Laboratory Manual

Statistical Methods and Algorithms PCL 105

Course offered to **MTech** students

Computer Science & Engineering, Mechanical Engineering, and Civil Engineering

Thapar Institute of Engineering & Technology, Patiala, Punjab

List of Experiments

1. **A Brief Introduction to Matlab**
   – will familiarize the students with basic commands and functions in Matlab.
   – 4-6 contact hours (2-3 weeks).

2. **Least Squares Regression Analysis for Predicting Energy Consumption**
   – application in electrical, energy and environmental engineering.
   – 2 contact hours (1 week).

3. **Fundamental Concepts of Markov Processes with Applications in Weather Prediction and Random Walks**
   – application in weather forecasting, random walks and Brownian motion.
   – 2 contact hours (1 week).

4. **Construction of a Markovian Model using the Viterbi Algorithm to predict Aerodynamic Control Laws of an Aircraft**
   – application in mechanical and aerospace engineering.
   – 4 contact hours (2 weeks).

5. **Analyzing Statistical Differences in Multi-Population Means using ANOVA to prioritize Post Disaster Reconstruction Projects**
   – application in civil and construction engineering.
   – 2 contact hours (1 week).

6. **Autoregressive Model of Time Series Data using the YuleWalker Equations to forecast Employment Growth Statistics**
   – application in logistics and data analytics.
   – 2 contact hours (1 week).

7. **Multivariate Principal Component Analysis for Image Compression**
   – application in computer science engineering, image and signal processing.
   – 2 contact hours (1 week).

Term Projects

Students (in groups of 3 or 4) will be required to submit a term project based on an engineering application of any of the statistical models and algorithms covered in the above mentioned laboratory experiments. To accomplish this they will have to work with actual data available online or field data collected by them. The projects will be graded based on novelty of application, extent of real engineering or scientific data used in their analysis, 10-15 minutes oral presentation and a team report on the respective projects. The plan (proposal) of the projects will have to be presented to the instructor in the form of a one page abstract and the approval of the instructor must be obtained before the end of the first week of class after the mid-semester examination. The oral presentations will be scheduled during the last two weeks of the academic session before the end semester examination.

Sessional points: Weekly lab experiments (5), project proposal (7), oral presentation (8), project report (10).

A Brief Introduction to Matlab

1 Mathematical operations

Try the following operations and commands on the command line in the command window of Matlab.

1.1 Defining an array and arranging in ascending and descending order

```
>> A = [2  4 -3 43 2 10 log(200)]

A =

    2.0000    4.0000   -3.0000   43.0000    2.0000   10.0000    5.2983

>> sort(A)

ans =

   -3.0000    2.0000    2.0000    4.0000    5.2983   10.0000   43.0000

>> sort(A,'descend')

ans =

   43.0000   10.0000    5.2983    4.0000    2.0000    2.0000   -3.0000
```

1.2 Defining a matrix and arranging in ascending and descending order

```
>> B = [2 3 1 4; -4 -2 0 -13; 66 3 0.2 1/3]

B =

    2.0000    3.0000    1.0000    4.0000
   -4.0000   -2.0000         0  -13.0000
   66.0000    3.0000    0.2000    0.3333

>> sort(B)

ans =

   -4.0000   -2.0000         0  -13.0000
    2.0000    3.0000    0.2000    0.3333
   66.0000    3.0000    1.0000    4.0000

>> sort(B,2)

ans =

    1.0000    2.0000    3.0000    4.0000
  -13.0000   -4.0000   -2.0000         0
    0.2000    0.3333    3.0000   66.0000
```

1.3 Powers of a matrix and its elements

```
>> C = [ 1 2 3; 4 2 3; 5 6 2]

C =

     1     2     3
     4     2     3
     5     6     2

>> C^2

ans =

    24    24    15
    27    30    24
    39    34    37

>> C.^2

ans =

     1     4     9
    16     4     9
    25    36     4

>> D = [ 3 3 2; 5 0 1; -2 3.5 6]

D =

    3.0000    3.0000    2.0000
    5.0000         0    1.0000
   -2.0000    3.5000    6.0000

>> C*D

ans =

    7.0000   13.5000   22.0000
   16.0000   22.5000   28.0000
   41.0000   22.0000   28.0000

>> C.*D

ans =

     3     6     6
    20     0     3
   -10    21    12
```

1.4 Finding maximum and minimum entries in an array (or matrix)

```
>> A = [ 4 2 -7 -0.2 33 12]

A =

    4.0000    2.0000   -7.0000   -0.2000   33.0000   12.0000

>> max(A)

ans =

    33

>> min(A)

ans =

    -7
```

Finding `argmax` and `argmin` in an array

```
>> find(A == max(A))

ans =

     5

>> find(A == min(A))

ans =

     3

>> D

D =

    3.0000    3.0000    2.0000
    5.0000         0    1.0000
   -2.0000    3.5000    6.0000

>> max(D)

ans =

    5.0000    3.5000    6.0000

>> max(D,[],2)
```

```
ans =

     3
     5
     6
```

Summing (and *cumulative* summing) rows and columns of a matrix

```
>> A = [ 3 1 -2 5; 0 2 11 -4; 9 0 7 6]

A =

     3     1    -2     5
     0     2    11    -4
     9     0     7     6

>> sum(A)

ans =

    12     3    16     7

>> sum(A,2)

ans =

     7
     9
    22

>> cumsum(A)

ans =

     3     1    -2     5
     3     3     9     1
    12     3    16     7

>> cumsum(A,2)

ans =

     3     4     2     7
     0     2    13     9
     9     9    16    22
```

Product of entries of a matrix

```
>> prod(A)
```

```
ans =

     0     0  -154  -120

>> prod(A,2)

ans =

   -30
     0
     0
```

1.5 Accessing certain sections of a matrix

```
>> A = [22 3 -4; 0 1 7; 2 -1 2]

A =

    22     3    -4
     0     1     7
     2    -1     2

>> A(:,2)

ans =

     3
     1
    -1

>> A(3,:)

ans =

     2    -1     2

>> [-2 -1 -3].*A(1,:)

ans =

   -44    -3    12
```

1.6 Some elementary mathematical functions

```
 >> exp(2)

ans =

    7.3891

>> sin(pi/4)
```

```
ans =

    0.7071

>> log(100)

ans =

    4.6052

>> log10(100)

ans =

     2

>> exp([1 2; 3 4])

ans =

    2.7183    7.3891
   20.0855   54.5982
```

1.7 Random number generator

```
 >> rand

ans =

    0.3112

>> rand()

ans =

    0.5285

>> rand(3)

ans =

    0.1656    0.6541    0.4505
    0.6020    0.6892    0.0838
    0.2630    0.7482    0.2290

>> rand(1,2)

ans =

    0.9133    0.1524
```

```
>> a=2

a =

     2

>> b=5

b =

     5

>> r = a+(b-a).*rand(5,1)          % random numbers between a and b
                                       % from uniform distribution

r =

    4.4775
    3.6150
    4.9884
    2.2345
    3.3280

>> randi([-3 3],5,1)

ans =

    -3
     3
     1
     0
    -2

>> randperm(4)

ans =

     3     4     2     1

>> nchoosek(5,2)

ans =

    10
>> perms([1 2 3])

ans =

     3     2     1
     3     1     2
     2     3     1
     2     1     3
     1     3     2
     1     2     3
```

Normally distributed random numbers

```
 >> randn(5)

ans =

    1.1275   -1.7502   -0.5336   -0.0348    1.3514
    0.3502   -0.2857   -2.0026   -0.7982   -0.2248
   -0.2991   -0.8314    0.9642    1.0187   -0.5890
    0.0229   -0.9792    0.5201   -0.1332   -0.2938
   -0.2620   -1.1564   -0.0200   -0.7145   -0.8479
```

1.8 Calculus and functional operations

Writing matlab script files

It must be noted that the `command window` is not the only interface in Matlab to perform mathematical operations. In fact there is a more optimal and preferred editor where one can write Matlab script files and define functions of their choice. To familiarize the user with the matlab editor, the following sequence of operations are written as a script file using the editor. In order to execute the code, you simply have to hit the `run` icon on the editor at the top. The answers and results of the code appear on the `command window` as well as in the `workspace`.

The commented sections of the code appear to the right of the % symbol and are non-executable portions of the code. The comments should be self explanatory.

```
% function handles and function evaluations
f = @(y) exp(y)
f(1)

syms x          % define x as a symbolic variable
g = 5*exp(x) + sin(x) % symbolic function definition
dg = diff(g)    % symbolic differentiation
subs(dg,x,pi)   % replace x with pi in the expression given by dg


% function handles and function evaluations
A = [1 2; 3 4] % A is a matrix whose eigen system we want to find
fn = @eig
%two alternate ways of finding eigenvector and eigenvalues
[EV ev] = fn(A)
% OR
[V e] = feval(fn,A)
```

2 Graphical representation of data

2.1 Plotting functions

The following piece of program gives you an idea of plotting functions graphically, assigning appropriate labels, defining boundaries of graphs and using desired font size and markers. Additionally, you will also learn how to generate multiple plots within a single graphic frame using the matlab `subplot` functionality. Tweak the parameters and practise to get a better understanding of the different available options for pictorially depicting data in matlab.

```
x=[0:0.05:4*pi];
y=sin(x);
z=cos(x);
figure, subplot(3,1,1);
plot(x,y,'r*-',x,z,'bo-');
xlim([0 4*pi]);
ylim([-1.3 1.3]);
xlabel('x');
ylabel('fundamental oscillations');
legend('sin(x)','cos(x)');
title('periodic functions');

subplot(3,1,2);
xx=[0:0.1:4*pi];
t=tan(xx);
stem(xx,t,'mx')
xlim([0 4*pi]);
ylim([-10 10]);
xlabel('x');
ylabel('tan x');
title('periodic singular function');

%% drawing rectangular pulse %%
% This example generates a pulse train using the default rectangular
% pulse of unit width. The repetition frequency is 0.5 Hz,
% the signal length is 60 s, and the sample rate is 1 kHz.
% The gain factor is a sinusoid of frequency 0.05 Hz.
t = 0:1/1e3:60;
d = [0:2:60;sin(2*pi*0.05*(0:2:60))]';
x = @rectpuls;
y = pulstran(t,d,x);
subplot(3,1,3);
plot(t,y,'-bs',...
    'LineWidth',2,...
    'MarkerSize',10,...
    'MarkerEdgeColor','g',...
    'MarkerFaceColor',[0.5,0.5,0.5]);
hold on
xlabel('Time (s)')
ylabel('Waveform')
title('rectangular pulse train');
```

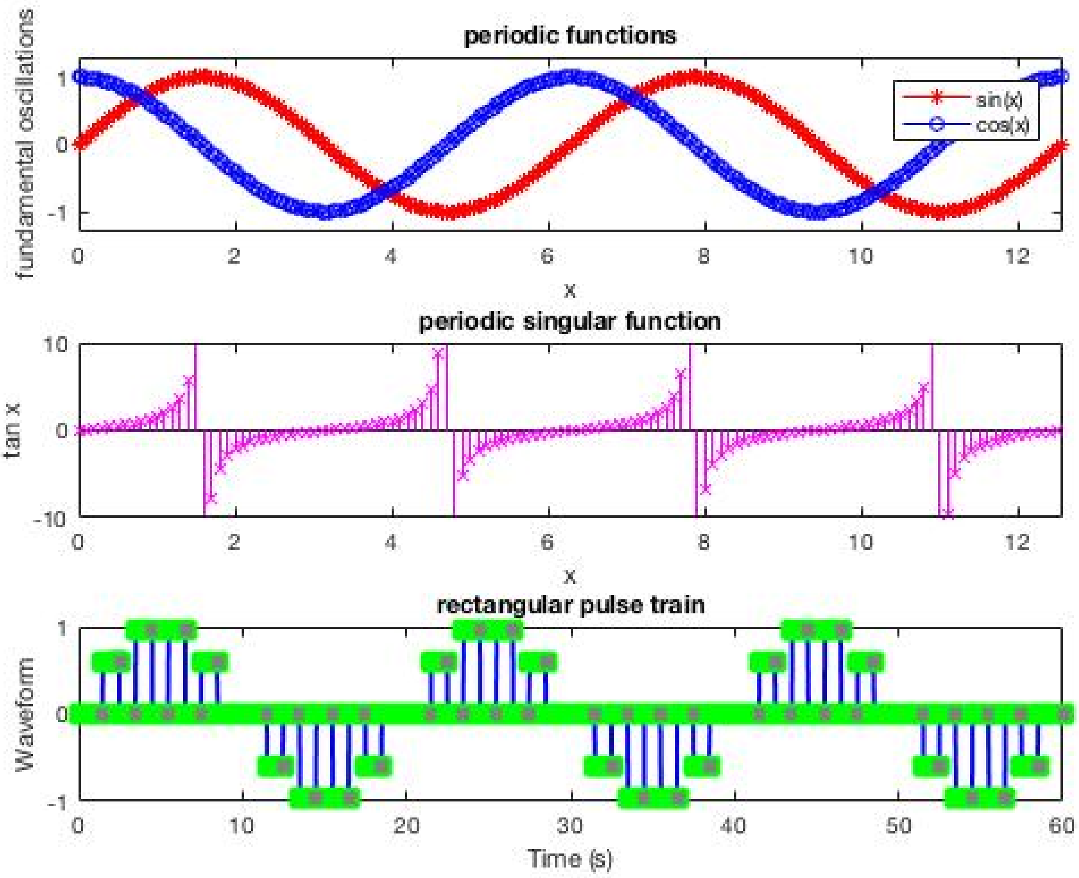

Figure 1: How to make subplots in matlab with labels and legends?

2.2 Plotting data on a bar graph

In a matlab script file, store data as a matrix as shown below. Then call a user defined function `mydataPlot` which takes the input matrix as its argument and draws a joint-histogram of the data over time (years). The data must be normalized on a scale of 0-2 and depicted as a bar graph.

The script file may contain the following lines of code.

```
 %%%%%%%%%%%%%%%% making histograms %%%%%%%%%%%
% Ip stores input data in 3 columns:
% (Year, Rainfall, Temperature) in appropriate units
Ip=[2009 1000 39; 2010 997 34;2011 1152 41; 2012 855 31; ...
    2013 1013 40; 2014 878 30; 2015 1243 43];
mydataPlot(Ip);
```

Writing user defined functions

`mydataPlot` is a user defined function which needs to be programmed by writing a short matlab function and saving it as `mydataPlot.m`. The function may be written as follows.

```
function [X] = mydataPlot(Ip)

A = Ip(:,1);
```

```
B = Ip(:,2);
C = Ip(:,3);
Bnew=(B(:)-min(B))/(max(B)-min(B));
Cnew=(C(:)-min(C))/(max(C)-min(C));
newData = [Bnew+1 Cnew+1]; % setting it on a normalized scale 0-2
figure, bar(A,newData(:,1:2));
xlabel('Year','fontsize',18);
ylabel('Normalized data on a scale of 0-2','fontsize',18);
legend('Amount of rainfall','temperature','Location','northwest');
title('Visualizing trend & correlation qualitatively using joint-histogram',...
    'fontsize',14);

end % denoting end of function
```

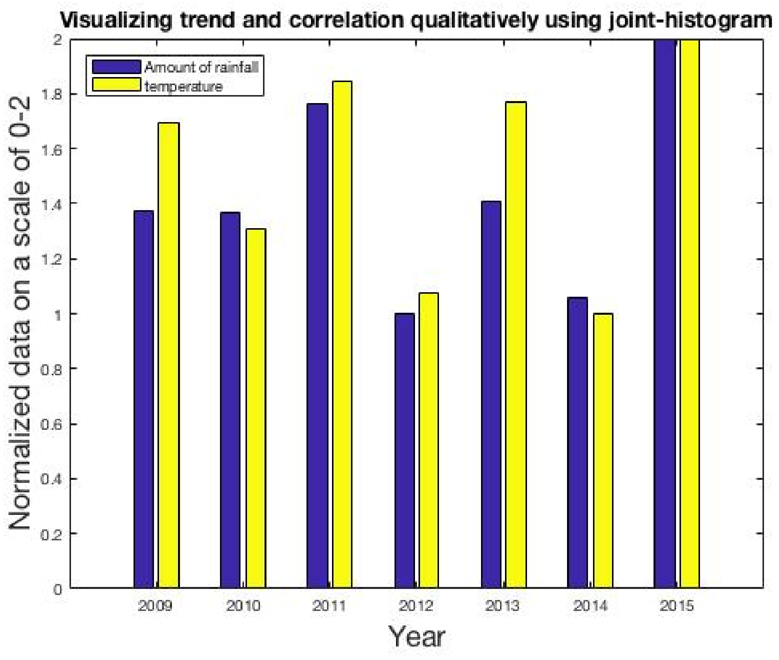

Figure 2: How to make bar graphs in matlab with labels and legends?

3 Loops and conditional statements

Matlab provides in-built *loop* functionalities using the following syntax.

```
for
   .........
   .........
end
```

or

```
while
   .........
   .........
end
```

3.1 `for` loop to generate a matrix with labels in lexicographical order

```
% use for loops to construct a matrix with entries
% that store the serial location (lexicographical order)
% also compute the sum of all entries of the same matrix

imax=4; jmax=5;
running_sum=0;
for i=1:imax
    for j=1:jmax
        A(i,j) = (j + (i-1)*jmax);
        running_sum = running_sum + A(i,j);
    end
end
```

3.2 Conditional statement using `if-else` for comparing results

```
if running_sum == sum(sum(A))
    disp('my calculation is correct: Hurrah!');
else
    disp('I made an error in calculation');
end
```

3.3 `for` loops for printing prime numbers less than or equal to N

Exercise: Following is a pseudocode for printing all the prime numbers less than or equal to N where $N = 30$ for example. Convert the pseudocode to matlab executable code and display your answer.

Pseudocode for printing prime numbers:

```
INPUT: $N$.

for all i from 2 to N
   reset prime number flag to default ON
   for all j from 2 to i/2
      if remainder of $i \div j$ is not equal to 0
         continue the current for loop
      else
         turn OFF prime number flag
         break from the current for loop
      end if-else condition
   end for loop for j
   if prime number flag is ON
      store prime number i in a dynamic array
   end if condition
end for loop for i

OUTPUT: display array containing the prime numbers.
```

While implementing the above algorithm you will learn to use the following matlab commands: `mod`, `continue` and `break`. Use `>>doc` ⟨`function name`⟩ or `>>help` ⟨`function name`⟩ on the command line to learn how to use them. Also, the comparative clause *'is not equal to'* is written in matlab as ∼=.

4 Data structures: loading, organizing, accessing and writing data

There are many different data structures available in matlab. Here we will discuss only some of the important ones that may be relevant to us.

4.1 Data as graphs of matrices

A *graph* is a data structure that consists of the following two components.

1. A finite set of *vertices* also known as *nodes*.
2. A finite set of ordered pairs of the form (u, v) known as *edges*. The pair is ordered because (u, v) is not same as (v, u) in case of a directed graph (di-graph). The pair of the form (u, v) indicates that there is an edge from vertex u to vertex v. The edges may contain weight/value/cost.

Graphs are used to represent many real-life applications: Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network. Graphs are also used in social networks like linkedIn, Facebook. For example, in Facebook, each person is represented with a vertex (or node). Each node is a structure and contains information like id, name, gender and location of a person.

4.1.1 Adjacency Matrix

A graph may be represented as an *adjacency matrix*. An *adjacency matrix* is a 2D array (or matrix) of size $V \times V$ where V is the number of vertices in a graph. Let the 2D array be $adj(i,j)$, a slot $adj(i,j) = 1$ indicates that there is an edge from vertex i to vertex j. Adjacency matrix for undirected graph is always symmetric. Adjacency Matrix is also used to represent weighted graphs. If $adj(i,j) = w$, then there is an edge from vertex i to vertex j with weight w.

Drawing a graph from adjacency matrix:

```
% Define a matrix adj.
adj = [0 1 1 0 ; 1 0 0 1 ; 1 0 0 1 ; 0 1 1 0];

% Draw a picture showing the connected nodes.
cla % clear current axis
subplot(1,2,1);
gplot(adj,[0 1;1 1;0 0;1 0],'.-');
% gplot(A,xy) plots the graph (as in 'graph theory') specified by A and xy
text([-0.2, 1.2 -0.2, 1.2],[1.2, 1.2, -.2, -.2],('1234')', ...
   'HorizontalAlignment','center')
axis([-1 2 -1 2],'off')
title('undirected graph','fontsize',14);

% Draw a picture showing the adjacency matrix.
subplot(1,2,2);
xtemp = repmat(1:4,1,4); ytemp = reshape(repmat(1:4,4,1),16,1)';
text(xtemp-.5,ytemp-.5,char('0'+adj(:)),'HorizontalAlignment','center');
```

```
line([.25 0 0 .25 NaN 3.75 4 4 3.75],[0 0 4 4 NaN 0 0 4 4])
axis off tight
title('adjacency matrix','fontsize',14);
```

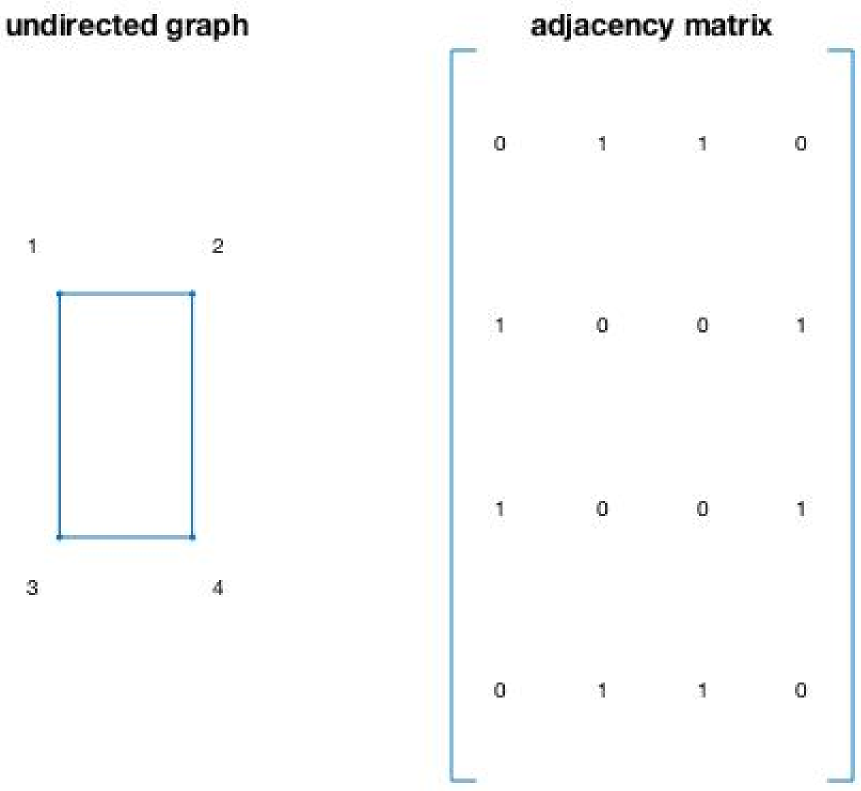

Figure 3: A *graph* and its corresponding *adjacency matrix*.

Drawing the adjacency matrix of a given graph:

For fun, we will first use the matlab in-built function `bucky` to first draw the graph of a *geodesic dome*.[1]

```
%% using bucky and find the adjacency matrix of a given graph
[B,V] = bucky;
G = graph(B);
figure,
p = plot(G);
```

In order to investigate the graph `G` in more detail, type `>> G.Edges` on the command line. Now, let us say we were given the graph `G` (and not the matrix `B`), and we had to find the corresponding adjacency matrix `A`, we would do as follows:

```
A = adjacency(G);
H = graph(A(1:10,1:10)); % graph only a section of the adjacency matrix
figure,
h = plot(H);
```

[1] https://en.wikipedia.org/wiki/Geodesic_dome

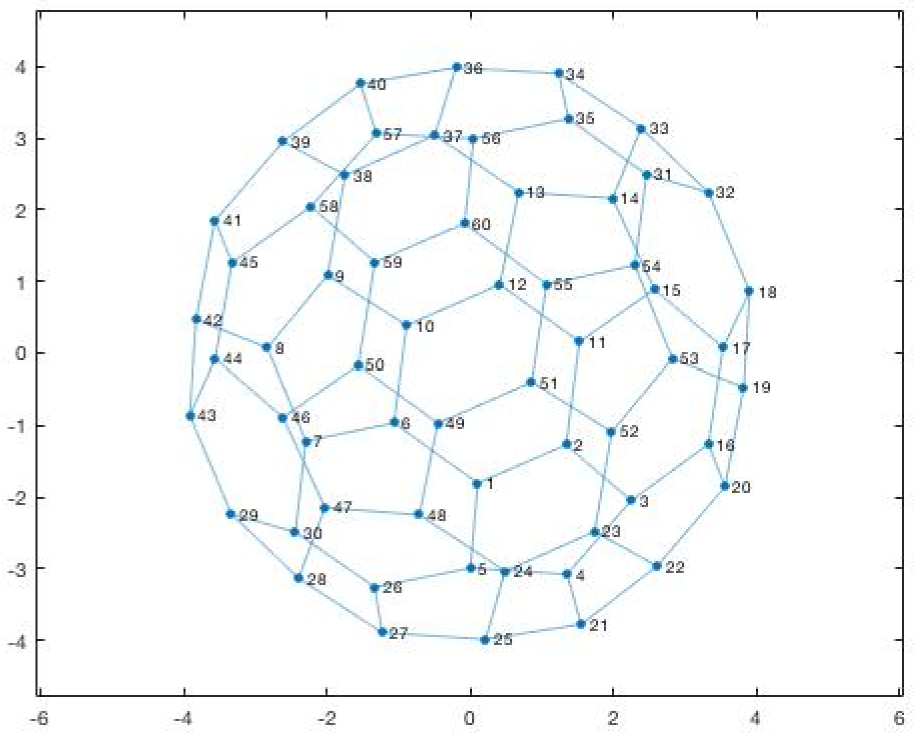

Figure 4: Graph of the geodesic dome using the matlab function `bucky`.

The new plot graphs the selected section of the adjacency matrix `A` specified by the first 10×10 entries of `A`. The new adjacency matrix `A` can be seen from the command line as $>>$ `A` and the new truncated graph is shown below. The information pertaining to the new truncated graph can be found thusly.

```
>> H.Edges

ans =

  112 table

    EndNodes     Weight
    ________     ______

    1     2        1
    1     5        1
    1     6        1
    2     3        1
    3     4        1
    4     5        1
    6     7        1
    6    10        1
    7     8        1
    8     9        1
    9    10        1
```

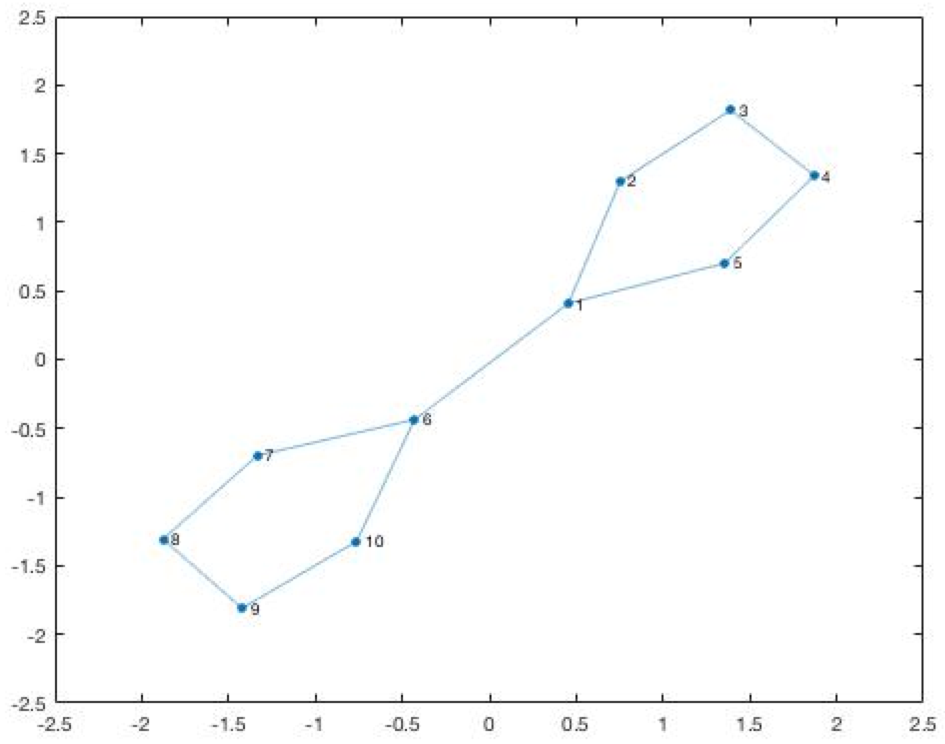

Figure 5: Graph of the truncated geodesic dome.

You can also use similar commands to generate the adjacency matrix of the graph in the previous example (the box graph) and compare it with the matrix `adj` you defined to begin with.

```
Gnew=graph(adj)
Anew = adjacency(Gnew);
Hnew = graph(Anew);
figure,
hnew = plot(Hnew);
```

Compare `Anew` and `adj` to check for consistency.

```
>> Anew

Anew =

   (2,1)        1
   (3,1)        1
   (1,2)        1
   (4,2)        1
   (1,3)        1
   (4,3)        1
   (2,4)        1
   (3,4)        1

>> adj
```

```
adj =

     0     1     1     0
     1     0     0     1
     1     0     0     1
     0     1     1     0
```

4.2 Data as tables

Often we will encounter situations where we may have to import data from an excel file (.csv file). This data in the excel file will be assumed to be stored in tabular (columnar) format under field headings. An example at hand is the file named

`EnergyConsumptionMP_1996-2018.csv`

where the data is stored in two columns under the field headings:

`Year`

and

`EnergySupply_MU_`

This data may be imported in matlab local workspace as follows:

```
%% reading data from .csv file
T = readtable('EnergyConsumptionMP_1996-2018.csv', 'ReadVariableNames', ...
    true, 'Format', '%f %f');
xdata = T.Year;
ydata = T.EnergySupply_MU_;
>> xdata

xdata =

        1996
        1998
        2000
        2002
        2004
        2006
        2008
        2010
        2012
        2014
        2016
        2018

>> ydata

ydata =

       27094
       28599
       30624
       32232
       33435
       36073
       35503
       35563
       38799
       42945
       47858
       51976
```

4.3 Local data structures in matlab

4.3.1 Structure arrays

Below is a code to show a single variable (named student) which contains two fields which are "sub-components" of what it means to be a student. Each field has its own name and its own type.

```
        students(1).name  = 'jim';
        students(1).age   = 21;

        students(2).name  = 'Jane';
        students(2).age   = 33;

        students(3).name  = 'Joe';
        students(3).age   = 25;

        students(4).name  = 'Janet';
        students(4).age   = 24;
```

A querry about the first student yields the following information.

```
>> students(1)

ans =

  struct with fields:

    name: 'jim'
     age: 21
```

4.3.2 Cell arrays

Cell arrays contain data in cells that you access by numeric indexing. Common applications of cell arrays include storing separate pieces of text and storing heterogeneous data from spreadsheets. For example, store temperature data for three cities over time in a cell array.

```
temperature(1,:) = {'2009-12-31', [45, 49, 0]};
temperature(2,:) = {'2010-04-03', [54, 68, 21]};
temperature(3,:) = {'2010-06-20', [72, 85, 53]};
temperature(4,:) = {'2010-09-15', [63, 81, 56]};
temperature(5,:) = {'2010-12-09', [38, 54, 18]};

>> temperature
```

```
temperature = 5x2 cell array
    {'2009-12-31'}    {1x3 double}
    {'2010-04-03'}    {1x3 double}
    {'2010-06-20'}    {1x3 double}
    {'2010-09-15'}    {1x3 double}
    {'2010-12-09'}    {1x3 double}
```

Plot the temperatures for each city by date.

```
allTemps = cell2mat(temperature(:,2));
dates = datetime(temperature(:,1));

plot(dates,allTemps)
title('Temperature Trends for Different Locations')
xlabel('Date')
ylabel('Degrees (Fahrenheit)')
```

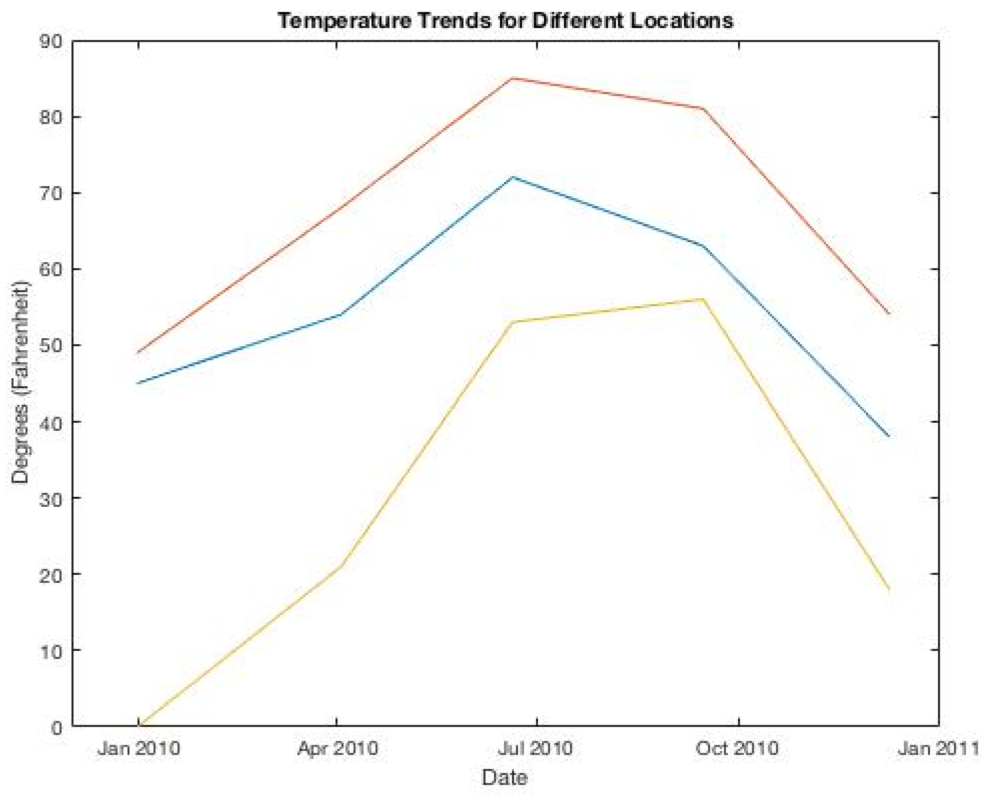

Figure 6: Graph of the truncated geodesic dome.

4.4 Writing to a binary file and reading from a binary file

Write a nine-element vector to a sample file, `nine.bin`.

```
fileID = fopen('nine.bin','w');
fwrite(fileID,[1:9]);
fclose(fileID);
```

Now, read the contents of the same file and store it in a local variable `A`.

```
fileID = fopen('nine.bin');
A = fread(fileID)
```

4.5 Reading a graphic image

Read a sample image.

```
A = imread('ngc6543a.jpg');
```

`imread` returns a 650-by-600-by-3 array, `A`. Now, display the image.

```
>> image(A)
```

Use
`>> doc imread`
to learn more about the function. Likewise, you may use the matlab command `imwrite` to write data to an image file.

4.6 Recording and playing a movie in matlab

For recording a set of graphic frames and constructing a movie, you may use `getframe` as shown below immediately after the graphic frame is generated each time in a loop.

```
while n <= 100
    .......
    .......
    plot(x,y,'r--');
    F(n) = getframe;
    n=n+1;
end
```

In order to make a movie and play it twice, use the following.

```
 >> figure, movie(F,2);
```

5 Final remarks

Finally, extensively use `>> doc doc` and `>> help ⟨function name⟩` to explore different functionalities available in matlab. Also visit the mathworks online help center at
`https://in.mathworks.com/help/`
in order to learn how to use matlab more effectively.

All the best. Have fun learning!

Amrik Sen
Autumn, 2019.

□

Least Squares Regression Analysis for Predicting Energy Consumption

Objective of the experiment: To forecast statistical data using regression analysis by using the method of least squares.

Learning concepts: Linear and nonlinear regression analysis, method of least squares, forecasting and data analytics.

Theoretical Concepts

I. **Introduction & overview**: Given a set of nodes (independent variables) `xdata` and corresponding data values (dependent variables) `ydata`, our objective is to find a *curve of best fit* that suitably predicts data values at points not specified in `xdata`. Let $\{x_i\}_{i=1}^n$ be the nodes in `xdata` and $\{y_i\}_{i=1}^n = \{y(x_i)\}_{i=1}^n$ be the corresponding data values in `ydata`, and let us suppose we want to find a curve of best fit of the form $y = a + bf(x) + cg(x)$ that most suitably describes the database (x_i, y_i); then one way to accomplish this is by using the method of least squares. Here a, b, c are constants that parametrize the model and f and g are some functions that we may choose to our liking depending on the dataset and the complexity of the model we wish to design. The method of least squares involves minimizing an objective function (known as the residual, r) with respect to the constant parameters a, b, c, i.e. $\min\limits_{a,b,c} e(a,b,c) := \min\limits_{a,b,c} r^2(a,b,c) := \min\limits_{a,b,c} \sum\limits_{i=1}^n \big(y_i - (a + bf(x_i) + cg(x_i))\big)^2$. This demands $\frac{\partial e}{\partial a} = 0$, $\frac{\partial e}{\partial b} = 0$ and $\frac{\partial e}{\partial c} = 0$ from which a, b, c may be estimated. Note that even though we wish to minimize the residual r, in essence this turns out to be the same as minimizing $e = r^2$ and hence follows the name- *least squares*.

The constraints $\frac{\partial e}{\partial a} = 0$, $\frac{\partial e}{\partial b} = 0$ and $\frac{\partial e}{\partial c} = 0$, upon simplification, reduces to

$$\underbrace{\begin{pmatrix} \sum_{i=1}^n 1 & \sum_{i=1}^n f(x_i) & \sum_{i=1}^n g(x_i) \\ \sum_{i=1}^n f(x_i) & \sum_{i=1}^n f^2(x_i) & \sum_{i=1}^n f(x_i)g(x_i) \\ \sum_{i=1}^n g(x_i) & \sum_{i=1}^n f(x_i)g(x_i) & \sum_{i=1}^n g^2(x_i) \end{pmatrix}}_{\mathbf{\Lambda}} \underbrace{\begin{pmatrix} a \\ b \\ c \end{pmatrix}}_{\alpha} = \underbrace{\begin{pmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n y_i f(x_i) \\ \sum_{i=1}^n y_i g(x_i) \end{pmatrix}}_{\chi}, \tag{1}$$

whence the regression parameters are given by $\boxed{\alpha = \mathbf{\Lambda}^{-1}\chi}$ as long as the matrix $\mathbf{\Lambda}$ is invertible.

Software Implementation

II. **Least squares regression algorithm**:

II.1. **Importing and reading data**: The file EnergyConsumptionMP_1996-2018.csv contains biennial data for energy consumption in the state of Madhya Pradesh starting from 1996 through 2018. The file has two columns, the first lists the year and the second enumerates the total energy consumption in that year in MU (million units). Use the matlab function `readtable` to import the data into the independent variable named `xdata` and the dependent variable `ydata`. Then use the matlab function `plot` to map the energy consumption of the respective years in a scatter plot.

II.2. **Constructing the Λ matrix, the χ vector and computing regression parameters**: Consider the curve of best fit to be of the form $y = a + bx + cx^2$, i.e. $f(x) = x$ and $g(x) = x^2$. Use the matlab function handles to define $f(x)$ and $g(x)$ as follows: `f = @(x)x` and `g = @(x)x.^2`. Then fill up the entries of $\mathbf{\Lambda}$ and χ with the help of matlab functions `sum` and `length`. Compute α by using the matlab function `inv` and eq (1).

II.3. **Plotting regression curve**: Overlay your curve of best fit (regression curve), using the regression parameters we have estimated in the previous step and for all years from 1996 through 2018, on the figure of II.1 using a different color. This is your *nonlinear* regression model using the method of least squares. If we choose a *line of best fit* of the form $y = a + bx$, and repeat the above steps, we would end up building a linear regression model instead. Thus the method of least squares can be used to build both linear and nonlinear regression models each of which are linear in the regression parameters.

II.4. **Plotting regression curve using matlab function `lsqcurvefit`**: Explore the matlab inbuilt function `lsqcurvefit` by typing `doc lsqcurvefit` on the command line. Re-plot the curve of best fit for the same data and the same model $y = a+bx+cx^2$ using the matlab function `lsqcurvefit` and a suitable initial guess for the regression parameters. Also check out a very similar matlab function `lsqnonlin`.

Questions

Answer the following questions.

1. Derive the matrix form of the equation for the least squares regression parameters given by eq (1).
2. Is the regression matrix $\mathbf{\Lambda}$ always invertible? If so, why? If not, explain the conditions when it is not invertible.
3. Implement the algorithm described in the above section in matlab. Compare the performance of your regression model with that of the matlab function `lsqcurvefit`. For II.4, use initial guess for the regression parameters to be $1.0e+08 * [1,\ -1.5,\ 3]$. Repeat with an initial guess of $[1,\ 1,\ 1]$. Comment on your observations.
4. Implement a linear regression model of the form $y = a + bx$ and plot your predictions on a separate figure.
5. Use this model to predict the total energy consumption in the state of Madhya Pradesh for the years 1999, 2007 and 2021.

□

Fundamental Concepts of Markov Processes with Applications in Weather Prediction and Random Walks

Objective of the experiment: To illustrate elementary concepts in modeling and simulation of Markov processes.

Learning concepts: State space, Markov property, probability transition matrix, random number generation, random walks, stochastic simulations.

Theoretical Concepts

I. **Introduction to Markov chains**: A stochastic process $\{X_n\}_{n\geq 0,\ n\in Z^+}$ has Markov property if

$$\text{Prob}(X_{n+1}=j\big|X_n=i, X_{n-1}=i_{n-1},...,X_0=i_0)=\text{Prob}(X_{n+1}=j\big|X_n=i)=:p_{ij}.$$

Further, a Markov process is entirely characterized by the initial probability distribution of states $\mu^{(0)}$, and the probability transition matrix $\mathbb{P}$. The superscript within parenthesis denotes the time stamp of the process with (0) representing the initial state. The probability distribution of states at the instant n (i.e. after n steps) is then given by $\mu^{(n)}=\mu^{(0)}\mathbb{P}^n$. Moreover, the *long time* (or *equilibrium*) distribution of states is prescribed by $\mu_{eq}=\lim_{n\to\infty}\mu^{(n)}$. In the first part of this laboratory experiment, we will devise a simple weather prediction model based on the Markov property and the above mentioned ideas.

II. **Random walks**: A random walk is a stochastic process where the state of the system at any given instant moves to one of two possible states with equal probability. The state of the system undergoes a slow gradual drift from the initial state and is phenomenologically related to *Brownian motion* which has diverse applications in the natural sciences, economics and engineering. The term random walk was first introduced by Karl Pearson in 1905.

In the second part of this laboratory experiment, we will investigate the behavior of a random walk by considering the fate of a squirrel, hopping to the left or right with equal probability at every instant of time, on a one-dimensional island that abruptly drops to pits (cliffs) on either ends.

Part I: Weather Prediction

Consider that the weather on a given day stays the same as the previous day 75% of the time and changes 25% of the time. For simplicity, let us only consider two weather states: viz., *sunny* and *rainy*. This weather model is a Markov process.

Software Implementation

III. **Weather prediction algorithm**:

- define the initial probability distribution $\mu^{(0)}$
- construct $\mathbb{P}$
- choose appropriate n as per model requirements
- $\mu^{(n)}=\mu^{(0)}\mathbb{P}^n$

- choose large n (say N)
- compute $\mu_{eq} = \mu^{(N)}$

Part II: Random walk on a lonely island

Consider a squirrel on a lonely one-dimensional island. At every instant of time, the squirrel makes a jump to the left or to the right with equal probability. His decision on the direction of his jump at any given instant is independent of all his previous such decisions; after all, he is a *happy go merry* squirrel ☺. Let us say the one-dimensional island can be explored on an n-point discrete lattice and that the squirrel commences his exploration at the position m units from the left (indexed 0 in our convention). This is a classic random walk model. Our objective is to investigate the following using the algorithm provided in the next section:

1. what is the probability that the squirrel eventually drifts to either end of the island and perishes?
2. what is his life expectancy in terms of expected number of steps to death?

Software Implementation

IV. **Random walk algorithm**:

Pseudocode of the random walk algorithm:

```
INPUT: grid_length, start_pos.

initialise curr_pos = start_pos;
initialise num_hops = 1;
while (curr_pos > 0 && curr_pos < grid_length)
   toss = rand(1);
   if (toss < 0.5)
      curr_pos = curr_pos - 1;
   elseif (toss >= 0.5)
      curr_pos = curr_pos + 1;
   end
   plot curr_pos and record graphic frame;
   num_hops = num_hops + 1;
end

OUTPUT: num_hops, play recorded animation.
```

Some useful matlab commands: `rand, stem, getframe, movie.`

Questions

Use the results of your code to answer the following questions.

Part I: Weather prediction

1. Starting with the current day's weather as sunny, what is the probability that it will rain day after tomorrow.
2. Starting with the same initial condition as above, what is the likely weather pattern after 100 days?

Part II: Random walk on a lonely island

1. Consider an island defined by a one-dimensional grid of length 100 units and the initial position of the squirrel is 40 units from the left (tagged 0), create a simulation of the squirrel hooping on the island over time.
2. Does the squirrel eventually fall of and die or does he just bounces on and off on the island in a never ending fashion? Play your simulation and justify your answer.
3. Does your answer above depend on the size of the island or the initial position of the squirrel? Repeat your experiment with different grid size and initial position to justify your answer.
4. What is the life expectancy of the squirrel? Does your answer tally with the theoretically predicted expected number of steps or is there a discrepancy? Explain why?

$\square$

Construction of a Markovian Model using the Viterbi Algorithm to predict Aerodynamic Control Laws of an Aircraft

Objective of the experiment: To build a computational stochastic model based on Markov chains to predict the most likely sequence of events using the Viterbi algorithm.

Learning concepts: Conditional probability, Markov property, stochastic optimizaion, dynamic programming.

Theoretical Concepts

I. **Introduction & overview**: We will consider a certain stochastic process with the following state space of dimension K, $S = \{s_1, s_2, ..., s_K\}$. Associated with this process is a T dimensional observation set $\mathbf{Y} = \{y_1, y_2, ..., y_T\}$ from amongst a possible N dimensional observation space $O = \{o_1, o_2, ..., o_N\}$. Note: $y_n \in O$. Further, consider an initial probability distribution given by $\mathbf{\Pi} = \{\pi_1, \pi_2, ..., \pi_K\}$. The probability transition matrix $\mathbb{P}$ is a $K \times K$ matrix with entries

$$p_{ij}(t) := \text{probability of transitioning from state } s_i \text{ to state } s_j = Prob(x_t = s_j \big| x_{t-1} = s_i),$$

and the emission matrix $\mathbb{E}$ is a $K \times N$ matrix with entries

$$e_{ij}(t) := \text{probability of observing } o_j \text{ from state } s_i = Prob(y_t = o_j \big| x_{t-1} = s_i).$$

Succinctly, we will often write $s_i \equiv i$ and $o_j \equiv j$ where it must be understood that $x_t = i$ refers to the random variable x_t taking the state s_i and $y_t = j$ refers to the random variable y_t being assigned the observable o_j. The goal of the prediction algorithm is to forecast the most likely sequence of states (events) $\mathbf{X} = \{x_1, x_2, ..., x_T\}, x_n \in S$ given a prescribed sequence of observables $\mathbf{Y}$, i.e. we need to compute

$$\text{argmax}_{\mathbf{X}} Prob(\mathbf{X} \big| \mathbf{Y}) = \text{argmax}_{\mathbf{X}} Prob(\mathbf{Y} \big| \mathbf{X}) Prob(\mathbf{X}) = \text{argmax}_{\mathbf{X}} Prob(\mathbf{Y}, \mathbf{X}).$$

Here argmax$\big(f(x)\big)$ returns the value of x at which the function $f(x)$ attains its maximum.

For convenience, you may think of a state space $S = \{$rainy, cloudy, sunny$\}$, an observational space $O = \{$walk, shop, clean$\}$ and a sequence of observations of activity patterns of Billoo, the handyman as $Y = \{$walk, walk, shop, walk, clean, walk, shop$\}$. The objective here is to find the most likely sequence of (hidden) states X corresponding to the sequence of observables Y. E.g., one possible likely outcome may be $X = \{$sunny, sunny, cloudy, sunny, rainy, sunny, cloudy$\}$. In this experiment, we will implement the Viterbi algorithm to predict the most likely sequence of states that corresponds to a sequence of associated observables assuming a Markovian stochastic model (also known as the *Hidden Markov Model* (HMM)).

II. **Construction and essential calculations of the Viterbi algorithm**:

In what follows, we will fix the notation $Prob(X_1 = x_1) \equiv Prob(x_1) \equiv \pi_1$. Note that if $T = 2$, then

$$\begin{aligned} Prob(\mathbf{Y}, \mathbf{X}) &\equiv Prob(y_1, y_2, x_1, x_2) \\ &= Prob(y_1, y_2, x_2 \big| x_1) Prob(x_1) \\ &= Prob(y_1, y_2 \big| x_2, x_1) Prob(x_2 \big| x_1) Prob(x_1) \\ &= Prob(y_1 \big| y_2, x_2, x_1) Prob(y_2 \big| x_2, x_1) p_{12} \pi_1 \\ &= Prob(y_1 \big| x_1, x_2, y_2) Prob(y_2 \big| x_2) p_{12} \pi_1 \\ &= Prob(y_1 \big| x_1) Prob(y_2 \big| x_2) p_{12} \pi_1 \end{aligned} \tag{1}$$

In general, we have

$$\begin{aligned} Prob(\mathbf{Y}, \mathbf{X}) &\equiv Prob(\mathbf{Y} = y_1, ..., y_T, \mathbf{X} = x_1, ..., x_T) \\ &= \underbrace{Prob(x_1)}_{\pi_1} Prob(y_1 \big| x_1) \underbrace{Prob(x_2 \big| x_1)}_{p_{12}} Prob(y_2 \big| x_2) \cdots Prob(y_T \big| x_T) \end{aligned} \tag{2}$$

The Viterbi algorithm involves *recursively* computing the Viterbi entries $V_{k,t}$

$$\begin{aligned} V_{k,t} &:= \max Prob\big((y_1, ..., y_t), (x_1, ..., x_t = k)\big) \\ &= \text{probability of the best (most likely) sequence of states (ending with state } k\text{, i.e. } x_t = k) \\ &\quad \text{corresponding to the sequence of observables } (y_1, ..., y_t). \end{aligned}$$

II.1. **Recursive computation of $V_{k,t}$**:

By comparing the terms on the right hand side of eq. (2) and the definition of the Viterbi entries above, we see that $V_{k,t}$ can be obtained recursively and consequently using the argmax function, we can find the most likely sequence of events. The algorithm includes calculation of the following three important terms.

- $V_{k,t} = \max\limits_{\alpha \in S}\big(Prob(y_t = j \big| x_t = k) p_{\alpha k} V_{\alpha, t-1}\big) = \max\limits_{\alpha \in S}\big(e_{kj} p_{\alpha k} V_{\alpha, t-1}\big)$
  with $V_{k,1} \overset{set}{=} Prob(y_1 = o_m \big| x_1 = k)\pi_k = e_{km}\pi_k$, and
- $x_T = \underset{\alpha \in S}{\text{argmax}}\big(V_{\alpha,T}\big)$.
- x_{t-1} = back_pointer(x_t, t) = value of x used to compute $V_{k,t}$ $\forall t > 1$.

Software Implementation

Pseudocode of the Viterbi algorithm:

```
INPUT: $S, \mathbf{\Pi}, \mathbb{E}, \mathbb{P}, \mathbf{Y} = \{y_1, y_2, ..., y_T\}$.

Part I: Initialization.
for each i of K states
   viterbi_prob(i,1) = $\pi_i * e_{iy_1}$
   viterbi_path(i,1) = $0$
end for

Part II: Compute Viterbi probabilities and Viterbi path.
for each j of T-1 observations starting with T=2
   for each i of K states
      viterbi_prob(i,j) = $\max_{\alpha\in S}\big(e_{iy_j} * p_{\alpha i}*$viterbi_prob$(\alpha, j-1)\big)$
      viterbi_path(i,j) = argmax$_{\alpha\in S}\big(e_{iy_j} * p_{\alpha i}*$viterbi_prob$(\alpha, j-1)\big)$
   end for
end for
$x_T = s_{z_T}$ where $z_T :=$ argmax$_{\alpha\in S}\big($viterbi_prob$(\alpha, T)\big)$
```

The appearance of e_{ij} in the computation of `viterbi_path(i,j)` is unnecessary because it is non-negative and independent of α (so you may choose to skip it).

```
Part III: Retracking the most likely path X.
for each j of T-1 observations from T to 2
   $x_{j-1} = s_{z_{j-1}}$ where $z_{j-1} =$viterbi_path$(z_j, j)$
end for

OUTPUT: $X = \{x_1, x_2, ..., x_T\}$
```

III. **Questions**: Implement the above algorithm in MATLAB and use your program to answer the following questions.

1. Consider there are only two specific types of weather states, viz., *rainy*, *sunny*. Our friend Billoo, the handyman decides to either go *walking*, *shopping* or undertake *cleaning* depending on the type of weather on a given day. Let us say that we have recorded his daily chores over the past five days and observed that he undertook the following sequence of activities on subsequent days: *walking*, *walking*, *shopping*, *walking*, *cleaning*. Use the Markovian model explained above to predict the weather for the last five days. Assume the initial weather distri-

bution $\mathbf{\Pi} = \{0.43, 0.57\}$, the probability transition matrix $\mathbb{P} = \begin{pmatrix} 0.2 & 0.8 \\ 0.4 & 0.6 \end{pmatrix}$, where state 1 is *rainy* and state 2 is *sunny*, and the probability emission matrix $\mathbb{E} = \begin{pmatrix} 0.2 & 0.4 & 0.4 \\ 0.3 & 0.25 & 0.45 \end{pmatrix}$, where the columns (observations) are labelled in order of *walking*, *shopping* and *cleaning*, respectively.

2. Aircraft sensor data from the Airbus A330 is used to predict the flight characteristics and accordingly modify control inputs. One such flight characteristic is pitch up and pitch down motions (observables) measured by the angle of attack sensors. Any error in the pitch measurements may inadvertently affect the primary flight control laws (state variables) and have major consequences in the aerodynamic performance of the plane. In any aircraft there are three primary control laws, viz., *normal*, *alternate* and *direct*, each of which demand distinct inputs by the pilot and the on-board flight computer system. The flight envelope and failure protection modes are also distinctly different depending on the type of control law governing the flight at any given instant, e.g., normal law may have automated low-speed anti-stall protection whereas the same may not be available while the aircraft is operated under direct law. Therefore, accurate real-time prediction of the prevailing control law is essential for continuing safe flight and is monitored carefully by the company at the Airbus engineering systems headquarters. At a certain time, the company receives the following sequence of pitch measurements at 5 minute intervals. Devise a model using the Viterbi algorithm to predict the corresponding sequence of control laws that will likely be activated during the same time instant.

   Pitch data: ‘up’, ‘down’, ‘down’, ‘down’, ‘down’, ‘up’, ‘up’, ‘down’, ‘down’, ‘down’, ‘down’.

   Consider the following probability transition matrix $\mathbb{P}$ and probability emission matrix $\mathbb{E}$ that is available from the Airbus database.

$$\mathbb{P} = \begin{pmatrix} 0.7 & 0.1 & 0.2 \\ 0.4 & 0.5 & 0.1 \\ 0.2 & 0.3 & 0.5 \end{pmatrix}, \mathbb{E} = \begin{pmatrix} 0.6 & 0.4 \\ 0.3 & 0.7 \\ 0.2 & 0.8 \end{pmatrix} \text{ and } \mathbf{\Pi} = \{0.8,\ 0.1,\ 0.1\}.$$

$\square$

Analyzing Statistical Differences in Multi-Population Means using ANOVA to prioritize Post Disaster Reconstruction Projects

Objective of the experiment: To investigate if there is any statistically significant difference in the mean response scores of multiple construction managers across six different cities on issues relating to the effectiveness of Post Disaster Reconstruction (PDR) projects.

Learning concepts: Analysis of Variance (ANOVA), difference of means, F distribution, decision analytics for Post Disaster Reconstruction (PDR) projects.

Theoretical Concepts

I. **Introduction & overview**: The goal of this laboratory experiment is to highlight the issues and challenges in Post Disaster Reconstruction (PDR) projects and to determine the significant differences between the issues and challenges in different locations where PDR projects are carried out. As a chief construction engineer of an international non-governmental organization, you are tasked with devising an emergency strategy for tackling the issues concerning the efficient implementation of PDR projects. Your decision making relies on an extensive database across six international cities[1] where project engineers have rated the most pressing issues that are responsible for delay in PDR projects. Your first task (and the objective of this laboratory experiment) is to identify those issues that are common across geographical locations and address them as a priority. In order to accomplish this, you are provided with a database of responses by construction engineers from six different international cities. Construction engineers who have worked in those cities in the past have rated the significance of the respective issues on a scale of 1-10 with 1 being *strongly disagree* and 10 being *strongly agree*. The issues that will be investigated are:

- *shortage of technical staff*,
- *land ownership and related laws*,
- *funding and aid for PDR projects*, and
- *community participation in rebuilding efforts*.

The ratings of the engineers on each of these challenges across cities are tabulated in the following spreadsheet files:

```
PostDisasterReconstruction_ShortTechStaff.csv
PostDisasterReconstruction_LandOwnership.csv
PostDisasterReconstruction_Funding.csv
PostDisasterReconstruction_CommunityParticipation.csv
```

II. **One-way ANOVA, difference of means and universal relevance of PDR issues**:

Null hypothesis, $H_0 : \mu_1 = \mu_2 = \cdots = \mu_6$
Alternate hypothesis, H_1 : not all means are equal.

The one-way analysis of variance (ANOVA) is used here to determine whether there are any statistically significant differences between the mean rating score of the engineers stationed at six different cities that constitute six independent (and unrelated) groups. In each group (city), there are six different

[1]Port-au-Prince (Haiti), Tacloban City (Philippines), Latur (India), New Orleans (USA), Kathmandu (Nepal) and Bagh City (Pakistan).

observations from six construction engineers who have been involved in PDR projects over the last many years. For each of the above mentioned issues, you have to perform a one-way ANOVA calculation and test if the data provided in the tables (in the spreadsheets) presents a statistical difference in the mean rating of the construction engineers between the six different cities. This will reveal if the issues plaguing the implementation of PDR projects is affected in an identical manner across the six different cities around the world.

Once the issues that are universally relevant have been identified, then the total mean rating score across all cities for a given issue should be computed and a decision on necessary corrective measure should be taken if this grand mean is greater than 5 (on a scale of 10).

Sample data table: The entries in the spreadsheets are similar to the table shown below.

Rating (scale: 1-10, 1: strongly disagree, 10: strongly agree)						
Cities	Mgr1	Mgr2	Mgr3	Mgr4	Mgr5	Mgr6
Port-au-Prince (Haiti)	$y_{11}=3$	$y_{12}=2$	$y_{13}=9$	$y_{14}=8$	$y_{15}=9$	$y_{16}=9$
Tacloban City	$y_{21}=5$	$y_{22}=9$	$y_{23}=10$	$y_{24}=5$	$y_{25}=8$	$y_{26}=9$
Latur	$y_{31}=6$	$y_{32}=7$	$y_{33}=10$	$y_{34}=5$	$y_{35}=7$	$y_{36}=8$
New Orleans	$y_{41}=8$	$y_{42}=9$	$y_{43}=9$	$y_{44}=8$	$y_{45}=2$	$y_{46}=8$
Kathmandu	$y_{51}=3$	$y_{52}=8$	$y_{53}=7$	$y_{54}=10$	$y_{55}=10$	$y_{56}=4$
Bagh City	$y_{61}=2$	$y_{62}=7$	$y_{63}=9$	$y_{64}=10$	$y_{65}=6$	$y_{66}=7$

ANOVA table:

source	degree of freedom	sum of sqs.	mean of sqs.	F_{cal}
between groups	$dfB = t-1$	SSB	MSB	$F_{cal} = \frac{MSB}{MSW}$
within groups	$dfW = \sum_i n_i - t$	SSW	MSW	
total	$\sum_i n_i - 1 = n - 1$	TSS = SSB+SSW		

Here number of groups = $t = 6$ and number of observations in group i = $n_i = 6$ and $n = 36$.

Software Implementation

Use matlab to construct the one-way ANOVA table and implement the following algorithm.

⇒ Compute the entries of the ANOVA table. One-way ANOVA computation involves the following calculations:

1. Compute $Y_{i.} = \sum_{j=1}^{n_i} y_{ij}, Y_{..} = \sum_{i,j} y_{ij}$.
2. Compute $SSB = \sum_i \frac{Y_{i.}^2}{n_i} - \frac{Y_{..}^2}{\sum_i n_i}, \;\; SSW = \sum_{i,j} y_{ij}^2 - \sum_i \frac{Y_{i.}^2}{n_i}, \;\; TSS = SSB + SSW.$
3. Set $MSB = \frac{SSB}{t-1}$ and $MSW = \frac{SSW}{\sum_i n_1 - t}$.
4. Compute $F_{cal} = \frac{MSB}{MSW}$.

⇒ Next, compare F_{cal} and $F_{tab} = F_\alpha(dfB, dfW)$ (F_{tab} is found from $F-$distribution table).

⇒ If $F_{tab} > F_{cal}$, then fail to reject H_0 (i.e. possibly all means (mean rating values) are *statistically* equal); else if $F_{tab} < F_{cal}$, then reject H_0 (and abandon dwelling on the issue for the time being).

⇒ If $F_{tab} > F_{cal}$ and if $\mu = \frac{\sum_{i,j} y_{ij}}{n} > 5.0$, then the issue is universally relevant across cities and demands corrective measures to successfully implement PDR projects.

Questions

1. State the assumptions of one-way ANOVA. Comment if these assumptions seem reasonable in the context of the PDR data provided here.
2. Implement the algorithm prescribed above in matlab. Specifically, write a matlab script to construct and display the ANOVA table for each of the dataset provided in the .csv files.
3. Compute F_{tab} for the given problem from the $F-$ distribution table corresponding to a level of significance of test $\alpha = 0.01$.
4. Based on the strategy prescribed in the algorithm, select and mention the issues that are universally relevant across different cities and that need immediate redressal.
5. Mention at least two drawbacks of one-way ANOVA test.

$\square$

Autoregressive Model of Time Series Data using the Yule Walker Equations to forecast Employment Growth Statistics

Objective of the experiment: To estimate the order (lag), $\tilde{p}$ of an autoregressive (AR) process and model a given time series data as an AR($\tilde{p}$) process.

Learning concepts: Time series modeling, AR($\tilde{p}$) process, autocorrelation and partial autocorrelation functions, Yule Walker equations.

Theoretical Concepts

I. **Introduction & overview**: Autoregressive models are used for time series forecasting when the underlying process is dependent (linearly) on values from previous time instants but may be offset by random shocks. Consequently, the formal structure of such a process takes the form

$$X_{t+1} = \phi_1^{(p)} X_t + \phi_2^{(p)} X_{t-1} + ... + \phi_p^{(p)} X_{t-p+1} + \epsilon_{t+1}, \quad (1)$$

where the $\phi_s^{(p)}$ are the AR coefficients of a p dimensional process and ϵ_t is Gaussian white noise that prescribes the effects of random shocks. Here, the effect of the linear dependence on previous states is significant up to lag p and truncated thereafter. We will consider a sample time series data that lists total number of jobs filled in a public sector enterprise every month over many years. Our goal is to fit an autoregressive (AR) model to this time series data of length N. This will essentially require us to estimate the order (lag) of the AR process by using the *Yule Walker* equations.

II. **Finding AR coefficients using the Yule Walker equations**:

We begin by multiplying eq (1) with X_t and subsequently compute the expected value, $\langle \cdot \rangle$, of the terms in the resulting equation to obtain the following:

$$\langle X_t X_{t+1} \rangle = \sum_{j=1}^{p} \phi_i^{(p)} \langle X_t X_{t-j+1} \rangle + \cancelto{0}{\langle X_t \epsilon_{t+1} \rangle}. \quad (2)$$

The last term above cancels to zero because the process X_t is independent of and uncorrelated with the random shocks ϵ_t. Upon division by $(N-1)$, the equation reduces to $c_1 = \sum_{j=1}^{p} \phi_j^{(p)} c_{j-1}$ which is the first autocovariance function. Normalization by c_0, gives the first autocorrelation function $r_1 = \sum_{j=1}^{p} \phi_j^{(p)} r_{j-1}$. Likewise, to compute the second autocorrelation function we multiply the terms in eq (1) with X_{t-1} and follow the same steps as above to obtain $r_2 = \sum_{j=1}^{p} \phi_j^{(p)} r_{j-2}$. Similarly, the k^{th} autocorrelation function is $r_k = \sum_{j=1}^{p} \phi_j^{(p)} r_{j-k}$. Putting this all together, we have the *Yule Walker* equations:

$$\begin{aligned}
r_1 &= \phi_1^{(p)} r_0 + \phi_2^{(p)} r_1 + \phi_3^{(p)} r_2 + \cdots + \phi_p^{(p)} r_{p-1}, \\
r_2 &= \phi_1^{(p)} r_1 + \phi_2^{(p)} r_0 + \phi_3^{(p)} r_1 + \cdots + \phi_p^{(p)} r_{p-2}, \\
&\ \ \vdots \\
r_{p-1} &= \phi_1^{(p)} r_{p-2} + \phi_2^{(p)} r_{p-3} + \phi_3^{(p)} r_{p-4} + \cdots + \phi_p^{(p)} r_1, \\
r_p &= \phi_1^{(p)} r_{p-1} + \phi_2^{(p)} r_{p-2} + \phi_3^{(p)} r_{p-3} + \cdots + \phi_p^{(p)} r_0.
\end{aligned}$$

The above set of equations can be written succinctly in matrix form as follows:

$$\mathbf{\Phi} = \mathbf{R}^{-1}\mathbf{r}, \tag{3}$$

where $\mathbf{\Phi}^{(p)} = \begin{pmatrix} \phi_1^{(p)} \\ \phi_2^{(p)} \\ \cdot \\ \cdot \\ \phi_{p-1}^{(p)} \\ \phi_p^{(p)} \end{pmatrix}$, $R = \begin{pmatrix} r_0 & r_1 & r_2 & \cdots & r_{p-2} & r_{p-1} \\ r_1 & r_0 & r_1 & \cdots & r_{p-3} & r_{p-2} \\ & & & \cdot & & \\ & & & \cdot & & \\ r_{p-2} & r_{p-3} & r_{p-4} & \cdots & r_0 & r_1 \\ r_{p-1} & r_{p-2} & r_{p-3} & \cdots & r_1 & r_0 \end{pmatrix}$ and $\mathbf{r} = \begin{pmatrix} r_1 \\ r_2 \\ \cdot \\ \cdot \\ r_{p-1} \\ r_p \end{pmatrix}$.

The matrix $\mathbf{R}$ is always invertible because it is full rank and symmetric and hence the Yule Walker equations are well posed and can be used to solve for the unknown AR coefficients ($\mathbf{\Phi}$). Here the auto-correlation functions (ACFs) are given by $\mathbf{r}$. It must be noted that $r_0 := 1$ and $c_{-k} = c_k$.

III. Computing partial autocorrelation functions and estimating order of AR model:

The correlation between X_t and X_{t+h} comprises of both *direct* and *indirect* dependencies. The indirect dependency between X_t and X_{t+h} arises due to the linear dependency between X_t and X_{t+1}, X_{t+1} and X_{t+2}, X_{t+2} and X_{t+3}, and so on, all the way through X_{t+h-1} and X_{t+h}. The partial auto-correlation functions (PACFs) prescribe the direct dependency between X_t and X_{t+h} with the effects of all the intermediary variables, X_{t+1} through X_{t+h-1}, removed. The PACFs, $\mathbf{\Psi}^{(p)}$ are negative of the last coefficients $\phi_k^{(k)}$s computed using the Yule Walker equations for every lag (order) k starting with 1 through p i.e., $\mathbf{\Psi}^{(p)} = \begin{pmatrix} \psi_1^{(p)} \\ \psi_2^{(p)} \\ \cdot \\ \cdot \\ \psi_{p-1}^{(p)} \\ \psi_p^{(p)} \end{pmatrix} = \begin{pmatrix} -\phi_1^{(1)} \\ -\phi_2^{(2)} \\ \cdot \\ \cdot \\ -\phi_{p-1}^{(p-1)} \\ -\phi_p^{(p)} \end{pmatrix}$. Consequently, the following table prescribes a strategy to estimate the order of various time series models using the ACFs and PACFs. Specifically, for the AR model, the rapid (abrupt) decay of the PACFs prescribes the order of the model.

Model	ACF $\left(r_h\right)$	PACF $\left(\psi_h\right)$
AR($\tilde{p}$)	decays infinitely as $r_h \xrightarrow{h\to\infty} 0$	truncates abruptly as $\psi_h = 0 \quad \forall h > \tilde{p}$
MA($\tilde{q}$)	truncates abruptly as $r_h = 0 \quad \forall h > \tilde{q}$	decays infinitely as $\psi_h \xrightarrow{h\to\infty} 0$
ARMA($\tilde{p}, \tilde{q}$)	decays as AR($\tilde{p}$) $\quad \forall h > \tilde{q}$	decays as MA($\tilde{q}$) $\quad \forall h > \tilde{p}$

Software Implementation

IV. Construction of the AR($\tilde{p}$) model:

The algorithm constitutes the following steps.

IV.1. **Importing data and dimensions of Yule Walker system**: Use the matlab function `readtable` to import the tabular data from the spreadsheet (JOBS.csv). The table has two columns, viz., *Month* and *TotalFilledJobs*. Plot the entries of the second column against the entries in the first column (or simply in sequence) in *blue* and then plot, in the same figure, the mean subtracted entries of the second column. Visually inspect the plotted mean subtracted time series data and check if it is periodic with period $(p+1)$. If so, use a system of p equations to construct the Yule Walker system (prescribed by eq (3)), else use some arbitrarily large enough number (say $N/4$) to construct the Yule Walker system.

IV.2. **Computing AR coefficients using the Yule Walker equations**: Here we will use two strategies:

- First, write your own matlab function, `myArYule` to compute $\mathbf{\Phi}$ as prescribed by eq (3). The input to your function should be the mean subtracted time series data and the dimension of the Yule Walker system selected as above. The output of the function should be the AR coefficients, $\mathbf{\Phi}$ and the partial autocorrelation functions, Ψ.

- Second, use the matlab function `aryule` to compute the AR coefficients and the partial autocorrelation functions. Compare the results of both these strategies for consistency.

IV.3. **Select the order of the AR model using the partial autocorrelation functions**: Practically, the PACF is considered zero (at a 5% significance level) if the value of ψ_h falls within the critical region defined by the upper and lower limits given by $\pm 1.96/\sqrt{N}$.[1] In your software routine, plot the PACFs in conjunction with the critical region set at 5% significance level. Finally, choose the order $\tilde{p}$ of the AR model in such a way that $\psi_h \neq 0$ for $h = \tilde{p}$ but $\psi_h = 0 \ \forall h > \tilde{p}$.

Pseudocode of the `myArYule` algorithm:

```
INPUT: data, max_order.

for current_order from max_order to 1
   initialize $\mathbf{R}$ to identity matrix
   compute ACFs $\mathbf{r}$ from data
   for i from 1 to current_order
      k=1;
      for j from i+1 to current_order
         $\mathbf{R}(i,j) = \mathbf{r}(k)$
         $k = k + 1$
      end
   end
compute remaining entries of $\mathbf{R}$ by using symmetry of $\mathbf{R}$
compute $\mathbf{\Phi}_{\text{temp}} = \mathbf{R}^{-1} * \mathbf{r}$
update $\mathbf{\Phi}_{\text{temp}}$ by concatenating 1 and $\mathbf{\Phi}_{\text{temp}}$
if current_order == max_order
   $\mathbf{\Phi} = \mathbf{\Phi}_{\text{temp}}$
else
   --- Are we having fun ☺ or what? ☹ ---
end
pacf(current_order) = negative of last entry of $\mathbf{\Phi}_{\text{temp}}$
end

OUTPUT: $\mathbf{\Phi}$, pacf.
```

[1]This approximation relies on the assumption that $N > 30$ and that the underlying process has finite second moment (finite variance).

Some useful matlab functions you may want to consider are: `xcorr, eye, inv, aryule, readtable, plot, subplot, xlabel, ylabel, length, stem, sqrt, title, grid, xlim, hold on`. Use the command `doc <function_name>` to study more about them.

Questions

Answer the following questions.

1. Implement the algorithm of sec. IV. outlined above in matlab.
2. Estimate the order $\tilde{p}$ of the AR model for the given time series data in JOBS.csv.
3. Based on the AR coefficients computed by your matlab routine, comment whether the AR($\tilde{p}$) model has a stable solution. Explain your answer in detail.
4. Use this model to forecast the additional jobs filled each month in the year 2012 for months March through December.

□

Multivariate Principal Component Analysis for Image Compression

Objective of the experiment: To investigate the efficacy of image compression using multivariate principal component analysis.

Learning concepts: Covariance matrix, principal component analysis, linear algebra, image and data compression.

Theoretical Concepts

I. **Introduction & overview**: Principal Component Analysis (PCA) is a technique used in multivariate statistics to reduce the dimensions of a given problem and hence provides a way to encode the basic features of a multivariate data (eg. an image) by using a few appropriate variables (dimensions). The reduction in the representation of the dataset to a few important dimensions is carried forth by a suitable linear transformation. Technically, this involves a change of basis in the representation of the given data matrix. PCA de-correlates the original data by finding the directions in which the variance is maximized and then utilizes these directions to find the new basis.

II. **Data matrix, principal components & covariance matrix**: Consider an $m \times n$ data matrix X where the n columns are the samples (observations) and the m rows are the variables (dimensions) such that for each k of the m dimensions, the n data points constitute a (row) vector $\mathbf{\tilde{x}_k}$ with mean zero. That is $\mathbf{\tilde{x}_k} \longleftarrow \left(\mathbf{\tilde{x}_k} - \sum_{i=1}^{n} \frac{\tilde{x}_{k,i}}{n}\right)$. Formally, $X = \begin{pmatrix} | & | & & | \\ \mathbf{x_1} & \mathbf{x_2} & \cdots & \mathbf{x_n} \\ | & | & & | \end{pmatrix} = \begin{pmatrix} \mathbf{\tilde{x}_1} \\ \mathbf{\tilde{x}_2} \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{\tilde{x}_m} \end{pmatrix}$ represents the mean subtracted data where each $\begin{matrix} | \\ \mathbf{x_k} \\ | \end{matrix}$ represents an m dimensional data point (vector) and there are n of them corresponding to n samples, and each of $\mathbf{\tilde{x}_k} = \left(\tilde{x}_{k,1}\ \tilde{x}_{k,2}\ \cdots \tilde{x}_{k,n}\right)$ represents the n sample observations (data points) of the k^{th} dimension. The data is then transformed into a new representation Y (also an $m \times n$ matrix) by a change of basis matrix P of dimensions $m \times m$ as follows:

$$Y = PX = \begin{pmatrix} \mathbf{p_1} \cdot \mathbf{x_1} & \mathbf{p_1} \cdot \mathbf{x_2} & \cdots & \mathbf{p_1} \cdot \mathbf{x_n} \\ \mathbf{p_2} \cdot \mathbf{x_1} & \mathbf{p_2} \cdot \mathbf{x_2} & \cdots & \mathbf{p_2} \cdot \mathbf{x_n} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \mathbf{p_m} \cdot \mathbf{x_1} & \mathbf{p_m} \cdot \mathbf{x_2} & \cdots & \mathbf{p_m} \cdot \mathbf{x_n} \end{pmatrix}. \tag{1}$$

Here $P = \begin{pmatrix} ---\mathbf{p_1}--- \\ ---\mathbf{p_2}--- \\ . \\ . \\ . \\ ---\mathbf{p_m}--- \end{pmatrix}$ where rows of P, represented here as $---\mathbf{p_k}---$, form the new basis. In fact, the rows of P turn out to be the *principal components*. The covariance matrix of the mean subtracted data is

$$C_X = \frac{1}{n-1}XX^T = \frac{1}{n-1}\begin{pmatrix} \mathbf{\tilde{x}_1}\mathbf{\tilde{x}_1}^T & \mathbf{\tilde{x}_1}\mathbf{\tilde{x}_2}^T & \cdots & \mathbf{\tilde{x}_1}\mathbf{\tilde{x}_m}^T \\ \mathbf{\tilde{x}_2}\mathbf{\tilde{x}_1}^T & \mathbf{\tilde{x}_2}\mathbf{\tilde{x}_2}^T & \cdots & \mathbf{\tilde{x}_2}\mathbf{\tilde{x}_m}^T \\ . & . & \cdots & . \\ . & . & \cdots & . \\ . & . & \cdots & . \\ \mathbf{\tilde{x}_m}\mathbf{\tilde{x}_1}^T & \mathbf{\tilde{x}_m}\mathbf{\tilde{x}_2}^T & \cdots & \mathbf{\tilde{x}_m}\mathbf{\tilde{x}_m}^T \end{pmatrix}. \tag{2}$$

By construction, C_X is a symmetric positive definite matrix of dimensions $m \times m$. The off-diagonal entries of a covariance matrix reveal how well correlated are two distinct variables (in this case, how well correlated are two distinct dimensions). If two dimensions are significantly correlated, then perhaps retaining both these dimensions is not necessary as they would provide redundant information. This entails that the optimal representation of the data should be such that the off-diagonal terms of the covariance matrix in that representational space must be zero. Further, this also underscores the importance of retaining those dimensions that lend themselves to exhibit the greatest *variety* of *independent and uncorrelated* information inherent within the data (maximal *variance* presents richer information). Therefore, the diagonal entries of the covariance matrix in the transformed space should be as large as possible, especially for the important dimensions that are to be retained in the reduced (transformed) representation of the data.

III. **Linear transformation of the data and covariance matrix of transformed data**: The objective here is to find a transformation P which transforms the data into an optimal representational space using $Y = PX$. This is done by constructing the covariance matrix of the transformed data C_Y by

1. maximizing the diagonal entries of C_Y (maximizing variance of every dimension), and
2. minimizing the off-diagonal entries of C_Y (minimizing covariance between dimensions).

This means our objective is to find P that makes C_Y diagonal. Also, since the columns of P must form the new basis, we must have that P is an orthonormal matrix ($PP^T = I$).

By definition,

$$C_Y = \frac{1}{n-1}YY^T = \frac{1}{n-1}(PX)(PX)^T = \frac{1}{n-1}(PX)(X^TP^T) = \frac{1}{n-1}P(XX^T)P^T = PSP^T$$

where $S = \frac{1}{n-1}XX^T$ is an $m \times m$ symmetric positive definite matrix. Since every square symmetric matrix is diagonalizable, we have $S = EDE^T = \frac{1}{n-1}XX^T$ where E is an $m \times m$ orthonormal matrix whose columns are the eigenvectors of S and D is a diagonal matrix with the respective eigenvalues along the diagonal. Now, if we choose the rows of P to be the eigenvectors of S whence $P = E^T$, then from above we find that $C_Y = PSP^T = E^T(EDE^T)E = D$ because $E^TE = PP^T = I$ as stated above. Summarizing, the essence of all this is, ***if we choose $P = E^T$ (choose the principal components or rows of P as the eigenvectors of XX^T), then we recover C_Y in the desired diagonal form***.

IV. **Singular Value Decomposition to find eigen-decomposition of $\frac{1}{n-1}XX^T$**: In linear algebra, there is a well known methodology to compute the eigenvalues and eigenvectors of Z^TZ by using the Singular Value Decomposition (SVD) of $Z = U\Sigma V^T$ where $Z := \frac{X^T}{\sqrt{n-1}}$, Σ is diagonal with the singular values of Z, and U, V are orthornormal matrices. The non-zero singular values of Z are the square roots of the nonzero eigenvalues of $Z^TZ = C_X$. So $Z^TZ = (U\Sigma V^T)^T(U\Sigma V^T) = V(\Sigma^T\Sigma)V^T$. So Z^TZ is similar to $\Sigma^T\Sigma$ and it has the same eigenvalues as that of $\Sigma^T\Sigma$. Therefore, ***the non-zero singular values of Z, upon rearrangement in descending order along the diagonal of Σ and upon squaring, give us the eigenvalues of Z^TZ (or equivalently the eigenvalues of $S = \frac{1}{n-1}XX^T$), also in descending order. Further, the columns of V, rearranged in accordance with the respective singular values of Z, are the same as the columns of E (rearranged) and constitute the rows of P which are the principal components. Therefore,*** $\boxed{P = E^T = V^T}$.

V. **Recovery of original data from transformed space**:
Since $Y = PX = V^TX$, $\boxed{X = VY}$ returns the original data X from the transformed data Y.

VI. **Image (data) compression**: Suppose that before projecting the data using the relation, $Y = V^TX$, we were to truncate the matrix, V so that we kept only the first $r < m$ columns. We would thus have a matrix of dimensions $m \times r$. The projection $\widehat{Y} = \widehat{V}^TX$ is the new reduced (transformed) data with dimensions $r \times n$. Suppose that we then wished to transform this data back to the original basis by computing $\widehat{X} = \widehat{V}\widehat{Y}$, we would recover the dimensions of X because $\widehat{X}$ also has dimensions $m \times n$ even though $\widehat{X} \neq X$.

The matrices, X and $\widehat{X}$ are of the same dimensions, but they are not the same matrix, since we truncated the matrix of principal components V in order to obtain $\widehat{X}$. It is therefore reasonable to conclude that the matrix, $\widehat{X}$ has in some sense, *less information* in it than the matrix X. Of course, in terms of memory allocation on a computer, this is certainly not the case since both matrices have the same dimensions and would therefore allot the same amount of memory. However, the matrix, $\widehat{X}$ can be computed as the product of two smaller matrices ($\widehat{V}$ and $\widehat{Y}$). This, together with the fact that the *essential* information in the matrix is captured by the first few principal components suggests a possible method for image compression.

VI.1. **Compression ratio**: Let us suppose we have an image of dimensions 512×512 and we want to reduce it to dimensions 512×40. So $\widehat{V}$ and $\widehat{Y}$ have dimensions 512×40 and 40×512 respectively. In addition to this, we have another vector of size 512×1 that stores the means from the mean-subtraction step. Therefore, we have reduced the number of columns required from 512 to $40 + 40 + 1 = 81$. This gives us a compression ratio of $512 : 81$ or approximately $6.33 : 1$. In the exercises below, you will be asked to investigate different compression ratios and a qualitative assessment of the amount of information retained.

Software Implementation

Pseudocode of image compression by PCA:

```
INPUT: image, numPCs.
    ➤ store image as a matrix    % use matlab command imread
    ➤ find $X$, mean subtracted data matrix    % use matlab commands mean and repmat
    ➤ construct $Z := \frac{1}{\sqrt{n-1}} X^T$    % dimension of $X$ is $m \times n$
    ➤ construct covariance matrix $C_X = Z^T Z$
    ➤ compute SVD of Z: $[U\ S\ V] = \texttt{svd}(Z)$    % $S$ stores singular values of Z
    ➤ compute eigenvalues of $Z^T Z$ by squaring singular values of $Z$
    ➤ store top numPCs eigenvectors: $\widehat{V} = V(:, 1 : \texttt{numPCs})$
    ➤ store transformed reduced data in $\widehat{Y} = \widehat{V}^T X$
    ➤ compute compression ratio: ratio = $\frac{n}{2(\texttt{numPCs})+1}$
    ➤ convert image data in original space: $\widehat{X} = \widehat{V}\widehat{Y}$
    ➤ add the subtracted means to $\widehat{X}$ to construct compressed image
OUTPUT: compressed image, ratio.
```

† Lines in green to the right of % symbol are comments, not executable lines of code.

Questions

Answer the following questions.

1. Use the image in the file `butterfly2.jpg` and apply the image compression algorithm given above to compress the given image by considering the top 40 principal components (PCs). Plot the top 40 eigenvalues as a bar graph. What is the compression ratio?
2. Repeat the above experiment by using only top 5 PCs. What is the compression ratio?
3. What can you conclude qualitatively from the above two experiments?
4. What happens if you replace $[U\ S\ V] = \texttt{svd}(Z)$ in your code by $[U\ S\ V] = \texttt{svd}(C_X)$? Why?

□