Exercices programmation – Matlab 2 + Corr

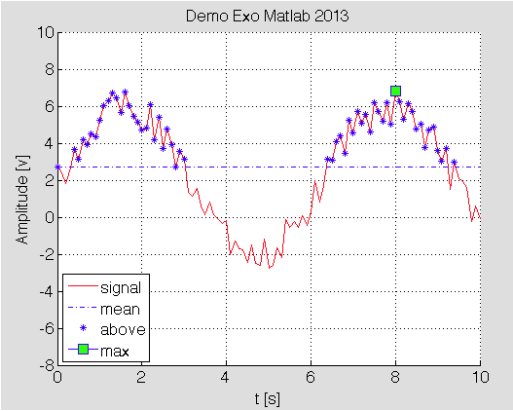
Rappel matlab

Le help contient l'entier de la documentation

Find()

X=[0.2, 0.6, -1.2, 3.1, 10.3, 100.5, 1000.8]
A l'aide de `find()` cherchez les éléments de X > 1.
Affichez uniquement ces éléments dans un graph à l'aide de `plot()`
Affichez uniquement les éléments dans un graph avec une échelle semi log en Y à l'aide de `semilogy()`

Plot()



Créez le plot ci-dessus à l'aide des commandes suivantes:
Utilisez `doc/help` pour connaître les fonctions manquantes et leurs paramètres.

```
clear all          % efface toutes les variables du workspace
close all          % ferme toutes les fenetres
figure;            % new figure

h=0.1;
X=0:h:10;          % x vector
```

Générez le vecteur Y =f(X), avec f()= *a**sin(x)+*b**rand + c; *a, b, c* à choix

```
Y=…

plot(X,Y,'r');     % plot Y vs X en rouge



hold on;           % stack plots
grid on;           % affiche la grille
xlabel('t [s]');   % affiche les labels et titre
ylabel('Amplitude [v]');
title('Demo Exo Matlab 2013')
```

Calculez la valeur moyenne de Y

```
m=…
```

Affichez la moyenne comme une droite (vecteur avec 2 pts) en pointillés bleus (`'-.b'`).

```
mY = [m m]
```

Quelles sont les valeurs de mX pour mY defini ci-dessus?

```
mX = …

plot(…'-.b')
```

Affichez un carré vert à l'emplacement (`maxX, maxY`) de la valeur maximum d' Y (à calculer)

```
maxX, maxY = …

plot(maxX, maxY, …
  'marker','square','markersize',12,'markeredgecolor','b','markerfacecolor','g')
```

Sélectionnez les valeurs plus grande que la valeur moyenne et mettez-les en évidence par une étoile bleu (`'b*'`). Utilisez `find()` pour sélectionner les indexes des valeurs puis générez les X et Y correspondants.

Changez les valeurs min et max de l'axe Y à l'aide de la function `axis()`. Comment faire pour garder les valeurs courantes de l'axe X ?

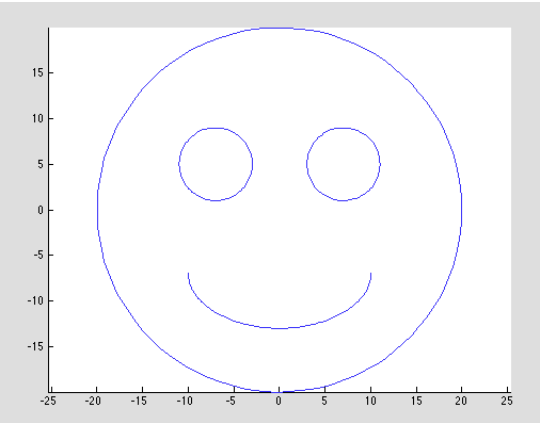
Créez une legend en bas à gauche (`'Location','SouthWest'`) à l'aide de `legend()`.

Créez un fichier pdf à partir de la figure courante

```
saveas(gca,'demoExo.pdf')
```

```
% exo 2021 plot

clear all       % efface toutes les variables du workspace
close all       % ferme toutes les fenetres

h=0.1;
X=0:h:10;     % x vector
Y=4*sin(X)+2*rand(1,length(X))+1;  % evaluate y for each x

figure;         % new figure
hold on;
grid on;
xlabel('t [s]');
ylabel('Amplitude [v]');
title('Demo Exo Matlab 2015')

ph=plot(X,Y,'r');% plot in red
m=mean(Y);      % compute the mean
plot([min(X) max(X)],[m,m], '-.b'); % plot the mean in blue dashed
line

aboveX=find(Y>m);
aboveY=Y(aboveX);

plot(h*(aboveX-1),aboveY,'*');

[mm mp]=max(Y);     % get the position and the value of the max
element
plot((mp-
1)*h,mm,'marker','square','markersize',12,'markeredgecolor','b','mar
kerfacecolor','g')


axis([min(X),max(X),-8,10]);
legend('signal','mean','above','max','Location','SouthWest') % has
to be defined after plot()

saveas(gca,'demoExo2015.pdf')
```

Function smile() - *optionnel*



Dans Matlab, écrire une fonction qui dessine un *smile* à l'aide de 3 cercles et une ellipse. Pour cela faite un script qui contient les fonctions suivantes :

Définir une fonction `mydrawCircle` () pour afficher cercle fait d'une courbe paramétrique où x=r*sin(v)+offsetX et y = r*cos(v)+offsetY avec v= -pi : 0.1*pi : pi.
center est un vecteur.

```
function mydrawCircle(radius, center)
v=…
x=…    % x=f(raduis,v,center)
y=…    % y=f(raduis,v,center)
plot(x, y);
end
```

Faire de même pour afficher une ellipse `mydrawElipse()`. La fonction a comme paramètre de retour (`boucheHandle`) un *handle* sur le plot qui nous permettra de changer les valeurs du plot par la suite. Avec x=r1*sin(v)+offsetX et y = r2*cos(v)+offsetY , avec v= -pi : 0.1*pi : 0

```
function [boucheHandle]=mydrawElipse(radius1,radius2, center)
…
end
```

Faire une procédure `drawSmiley()` qui dessine le *smile*. La taille de la tête est passée en paramètre (size). Employer les commandes

hold on pour afficher plusieurs plots sur la même figure
axis equal pour avoir de vrais cercles (échelle x = échelle y)
f=figure pour créez une nouvelle fenêtre
figure(f) pour la mettre au 1er plan

```
function [boucheHandle] = drawSmiley(size)
f=figure; % nouvelle fenetre
hold on   % stocker les graphiques
axis equal; % Sinon le cercle apparait oval

…   % dessin du smile

figure(f) % amene la fenetre au premier plan
end
```

Animation du Smile

Ecrire une fonction `doSmile()` qui permet de modifier la forme de la bouche, i.e. changer la valeur du rayon de la demi ellipse (r =0 -> ligne horizontale, r>0 ellipse)
Le paramètre de sortie *boucheHandle* retourne une référence sur le plot de la bouche du *smile*.
Pour cela accéder aux points X et Y du plot définissant la bouche, utiliser la fonction , **get(bouchehandle, 'YData', ...)** et **set(bouchehandle, 'YData', ...)** voir slide 35 du cours.
Les X de l'ellipse ne change pas, recalculer les Y en fonction du nouveau rayon, ne pas oublier d'ajouter l'offset en Y qui se calcul en prenant le 1er point en Y. Le vecteur d'angles *v* doit être de la même taille que **length(Yoffset).**

```
function doSmile(boucheHandle, radius2)
Yoffset=get(boucheHandle,'YData');  % position du 1er pt, defini l'offset en Y
v = …
y = …  % y=f(raduis2,v,Yoffset)

set(boucheHandle,'YData',y);
end
```

Faite sourire votre *smile* en modifiant l'ouverture de la bouche via `doSmile()`.

```
function MySmiley
boucheHandle=drawSmiley(20);
 for n=-1:6
    doSmile(boucheHandle,n);  % anime la bouche
    pause(0.1);
 end
end
```

```
% la fonction MySmiley contient l'equivalent des commandes qui devrait
% etre entrees dans la fenetre de commande

function MySmiley
bouche=drawSmiley(20);
 for n=-1:6
    doSmile(bouche,n);  % anime la bouche
    pause(0.1);
 end
end

function [bouche] =   drawSmiley(size)
f=figure; % nouvelle fenetre
hold on   % stocker les graphiques
axis equal; % Sinon le cercle apparait oval.
mydrawCircle(size,[0 0]) %tete
mydrawCircle(size/5,[-7 5]) % oeil gauche
mydrawCircle(size/5,[7 5])  % oeil droite
bouche=mydrawElipse(size/2,-1,[0 -7]) % bouche initiale
figure(f) % amene la fenetre au premier plan
end

function doSmile(f, radius2)
l=length(get(f,'XData'));
angle = linspace(-pi, 0, l); % Vecteur lignes avec les angles.
Yoffset=get(f,'YData');  % position du 1er pt, d?fini l'offset en Y
y = radius2 * sin(angle) + Yoffset(1);
set(f,'YData',y);
end

function mydrawCircle(radius, center)
nPoints = 100; % Nombre de points composant le cercle.
angle = linspace(0, 2*pi, nPoints); % Vecteur lignes avec les angles.
x = radius * cos(angle) + center(1);
y = radius * sin(angle) + center(2);
plot(x, y);
end

function [p]=mydrawElipse(radius1,radius2, center)
nPoints = 100; % Nombre de points composant le cercle.
angle = linspace(-pi, 0, nPoints); % Vecteur lignes avec les angles.
x = radius1 * cos(angle) + center(1);
y = radius2 * sin(angle) + center(2);
p=plot(x, y);
end
```