .* mes_070(:,2);
Z_075 = mes_075(:,3) + 2*pi*1i*f .* mes_075(:,2);
Z_080 = mes_080(:,3) + 2*pi*1i*f .* mes_080(:,2);
Z_085 = mes_085(:,3) + 2*pi*1i*f .* mes_085(:,2);
Z_090 = mes_090(:,3) + 2*pi*1i*f .* mes_090(:,2);
Z = [Z_050, Z_055, Z_060, Z_065, Z_070, Z_075, Z_080, Z_085, Z_090 ];

%Vecteur distance
x = 0.5:0.05:0.9;
figure;
```

Inductance en fonction de la fréquence L(f)

```
% Inductances
Lb_050 =  mes_050(:,2);
Lb_055 =  mes_055(:,2);
Lb_060 =  mes_060(:,2);
Lb_065 =  mes_065(:,2);
Lb_070 =  mes_070(:,2);
Lb_075 =  mes_075(:,2);
Lb_080 =  mes_080(:,2);
Lb_085 =  mes_085(:,2);
Lb_090 =  mes_090(:,2);
Lb = [Lb_050, Lb_055, Lb_060, Lb_065, Lb_070, Lb_075, Lb_080, Lb_085, Lb_090 ];
% Graphique
plot(f, Lb)
```

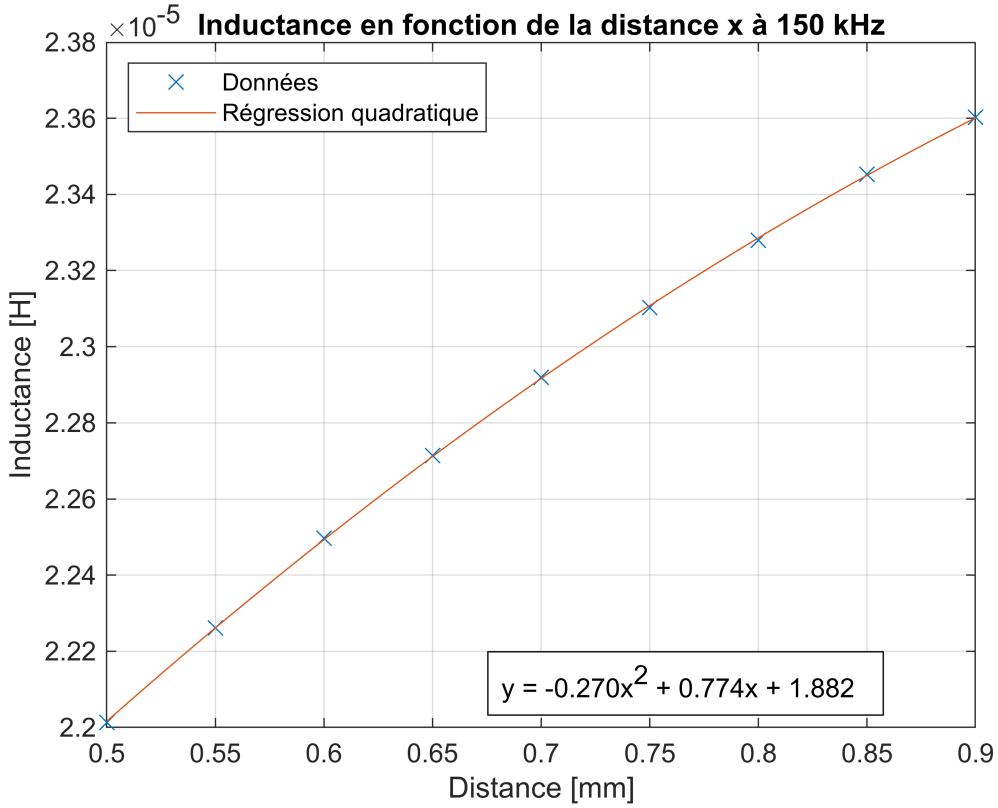
```
title('Inductance en fonction de la fréquence ');
ylabel("Inductance [H]")
xlabel("fréquence [Hz]")
xline(6.767*1000000,'--r', '6,77 MHz');
legend('0.50 mm', '0.55 mm', '0.60 mm', '0.65 mm', '0.70 mm', '0.75 mm', '0.80 mm','0.85 mm', '
xlim([8e5 1e7]);
set(gca, 'XScale', 'log')
xlim([100e3 10^7])
grid;
```



Inductance à 150 kHz en fonction de la distance L(x)

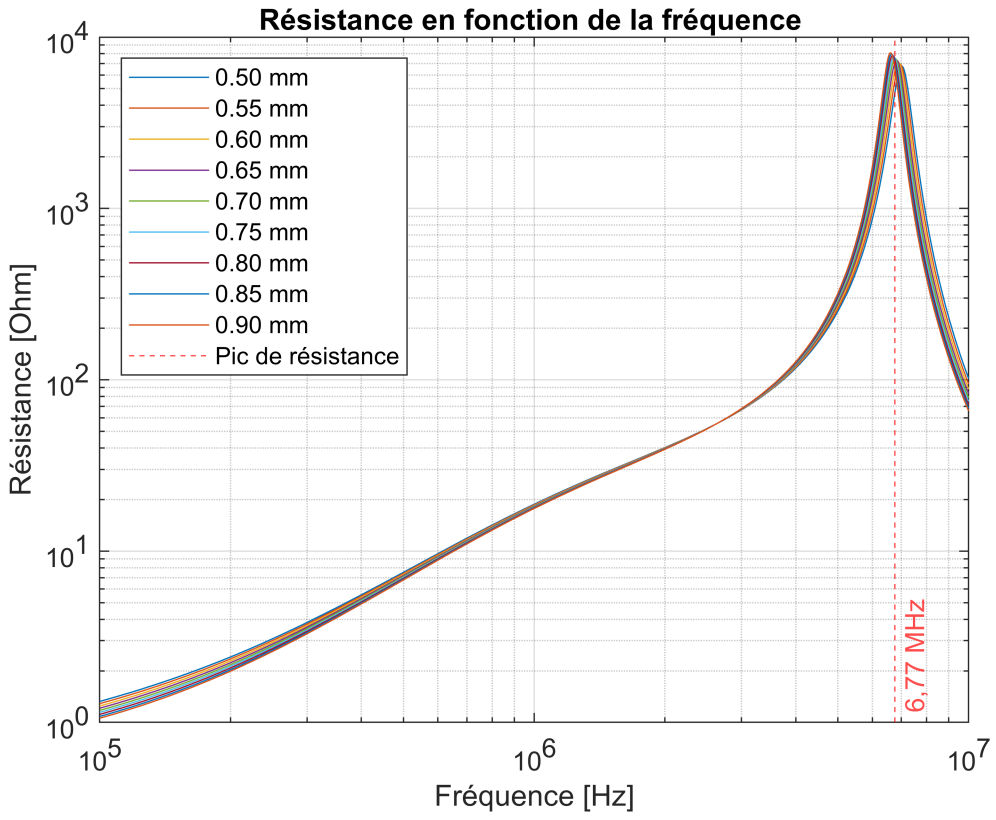
```
% Inductances à 150kHz
Lbx_050 =  mes_050(26,2);
Lbx_055 =  mes_055(26,2);
Lbx_060 =  mes_060(26,2);
Lbx_065 =  mes_065(26,2);
Lbx_070 =  mes_070(26,2);
Lbx_075 =  mes_075(26,2);
Lbx_080 =  mes_080(26,2);
Lbx_085 =  mes_085(26,2);
Lbx_090 =  mes_090(26,2);
Lbx = [Lbx_050, Lbx_055, Lbx_060, Lbx_065, Lbx_070, Lbx_075, Lbx_080, Lbx_085, Lbx_090];
```

```
%Graphique

% Régression quadratique
coeffs1 = polyfit(x, Lbx, 2); % Ajustement d'une courbe d'ordre 2
x_fit = linspace(min(x), max(x), 100); % Points pour tracer la courbe lisse
Lbx_fit = polyval(coeffs1, x_fit); % Calcul des valeurs ajustées
% Tracé des données et de la régression
figure;
plot(x, Lbx, 'x', 'MarkerSize', 8, 'DisplayName', 'Données'); % Données mesurées
hold on;
plot(x_fit, Lbx_fit, '-', 'DisplayName', 'Régression quadratique'); % Courbe ajustée
grid on;
% Ajouter les labels et le titre
title('Inductance en fonction de la distance x à 150 kHz');
ylabel('Inductance [H]');
xlabel('Distance [mm]');
% Ajouter l'équation comme annotation
eqn1 = sprintf("y = %.3fx^2 + %.3fx + %.3f", coeffs1(1)*100000, coeffs1(2)*100000,coeffs1(3)*10
dim = [0.47 0.1 0.1 0.1]; % Position de l'annotation (valeurs normalisées entre 0 et 1)
annotation('textbox', dim, 'String', eqn1, 'FitBoxToText', 'on', 'BackgroundColor', 'w');
% Légende
legend('Location','northwest');
hold off;
```



Résistance en fonction de la fréquence R(f)

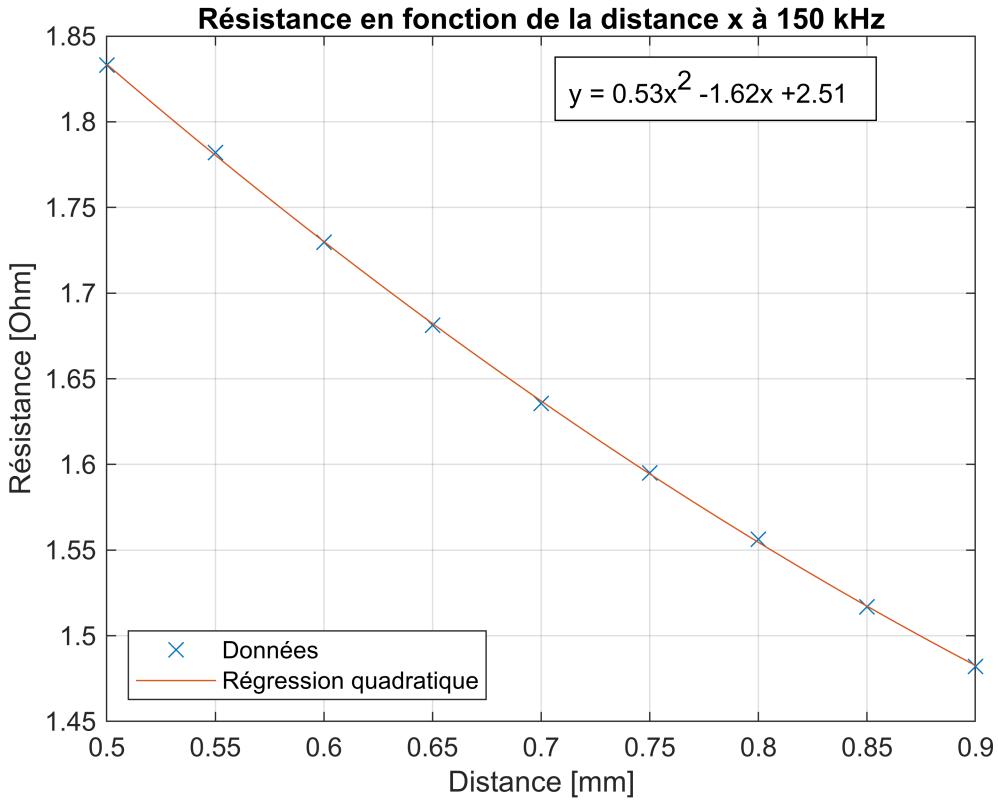
```
% Résistances
Rb_050 =  mes_050(:,3);
Rb_055 =  mes_055(:,3);
Rb_060 =  mes_060(:,3);
Rb_065 =  mes_065(:,3);
Rb_070 =  mes_070(:,3);
Rb_075 =  mes_075(:,3);
Rb_080 =  mes_080(:,3);
Rb_085 =  mes_085(:,3);
Rb_090 =  mes_090(:,3);
Rb = [Rb_050, Rb_055, Rb_060, Rb_065, Rb_070, Rb_075, Rb_080, Rb_085, Rb_090 ];

% Graphique échelle log/log
loglog(f, Rb);
grid;
title('Résistance en fonction de la fréquence ');
xl = xline(6.767*1000000,'--r', '6,77 MHz');
xl.LabelVerticalAlignment = 'bottom';
legend('0.50 mm', '0.55 mm', '0.60 mm', '0.65 mm', '0.70 mm', '0.75 mm', '0.80 mm','0.85 mm',
xlim([100e3 10^7])
ylabel("Résistance [Ohm]")
xlabel("Fréquence [Hz]")
```



Résistance à 150 kHz en fonction de la distance R(x)

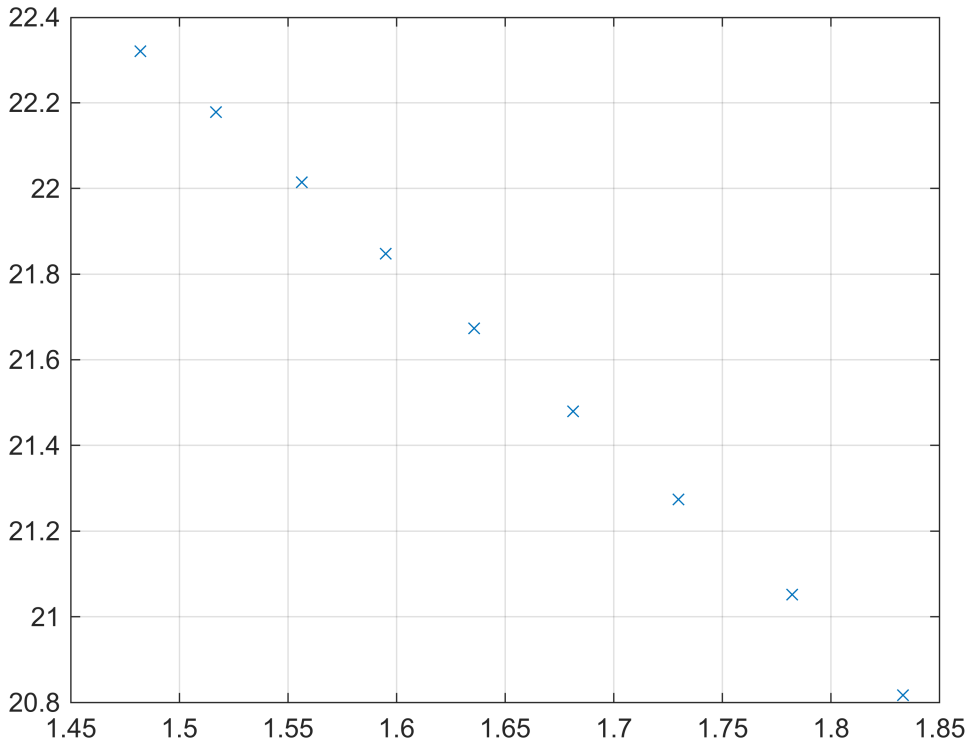
```
%Résistances à 150 kHz
```

```
Rbx_050 =  mes_050(26,3);
Rbx_055 =  mes_055(26,3);
Rbx_060 =  mes_060(26,3);
Rbx_065 =  mes_065(26,3);
Rbx_070 =  mes_070(26,3);
Rbx_075 =  mes_075(26,3);
Rbx_080 =  mes_080(26,3);
Rbx_085 =  mes_085(26,3);
Rbx_090 =  mes_090(26,3);
Rbx = [Rbx_050, Rbx_055, Rbx_060, Rbx_065, Rbx_070, Rbx_075, Rbx_080, Rbx_085, Rbx_090 ];

% Graphique

% Régression quadratique
coeffs2 = polyfit(x, Rbx, 2); % Ajustement d'une courbe d'ordre 2
x_fit = linspace(min(x), max(x), 100); % Points pour tracer la courbe lisse
Rbx_fit = polyval(coeffs2, x_fit); % Calcul des valeurs ajustées
% Tracé des données et de la régression
figure;
plot(x, Rbx, 'x', 'MarkerSize', 8, 'DisplayName', 'Données'); % Données mesurées
hold on;
plot(x_fit, Rbx_fit, '-', 'DisplayName', 'Régression quadratique'); % Courbe ajustée
grid on;
% Ajouter les labels et le titre
title('Résistance en fonction de la distance x à 150 kHz');
ylabel('Résistance [Ohm]');
xlabel('Distance [mm]');
% Ajouter l'équation comme annotation
eqn2 = sprintf("y = %.2fx^2 %+.2fx %+.2f", coeffs2(1), coeffs2(2), coeffs2(3)); % Formater l'éq
dim = [0.53 0.8 0.1 0.1]; % Position de l'annotation (valeurs normalisées entre 0 et 1)
annotation('textbox', dim, 'String', eqn2, 'FitBoxToText', 'on', 'BackgroundColor', 'w');
% Légende
legend('Location', 'southwest');
hold off;
```

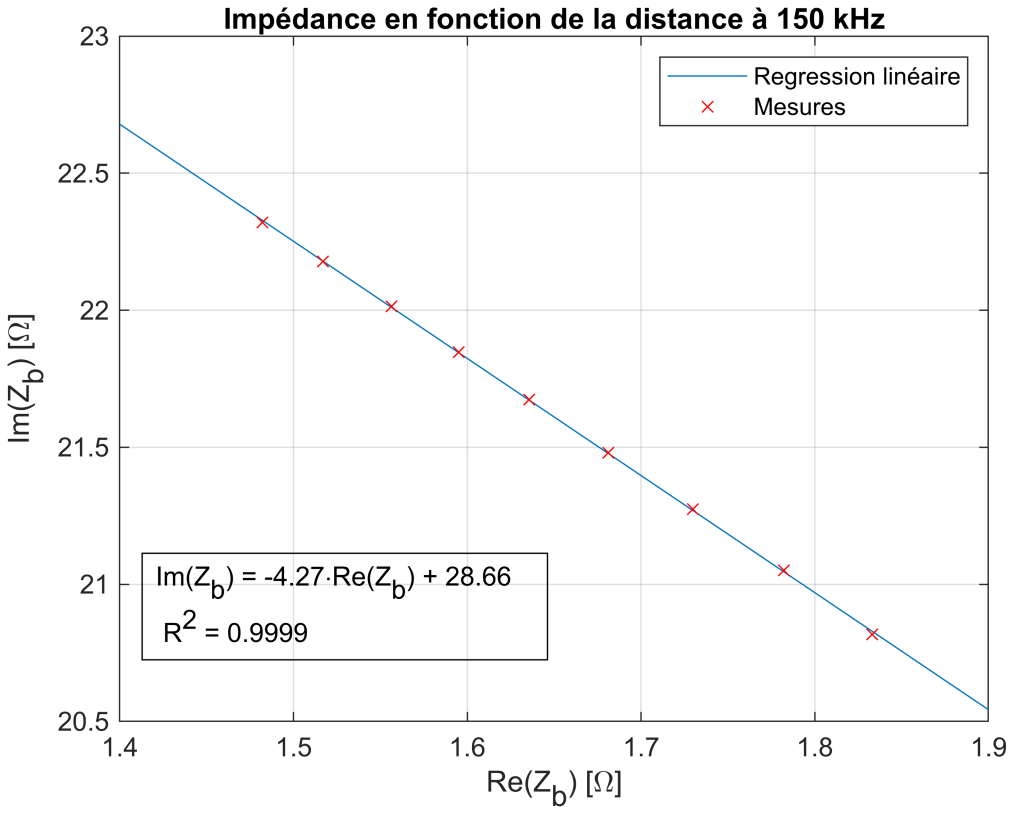


Impédence à 150 kHz

```
% Impédance
Z150 = [Z_050(26), Z_055(26), Z_060(26), Z_065(26), Z_070(26), Z_075(26), Z_080(26), Z_085(26),

% Graphique
figure;
plot(real(Z150),imag(Z150), 'x');
%Regression lineaire
x_reg = linspace(1.4,1.9,1000);
reg = polyfit(real(Z150(1,:)),imag(Z150(1,:)),1);
%Parametres
a =reg(1,1);
b= reg(1,2);
r = fitlm(real(Z150(1,:)),imag(Z150(1,:)));
R2 = r.Rsquared.Ordinary;
eqn3 = sprintf("Im(Z_{b}) = %.2f\\cdotRe(Z_{b}) + %.2f \n R^{2} = %.4f",reg(1),reg(2),R2);
grid on;
```

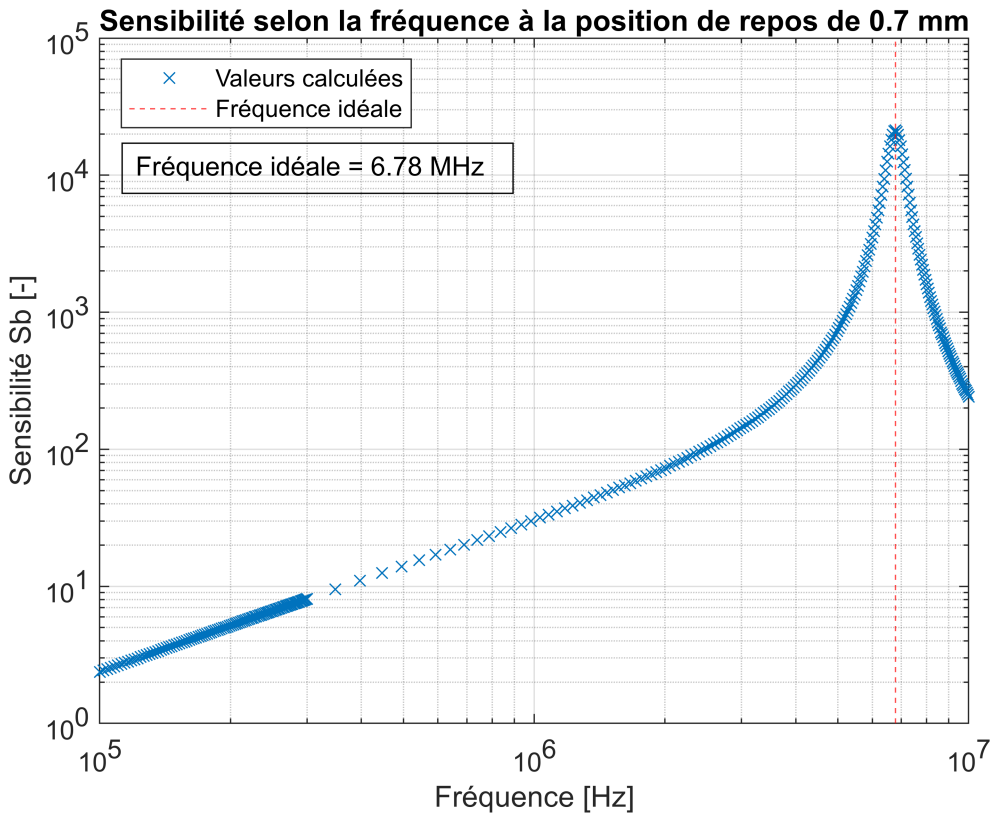



```
% Impédance avec regression lineaire
imZb = polyval(reg,x_reg);
plot(x_reg,imZb,"-",real(Z150(1,:)),imag(Z150(1,:)),"rx")
title('Impédance en fonction de la distance à 150 kHz');
annotation("textbox",[0.15 0.01 0.3 0.3],'string',eqn3,'FitBoxToText','on');
legend('Regression linéaire','Mesures','location','northeast');
xlabel('Re(Z_{b}) [\Omega]');
ylabel('Im(Z_{b}) [\Omega]');
grid on;
```



Sensibilité

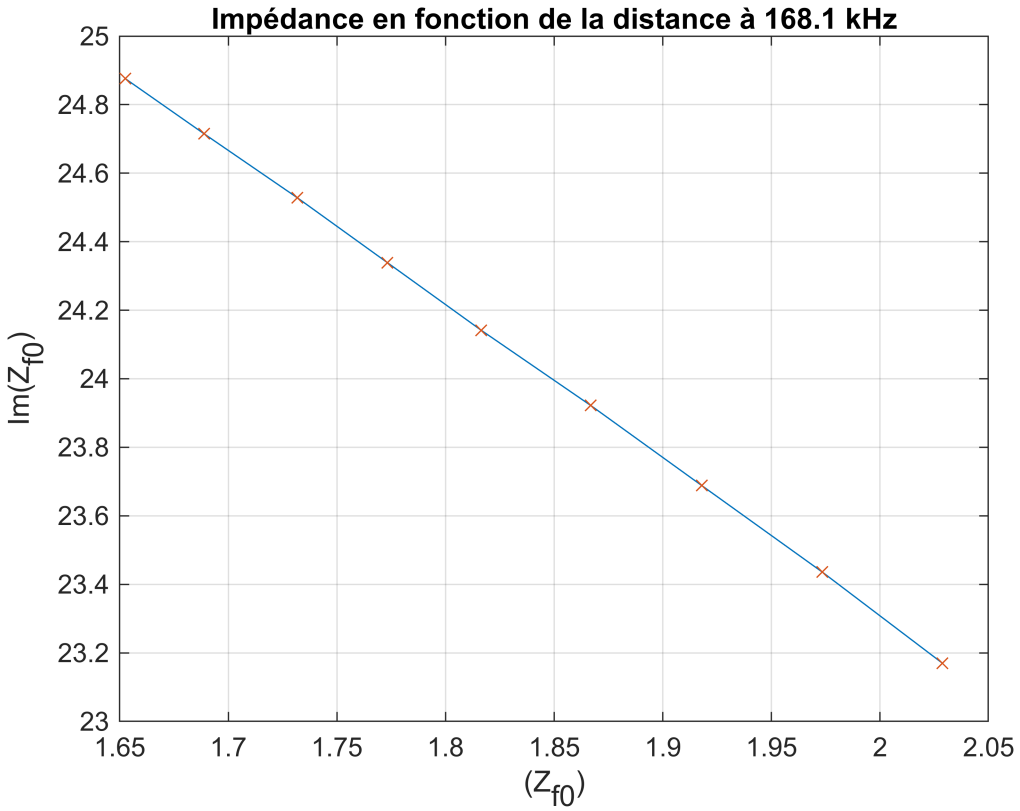
```
S = abs(Z(:, 4)-Z(:, 6))/0.1;
% Trouver le max
[vmax, indicefmax] = max(S);
fmax = f(indicefmax);
fmaxstr=sprintf("Fréquence idéale = %.2f MHz",fmax/(1e6));
% Graphique
loglog(f,S,"x")
xlabel('Fréquence [Hz]');
ylabel('Sensibilité Sb [-]');
title('Sensibilité selon la fréquence à la position de repos de 0.7 mm');
xline(fmax,'--r');
annotation("textbox",[0.15 0.5 0.3 0.3],'string',fmaxstr,'FitBoxToText','on');
legend('Valeurs calculées','Fréquence idéale','Location','northwest');
%%set(gca, 'YScale', 'log')
grid on;
```



Seance 2

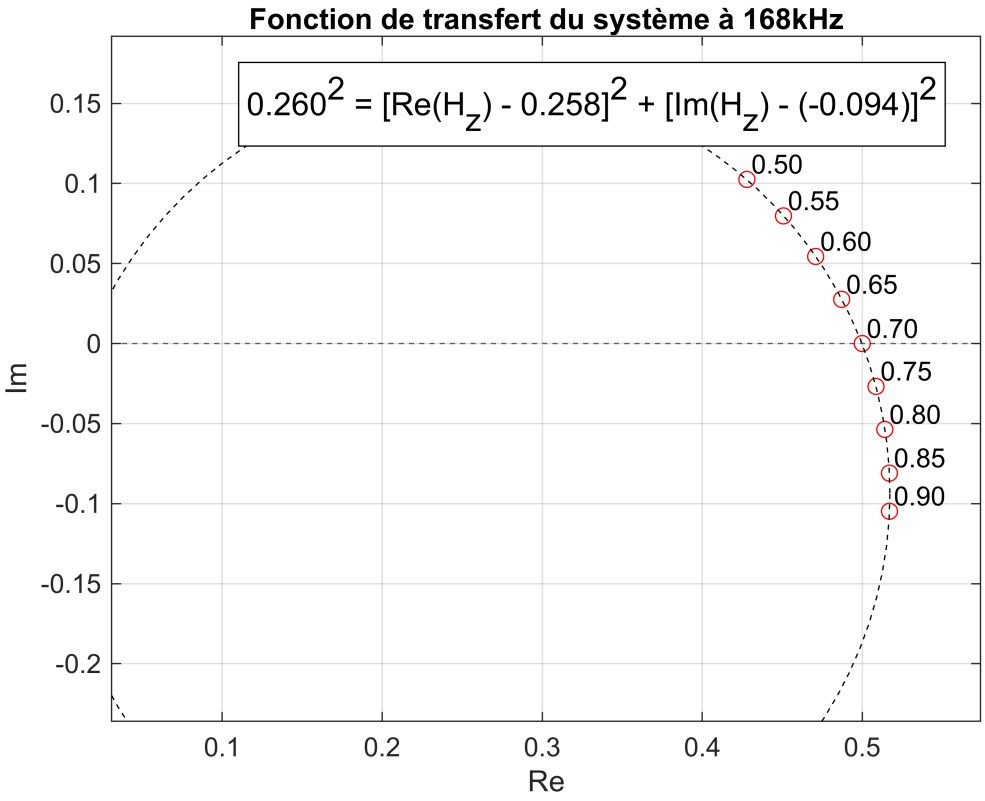
```
f0_mes = 166.5;     %170.8e3 ; %[Hz] (old)
Z(101, :) = [];
Z(300, :) = [];

% Calcul de la fréquence de résonnance théorique
f0 = 150.3 * 1e3; % [kH] fréquence résonnance mesurée
C = 39 * 1e-9;
f(101,:) = [];
f(300, :) = [];

for aaaa = 1 : 20
z_f0_70 = interp1(f,Z(:,5), f0);
Lb = imag(z_f0_70)/(f0*2*pi);
Rb = real(z_f0_70);
omega0 = sqrt(1/(Lb*C) - Rb^2/Lb^2);
f0 = omega0/(2*pi);
%  U excitation
end
z_f0 = interp1(f,Z,f0);
plot(real(z_f0),imag(z_f0),real(z_f0),imag(z_f0),'x')
ylabel('Im(Z_{f0})')
xlabel('(Z_{f0})')
title('Impédance en fonction de la distance à 168.1 kHz')
```

```
grid;
```



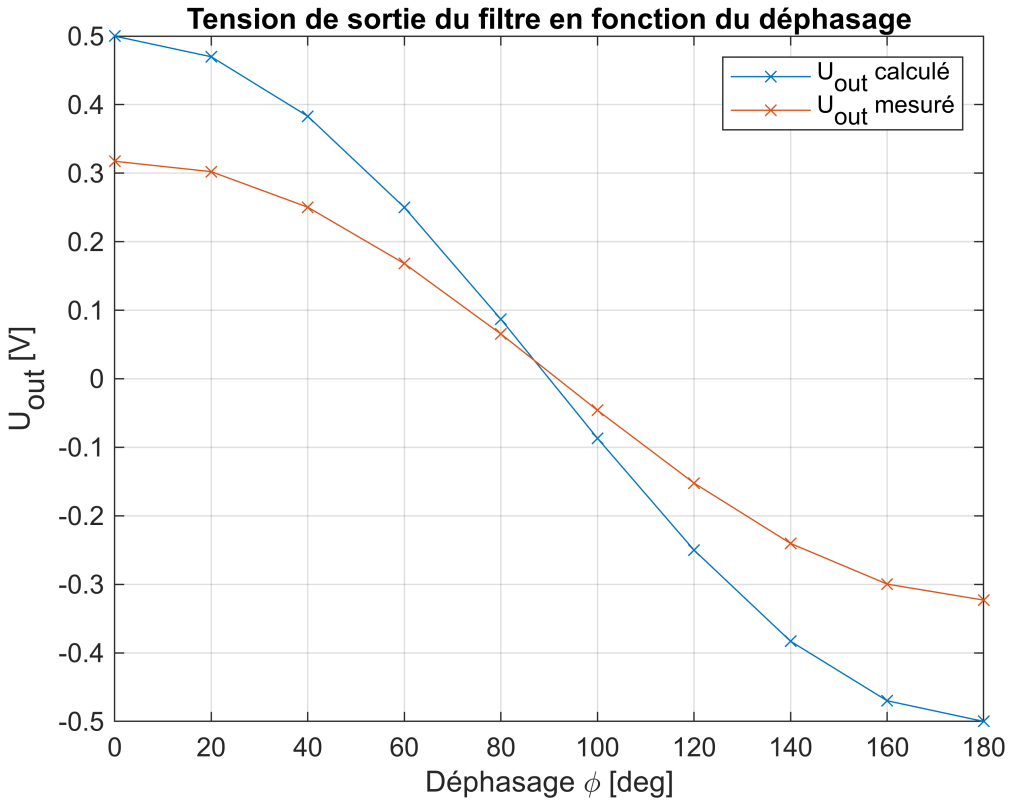

```
Q = omega0*Lb/Rb;
Rp = Rb*(1+Q^2);

Rmes = 324; %[Ohm]

UmodRMS = 1.074;
Umod = 1.519;

H_z_f0 = z_f0./((1+1i.*C.*Rp.*omega0).*z_f0 + Rp);
% Calcul des coordonnées réelles et imaginaires
x = real(H_z_f0);
y = imag(H_z_f0);

% Calcul du cercle passant par ces points (utilisation des moindres carrés pour ajuster un cerc
A = [2*x', 2*y', ones(length(H_z_f0), 1)];
B = x'.^2 + y'.^2;

% Résolution du système linéaire pour obtenir les paramètres du cercle (méthode des moindres ca
params = A \ B;  % Résolution du système de manière optimale en termes de moindres carrés

% Paramètres du cercle
xc = params(1);  % Centre du cercle en x
yc = params(2);  % Centre du cercle en y
r = sqrt(params(3) + xc^2 + yc^2);  % Rayon du cercle
```

```
% Affichage des points et du cercle
theta = linspace(0, 2*pi, 100);  % Angles pour dessiner le cercle
x_circle = xc + r * cos(theta);
y_circle = yc + r * sin(theta);

labels = {'0.50', '0.55', '0.60', '0.65', '0.70','0.75', '0.80', '0.85', '0.90'};

% Tracer
figure;
plot(x, y, 'ro');  % Points d'origine

for foo = 1:length(x)
    text(x(foo) + 0.035, y(foo), labels{foo}, 'VerticalAlignment', 'bottom', 'HorizontalAlignme
end
eqn4 = sprintf("%.3f^{2} = [Re(H_{z}) - %.3f]^{2} + [Im(H_{z}) - (%.3f)]^{2}",r,xc,yc);
hold on;
plot(x_circle, y_circle, 'k--');  % Cercle
axis equal;
xlabel('Re');
ylabel('Im');
title('Fonction de transfert du système à 168kHz');
grid on;
text(0.95, 0.95, eqn4, ...
    'Units', 'normalized', ... % Utiliser les coordonnées normalisées
    'HorizontalAlignment', 'right', ... % Alignement horizontal
    'VerticalAlignment', 'top', ... % Alignement vertical
    'FontSize', 12, ... % Taille de la police
    'BackgroundColor', 'w', ... % Couleur de fond (optionnel)
    'EdgeColor', 'k'); % Bordure autour du texte (optionnel)
hold off;
yline(0, '--')
xlim([0.031 0.574])
ylim([-0.236 0.192])
```



Seance 3

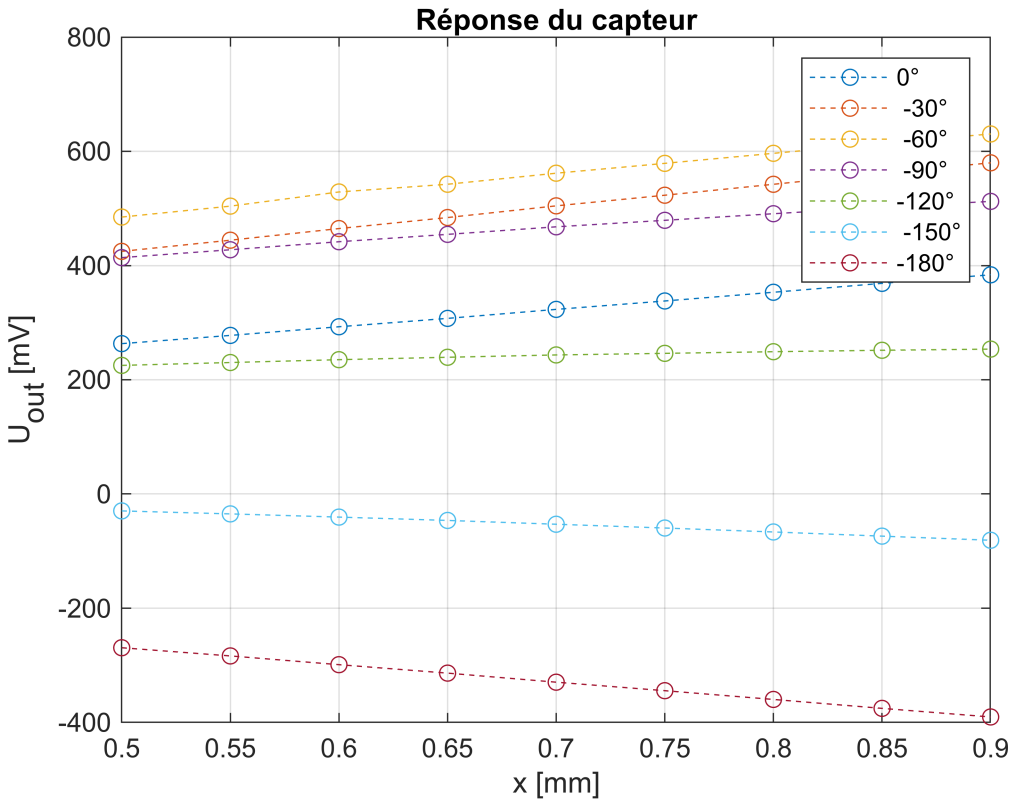
```
x = 0.5:0.05:0.9;
%Vecteur d'angle
angle = [0, 20, 40, 60, 80, 100, 120, 140, 160, 180];
%Tension de sortie mesurée
Umes = [317.4, 302.2, 250.3, 168.1, 65.28, -46.02, -152.2, -240.3, -299.6, -323];
% Tension de sortie mesurée
Uth = 0.5 * cos(angle*((2*pi)/360));
% Graphique
plot(angle, Uth,'-x', angle, Umes/1000,'-x')
xlabel('Déphasage \phi [deg]');
ylabel('U_{out} [V]');
legend('U_{out} calculé','U_{out} mesuré');
grid on;
title('Tension de sortie du filtre en fonction du déphasage');
```



Seance 4

```
opts = delimitedTextImportOptions("NumVariables", 8);

opts.DataLines = [2, Inf];
opts.Delimiter = ";";

opts.VariableNames = ["Var1", "d0", "d30", "d60", "d90", "d120", "d150", "d180"];
opts.SelectedVariableNames = ["d0", "d30", "d60", "d90", "d120", "d150", "d180"];
opts.VariableTypes = ["string", "double", "double", "double", "double", "double", "double", "do

opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";

opts = setvaropts(opts, "Var1", "WhitespaceRule", "preserve");
opts = setvaropts(opts, "Var1", "EmptyFieldRule", "auto");

tbl = readtable("C:\Users\romai\Bureau\HEIG-VD\S5\Capteurs\Projet\Seance4\Projet_Capteur4_v2.cs

d0 = tbl.d0;
d30 = tbl.d30;
d60 = tbl.d60;
d90 = tbl.d90;
d120 = tbl.d120;
d150 = tbl.d150;
```

```
d180 = tbl.d180;

clear opts tbl

plot(x,d0,'--o',x, d30,'--o',x,d60,'--o',x, d90,'--o',x,d120,'--o',x, d150,'--o',x,d180,'--o');
legend("0°"," -30°"," -60°"," -90°","-120°"," -150°","-180°")
title('Réponse du capteur')
xlabel('x [mm]')
ylabel('U_{out} [mV]')
grid;
```




```
pl0 = polyfit(x,d0,1);
pl30 = polyfit(x,d30,1);
pl60 = polyfit(x,d60,1);
pl90 = polyfit(x,d90,1);
pl120 = polyfit(x,d120,1);
pl150 = polyfit(x,d150,1);
pl180 = polyfit(x,d180,1);

pl0_reg = polyval(pl0,x)';
pl30_reg = polyval(pl30,x)';
pl60_reg = polyval(pl60,x)';
pl90_reg = polyval(pl90,x)';
pl120_reg = polyval(pl120,x)';
pl150_reg = polyval(pl150,x)';
pl180_reg = polyval(pl180,x)';
```

```
EL0 = 100*max(abs(d0 - pl0_reg))./(max(d0)- min(d0))
```

```
EL0 = 0.4706
```

```
EL30 = 100*max(abs(d30-pl30_reg))./(max(d30)-min(d30))
```

```
EL30 = 0.8025
```

```
EL60 = 100*max(abs(d60-pl60_reg))./(max(d60)-min(d60))
```

```
EL60 = 3.2960
```

```
EL90 = 100*max(abs(d90-pl90_reg))./(max(d90)-min(d90))
```

```
EL90 = 2.5606
```

```
EL120 = 100*max(abs(d120-pl120_reg))./(max(d120)-min(d120))
```

```
EL120 = 7.6658
```

```
EL150 = 100*max(abs(d150-pl150_reg))./(max(d150)-min(d150))
```

```
EL150 = 3.0767
```

```
EL180 = 100*max(abs(d180-pl180_reg))./(max(d180)-min(d180))
```

```
EL180 = 0.5303
```

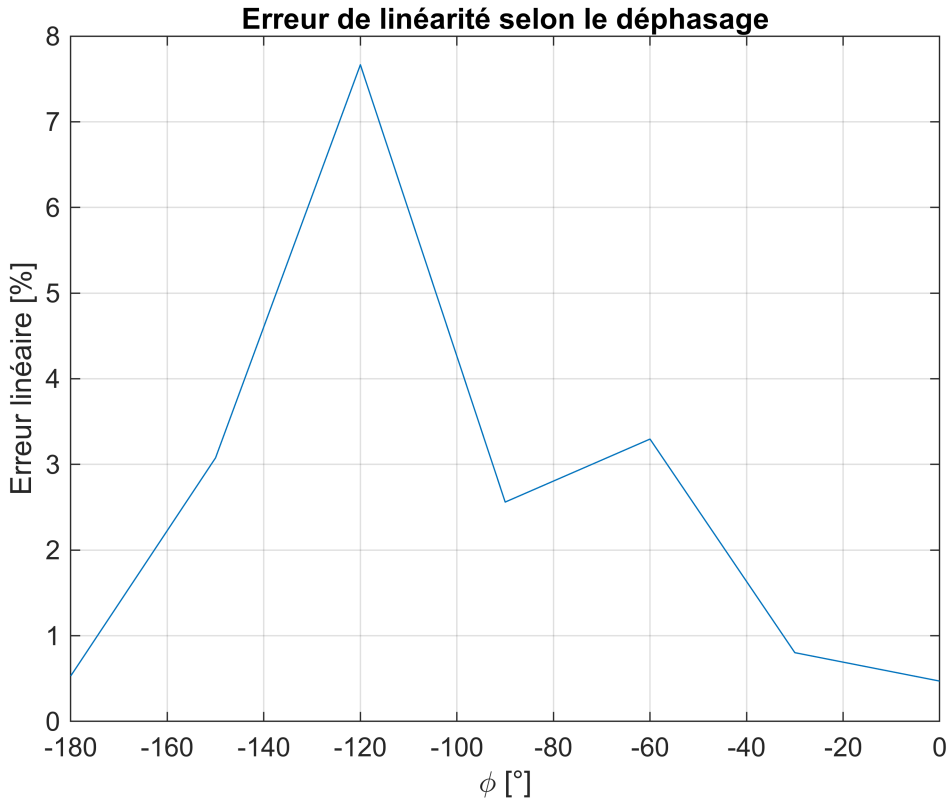
```
EL = [EL0 EL30 EL60 EL90 EL120 EL150 EL180]
```

```
EL = 1×7
    0.4706    0.8025    3.2960    2.5606    7.6658    3.0767    0.5303
```

```
d = 0 :-30 :-180
```

```
d = 1×7
     0   -30   -60   -90  -120  -150  -180
```

```
plot(d,EL)
title('Erreur de linéarité selon le déphasage')
xlabel('\phi [°]')
ylabel('Erreur linéaire [%]')
grid;
```



Séance 5

```
pente_reele = pl0(1) *10^-3 %V/mm
```

```
pente_reele = 0.3023
```

```
pente_voulue = 10; %V/mm
gain = pente_voulue/pente_reele
```

```
gain = 33.0775
```

```
r13 = 10E3/gain
```

```
r13 = 302.3200
```

```

% Résistance utilisée = 270 + 302 Ohms

r13_mes = 298.7;

gain_reel = 10E3/r13_mes
```

```
gain_reel = 33.4784
```

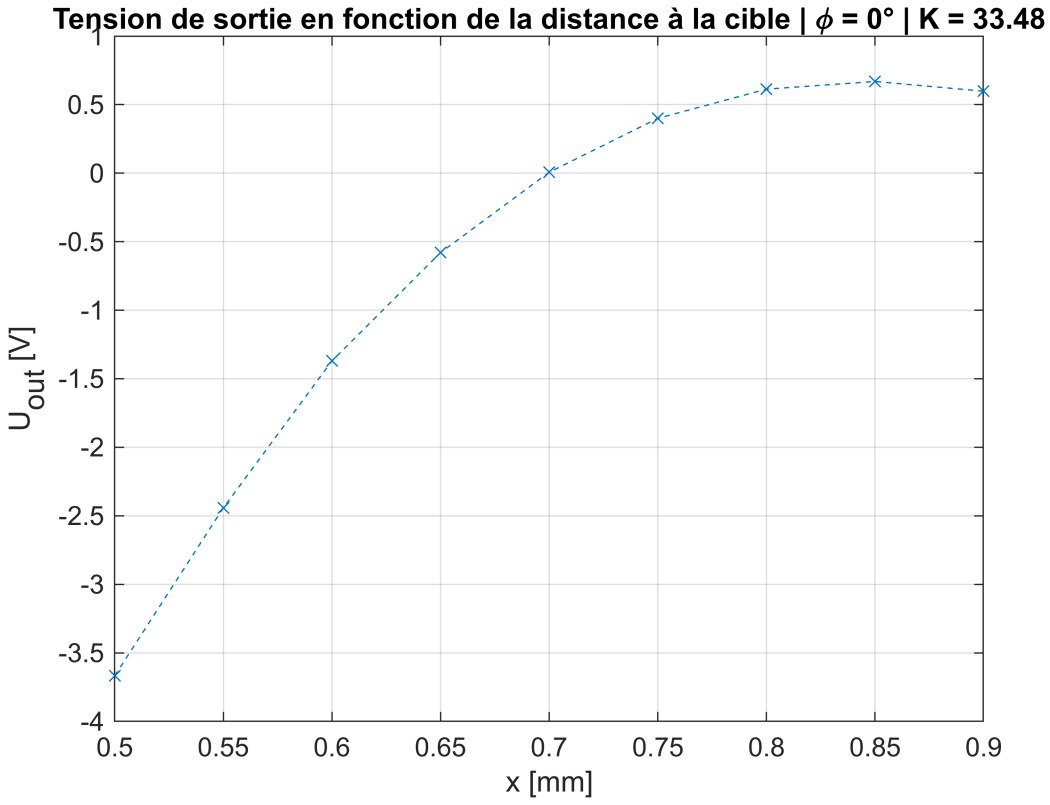
```
pente_reele*gain_reel
```

```
ans = 10.1212
```

```
u_out = [-3.667 -2.442 -1.370 -0.579 0.007 0.400 0.613 0.669 0.598];

Pu_out = polyfit(x,u_out,1);
u_out_reg = polyval(Pu_out,x);

ELu_out = 100*max(abs(u_out-u_out_reg))./(max(u_out)-min(u_out))
```

```
ELu_out = 21.6000
```

```
plot(x, u_out,"x--")
title("Tension de sortie en fonction de la distance à la cible | \phi = 0° | K = 33.48")
grid
xlabel ("x [mm]")
ylabel ( "U_{out} [V]")
```




```
x_half = 0.5:0.1:0.9;
u_out = [-0.267 -0.387 -0.487 -0.572 -0.640];
P_u_out = polyfit(x_half,u_out,1)
```

```
P_u_out = 1×2
   -0.9310    0.1811
```

```
u_out_r = polyval(P_u_out,x_half)
```

```
u_out_r = 1×5
   -0.2844   -0.3775   -0.4706   -0.5637   -0.6568
```

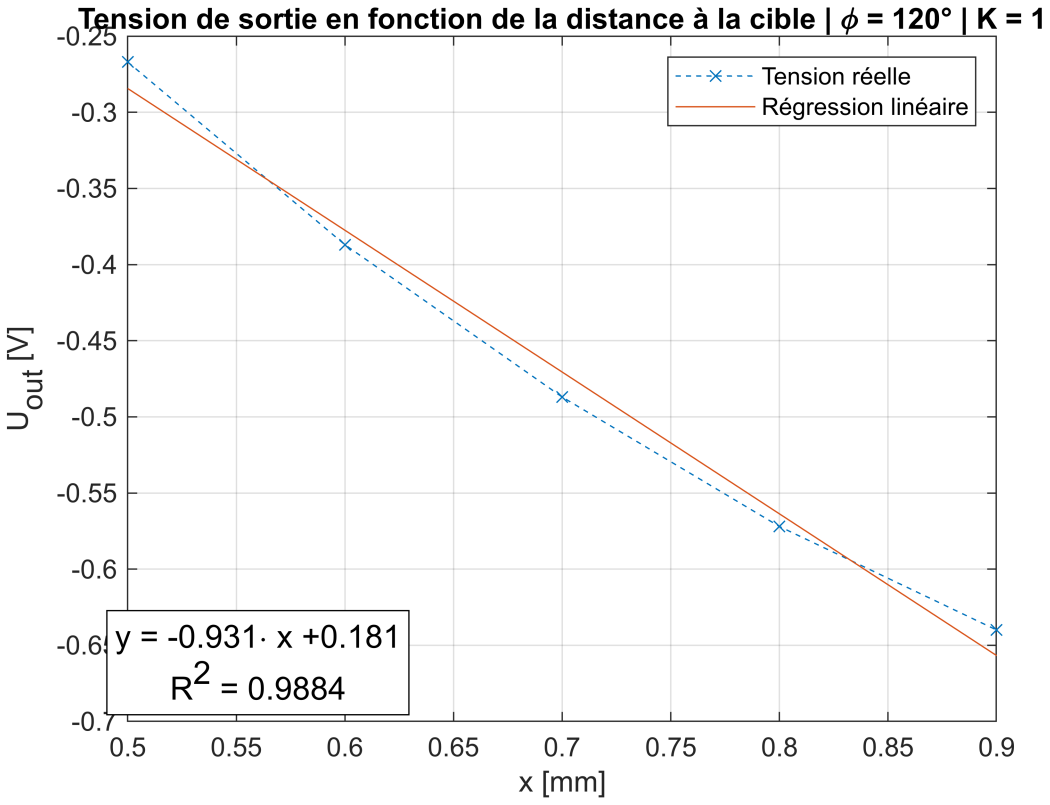
```
rr = fitlm(x_half, u_out);
RR2 = rr.Rsquared.Ordinary
```

```
RR2 = 0.9884
```

```
plot(x_half, u_out,"x--", x_half, u_out_r)
title("Tension de sortie en fonction de la distance à la cible | \phi = 120° | K = 1")
grid
legend("Tension réelle","Régression linéaire")
eqn5 = sprintf("y = %.3f\\cdot x %+.3f\nR^{2} = %.4f", P_u_out(1), P_u_out(2), RR2)
```

```
eqn5 =
    "y = -0.931\cdot x +0.181
     R^{2} = 0.9884"
```

```
text(0.15, 0.15, eqn5, ...
    'Units', 'normalized', ... % Utiliser les coordonnées normalisées
    'HorizontalAlignment', 'center', ... % Alignement horizontal
    'VerticalAlignment', 'top', ... % Alignement vertical
    'FontSize', 12, ... % Taille de la police
    'BackgroundColor', 'w', ... % Couleur de fond (optionnel)
    'EdgeColor', 'k'); % Bordure autour du texte (optionnel)
xlabel ("x [mm]")
ylabel ( "U_{out} [V]")
```




```
d300 = [-2.179 -1.597 -1.054 -0.506 0.042 0.524 0.999 1.413 1.788];
```

```
pl300 = polyfit(x,d300,1);
pl300_reg = polyval(pl300,x);

rr = fitlm(x, d300);
RR2 = rr.Rsquared.Ordinary;

EL300 = 100*max(abs(d300-pl300_reg))./(max(d300)-min(d300));
pl300_minus = pl300(1)*x + pl300(2)*(1-EL300/200)
```

```
pl300_minus = 1×9
   -1.9311   -1.4305   -0.9299   -0.4294    0.0712    0.5718    1.0723    1.5729 ···
```

```
pl300_plus = pl300(1)*x + pl300(2)*(1+EL300/200)
```

```
pl300_plus = 1×9
   -2.2001   -1.6996   -1.1990   -0.6984   -0.1979    0.3027    0.8033    1.3038 ···
```

```
plot(x, d300,"x--", x, pl300_reg, x, pl300_minus, "k--",x, pl300_plus, "k--")
title("Tension de sortie en fonction de la distance à la cible | \phi = 300° | K = 10")
grid
legend("Tension réelle","Régression linéaire",'Location', 'northwest')
eqn5 = sprintf("y = %.3f\\cdot x %+.3f\nR^{2} = %.4f\nNL = %.3f%%", pl300(1), pl300(2), RR2, EL
text(0.85, 0.15, eqn5, ...
    'Units', 'normalized', ... % Utiliser les coordonnées normalisées
    'HorizontalAlignment', 'center', ... % Alignement horizontal
    'VerticalAlignment', 'top', ... % Alignement vertical
    'FontSize', 12, ... % Taille de la police
    'BackgroundColor', 'w', ... % Couleur de fond (optionnel)
    'EdgeColor', 'k'); % Bordure autour du texte (optionnel)
xlabel ("x [mm]")
ylabel ( "U_{out} [V]")
```

Tension de sortie en fonction de la distance à la cible | ϕ = 300° | K = 10