Matlab Homework

Contents

1 Intérêt financier et de bonheur

Dans cet exercice, je vais te demander de m'écrire un programme permettant de trouver l'argent et le bonheur gagnés en investissant dans les banques UBS, BAS et BCV.

Imaginons qu'un personnage investit X CHF un jour (jour 0) et qu'il retire son argent un jour J, quelle sera son bonheur et sa fortune?

Les intérêts financiers ***annuels*** des banques susmentionnées sont de 3.5%, 2% et -0.5%.
Toutefois, le bonheur compte également et celui-ci, démarrant à 50%, varie annuellement de -2%, -0.5% et 4% pour chacune des banques (UBS, BCV & BAS).

Petite précision: Les banques versent les intérêts financiers et de bonheur chaque matin à l'investisseur.

2 Approximation de la racine carrée d'un nombre

Dans cet exercice, je vais te demander d'analyser l'erreur d'une méthode d'approximation du carré.
L'approximation consiste à simplifier le carré d'un nombre selon la méthode suivante:
- Trouver le carré supérieur et inférieur au nombre
- Calculer la différence et faire une interpolation linéaire

Par exemple:
$\sqrt{153}$?
$\Rightarrow 12^2 = 144,\ 13^2 = 169$
$\Rightarrow \sqrt{153} \approx 12 + \frac{153-144}{169-144} = 12.36$ En réalité, le résultat est égal à 12.3693, soit une erreur d'environ 0.075%.

Je désire que tu m'écrives un petit programme qui permet de trouver de cette manière tous les racines carrées de 1 à 100'000.
De plus, je veux que tu me fasses un plot de ton erreur en fonction du nombre (en échelle logarithmique, see help plot / loglog / semilogy).

Finalement, à l'aide du "tic toc" de Matlab, trouve le temps requis pour:
- Calculer 10'000 racines carrées à l'aide de cette approximation et une boucle for
- Calculer 10'000 racines carrées à l'aide de sqrt() et une boucle for
- Calculer 10'000 racines carrées à l'aide de cette approximation en calculant pour tout le vecteur à la fois (sans boucle for)
- Calculer 10'000 racines carrées à l'aide de sqrt() en calculant pour tout le vecteur à la fois (sans boucle for).

3 Nombres premiers

3.1 Introduction

Dans cet exercice, je vais te demander de m'écrire un programme permettant de trouver tous les nombres premiers. Deux alternatives sont possibles:

1. Trouver tous les nombres premiers entre 0 et ∞ un à un et les afficher dans la command window. Arrêt du programme au bout d'un certain nombre d'itérations entre chaque nombre premier trouvé ou au bout d'un certain nombre d'opérations au total (up to you)

2. Trouver tous les nombres premiers entre 0 et N et les afficher soit un à un dans la command window, soit tous dans un vecteur à la fin, soit en affichant un à un et enregistrant ceux-ci dans un vecteur.

3.2 Rappel sur les nombres premiers

Pour rappel, un nombre premier ne se divise uniquement par lui-même et par 1.

3.3 Optimisation

Proposer deux manières d'optimiser la recherche "brutale", qui fait un filtre de tous les nombres. Comparer les méthodes avec tic et toc (Recall: tic at the beginning of program, toc at the end of program)

Hint:
La première optimisation permet de diminuer le temps de calcul en cas d'erreur.
La deuxième optimisation permet d'éviter des calculs inutiles si le résultat est juste.

3.4 Opération inverse

Maintenant écrire un programme qui vérifie si un nombre est premier ou non. De plus, s'il n'est pas premier, trouver les multiples de ce nombre.

3.5 Transformation en fonction

Transformer ces deux petits programmes en deux fonctions (find_NbrP et find_MultP) tel que:

```
%% Call your functions in the "main" program

Prime_list=find_NbrP(10000) % Finds prime numbers up the number 100'000 the 100'000 first
    prime numbers (details written in the homemade help of the function)
[P237, Multiple237]=find_MultP(237) % Says if the 237 is prime or not and finds all the
    multiple of the number if it is not prime
[P57]=find_MultP(57) % Only says if the 57 is prime or not and finds all the multiple of the
    number if it is not prime

%% Functions (Put these below your main program or in another .m file)

function Nbrs_prime = find_NbrP (N) % Where N is the max prime number to search for or the max
     number of iterations
% Function help: Usually describe quickly function goal, inputs and outputs

%% Put your code here with N as an input and the vector Nbrsprime as an output

```

```
14 end
15
16 function [Prime, Mult_prime] = find_MultP (Nbr) % Where NbrP is the prime number from which we
       want to find the multiples.
17 % Function help: Usually describe quickly function goal, inputs and outputs.
18
19 %% Put your code here with Nbr as an input and the vector Multprime as an output (multiples of
       the number) and the boolean Prime that tells if the number is a prime or not.
20
21 end
```

4 Multiplication & Addition

4.1 Introduction

Dans cet exercice, je vais te demander de m'écrire un programme permettant de multiplier et diviser uniquement en utilisant l'addition et la soustraction. Important de faire un petit peu de gestion d'erreur (dans un 2ème temps)!

4.2 Structure requise

Voici ci-dessous un canvas de la structure requise pour ce programme (Utilisation de fonctions)

```
% To check if it works:
a1=8;
a2=2;
sol1=div(a1,a2) % Change test values to check other scenarios
sol2=mult(a1,a2)

% Test values for robustness:
% Uncomment the call to the following function when you want to check the robustness of the program, only once it works (gestion d'erreurs)

% Err = Errorchk % 0 if good, 1 if div is not robust enough, 2 if mult is not robust enough and 3 if mult & div are not robust enough

function [div_result] = div (a,b)
% function div: Divides a and b such as div_result=a/b without using the division symbol
% Inputs: scalar (double) a and b
% Output: scalar result div_result

%% Add here the program for your dividing function

end

function [mult_result] = mult (a,b)
% function mult: Multiplication of a and b such as mult_result=a*b without using the multiplication symbol
% Inputs: scalar (double) a and b
% Output: scalar result mult_result

%% Add here the program for your multiplication function

end

function [error, a, b] = Errorchk
% function Errorchk : Checks robustness of code, ouput is defined as 0 if good, 1 if div is not robust enough (checked 1st), 2 if mult is not symetric, 3 if mult is not robust enough
% Also outputs a and b which were the values that broke the program
A1=[5, -1, 0, 100, 1, 2, -2, 0.1];
A2=[8, 10^16, 10^-309, 0, 10^308 -1, 10^-15, -10^-16];
error=0;
for i=1:length(A1)
    for j=1:length(A2)
        sol_1=div(A1(i),A2(j));
        sol_2=div(A2(i),A1(j));
        sol_d1=A1(i)/A2(j);
        sol_d2=A2(i)/A1(j);
        sol_3=mult(A1(i),A2(j));
        sol_4=mult(A2(i),A1(j));
        sol_m=A1(i)*A2(j);

        if sol_1~=sol_d1
            a=A1(i);
            b=A2(j);
            error=1;
            return
        elseif sol_2~=sol_d2
            a=A2(j);
```

```
53              b=A1(i);
54              error=1;
55              return
56          end
57
58          if sol_3~=sol_4
59              a=A1(i);
60              b=A2(j);
61              error=2;
62              return
63          elseif sol_3~=sol_m
64              a=A1(j);
65              b=A2(j);
66              error=3;
67              return
68          end
69      end
70  end
71  end
```

4.3 Gestion d'erreur, que faire?

Est-ce que le programme réagit comme il faut lorsque les entrées sont "étranges"?
Par exemple, que se passe-t-il avec le programme lorsque le nombre est égal à 0, ∞, ou lorsqu'il n'est pas entier? Que se passe-t-il lorsque le nombre est négatif?

(Bonus questions): Nombre complexe? Vecteurs ou matrices de même dimension à la place de scalaires? Vecteurs ou matrices de dimensions différentes?

En résumé, il faut s'assurer que dans la plupart des cas (plausibles), le programme soit prêt à retourner une réponse sensée. Cela peut être une réponse adaptée (i.e. dans le cas des vecteurs ou matrices de taille différente => output error), le programme peut alternativement être capable de répondre à la demande selon la logique, par exemple en sortant NaN (Not a number) lorsqu'il y a quelque chose qu'il ne sait pas faire.

5 Algorithmes de tri

5.1 Introduction

Dans cet exercice, je vais te demander de m'écrire un algorithme permettant de trier une liste de $N \times M$ éléments[1] de la manière la plus efficace possible, ainsi que de comparer l'efficacité de ton algorithme.
Pour écrire cet algorithme, interdiction d'utiliser la fonction intégrée dans Matlab "find", "sort" ou autre fonction similaire.
De plus, ton algorithme doit retourner la taille de la liste et doit être trié en fonction de la colonne m de la liste.

5.2 Liste non triée

Voici une liste de colis en fonction de leur ID, poids et dimension. Tout d'abord, je vais te demander de rajouter une composante du volume et densité du colis à la liste puis de trier selon ces deux derniers (m=6 et m=7 respectivement). N'oublie pas de convertir les valeurs selon les SI units!

ID #	Weight [kg]	Height [cm]	Width [cm]	Length [cm]
1	10.6	53	149	19
2	10.1	60	35	58
3	27.0	194	143	186
4	20.2	197	163	61
5	11.5	60	167	45
6	4.9	16	24	196
7	2.9	11	153	35
8	3.4	18	16	13
9	7.7	42	137	60
10	4.8	46	146	56
11	6.3	20	40	30
12	3.5	10	18	12

5.3 Comparaison de l'efficacité de l'algorithme

***Attention*:**
Section à ne pas lire avant d'avoir effectué ton algorithme!
Ceci est dû au fait que tu as la possibilité de faire un des algorithmes présentés ci-dessous.

[1] N pour nombre de lignes et M pour le nombre de colonnes.

Remplir le tableau ci-dessous en comparant l'algorithme effectué ci-dessus avec la fonction "sort" et deux autres algorithmes ("BogoSort" et "Bubble sort"). Le worst-case scenario est très utilisé (en plus du average case scenario) en informatique car il permet de comparer les temps maximaux de deux algorithmes avec le nombre d'opérations effectuées par chacun au maximum, basé sur un certain nombre d'inputs N.

Pour utiliser la fonction "sortrows", taper 'help sortrows' dans la command window, 'doc sortrows' ou doc 'sort pour voir la différence entre ces deux derniers. Concernant l'algorithme "BogoSort", celui-ci trie les éléments de manière aléatoire puis vérifie si c'est trié ou non.
L'algorithme "BubbleSort" fonctionne comme suit:
- Comparaison et tri des deux premiers éléments de la liste.
- Comparaison et tri des $2^{ème}$ et $3^{ème}$ éléments de la liste.
- etc. jusqu'au dernier ($N^{ème}$ élément). Celui-ci est alors le plus grand de la liste.
- Répéter l'action pour les N-1 éléments restants.

Algorithme utilisé:	"Maison"	"BubbleSort"	"BogoSort"	Matlab "sortrows"
Comparaison des algorithmes selon la colonne "poids"				
Temps de calcul (with "tic toc")				
Nombre d'opérations effectuées (prédiction à la main)				-
Nombre d'opérations effectuées (confirmation via Matlab)				
Comparaison des algorithmes selon la colonne "densité"				
Temps de calcul (with "tic toc")				
Nombre d'opérations effectuées (calcul avec Matlab)				
Comparaison des algorithmes selon le "worst-case" scenario (i.e. tout est dans le désordre)				
Nombre d'opérations effectuées (pour 4 éléments)				
Nombre d'opérations effectuées (pour N éléments)				
Nombre d'opérations effectuées (pour 100'000 éléments)				
Estimer la durée du tri (pour 100'000 éléments)				

Voici quelques liens expliquant en détail ces algorithmes:
- *Overview de tous les algorithmes*
- *Algorithme QuickSort, utilisé dans la fonction "sort" de Matlab*
- *Algorithme BubbleSort, Sorting*
- *Algorithme BubbleSort, Explanation*

5.4 Bonus: Utilisation du tri pour un problème d'optimisation

Une fois la liste triée, optimiser la livraison des colis (Problème à la main et Matlab).

La poste a à disposition deux petites camionnette et quatre vélos-cargo électriques, qui peuvent transporter respectivement 80 kg et 20 kg. Tout doit être livré entre 8h et 9h30.
Le volume disponible dans la petite camionnette est de 6 m^3 et dans les vélos cargo de 2 m^3.
Une tournée dure 1h pour la camionnette et 1h30 à vélo (quelque soit le nombre de colis) et le salaire-horaire d'un facteur est de 30 CHF. Les coûts d'entretien de la camionnette sont égaux à $Coûts = 5 + \frac{Poids}{5}$ et les coûts d'entretien des vélos sont égaux à $Coûts = 0.5 + \frac{Poids}{20}$.
Trouver la configuration la moins chère pour la poste.

Indices:

- Calculer le poids total et volume total requis.
- Définir les différentes options d'engins utilisables.
- Comparer les coûts par quantité de masse transportée lors d'une tournée. (use plot)
- Choisir les engins pour la tournée en fonction de la masse totale à transporter.
- Alternativement, utiliser la force brute en vérifiant chaque configuration possible.

6 Variation de gravité

6.1 Introduction

Comme tu le sais probablement, la force gravitationnelle définie par Newton est comme suit:

$$F_{12} = G \cdot \frac{m1 \cdot m2}{r^2} \tag{1}$$

où G est la constante universelle de gravité, tel que $G = 6.674 \cdot 10^{-11} \frac{m^3}{kg \cdot s^2}$. la distance r correspond à la distance entre deux corps, soit la distance entre le centre de gravité de deux corps.
Dans cet exercice, je vais te demander de jouer avec cet équation pour différentes planètes de différentes manières.

6.2 But, Outputs désirés

Voici les tâches qui sont requises par le programme:

1. Retrouver la constante de gravité exacte de la terre g pour un rayon de la terre moyen.
2. Étudier les variations de la constante de gravité de la terre:
   - Sachant que la constante de gravité g varie environ entre 9.7639 et 9.8337 $\frac{m}{s^2}$, quelle est la hauteur par rapport au rayon de la terre moyen pour ces deux constantes de gravité?
   - Est-ce logique?
   - Si la planète était ronde, quelle serait la gravité au sommet de l'Everest?
   - Comment expliquer ces résultats?
   - Où se situent donc probablement les points de gravité maximale et minimale sur la terre?
3. Calculer les constantes de gravité des corps célestes suivants (En assumant qu'ils sont tous parfaitement sphériques): Soleil, Mercure, Venus, Mars, Jupiter, Lune, Pluton.
4. (Hard) Le plus gros astéroïde ayant jamais impacté la terre a créé le cratère de Vredefort. Le rayon et la densité de cet astéroïde avant son impact sont mal connus.
   - Prendre 1'000 valeurs de densité possibles pour l'astéroïde ainsi que 1'000 valeurs possibles de rayon
   - Faire un graphe (doc surf) de la valeur de g pour chaque combinaison de rayon-densité
   - Trouver la valeur de la constante de gravité maximale et minimale possible sur cet asteroïde
5. Calculer le poids équivalent (en kg) d'un homme sur toutes les planètes (+astéroïde, gravité max et min). Cet homme a une masse de 80 kg et a donc un poids équivalent sur la terre de 80 kg (avec $g = 9.81 \frac{m}{s^2}$).
6. Faire un "bar plot" (help/doc bar) avec les différents poids équivalents sur les différentes planètes

6.3 Input data

- Rayon moyen de la terre: $R_T = 6'371.008\,km$
- Masse volumique moyenne de la terre: $\rho_T = 5'515\,\frac{kg}{m^3}$
- $g = G \cdot \frac{M}{r}$
- Rayon moyen du Soleil: $R_S = 696'342\,km$
- Masse du Soleil: $M_S = 1.9919 \cdot 10^{30}\,kg$
- Rayon moyen de Venus: $R_V = 6'051.8\,km$
- Masse de Venus: $M_V = 4.867 \cdot 10^{24}\,kg$
- Rayon moyen de la lune: $R_l = 1'737.4\,km$
- Masse de la lune: $M_l = 7.342 \cdot 10^{22}\,kg$
- Rayon moyen de Mars: $R_M = 3'396.2\,km$
- Masse de Mars: $M_V = 6.419 \cdot 10^{23}\,kg$
- Rayon moyen de Jupiter: $R_J = 71'492\,km$
- Masse de Jupiter: $M_J = 1.898 \cdot 10^{27}\,kg$
- Rayon moyen de Pluton: $R_P = 1'185\,km$
- Masse de Pluton: $M_V = 1.314 \cdot 10^{22}\,kg$
- Rayon moyen de l'asteroïde de Vredefort: $R_a = 10 - 15\,km$
- Masse volumique d'un astéroïde: $\rho_a = 2'000 - 5'000\,\frac{kg}{m^3}$

7 General Help

```
%% To start with:
clc; % Clear command window
clear all; % Clear workspace
close all; % Closes all figures
dbstop if error % Stops before error

%% To define a scalar, vector or matrix:
a=5; % scalar a, a defined as 5
A1=[1 2 3 4]; % Vector A1 (1 line x 4 columns)
A1=[1, 2, 3, 4]; % Vector A1 (1 line x 4 columns), same as before
A2=[1;2;3;4]; % Vector A2 (4 lines x 1 column), similar as before
B=[1 2 3; 4 5 6; 7 8 9]; % Matrix B, same as phone dials

c=A1(3); % c is defined as the 3rd element of A1, which is 3
c=A2(3); % c is defined as the 3rd element of A2, which is also 3
d=B(2,3); % d is defined as the element in 1st line and 3rd column of B, which is 6
D1=B(3,:); % D1 is defined as the 3rd line of B, which is [7 8 9]
D2=B(:,2); % D2 is defined as the 2nd column of B, which is [2;5;8]

X=1:100 % defines a vector X from 1 to a 100 with an (default) increment of 1
X=a:b:c % defines a vector X between a and c, with an increment of b between each value
X=linspace(1,100,100) % defines a vector X from 1 to 100 with 100 values, no increment value
    defined by user, in this case increment is equal to ((100-1)+1)/100=100
X=linspace(a,c,b) % defines a vector X between a and c with b values, no increment value
    defined by user, in this case increment is equal to ((c-a)+1)/b

%% To initialize something:
a=0; % initializes scalar a to 0
A=[]; % initializes vector/matrix A to a void vector/matrix
B=zeros(a,b) % initializes vector/matrix B to a axb matrix (vector if a or b equal to 1) full
    of zeros
C=ones(a,b) % initializes vector/matrix C to a axb matrix (vector if a or b equal to 1) full
    of ones

%% Structures
A.a=1;
A.b=2;
A.c=3; % Stores a, b, c in the structure A

%% Loops

% While Loop: do something until condition is true
while condition % condition with comparison, i.e. i==0, i<=0, i~=0, ...
    do something % Put normal instructions here
    change condition requirements % If needed, change the condition variable in order to
    ensure you don't have an infinite loop
end

% If Loop: do something (once) if condition is true
if condition1
    do something % Put normal instructions here
elseif condition2 % if 1st condition is false (as a nested if loop), check if 2nd condition is
     true
    do something else % Put other instructions here
else % If condition 1 and 2 are false, then executes this
    do something else % Put other instructions here
end

% For Loop: do something n-times
for i=1:n % or i=0:2:2n, or i=2:1:n+2,...
% Usually for indexing loops i,j,k,l,m,n,o,p,q, etc. are used, in order
    do something % Put normal instructions here
end

% Break or continue, to use with caution!
break; % In a while or for loop, allows to "break" the loop and get out of it
continue; % Jumps to the next iteration of a while or for loop
% Both work only for the current loop (if nested loops, breaks or continues the closest)

```

```
% Nested loops:
A=zeros(3,3) % Initializes A (because loop afterwards depend on size of A)
for i=1:length(A(1,:)) % length of first line of A, or size(A,1)
% Does a for loop for every line of A
    for j=1:size(A,2) % size of lines of A, or length(A(1,:))
        % Does a nested for loop for every column of A
        A(i,j)=i+j; % Creates the matrix A with each element equal to the sum of its indices
    end
end

%% Conditions:
if A && B
    do something % When A and B are true
end
if A || B
    do something % When A or B are true
end
if A==B
    do something % When A equals to B
end
if A~=B
    do something % When A is not equal to B
end
if A>=B
    do something % When A is greater or equal to B
end

%% Multiplying matrices
A=[1 3; 4 2];
B=[10 8; 5 1];
C=A*B; % Gives the matrix product between A and B, thus equal to:
C=[A(1,1)*B(1,1)+A(1,2)*B(2,1), A(1,1)*B(1,2)+A(1,2)*B(2,2);
   A(2,1)*B(1,1)+A(2,2)*B(2,1), A(2,1)*B(1,2)+A(2,2)*B(2,2)]
D=A.*B; % Gives the multiplication of each element of A with each element (with the same index
    ) of B
D=[A(1,1)*B(1,1), A(1,2)*B(1,2); A(2,1)*B(2,1), A(2,2)*B(2,2)];
% Also used when you want to define functions for plots
% Example
f = sin(X.*X); % If X*X, then f would be a matrix where x*x would be the matrix product of x
    with x
% Also needed for division and power
g = (X.^2)./(2*Y); % See that not needed when multiply by a constant. Btw, constants directly
    multiply every element of matrix/vector

% Solve a linear system Ax=b, where A is a matrix and b a vector
x=A\b; % Careful, the backslash is the other way! (Improved version of x=inv(A)*b)

%% Plots:
X=1:0.001:2*pi; % vector between 1 and 2*pi with increment of 0.001
Y=sin(X); % sinus value for each number in vector defined before, creates a vector Y
% an alternative is to create a symbolic fct y such as:
    y = @(x) sin(x);
    Y=y(X);
% This way allows to reuse the function again with other values
plot(X,Y); % Plots all points with coordinates X(i), Y(i) for the whole vector

X=-1:0.1:1
Y=-1:0.1:1
Z=x^2+y^3;
surf(X,Y,Z)

X=-1:0.0001:1
Y=-1:0.0001:1
Z=x^2+y^3;
surf(X,Y,Z) % Same as before surface but with much finer mesh

% Tips for nice format:
set(0,'DefaultFigureWindowStyle','docked') % Docks windows in Matlab
set(0,'defaultAxesFontSize',18) % Sets bigger font for printing figures

%% Functions:
% In the main program
```

```
133
134 B1=Namefct1(A);
135 B2=Namefct2(A);
136
137 % Usually below the main program in the same file or in other .m files (Needs to be in the
        same folder)
138
139 function [Output_1, Output_2, ..., Output_N] = Namefct1 (Input_1, ... , Input_N)
140 % Describe function (used also for help)
141 do instructions in the function
142 end
143
144 function Output = Namefct2 (Input) % Bracket for output not necessary if unique
145 % Describe function (used also for help)
146 do instructions in the function
147 end
148
149 %% Output a message in the command window
150 disp('Here''s my message') % double '' to output '
151 fprintf(%d,a) % Outputs the decimal value of a as a message in the command window
152 disp([num2str(N),' uniformly distributed random points on a rectangle']) % Outputs the value
        of N with a message after (message before possible as well)
153
154 % Important:
155 % Explore some unknown function:
156 help namefunction % Works offline, gives a first insight (Can also work with your descriptions
        of your own functions)
157 doc namefunction % Works online, gives a deep understanding of the function with definition of
        parameters and examples.
158
159 % For instance, try:
160 help plot
161 doc plot
162 help surf
163 doc surf
```