TP Traitement du signal

Dr. TALEB Sarra

1.0 2023/2024

Université de Tlemcen

Table des matières

Objectifs

Ce TP est destiné aux étudiants de troisième année licence en télécommunications. Les textes de travaux pratiques présentés sont constitués d'un rappel théorique suivi d'une série de manipulations numériques. Ces manipulations permettent à l'étudiant de :

- **Niveau connaissance** : Vérifier pratiquement sur machine les différentes notions théoriques présentées dans le cours "traitement du signal".
- **Niveau compréhension** : Intégrer l'étudiant avec les techniques de traitement numérique du signal.
- **Niveau d'application** : Pratiquer l'étudiant à la programmation.
- **Niveau d'application :** Appliquer des opérations mathématiques élémentaires sur les signaux temporels pour les modifier.
- **Niveau d'évaluation** : Évaluer l'effet des opérations appliquées sur les propriétés des signaux temporels.

TP 1.2 : Instructions de contrôle

Les instructions de contrôle sont des éléments essentiels de tout langage de programmation, car elles permettent de gérer le flux d'exécution d'un programme. En d'autres termes, elles permettent de déterminer l'ordre dans lequel les instructions du programme doivent être exécutées en fonction de conditions spécifiques.

MATLAB propose un ensemble d'instructions de contrôle puissantes et flexibles qui permettent de créer des programmes structurés, efficaces et lisibles.

1. Objectifs spécifiques

Ce TP a pour objectifs de :

- **Niveau connaissance** : Vérifier les concepts fondamentaux des structures de contrôle et leur rôle dans la programmation.
- **Niveau d'application** : Mettre en pratique les structures de contrôle pour gérer les différents scénarios d'un programme.
- **Niveau d'évaluation** : Évaluer les différentes structures de contrôle et sélectionner la plus adaptée au contexte du problème.

2. Control conditionnel : « if... elseif ... else ... end » et

La commande **if** permet de faire une évaluation d'une expression logique et exécute une instruction quand cette expression est vraie. Elseif et else sont utilisé lorsqu'on a plusieurs conditions et end pour terminer.

La boucle conditionnelle la plus simple a la forme suivante :

Syntaxe :

If expression logique

<séquence d'instructions>;

end

Il existe une séquence conditionnée sous forme d'alternatives :

Syntaxe :

If expression logique

<séquence d'instructions 1>;

else

<séquence d'instructions 2>;

end

Exemple : Écrivez le script suivant et exécutez le :

```
n=input('entrez une valeur de n')
if n<0
disp('n est negatif')
elseif n==0
disp('n est nul')
else
disp('n est positif')
end

```

3. Control conditionnel : « switch ... case »

La commande **switch** permet d'exécuter une instruction après la vérification de la valeur d'une expression.

La boucle switch est une alternative de l'utilisation d'une séquence d'instructions conditionnées pour effectuer un choix en cascade existe.

Syntaxe :

switch var

case cst1,

<séquence d'instructions 1>;

case cst2,

<séquence d'instructions 2>;

...

case cstN,

<séquence d'instructions N>;

otherwise

<séquence d'instructions par défaut>;

end

Exemple : Écrivez le script suivant et exécutez le :

```
n= input('choisir une valeur de n parmi : 1 2 ou 3')
switch n
case 1
x=10
case 2
x=100
case 3
x=1000
otherwise
error('la valeur de n doit être 1 2 ou 3')
end

```

Cliquez ici[1] pour voir la vidéo:

4. Control de boucle « for ... end »

La commande **for** permet d'exécuter une instruction plusieurs fois, avec un nombre déterminer de répétition.

Cette boucle consiste à effectuer une boucle pour les valeurs d'un indice, incrémenté à chaque itération, variant entre deux bornes données (borne supérieur et borne inférieur).

Syntaxe :

for indice = borne_inf : pas: borne_sup

<Séquence d'instructions>;

end

Exemple : le programme suivant calcule la somme des éléments d'un vecteur

```
r=rand(1,8)
som = 0;
for i=1:8
som = som +r(i)
end
disp(som)

```

5. Control de boucle « while ... end »

La commande **while** permet d'exécuter une instruction plusieurs fois aven un nombre indéterminé de répétition mais avec une condition d'arrêt.

Dans cette boucle les instructions sont exécutées de façon répétée tant que la condition est satisfaite.

Syntaxe :

while expression logique

<séquence d'instructions>;

end

Exemple : ce programme calcule la somme des nombres entiers inférieur ou égale à 5 ,

```
Som = 0 ;
i =1 ; 1
while i <= 5
som = som+i
i=i+1;
end
disp(som)

```

[1] https://www.youtube.com/watch?v=2nLoP2n0hrM

Cliquez ici[1] pour voir la vidéo:

6. Exercices d'application :

Exercice 1 :

Créez un programme qui donne les solution d'une équation de 2 degré :

$a\,x^2 + b\,x + c = 0$

les solutions sont :

1- Entrez les valeurs des coefficients a, b et c

2- Si la valeur a est nulle, alors la solution est x= -c/b

3- Si la valeur de a est non nulle, alors on calcule le delta ($d=b^2-4ac$)

4- Maintenant si la valeur de d est positive alors, la solution est $x=(-b\pm\sqrt{d})/2a$

5- si la valeur de d est nul, alors x=-b/2a

6- Et si la valeur de d est négative alors, l'équation n'a pas de solution.

Exercice 2 :

Ecrire un programme qui calcule la suite de fibonnaci (inferieur à 100), cette série commence par deux éléments 1 et 1, et chaque élément suivant est la somme des deux précédant éléments :

1, 1, 2, 3, 5, 8, 13, 21, 34, ...

1- Initialiser les deux premiers éléments

2- Calculer la boucle a partir de 3 ^{ème} élément

Faire ce programme par deux méthodes (en utilisant for et while)

7. Solutions :

Exercice 1 :

```
% Résolution d'une équation de second degré : ax^2 + bx + c = 0
disp('Résolution d''une équation de second degré : ax^2 + bx + c = 0');
a = input('Entrez la valeur de a : ');
b = input('Entrez la valeur de b : ');
c = input('Entrez la valeur de c : ');

if a == 0
    if b == 0
        if c == 0
            disp('L''équation a une infinité de solutions.');
        else
            disp('L''équation n''a pas de solution.');
        end
    else
        x = -c / b;
        fprintf('La solution de l''équation est x = %f\n', x);
    end
else
    delta = b^2 - 4 * a * c;
    if delta > 0
        x1 = (-b + sqrt(delta)) / (2 * a);
        x2 = (-b - sqrt(delta)) / (2 * a);
        fprintf('Les solutions de l''équation sont x1 = %f et x2 = %f\n', x1, x2);
    elseif delta == 0
```

[1] https://www.youtube.com/watch?v=zYWQy0_7opA&t=1293s

```
        x = -b / (2 * a);
        fprintf('L''équation a une solution double : x = %f\n', x);
    else
        disp('L''équation n''a pas de solution réelle.');
    end
end

```

```
Résolution d'une équation de second degré : ax^2 + bx + c = 0
Entrez la valeur de a : 15
Entrez la valeur de b : 66
Entrez la valeur de c : 47
Les solutions de l'équation sont x1 = -0.893605 et x2 = -3.506395
```

Exercice 2 :

```
% Utilisation d'une boucle for pour générer la suite de Fibonacci
n = 11; % Limite jusqu'à laquelle vous voulez calculer Fibonacci (n < 100)
fib = zeros(1, n);
fib(1) = 1;
fib(2) = 1;

for i = 3:n
    fib(i) = fib(i - 1) + fib(i - 2);
end
disp(fib(1:i));
```

```
% Utilisation d'une boucle for pour générer la suite de Fibonacci
n = 11; % Limite jusqu'à laquelle vous voulez calculer Fibonacci (n < 100)
fib = zeros(1, n);
fib(1) = 1;
fib(2) = 1;

for i = 3:n
    fib(i) = fib(i - 1) + fib(i - 2);
end
disp(fib(1:i));
```

```
1     1     2     3     5     8    13    21    34    55    89
```

Ce TP vous a permis de découvrir et d'expérimenter les principales instructions de contrôle de MATLAB, notamment les instructions conditionnelles (if et switch), les boucles (for et while),

En maîtrisant ces instructions de contrôle, vous pouvez écrire des programmes MATLAB plus structurés, plus efficaces et plus lisibles. Cela vous permettra de résoudre des problèmes plus complexes et de développer des applications puissantes.