Université du Havre
UFR des Sciences et Techniques

Master EEEA - M2 SFSI

TP Diagnostic, approche traitement du signal.

Un rapport devra être rendu en fin de séance. Les salles de TP sont équipées du logiciel Open Office qui pourra être utilisé pour la rédaction du rapport.

1. Filtrage

1.1 Filtrage par fenêtre glissante

Théorie

- Expliquer la construction de ce filtrage et donner l'algorithme de la fonction correspondante. On doit pouvoir régler la taille de la fenêtre.
La fenêtre peut être appliquée tous les P points. Discuter de l'influence de P, on choisira P = 1 (expliquer l'intérêt).

Pratique

- Tester votre fonction sur le signal **sig1** quels sont les paramètres retenus ?

1.2 Carte EWMA

Théorie

- Rappeler la méthode de construction de la carte de contrôle à moyenne mobile EWMA. Quel est son rôle.

On choisira comme estimateur de la variance de la série, la valeur de la variance des échantillons observés pondérée par la valeur de $\frac{\lambda}{2-\lambda}$ où λ est le paramètre lié au facteur d'oubli compris entre 0.2 et 0.5 en général.

On initialisera la moyenne mobile à partir des premières valeurs de la série de mesures, expliquer pour qu'elle raison l'initialisation avec la première valeur uniquement pose t-elle problème ?

Les limites haute et basse de la carte sont données sous la forme :

$$Lim = M_0 \pm k.\sigma$$

- Expliquer les différents termes et donner l'influence de k.

- Commenter le code de la fonction Matlab correspondante **fct_ewma_TP.m** (voir fichier)

Pratique

- Tester avec le signal S suivant :

```
s =  [rand(1,100)*0.2+3 rand(1,100)*0.4+4];
```

Afin de déterminer le rôle des différents paramètres :
- modifier la fonction de telle sorte à pouvoir faire varier la largeur de la fenêtre.
- modifier la fonction de telle sorte à pouvoir faire varier les seuils de détection, quel paramètre doit on ajouter à la fonction ? justifier.

- Tester votre carte avec les données enregistrées dans le fichier ***mesure.mat*** correspondant à l'enregistrement de la sortie d'un système commutant entre trois type de comportements S1, S2 et S3 au cours du temps ainsi que sur celles du fichier ***Donnees.mat***. On souhaite déterminer les instants de commutation.

- Adapter la largeur de la fenêtre pour faire apparaitre les 3 comportements.

- Adapter la valeur du paramètre lambda pour permettre de détecter les 2 changements de comportements des différents signaux.

- Tester sur les quatre exemples pour déterminer les dates des changements de comportements.

- Quelles sont les influences sur la détection et sur le retard à la détection de :
  - la largeur de la fenêtre
  - la valeur de lambda

2. ACP

Le but est de construire un outil permettant de détecter les différents chiffres. On s'appuiera sur les signatures des signaux correspondant aux chiffres, l'ACP permettra de projeter ces signatures dans un espace réduit afin de caractériser un signal sain du chiffre. Les autres signatures seront considérées comme des défauts par rapport à ce chiffre. Un calcul de distance permettra de déterminer la classe d'appartenance de la signature inconnue.

Différentes fonctions Matlab sont fournies :

src_test_TP.m	permet de simuler un exemple de fonctionnement
fct_signature	transforme le vecteur de coordonnées en signature
fct_base_TP	
fct_deter_base_TP	permet de déterminer une base caractéristique
fct_err_TP	
fct_deter_err_TP	calcule les distances entre la signature inconnue et les classes
TracerSurFenetre	détermine le vecteur de coordonnées d'un chiffre tracé à l'aide de la souris.

- Rappeler succinctement le principe de l'ACP,

- Tester la fonction TracerSurFenetre
- Déterminer les différentes variables concernées ainsi que leur type

- Ecrire l'algorithme général (en utilisant les fonctions de bases fournies, sans entrer dans le détail) permettant de déterminer la classe d'un chiffre écrit à la souris

- Expliquer clairement les étapes permettant d'obtenir la projection d'un signal quelconque dans l'une des bases représentant les différentes classes.
- A quoi sert cette projection ?

- Expliquer le principe de détermination de la signature à partir du tracé du chiffre
- Ecrire les instructions clefs sous forme mathématique en précisant les types des variables.

- Dans la fonction fct_deter_err_TP, montrer que l'erreur correspond aussi à l'expression suivante :

```
% Err =  ((I -  Vect*Vect')*(Phi - phi))'*((I -  Vect*Vect')*(Phi - phi))
```

Indiquer la dimension de I
Quelles sont les différences entre phi, Phi et Phir

- Compléter la base pour permettre de reconnaitre d'autres chiffres …

Complément Matlab

Il est vivement recommandé de consulter l'aide en ligne de Matlab concernant les différentes fonctions ci-dessous, la commande <**help** nom_fonction> permet d'afficher cet aide. Un aide-mémoire comportant ces fonctions peut être établi.
Sous Matlab, dans l'espace de travail (workspace) les données stockées peuvent être listées à l'aide de la commande whos.
La commande dir permet d'afficher le contenu du répertoire courant qui est celui noté dans le cartouche supérieur "current folder"
Pour ouvrir des fichier ***.mat** utiliser la fonction **load**, les différentes variables comprises dans le fichier seront chargées sous l'espace de travail

load nom_fichier.mat

Pour sauvegarder un ensemble de variables sous un fichier ***.mat** utiliser la fonction **save**

save nom_fichier(.mat) <liste des variables à sauvegarder>

Fonctions Matlab

Afin de pouvoir répéter une simulation à l'identique, on utilise la fonction **randn**, avec la propriété 'seed' :

randn ('seed', 0)

Voir la fonction **RandStream**
La fonction **find** permet de retourner les indices des coordonnées d'un vecteur vérifiant un test donné

```
[indices] = find (Vect > 1);          % retourne les indices des coordonnées de Vect supérieures à 1
```

Fonctions affichage

La fonction **subplot** permet d'afficher plusieurs graphiques sur une même fenêtre.
Voir les fonctions annexes, **Title, grid**, **Xlabel, Ylabel, Legend** qui permettent de renseigner une figure.
La fonction **pause** (m) permet d'attendre pendant m secondes, m peut être un nombre réel.
La fonction **Axes** permet de cadrer un affichage

Modification des propriétés des objets graphiques sous Matlab

Les propriétés de chaque objet affiché peuvent être consultées et modifiées. Pour cela au moment de la création d'un objet, un identificateur doit lui être attaché :
lig = **line** ([1 N , LC LC]);
Les fonction **get** et **set** permettent d'afficher et de modifier les propriétés en passant par l'identificateur de ces objets.

```
get (lig, 'Color'),          % retourne la couleur de la ligne
set (lig, 'Color', 'r');     % modifie la couleur de la ligne
get (lig)                    % retourne l'ensemble des propriétés de l'objet lig
```

Type de données

Tab[i] : élément i d'un tableau, cela peut être un autre tableau, tous les éléments i doivent être de même type
Liste{i} : élément i d'une liste, cela peut être un autre tableau, les éléments i peuvent être de type différent
Liste{i}.coord : champ « coord » de la structure correspondant à l'élément i de la liste « Liste »

```
function [Mi, Sigest] = fct_EWMA_TP (Seq, lambda)

H = 500;
Mi(1:H) = mean (Seq(1:H)); % M0 = x0
N = max(size(Seq));
Sigi = std(Seq(1:H));
Sigest = Sigi*sqrt(lambda/(2-lambda));
LS = mean (Seq(1:H)) + 3*Sigest;
LI = mean (Seq(1:H)) - 3*Sigest;
for i=H+1:N
    Mi(i) = lambda*Seq(i) + (1-lambda)*Mi(i-1);
end
fprintf ('\nEcart-type : %f\n', Sigi);
fprintf ('\nEcart-type estimé : %f\n', Sigest);
N = N-H+1;
figure (3);
hold off
plot ([1,N],[Mi(1) Mi(1)], 'b');
hold on
line ([1,N],[LS LS]);
line ([1,N],[LI LI]);
plot (Mi,'m');
```

```
% src_test_TP
% script de test

load coord_chiffre   % (1, 2, 3 et 4)
clear global coord
global coord


[Sig] = fct_signature (coord_sauv);
[Vect, val, Xmoy] = fct_base_TP (Sig);
TracerSurFenetre (2);  % ou autre chiffre, attention : exécuter cette instruction avant les 2 dernières
[sigtest] = fct_signature (coord);
[clas] = fct_err_TP (sigtest{2}, Vect, Xmoy);
```

```
function [Sig] = fct_signature (coord)

%  TracerSurFenetre.m
%  fct_deter_ind
%  fct _init_acp

%  Sig{chiffre}=[Sig{chiffre} ; signature]
% signature : tableau des 9 x 9 cases : nombre binaire

% à partir du tableau de structures créé par TracerSurFenetre.m création des individus correspondant aux différents
% chiffres
% fct_signature permet de transformer les coord en signatures sur une grille de 10 x 10 cases
% une troisieme fonction permettra de déterminer le chiffre à partir de sa signature

for i=1:9
    Sig{i} = [];
end

Nb = size(coord, 2);    % nombre de tests
fprintf ('\nSignature de %d individus\n',Nb);
for i=1:Nb
    tracer = coord{i}.coord;        % tableau des points
    minx = min(tracer(:,1));  % on convertit le tracer en grille binaire
    miny = min(tracer(:,2));
    maxx = max(tracer(:,1));
    maxy = max(tracer(:,2));
    grille = zeros(9,9);
    for k=1:max(size(tracer))  % pour tous les points
        cx = round((tracer(k,1) - minx)/(maxx - minx)*9+1);
        cy = round(10 - (tracer(k,2) - miny)/(maxy - miny)*9);  % change orientation axe
       grille (cy, cx) = 1;  % permute x et y
    end
    signature = []; % transformation de grille en vecteur
    for k=1:10  % pour toutes les lignes
        signature = [signature  grille(k,:)];
    end
    fprintf ('\nIndividus : %d\n', coord{i}.chif);
    % ajout des signatures les une sous les autres
    Sig{coord{i}.chif} = [Sig{coord{i}.chif} ; signature];
End
```

```
function [Vect, val, Xmoy] = fct_base_TP (Sig, nb)
if (~exist('nb'))
    nb = 0;
end
N = size(Sig,2);
for i=1:N
    if (~isempty(Sig{i}))
        [R, L, phi]  = fct_deter_base_TP (Sig{i}, nb);
        Vect{i} = R;
        val{i} = L;
        Xmoy{i} = phi;
    end
end
```

```
function  [clas] = fct_err_TP (TSig, Vect, Xmoy)
Class = size(Vect,2);
for i=1:Class
   if (~isempty(Vect{i}))
        Err(i) = fct_deter_err_TP (TSig, Vect{i}, Xmoy{i});
        fprintf ('\nErreur par rapport à Ind%d : %6.2f', i,Err(i));
    else
        Err(i)=inf;
    end
end
Err = round(100*Err)/100;
[m, clas] = min(Err);

poss = find(Err == m);
if (max(size(poss) > 1))
    fprintf ('\nClasse %d', poss);
else
    fprintf ('\nClasse %d', clas);
end
fprintf ('\n\n');
```

```
function [R, L, phi] = fct_deter_base_TP (X, nb)

[N,M] = size(X);
% vect moyen
phi = mean(X)';
ec_type = std(X,1)';

X = fct_centre_reduit (X, 0);  %  (Ind, 0) donnees uniquement centrées
% le centrage est obligatoire, la réduction ne l'est pas

[R, L] = eig(X'*X/N);
[L, pos] = sort(diag(L),'descend');
R = R(:,pos);

 if (nb==0)
      indic = find (L > 0.1*max(L));
 else
     indic = [1:nb];
 end
 R = R(:,indic);
 L = L(indic);
```

```
function [Err] = fct_deter_err_TP (Phi, Vect, phi)

Phi = Phi';
Om = Vect'*(Phi - phi);
Phir = Vect*Om;
Err =1/(max(size(phi)))* ((Phi - phi) - Phir)'*((Phi - phi) - Phir);
```