République Populaire Démocratique Algérienne
Ministère d'enseignement supérieur et de la recherche scientifique
Université Djillali Liabes -Sidi Bel Abbes-

Faculté des sciences exactes :

Département de mathématique

Résolution du 5 -ème et 6 -ème TP du module outils de la programmation pour les mathématiques

Réaliser par :

DIB Abdelkrim Yassine Taki Eddine

Année universitaire :

2022/2023

Table des matières

Introduction

L'utilisation de Matlab est très répandue dans les domaines scientifiques et d'ingénierie. Matlab est un outil puissant qui permet d'effectuer des analyses numériques, de résoudre des problèmes d'optimisation, de tracer des graphiques et de simuler des systèmes complexes. En outre, Matlab est un langage de programmation qui permet aux utilisateurs de créer des fonctions personnalisées et de résoudre des problèmes complexes en quelques lignes de code.

Dans le cadre de ce travail, nous allons utiliser Matlab pour résoudre des problèmes mathématiques complexes. Nous allons explorer les différentes fonctions et commandes de Matlab pour résoudre des équations différentielles, des équations algébriques et des problèmes d'optimisation. Nous allons également apprendre à tracer des graphiques, à visualiser des données et à créer des fonctions personnalisées pour résoudre des problèmes spécifiques.

Dans ce travail, nous allons utiliser des exemples concrets pour illustrer l'utilisation de Matlab. Nous allons commencer par les bases de Matlab et progresser vers des exemples plus avancés. Nous allons également expliquer les différentes étapes impliquées dans la résolution de problèmes en utilisant Matlab, y compris l'importation de données, la manipulation de données, la résolution de problèmes et la visualisation des résultats.

En résumé, ce travail nous permettra d'explorer les nombreuses possibilités offertes par Matlab pour résoudre des problèmes mathématiques complexes. Nous allons découvrir comment utiliser Matlab pour résoudre des problèmes, simuler des systèmes et visualiser des données. Nous allons également apprendre à créer des fonctions personnalisées pour résoudre des problèmes spécifiques.

Idée générale sur le module

Le module "Outils de programmation pour les mathématiques" est un ensemble de fonctionnalités de programmation intégrées dans le logiciel Matlab. Ces outils permettent aux utilisateurs de créer des programmes, des scripts et des fonctions personnalisés pour résoudre des problèmes mathématiques complexes.

Les outils de programmation pour les mathématiques incluent des fonctions mathématiques de base telles que les fonctions trigonométriques, les fonctions logarithmiques, les fonctions exponentielles, les fonctions polynomiales et les fonctions de traitement de signal. Ils incluent également des fonctions de visualisation telles que les graphiques en 2D et en 3D, les diagrammes en secteurs et les diagrammes à barres. **[01]**

MATLAB

Matlab est un langage de programmation conçu pour les applications scientifiques et techniques. Il a été développé par MathWorks en 1984 et est maintenant utilisé dans de nombreux domaines tels que les mathématiques, la physique, la chimie, l'ingénierie et la finance. Il est largement utilisé pour résoudre des problèmes complexes qui nécessitent des calculs numériques et des analyses de données. **[02]**

L'un des avantages de Matlab est sa facilité d'utilisation et sa syntaxe simple et intuitive. Les utilisateurs peuvent créer des programmes, des scripts et des fonctions personnalisés pour résoudre des problèmes spécifiques, et utiliser des outils graphiques pour visualiser les résultats de leurs calculs.

Matlab offre également de nombreuses fonctionnalités avancées telles que la manipulation de matrices, la modélisation et la simulation, l'analyse statistique, la communication sans fil et le traitement du signal et de la vision par ordinateur.

En outre, Matlab dispose d'une vaste communauté de développeurs qui partagent des astuces, des codes et des solutions de problèmes. Cela en fait une ressource précieuse pour les chercheurs, les ingénieurs et les scientifiques dans une variété de domaines. **[03]**

Applications des outils de programmation en mathématiques dans les autres domaines :

Les outils de programmation en mathématiques de Matlab peuvent être appliqués dans de nombreux domaines, tels que la physique, l'ingénierie, la finance, la biologie et l'informatique. Voici quelques exemples d'applications : **[04]**

- En physique, les outils de programmation de Matlab sont utilisés pour résoudre des équations différentielles, des problèmes de mécanique quantique et des problèmes de traitement du signal.
- En ingénierie, les outils de programmation de Matlab sont utilisés pour la conception et l'analyse de systèmes de contrôle, la modélisation et la simulation de systèmes électroniques, la conception de circuits et la conception de structures.
- En finance, les outils de programmation de Matlab sont utilisés pour la modélisation des marchés financiers, l'analyse des risques et des performances des portefeuilles, et la conception de stratégies d'investissement.
- En biologie, les outils de programmation de Matlab sont utilisés pour l'analyse de données génomiques, l'analyse de signaux électrophysiologiques, la modélisation de réseaux de régulation génétique et la simulation de modèles de dynamique de population.
- En informatique, les outils de programmation de Matlab sont utilisés pour la reconnaissance d'images, l'apprentissage automatique, la modélisation de systèmes complexes et la simulation de réseaux de communication.

Fiche de TP 05

Travail en mode exécutif

Exercice 01 :

Exercice 1 :

- Écrire un script matlab qui retourne le quotient et le reste de la division euclidienne de a par b. Pour cette question, il est interdit d'utiliser la fonction mod existant dans matlab : vous n'avez le droit qu'à des soustractions, des comparaisons et l'utilisation de boucle while.

Résolution du problème :

```
a = input('Entrez la valeur de a : '); % Saisie de la valeur de a par l'utilisateur
b = input('Entrez la valeur de b : '); % Saisie de la valeur de b par l'utilisateur

% Initialisation des variables
quotient = 0;
reste = a;

% Tant que le reste est supérieur ou égal à b
while reste >= b
    reste = reste - b; % Soustraction de b au reste
    quotient = quotient + 1; % Incrémentation du quotient
end

% Affichage des résultats
fprintf('Le quotient de la division de %d par %d est %d et le reste est %d\n', a, b,
quotient, reste);
```

Exercice 02 :

Exercice 2 :

- Ecrire un script qui calcule le pgcd de deux nombres, en utilisant une boucle while.
- Ecrire un script qui calcule le ppcm de deux nombres, en utilisant une boucle for.

Résolution du problème :

```
a = input('Entrez la valeur de a : '); % Saisie de la valeur de a par l'utilisateur
b = input('Entrez la valeur de b : '); % Saisie de la valeur de b par l'utilisateur

% Initialisation des variables
pgcd = 1;
i = 1;

% Tant que i est inférieur ou égal à a et b
while (i <= a) && (i <= b)
    % Si i est un diviseur commun de a et b
    if (mod(a,i) == 0) && (mod(b,i) == 0)
        pgcd = i; % Mise à jour du PGCD
    end
    i = i + 1; % Incrémentation de i
end

% Affichage du PGCD
fprintf('Le PGCD de %d et %d est %d\n', a, b, pgcd);

a = input('Entrez la valeur de a : '); % Saisie de la valeur de a par l'utilisateur
b = input('Entrez la valeur de b : '); % Saisie de la valeur de b par l'utilisateur

% Recherche du plus grand des deux nombres
if a > b
    max_num = a;
else
    max_num = b;
end

% Boucle for pour tester tous les multiples du plus grand nombre
```

```
for i = max_num : a * b
    % Si i est un multiple commun de a et b
    if (mod(i,a) == 0) && (mod(i,b) == 0)
        ppcm = i; % Mise à jour du PPCM
        break; % Sortie de la boucle for dès que le PPCM est trouvé
    end
end

% Affichage du PPCM
fprintf('Le PPCM de %d et %d est %d\n', a, b, ppcm);
```

Exercice 03 :

Exercice 3 :

La formule de conversion des températures exprimées en degré Celsius en degré Fahrenheit est :

$$C = 5/9\ F - 32$$

- Ecrire un script permettant de calculer une liste d'équivalence pour des températures comprises entre 0°F et 300°F avec un incrément de 10°F (sous la forme d'un tableau à 2 colonnes : la première colonne donnera les degrés Celsius, la seconde les degrés Fahrenheit). en utilisant une boucle for.

Résolution du problème :

```
borne_inf = input('Entrez la borne inférieure en degrés Fahrenheit : ');
borne_sup = input('Entrez la borne supérieure en degrés Fahrenheit : ');
increment = input('Entrez l''incrément en degrés Fahrenheit : ');

% Calcul du nombre d'itérations nécessaires
nb_iterations = (borne_sup - borne_inf) / increment + 1;

% Initialisation du tableau de résultats
equivalence = zeros(nb_iterations, 2);

% Boucle for pour calculer les équivalences
for i = 1:nb_iterations
    fahrenheit = borne_inf + (i - 1) * increment; % Conversion en degrés Fahrenheit
    celsius = (fahrenheit - 32) * 5 / 9; % Conversion en degrés Celsius
    equivalence(i, 1) = celsius; % Enregistrement de la valeur de Celsius
    equivalence(i, 2) = fahrenheit; % Enregistrement de la valeur de Fahrenheit
end

% Affichage du tableau de résultats
fprintf('Tableau d''équivalence Celsius-Fahrenheit pour des températures entre %d°F et %d°F avec un incrément de %d°F\n', borne_inf, borne_sup, increment);
disp('---------------------------------------');
disp('  Celsius    Fahrenheit');
disp('-----------------------');
disp(equivalence);
```

Exercice 04 :

Exercice 4 :

- A l'aide de trois boucles for imbriquées, écrire un script réalisant la multiplication de 2 matrices de taille 5x5.
- Comparez son temps d'exécution (commandes tic et toc) par rapport à l'opérateur * de MATLAB

Résolution du problème :

```
% Initialisation des matrices à multiplier
A = randi([1, 10], 5, 5);
B = randi([1, 10], 5, 5);

% Initialisation de la matrice de résultat
C = zeros(5, 5);

% Boucles for pour la multiplication
tic % Mesure du temps d'exécution
for i = 1:5 % Pour chaque ligne de A
    for j = 1:5 % Pour chaque colonne de B
        for k = 1:5 % Pour chaque colonne de A et chaque ligne de B
            C(i,j) = C(i,j) + A(i,k) * B(k,j);
        end
    end
end
temps_execution = toc; % Mesure du temps d'exécution

% Affichage du temps d'exécution
fprintf('Temps d''exécution de la multiplication des matrices avec 3 boucles for imbriquées : %.4f secondes\n', temps_execution);

% Comparaison avec l'opérateur *
tic % Mesure du temps d'exécution
C_matlab = A * B; % Multiplication avec l'opérateur *
temps_execution_matlab = toc; % Mesure du temps d'exécution

% Affichage du temps d'exécution avec l'opérateur *
fprintf('Temps d''exécution de la multiplication des matrices avec l''opérateur * : %.4f secondes\n', temps_execution_matlab);
```

```
% Comparaison des résultats
disp('Vérification que les résultats sont identiques :');
if isequal(C, C_matlab)
    disp('Les résultats sont identiques.');
else
    disp('Les résultats ne sont pas identiques.');
end
```

Fiche de TP 06

Travail en mode exécutif

Exercice 01 :

Exercice 1 :

- Écrire un M_file pour la fonction : $f_1(x) = \frac{x^5-3}{\sqrt{x^2+1}}$
- Tester la fonction sur quelques valeurs, par exemple,
- $f_1(1) = -1.41, f_1(0) = -3.$
- Créer un tableau x d'abscisses de -5 à 5 par pas de 0.1
- Représenter la fonction f_1 aux points x_i , avec la commande plot(x,f1(x))

Résolution du problème :

```
% Définition de la fonction
function y = myFunction(x)
    y = ((x^5) - 3) / sqrt((x^2) + 1);
end

% Tester la fonction sur quelques valeurs
disp("f(1) = " + myFunction(1));
disp("f(0) = " + myFunction(0));
disp("f(-1) = " + myFunction(-1));

% Créer un tableau d'abscisses de -5 à 5 par pas de 0.1
x = -5:0.1:5;

% Tracer la représentation graphique de la fonction f aux points xi
plot(x, myFunction(x));
title("Représentation graphique de f(x)");
xlabel("x");
ylabel("f(x)");
```

Exercice 02 :

Exercice 2:

- Ecrire une fonction *factoriel* qui calcule itérativement $n!$.
- Ecrire une seconde *factoriel_rec* qui calcule de manière récursive $n!$.
- Comparer les temps d'exécution des deux méthodes.

Résolution du problème :

```
% Définition de la fonction factoriel itérative

function res = factoriel(n)
   res = 1;
   for i = 2:n
      res = res * i;
   end
end

% Définition de la fonction factoriel récursive
function res = factoriel_rec(n)
   if n == 0
      res = 1;
   else
      res = n * factoriel_rec(n - 1);
   end
end

% Comparaison des temps d'exécution
n = 15;
tic;
factoriel(n);
temps_iteratif = toc;
tic;
factoriel_rec(n);
temps_recursif = toc;
disp("Temps d'exécution itératif : " + temps_iteratif);
disp("Temps d'exécution récursif : " + temps_recursif);
```

Exercice 03 :

Exercice 3 :

La suite de Fibonacci est une suite d'entiers dans laquelle chaque terme est la somme des deux termes qui le précèdent. Elle est définie comme suit :

$$f(2) = f(1) = 1$$
$$f(n+2) = f(n+1) + f(n); \quad \forall n \in \mathbb{N}$$

- Ecrire une fonction fibonacci_rec qui calcule de manière récursive les n premiers termes de la suite.

Résolution du problème :

```
function fib = fibonacci_rec(n)
   if n == 1 || n == 2
      fib = 1;
   else
      fib = fibonacci_rec(n-1) + fibonacci_rec(n-2);
   end
end
```

Exercice 04 :

Exercice 4 :

- Ecrire une fonction **récursive** qui inverse une chaine donnée.
- Exemple :
  Chaine introduite : Ch= 'ruojnoB'
  Chaine affiché : Ch='Bonjour'

Indication : Utiliser les commandes `length, if … else … end` dans une fonction utilisateur : `inverse(Chaine)`

Résolution du problème :

```
function str_inv = inverse(chaine)
   len = length(chaine);
   if len <= 1
      str_inv = chaine;
   else
      str_inv = [chaine(len), inverse(chaine(1:len-1))];
   end
end
```

Bibliographie

[01] https://www.mathworks.com/products/matlab/programming-math.html

[02] https://www.mathworks.com/products/matlab.html

[03] https://en.wikipedia.org/wiki/MATLAB

[04] https://www.mathworks.com/solutions/mathematics.html