Lab 10

ENED 1090: MODELS I
Week 10 Laboratory

Submit Week 11 during Lab

姓名:	易弘睿
Xìngmíng (拼音):	Yi Hongrui
English Name:	Horace
CQU Student ID:	20186103

INSTRUCTIONS

Complete each question below by typing your answer or copying from the output in MATLAB or Excel.

This assignment is to be completed outside of class. You will submit a digital copy to your TA during the lab session next week.

!!! **To receive points for this assignment, add your name to the filename. For example, if my name is Lin Yali, I will change the filename to**

Wk10_ened1090_laboratory_LinYali.doc

OBJECTIVES

For this assignment, students will demonstrate

- Arrays

For this assignment,: you will apply the instructions to MATLAB (or Octave)

PROBLEM 1

Consider the following Mathematics principle:

A "dot product" is a special multiplication of two vectors **a** and **b** such that

$$c = a \cdot b = a_1b_1 + a_2b_2 + a_3b_3$$

A "cross product" is a special multiplication of two vectors **a** and **b** such that

$$c = a \times b = \begin{bmatrix} a_2b_3 - b_2a_3 \\ a_3b_1 - b_3a_1 \\ a_1b_2 - b_1a_2 \end{bmatrix}$$

Create a MATLAB script that does the following:

- Ask the user for two vectors **a** and **b** and compute the "dot product" and the "cross product"

Some suggestions:

- Name your script – **VectorProduct**
- Make sure you have a program header, GPP, and place comments in your code
- **Make sure you include clear; clc at the beginning of your code**

Run the code with the following values and complete the table

a	b	$c = a \cdot b$	$c = a \times b$
-1.1710	1.0748	-2.3700	-0.5591
0.8894	0.0270		1.4839
1.9994	-0.5679		-0.9875
0.8854	-0.5609	2.5513	-1.8911
-0.8069	0.2870		-2.5710
2.0163	1.6265		-0.1985

Copy your **script** here.

```
%% VectorProduct
% Name: Horace
% Date: 10 Nov 2018
```

```
% Description: This script use dot product and cross product to
calculate
% the multiplication of two vectors a and b.

%% Code
%clear processor
clear; clc;

%Display a1 and b1
a1 = [-1.1710;0.8894;1.9994];b1 = [1.0748;0.0270;-0.5679];
fprintf('a1:            b1: \n')
for i = 1:length(a1)
fprintf('%f \t %f \n \n',a1(i),b1(i))
end
%use two types of multiplication
c1dot = a1(1)*b1(1)+a1(2)*b1(2)+a1(3)*b1(3);
c1cross = [a1(2)*b1(3)-b1(2)*a1(3) a1(3)*b1(1)-b1(3)*a1(1)
a1(1)*b1(2)-b1(1)*a1(2)]';
%display the result
disp('The dot multiplication is ');disp(c1dot);
disp('The cross mulptiplication is ');disp(c1cross);

%Display a2 and b2
a2 = [0.8854;-0.8069;2.0163];b2 = [-0.5609;0.2870;1.6265];
fprintf('a2:            b2: \n')
for i = 1:length(a2)
fprintf('%f \t %f \n \n',a2(i),b2(i))
end
%use two types of multiplication
c1dot = a2(1)*b2(1)+a2(2)*b2(2)+a2(3)*b2(3);
c1cross = [a2(2)*b2(3)-b2(2)*a2(3) a2(3)*b2(1)-b2(3)*a2(1)
a2(1)*b2(2)-b2(1)*a2(2)]';
%display the result
disp('The dot multiplication is ');disp(c1dot);
disp('The cross mulptiplication is ');disp(c1cross);
```

PROBLEM 2

Consider the following Statistics Problem:

> MATLAB has a few different random number generators. We have already seen the **randi** command. Another command, **randn**, creates a different type of random number. We are going to create a script that explores what these commands do by generating many numbers and sorting them numbers into number bins:
>
> the command **randi([Bmin, Bmax],1,N)** will give a random number between the minimum and maximum values based on a *uniform* distribution
>
> the command **randn(1,N)** will give a random number based on a *normal* distribution
>
> **Note**: this program will also demonstrate some a cool tricks that can be done within the "wrapper" loop.

Create a MATLAB script that does the following:

- Set some initial values for
  - bin minimum, **Bmin = -10**
  - bin maximum, **Bmax = 10**

- number of bins, **Nbin = 23**
- number of values, **N = 25**

- Begin the "**wrapper**" loop – this time just use the *infinite loop* **while 1**
  (all the remaining commands will be in this infinite loop)
- Use a **menu** to ask the user to select from the following options
  - RANDI
  - RANDN
  - Set new bin minimum
  - Set new bin maximum
  - Set new bin count
  - Set the number of values
  - QUIT
- Use a **switch block**
  1. Create a **randi** array using the form at the beginning of this problem
  2. Create a **randn** array using the form at the beginning of this problem
  3. Use **fprintf** to display current bin minimum; use **input** to ask user for new minimum; add a **continue**
  4. Use **fprintf** to display current bin maximum; use **input** to ask user for new maximum; add a **continue**
  5. Use **fprintf** to display current bin count; use **input** to ask user for new count; add a **continue**
  6. Use **fprintf** to display the current number of values; use **input** to ask user for new number; add a **continue**
  7. Use **fprintf** to say goodbye; add a **return**

Suggestion: for 3 – 6
You can build your code so the value is only updated if the user enters a number. Try using the following block for case 3.

```
fprintf(['The current bin minimum is %0.4f \n', ...
         'Press ENTER or type a new value. \n'], Bmin)
user = input('New minimum? ');
if ~isempty(user); Bmin = user; end
continue;
```

- Sorting:
  - Use **linspace** to create an array with the bin boundaries **linspace(Bmin,Bmax,Nbin+1)**
  - Use **zeros** to create an array of the total values in each bin
  - Create a sorting process that will assign all the random numbers to the correct bin and keep track of the number of values in each bin
- Use **fprintf** to show which bin has the largest number of values
  Hint: using **[A, B] = max(total)**, **A** stores the number of values, **B** stores the index number where the maximum is

Some suggestions:

- Name your script – **RandomNumberDistribution**
- ** You will use your program again during Wk 11 **
- Make sure you have a program header, GPP, and place comments in your code
- **Make sure you include clear; clc at the beginning of your code**

Run the sorting many times. And experiment by changing the values from the menu. Comparing **randi** with **randn** what can you say about the where the most numbers are generated?

In the randi command: The numbers are generated in average.
In the randn command: The most numbers are generated in the middle of the bins.

Copy your **script** here.

```
%% RandomNumberDistribution
% Name: Horace
% Date: 10 Nov 2018
% Description: This script explores what randi and randn do by
generating
% many numbers and sorting them numbers into number bins.

%% Code
%clear processor
clear; clc;
```

```
%Set some initial values
Bmin = -10;             %bin minimum
Bmax = 10;              %bin maximum
Nbin = 23;              %number of bins
N = 25;                 %number of values

%Begin the ¡°wrapper¡± loop
yes = 1;
while yes == 1

%Use a menu to ask the user to select from the options
options = menu('Welcome to the Random World',...
                'RANDI',...
                'RANDN',...
                'Set new bin minimum',...
                'Set new bin maximum',...
                'Set the number of values',...
                'QUIT');
switch options
    case 1
        %Create a randi array
        number = randi([Bmin, Bmax],1,N);

    case 2
        %Create a randn array
        number = randn(1,N);

    case 3
        %Display current bin minimum
        fprintf(['The current bin minimum is %0.4f \n',...
                 'Press ENTER or type a new value. \n'],...
                 Bmin)
        user = input('New minimum? ');
            if  ~isempty(user)
                Bmin = user;
            end
            continue

    case 4
        %Display current bin maximum
        fprintf(['The current bin maxmum is %0.4f \n',...
                 'Press ENTER or type a new value. \n'],...
                 Bmax)
        user = input('New maxmum? ');
            if  ~isempty(user)
                Bmax = user;
            end
            continue

    case 5
         %Display current bin count
```

```
        fprintf(['The current bin count is %0.4f \n',...
                 'Press ENTER or type a new value. \n'],...
                 Nbin)
        user = input('New bin count? ');
           if  ~isempty(user)
               Nbin = user;
           end
           continue

    case 6
        %Display the current number of values
        fprintf(['The current number of values %0.4f \n',...
                 'Press ENTER or type a new value. \n'],...
                 N)
        user = input('New number of values? ');
           if  user~=empty
               N = user;
           end
           continue

    case 7
        %Say goodbye
        fprintf('Good bye!')
        return
end

%Sorting
 bins = linspace(Bmin,Bmax,Nbin+1);
 total = zeros(1,Nbin);
    for n = 1:N
        for i = 1:Nbin
           if bins(i)<=number(n) && number(n)<=bins(i+1)
              total(i) = total(i)+1;
           end
        end
    end
    disp('the total numbers');
    disp(total);
    [A,B] = max(total);
    fprintf(['------------------outcome-----------------\n',...
             'The bin with most numbers of is %i \n',...
             'This bin has %i values. \n',...
             '-----------------------------------------\n'],...
             B,A)

          yes = menu('Do you want to try new values?',...
          'Yes','No ');
end
```

PROBLEM 3

Consider the following Biology Problem:

In a predator-prey simulation, you can compute the annual populations increase of predators and prey using the following equations

$$P_{1,n+1} = P_{1,n} \times (1 + A - B \times P_{2,n})$$
$$P_{2,n+1} = P_{2,n} \times (1 - C + D \times P_{1,n})$$

where

- $P_{1,1} = 500$, the initial population of the prey
- $P_{2,1} = 55$, the initial population of the predator
- $A = 0.01$, the annual rate of prey population increase by natural causes
- $B = 0.0001$, the annual rate that prey are hunted by the predator
- $C = 0.01$, the annual rate of predator population decrease by natural causes
- $D = 0.0001$, the annual rate that predators thrive on abundant food sources

Create a MATLAB script that does the following:

- Create a loop to simulate the population increase of both species
  - Use a single array for P
  - Use a **single vector** to store A, B, C, D
  - After each calculation, round the values down to the nearest integer using the **floor** function
  - Exit the loop if
    - One population dies
    - Simulation reaches 100 years

Some suggestions:

- Name your script – **PredatorPrey**
- ** You will use your program again during Wk 11 **
- Make sure you have a program header, GPP, and place comments in your code
- **Make sure you include clear; clc at the beginning of your code**

Run the code then check the values of P to complete the table

Year	P_1	P_2
11	513	76
28	502	140
37	470	191
56	340	335
75	186	446

Copy your **script** here.

```
%% PredatorPrey3
% Name: Horace
% Date: 12 Nov 2018
% Description: This script

%% Code
%clear processor
clear; clc;

%Create an array for P
P = zeros(2,100);

%Code Dictionary
A = 0.01;           %The annual rate of prey population increase by
natural causes
B = 0.0001;         %The annual rate that prey are hunted by the
predator
C = 0.01;           %The annual rate of predator population
```

```
decrease by natural causes
D = 0.0001;          %The annual rate that predators thrive on
abundant food sources

%Give the initial population of the prey and the predator
P(1,1) = 500;        %The initial population of the prey
P(2,1) = 55;         %The initial population of the predator

%Simulate the population increase of both species
for n = 1:99
    P(1,n+1)=floor(P(1,n)*(1+A-B*P(2,n)));
    P(2,n+1)=floor(P(2,n)*(1-C+D*P(1,n)));
    if P(1,n+1)==0 || P(2,n+1)==0
        break
    end
end

%Display the array
disp(P)
```

PROBLEM 4

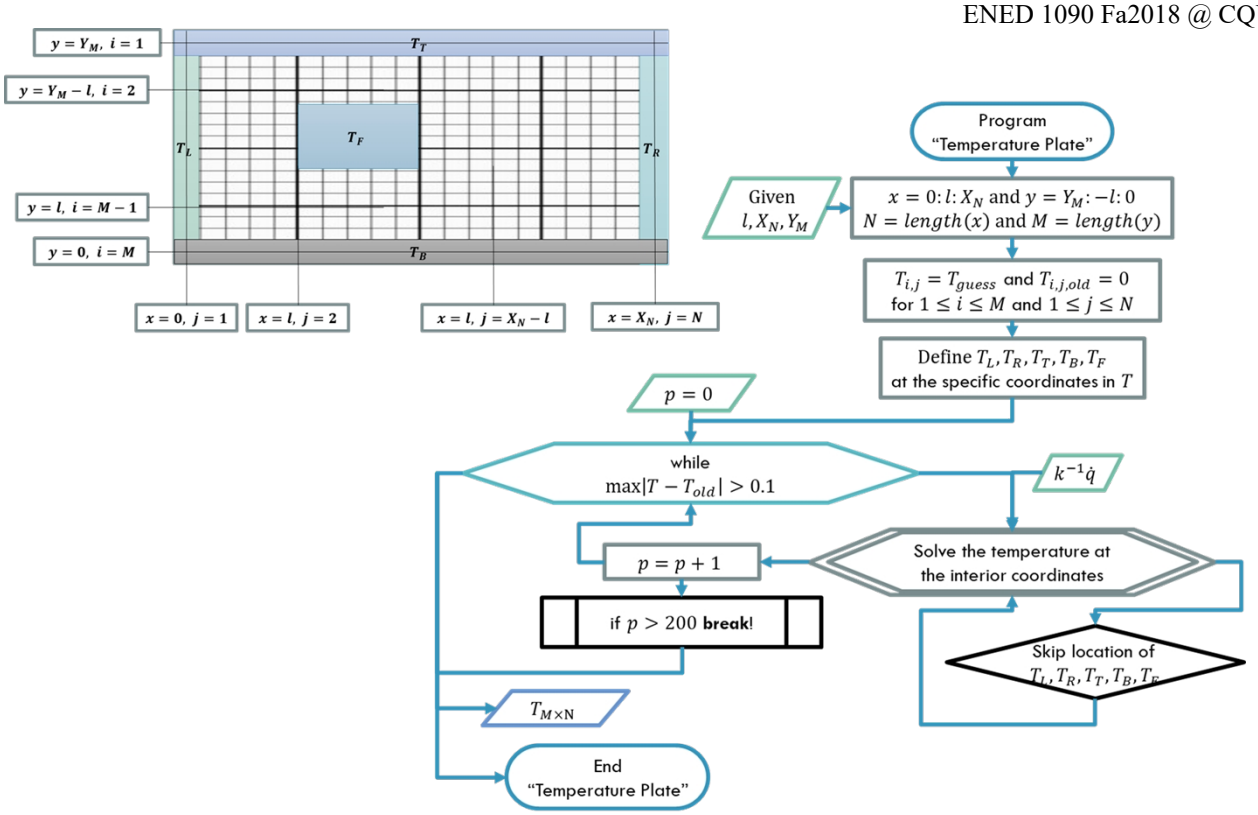
Consider the following Engineering Problem:

For steady, two-dimensional conduction with heat generation through a flat plate, the temperature at any coordinate on the plate is calculated as

$$T_{i,j} = \frac{T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} + l^2 k^{-1} \dot{q}}{4}$$

Because of the dependence on the surrounding coordinates, an iterative method (**while** loop) is required to determine the temperature values at all coordinates.

A diagram and flow chart of the method is shown below.



Create a MATLAB script that does the following:

- Calculate the temperature at each coordinate on the plate.

Some suggestions:

- Name your script – **TemperaturePlate**
- ** You will use your program again during Wk 11 **
- Make sure you have a program header, GPP, and place comments in your code
- **Make sure you include clear; clc at the beginning of your code**

- Use the following information in your program

$$l = 0.1, \quad X_N = 10, \quad Y_M = 5$$
$$k^{-1}q = 5$$
$$T_{guess} = 25$$
$$T_L = T_T = 40, \quad T_R = T_B = 10$$
$$T_F = 80, \text{ located at } 2 \le x \le 4 \text{ and } 2 \le y \le 3$$

Run the code then check the values of T to complete the table

i	j	$T_{i,j}$
25	25	80
95	38	1.611366e+01
7	8	4.061220e+01
88	17	2.668330e+01
78	45	1.765150e+01

Copy your **script** here.

```
%%Temperature
% Name: Horace
% Date: 12 Nov 2018
% Description:calculate the temperature at any coordinate on the
plate

%% code
 clear;clc;
```

```
 %the needed ivalue for caculation
 l=0.1;                    %Give the graduation value
 Xn=10;                    %Give the minimum of x
 Ym=5;                     %Give the minimum of y
 kq=5;                     %Give the value to k^(-1)*q
 Tguess=25;                %Give the guess value to T
 TL=40;                    %Give the value to the left of T
 TT=40;                    %Give the value to the top of T
 TR=10;                    %Give the value to the right of T
 TB=10;                    %Give the value to the bottom of T5
 TF=80;                    %Give the vaue to the middle of T

 %Create the array for T guess and T old
 xnew=0:l:Xn;
 ynew=Ym:-l:0;
 N=length(xnew);
 M=length(ynew);
 T=zeros(M,N);
 T(:,:)=Tguess;
 Told=zeros(M,N);

 %Locate the TT TB TL TR
 T(1,:)=TT;
 T(M,:)=TB;
 T(:,1)=TL;
 T(:,N)=TR;

 %Locate the TF
 for x=1:N
     for y=1:M
         if 2<=xnew(x) && xnew(x)<=4
             if 2<=ynew(y) && ynew(y)<=3
                T(y,x)=80;
             end
         end
     end
 end

%% Solve the temperature at each coordinate on the plate
 yes = 1;
while yes == 1
 %input the T(i,j) you want to get
 e=input('i=');
 d=input('j=');
    p=0;
 % caculate the temperature
while max(max(abs(T-Told)))>0.1 %compare T and T old
     p=p+1;
     Told = T;
    for i=2:M-1 %skip the TT TR TL TB
```

```
        for j=2:N-1
            %skip the TF
            if ~((2<=xnew(j) && xnew(j)<=4) &&  (2<=ynew(i) &&
ynew(i)<=3))
             %the fomula
             T(i,j)=(T(i+1,j)+T(i-1,j)+T(i,j-
1)+T(i,j+1)+kq*(l^2))/4;
            end
        end
    end
    if p>=200
        break
    end
end

 %output the T(i,j)
 fprintf('T(%i,%i) = %i \n',e,d,T(d,e));
 yes = menu('Do you want to try new values?',...
           'Yes','No');
end
```

PROBLEM 5

Consider the following Mathematics Principle.

A *telescoping* series is a series whose partial sums eventually only have a fixed number of terms after cancellation and approach a constant value. For example,

$$S_i = \sum_{n=1}^{i} \left[\frac{1}{n(n+1)}\right] \Rightarrow 1, \qquad as\ i \to \infty$$

A class of telescoping series, known as the **p-series** takes the form

$$S_N = \sum_{n=0}^{N} \frac{1}{(n+c)^p}$$

Create a MATLAB script that does the following:

- Calculate and store the *partial sums* of the p-series above in a **three-dimensional** array to test different values of n, c, p

Some suggestions:

- Name your script – **TelescopePSeries**
- ** You will use your program again during Wk 11 **
- Make sure you have a program header, GPP, and place comments in your code
- **Make sure you include clear; clc at the beginning of your code**

- Consider using **Sn = @(p, c, N) (n+c)^(-p)** at the beginning of the program to make it easier to read.
- Use the following information in your program

$$N = 0{:}100$$
$$c = logspace(-3,3)$$
$$p = linspace(0,5,5)$$

Copy the output **SN** for the set of results

- **all p, c(33), N(80)**
- **all p, c(33), N(90)**
- **all p, c(33), N(100)**

```
80.0000
   1.0873
   0.0298
   0.0013
   0.0001
```

```
90.0000
   1.1220
   0.0299
   0.0013
   0.0001
```

```
100.0000
   1.1526
   0.0300
   0.0013
   0.0001
```

Which values of **p** appear to be approaching a constant value?

```
3.75
5
```

Copy your **script** here.

```
%% TelescopePSeries
% Admin: Horace
% Date: 2018/11/20
% Description: This program alculate and store the partial sums of
the p-series above in a c
%              three-dimensional array to test different values of
n, c, p.

%% Code
clear;clc;
% Everyone does this, help you start a brand-new world!

%% Data Dictionary
% N             number of times superimposed
% c             a given array
% p             a given array

% Copy the known condition from the problem.
N = 0:100;
c = logspace(-3,3);p = linspace(0,5,5);
% Set up the function.
S_N = @(p,c,n) (n+c)^(-p);
% Record the length of the arrays.
L_N = length(N);L_c = length(c);L_p = length(p);
% Create a three-dimensional array to store different values of n,
c, p.
SN = zeros(L_p, L_c, L_N);
% Set up p and c when N=0 into the loop.
for x=1:L_p
    for y=1:L_c
        SN(x,y,1)=S_N(p(x),c(y),N(1));
    end
```

```
end
% Set up to do the function.
for x=1:L_p
    for y=1:L_c
        for i=2:L_N
            SN(x,y,i)=SN(x,y,i-1)+S_N(p(x),c(y),N(i));
        end
    end
end
% Program and reslut check.
disp(p)
disp(SN(:,33, 80));disp(SN(:,33, 90));disp(SN(:,33,100));
```