LINEAR ALGEBRA ASSIGNMENT

Unit 5

Applications of Linear algebra for Page Rank Algorithm.

a)

Creating a graph that illustrates how each node confers its PageRank score to the other nodes in the graph.

Code

```
s = {'a' 'a' 'a' 'b' 'b' 'c' 'd' 'd' 'd'};
t = {'b' 'c' 'd' 'd' 'a' 'b' 'c' 'a' 'b'};
G = digraph(s,t);
labels = {'a/3' 'a/3' 'a/3' 'b/2' 'b/2' 'c' 'd/3' 'd/3'
'd/3'};
p = plot(G,'Layout','layered','EdgeLabel',labels);
highlight(p,[1 1 1],[2 3 4],'EdgeColor','g')
highlight(p,[2 2],[1 4],'EdgeColor','r')
highlight(p,3,2,'EdgeColor','m')
title('PageRank Score Transfer Between Nodes')
```

output

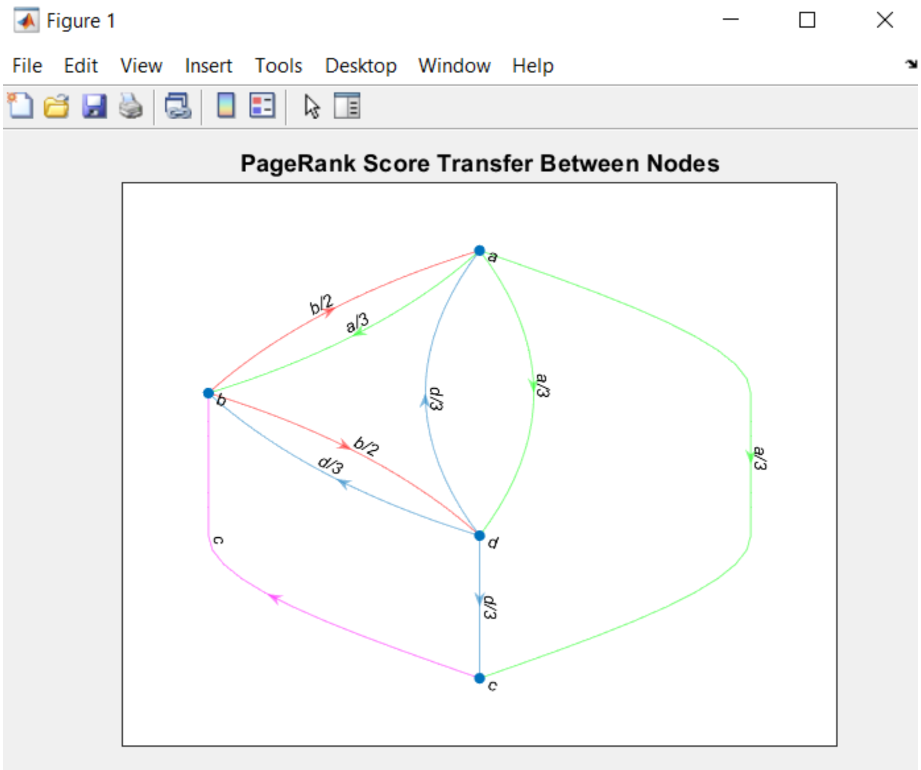


Figure 1
File Edit View Insert Tools Desktop Window Help
PageRank Score Transfer Between Nodes
a
b
c
d
c
b/2
a/3
d/3
a/3
a/3
b/2
d/3
c
d/3

b)

PageRank with 6 Nodes

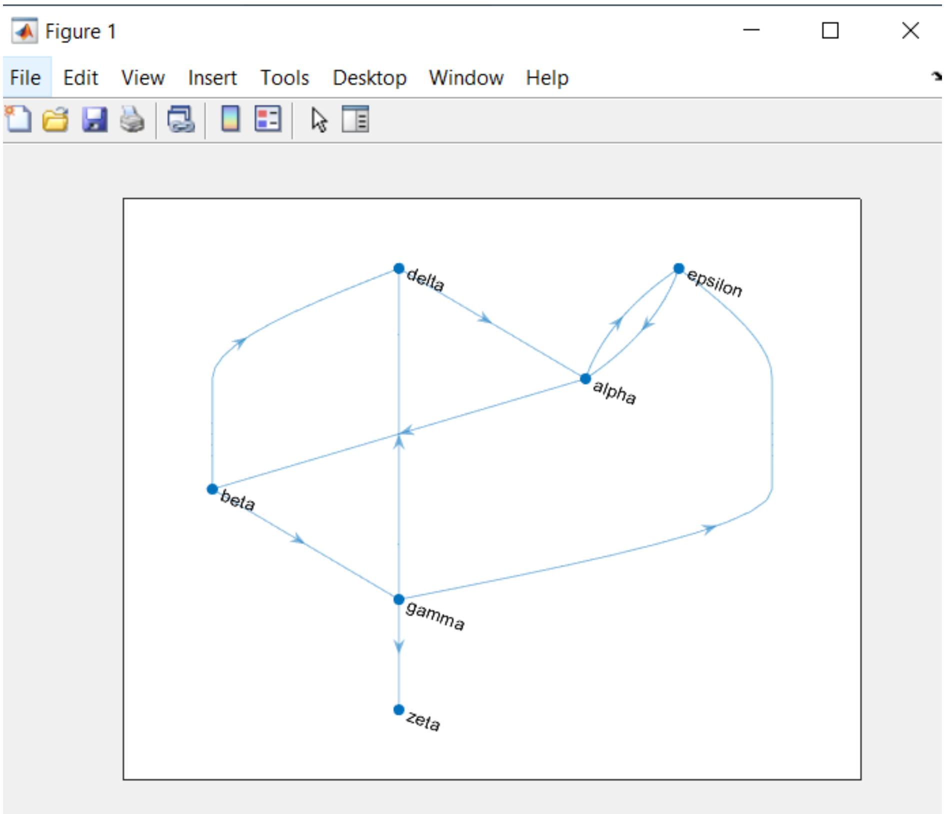
Code

```
s = [1 1 2 2 3 3 3 4 5];
t = [2 5 3 4 4 5 6 1 1];
names = {'http://www.example.com/alpha',
'http://www.example.com/beta', ...
         'http://www.example.com/gamma',
'http://www.example.com/delta', ...
         'http://www.example.com/epsilon',
'http://www.example.com/zeta'};
```

```
G = digraph(s,t,[],names)
plot(G,'Layout','layered', ...

'NodeLabel',{'alpha','beta','gamma','delta','epsilon','zeta'
})
```

output

```
G =

  digraph with properties:

    Edges: [9×1 table]
    Nodes: [6×1 table]
```



➤ Calculating the PageRank centrality score for the above graph

Code

```
pr = centrality(G,'pagerank','FollowProbability',0.85)
```

output

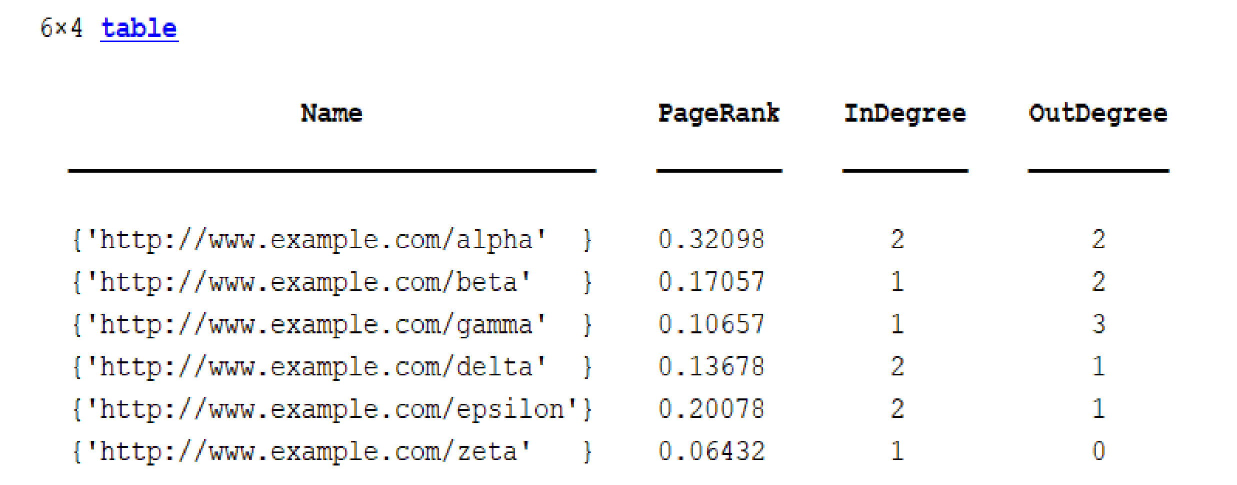
```
pr =


    0.3210
    0.1706
    0.1066
    0.1368
    0.2008
    0.0643
```

➤ View the PageRank scores and degree information for each page.

code

```
G.Nodes.PageRank = pr;
G.Nodes.InDegree = indegree(G);
G.Nodes.OutDegree = outdegree(G);
G.Nodes
```

output



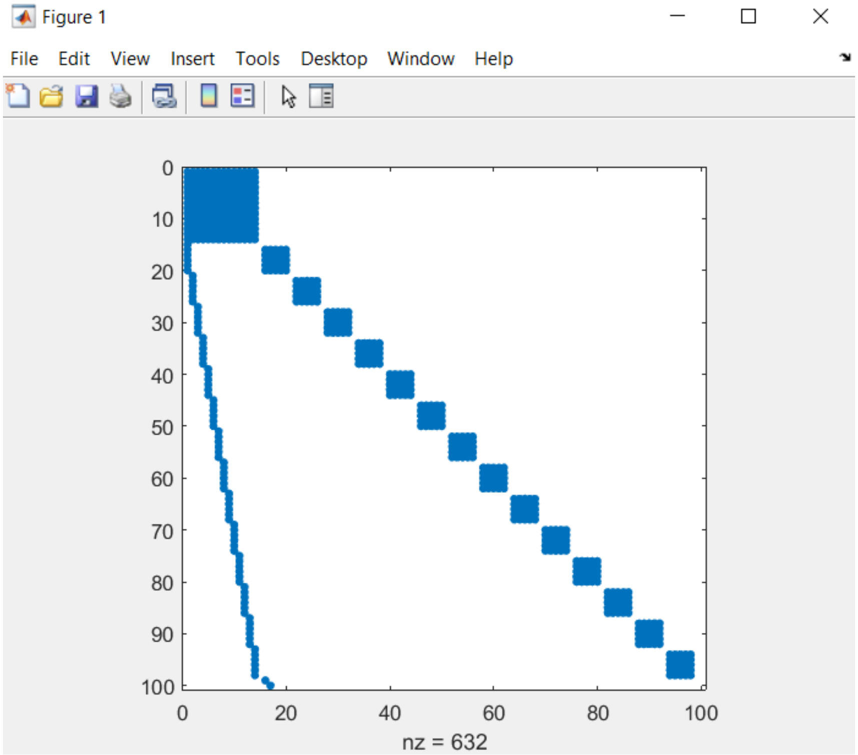
```
  6×4 table

                  Name                   PageRank    InDegree    OutDegree
    _________________________________    ________    ________    _________

    {'http://www.example.com/alpha'  }   0.32098        2            2
    {'http://www.example.com/beta'   }   0.17057        1            2
    {'http://www.example.com/gamma'  }   0.10657        1            3
    {'http://www.example.com/delta'  }   0.13678        2            1
    {'http://www.example.com/epsilon'}   0.20078        2            1
    {'http://www.example.com/zeta'   }   0.06432        1            0
```

c) PageRank of Websites on mathworks.com

Code

```
load mathworks100.mat
spy(A)
```

output

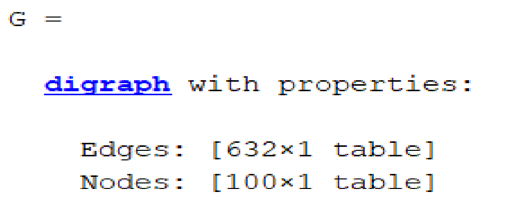


d) Create a directed graph with the sparse adjacency matrix, A, using the URLs contained in U as node names.

Code

```
G = digraph(A,U)
```

output



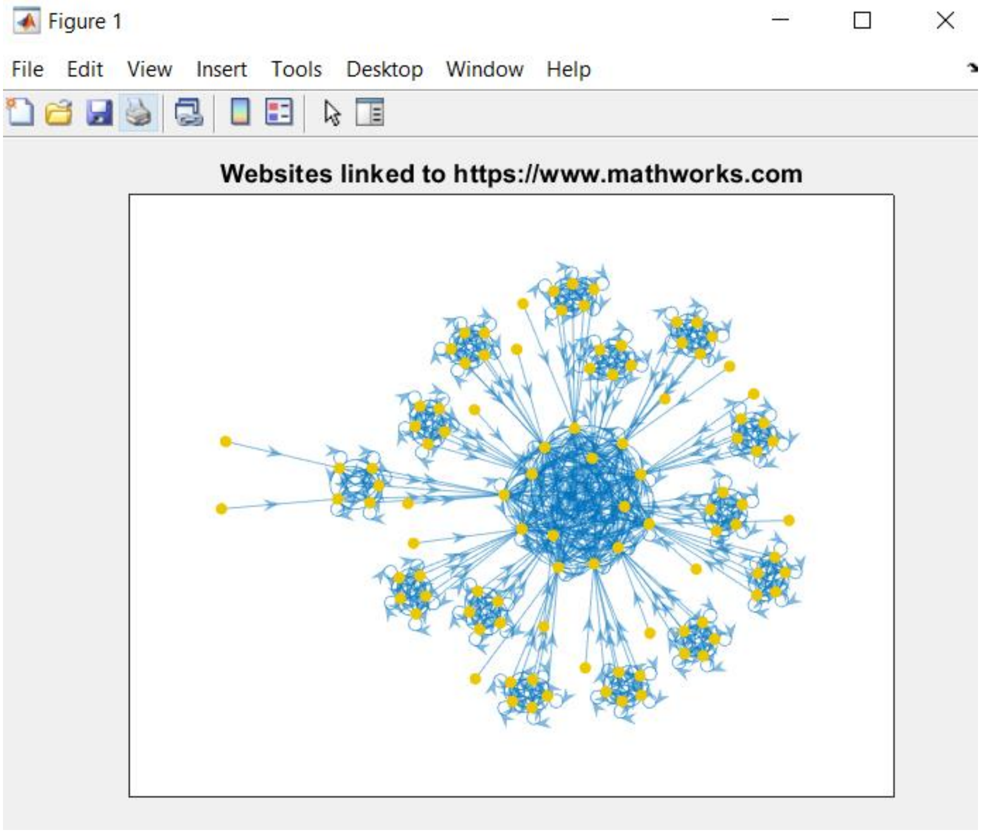
```
G = 

  digraph with properties:

    Edges: [632×1 table]
    Nodes: [100×1 table]
```

➢ Plotting the graph using the force layout.

Code

```
plot(G,'NodeLabel',{},'NodeColor',[0.93 0.78 0],'Layout','force');
title('Websites linked to https://www.mathworks.com')
```

output

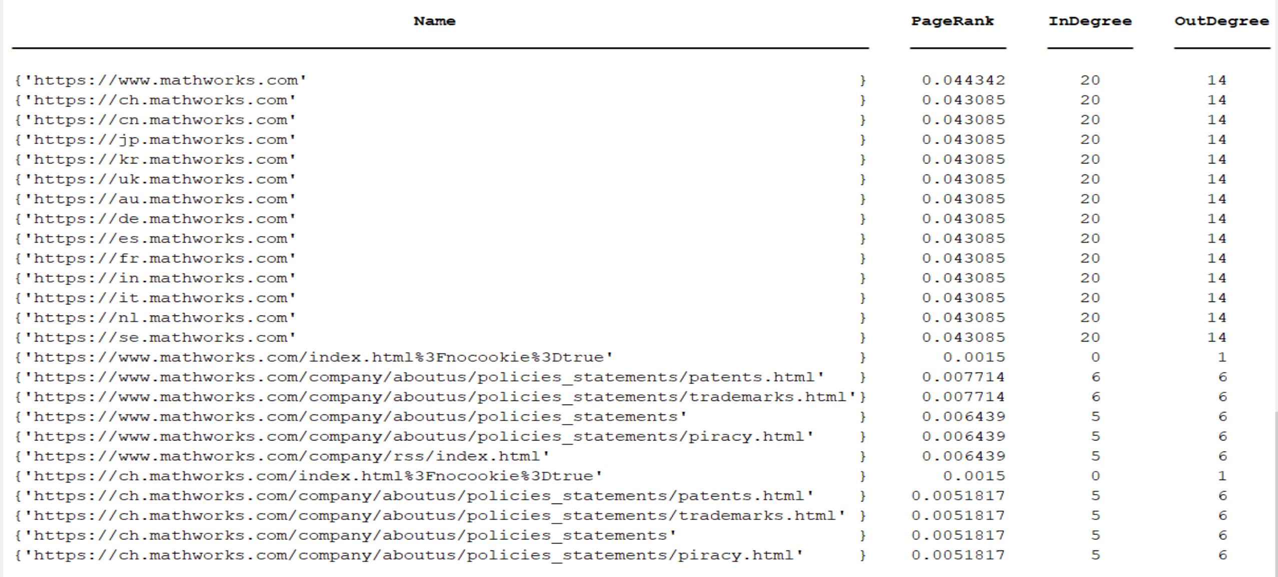


e) Computing the PageRank scores for the graph, G, using 200 iterations and a damping factor of 0.85. Add the scores and degree information to the nodes table of the graph.

Code

```
pr =
centrality(G,'pagerank','MaxIterations',200,'FollowProbabili
ty',0.85);
G.Nodes.PageRank = pr;
G.Nodes.InDegree = indegree(G);
G.Nodes.OutDegree = outdegree(G);
G.Nodes(1:25,:) %Viewing the top 25 resulting scores.
```

output



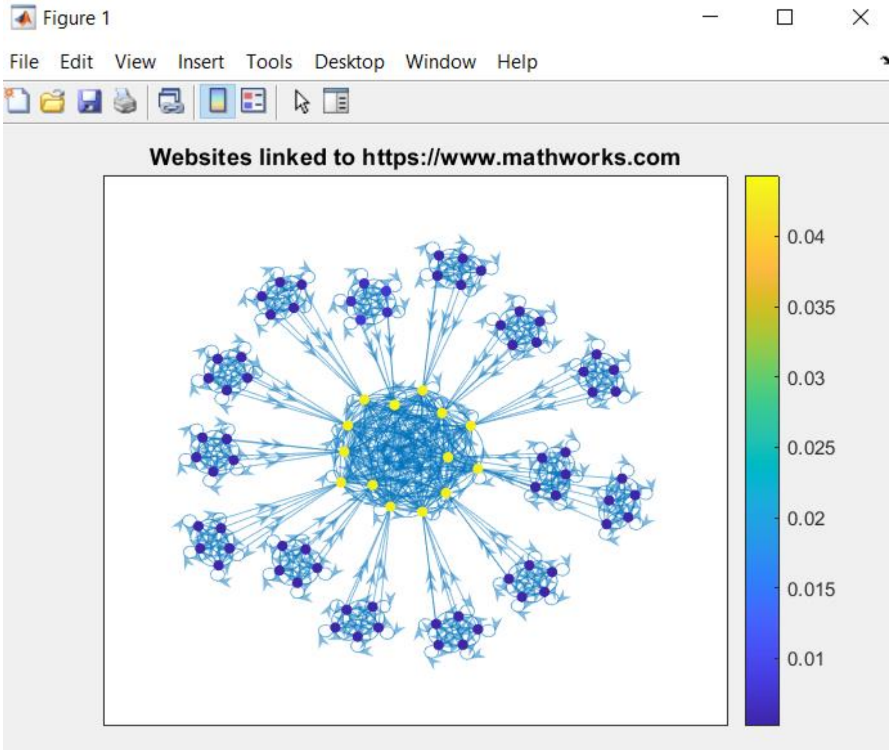
```
                                   Name                                              PageRank    InDegree    OutDegree
    ___________________________________________________________________________    _________    ________    _________

    {'https://www.mathworks.com'                                               }     0.044342       20           14
    {'https://ch.mathworks.com'                                                }     0.043085       20           14
    {'https://cn.mathworks.com'                                                }     0.043085       20           14
    {'https://jp.mathworks.com'                                                }     0.043085       20           14
    {'https://kr.mathworks.com'                                                }     0.043085       20           14
    {'https://uk.mathworks.com'                                                }     0.043085       20           14
    {'https://au.mathworks.com'                                                }     0.043085       20           14
    {'https://de.mathworks.com'                                                }     0.043085       20           14
    {'https://es.mathworks.com'                                                }     0.043085       20           14
    {'https://fr.mathworks.com'                                                }     0.043085       20           14
    {'https://in.mathworks.com'                                                }     0.043085       20           14
    {'https://it.mathworks.com'                                                }     0.043085       20           14
    {'https://nl.mathworks.com'                                                }     0.043085       20           14
    {'https://se.mathworks.com'                                                }     0.043085       20           14
    {'https://www.mathworks.com/index.html%3Fnocookie%3Dtrue'                  }       0.0015        0            1
    {'https://www.mathworks.com/company/aboutus/policies_statements/patents.html'   }     0.007714        6            6
    {'https://www.mathworks.com/company/aboutus/policies_statements/trademarks.html'}     0.007714        6            6
    {'https://www.mathworks.com/company/aboutus/policies_statements'           }     0.006439        5            6
    {'https://www.mathworks.com/company/aboutus/policies_statements/piracy.html'    }     0.006439        5            6
    {'https://www.mathworks.com/company/rss/index.html'                        }     0.006439        5            6
    {'https://ch.mathworks.com/index.html%3Fnocookie%3Dtrue'                   }       0.0015        0            1
    {'https://ch.mathworks.com/company/aboutus/policies_statements/patents.html'    }    0.0051817        5            6
    {'https://ch.mathworks.com/company/aboutus/policies_statements/trademarks.html' }    0.0051817        5            6
    {'https://ch.mathworks.com/company/aboutus/policies_statements'            }    0.0051817        5            6
    {'https://ch.mathworks.com/company/aboutus/policies_statements/piracy.html'     }    0.0051817        5            6
```

➢ Extracting and plot a subgraph containing all nodes whose score is greater than 0.005. Colour the graph nodes based on their PageRank score.

Code

```
H = subgraph(G,find(G.Nodes.PageRank > 0.005));
plot(H,'NodeLabel',{},'NodeCData',H.Nodes.PageRank,'Layout',
'force');
title('Websites linked to https://www.mathworks.com')
colorbar
```

output

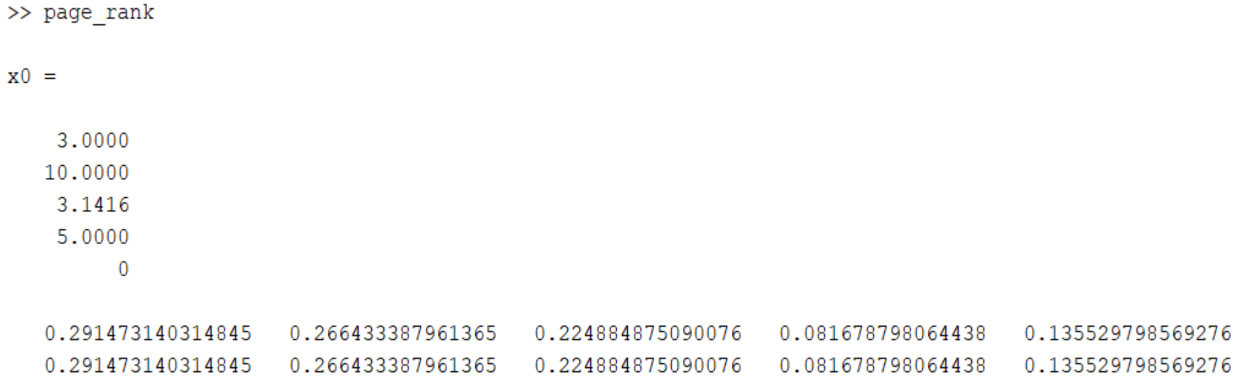


f) **Page rank algorithm**

Code

```
% PageRank algorithm
%% Constructing adjancency matrix A
A = [...
    0  1  0  1  1;
    0  0  1  1  1;
    1  0  0  1  0;
    0  0  0  0  1;
    0  0  1  0  0];

%% Solveing the eigenvalue problem
% Diagonal matrix D contains the eigenvalues of A in the
diagonal.
% The columns of matrix V are the eigenvectors of A
[V,D] = eig(A);

% Eigenvalues of A are in this vector. The first
% one is the only real eigenvalue; that's what we
% need.
evals = diag(D);
eval1 = evals(1);

% Find proportionality constant alpha
alpha = 1/eval1;

% Find ranking vector r as the eigenvector corresponding to
% the first eigenvalue
r = V(:,1);

% Normalize the ranking vector
r = r/sum(r);

% trying the same with the power method!

% First, pick some 5-vector
x0 = [3,10,pi,5,0];
x0 = x0(:) % Make vertical

% Second, iterate!
x = x0;
for iii = 1:500
    x = A*x;
end

% Third, normalize
x = x/sum(x);

% Show the result to compare
format long
disp([r.';x.'])
```

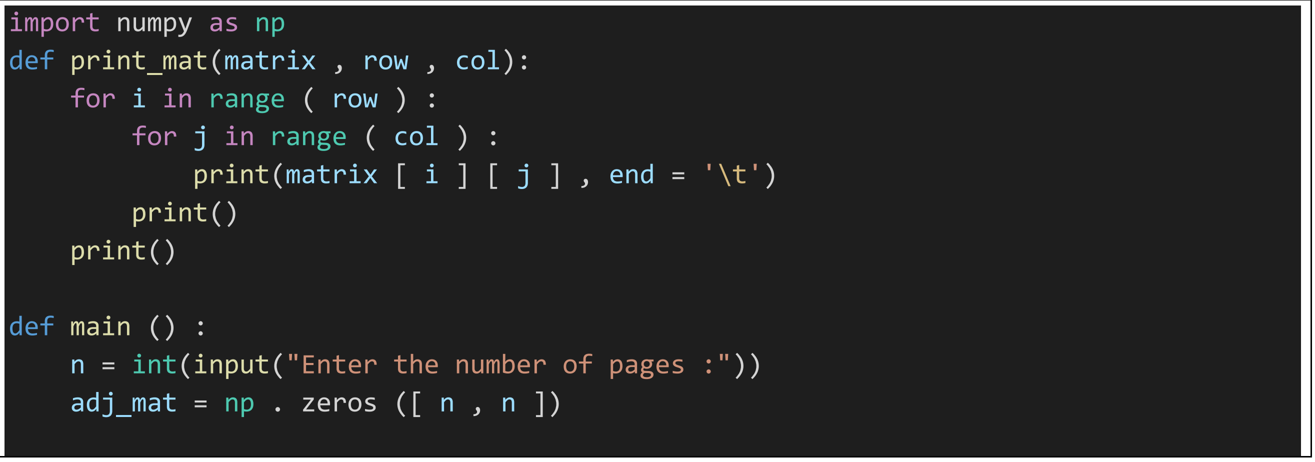
output



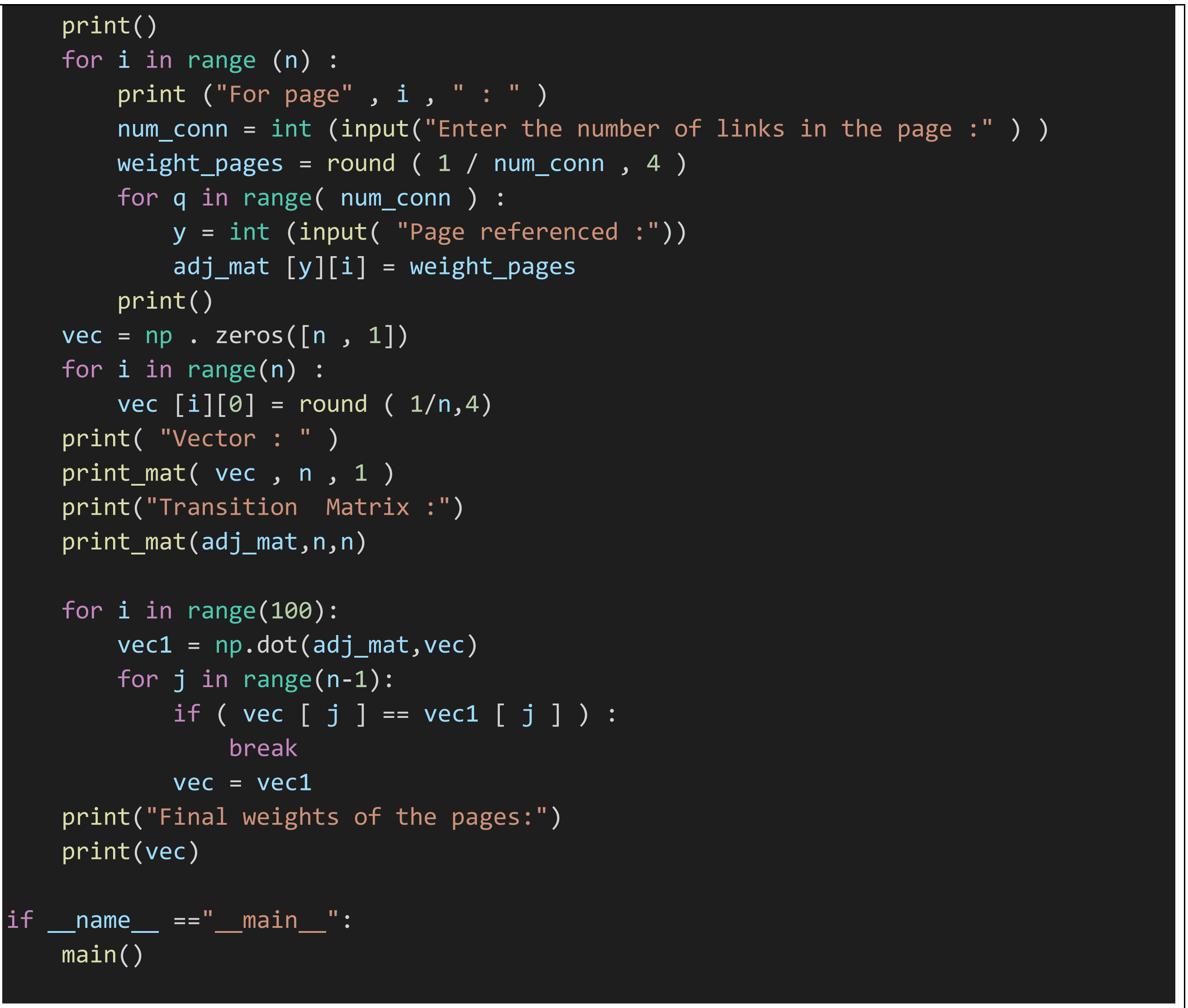
```
>> page_rank

x0 =

    3.0000
   10.0000
    3.1416
    5.0000
         0

   0.291473140314845   0.266433387961365   0.224884875090076   0.081678798064438   0.135529798569276
   0.291473140314845   0.266433387961365   0.224884875090076   0.081678798064438   0.135529798569276
```

Python Code

Code

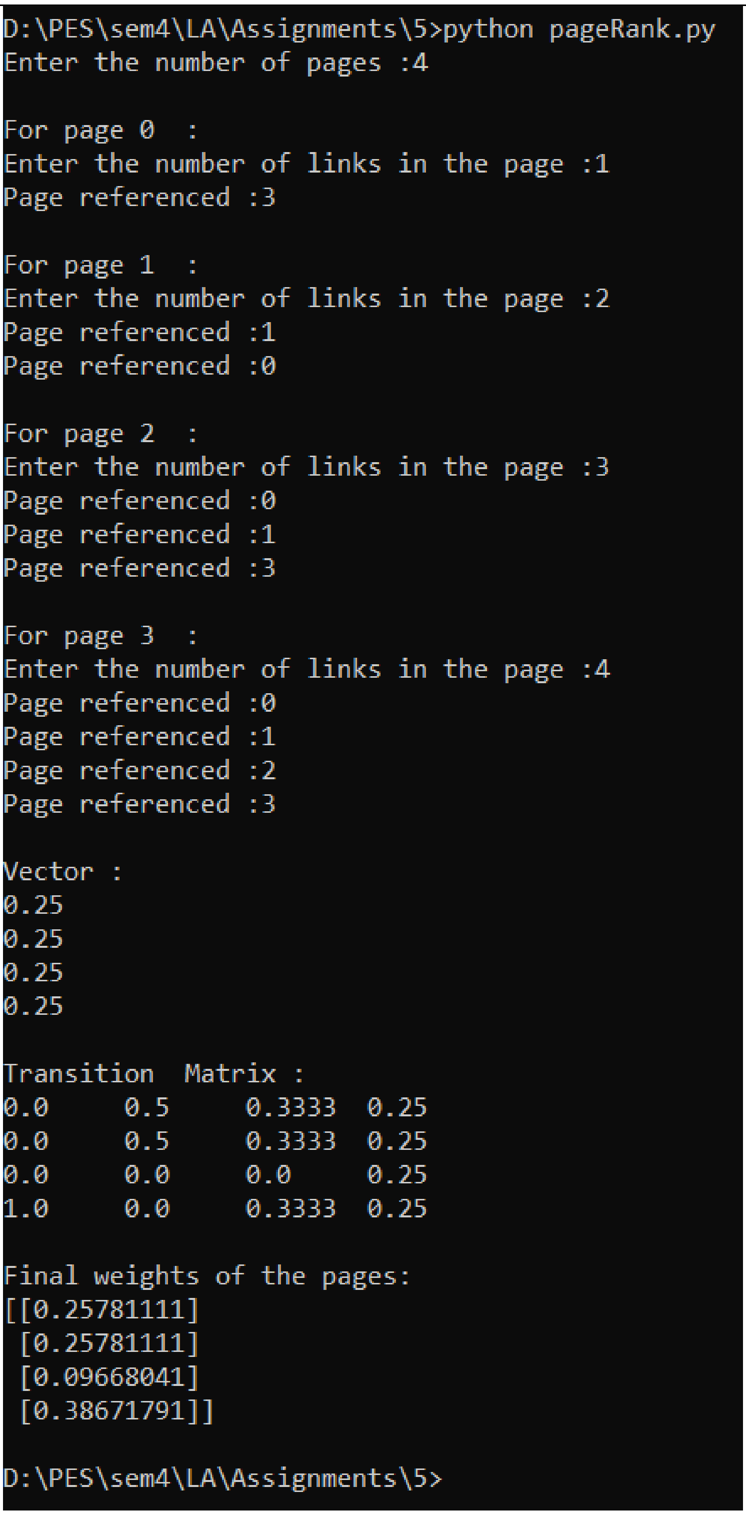


```
import numpy as np
def print_mat(matrix , row , col):
    for i in range ( row ) :
        for j in range ( col ) :
            print(matrix [ i ] [ j ] , end = '\t')
        print()
    print()

def main () :
    n = int(input("Enter the number of pages :"))
    adj_mat = np . zeros ([ n , n ])
```



```python
    print()
    for i in range (n) :
        print ("For page" , i , " : " )
        num_conn = int (input("Enter the number of links in the page :" ) )
        weight_pages = round ( 1 / num_conn , 4 )
        for q in range( num_conn ) :
            y = int (input( "Page referenced :"))
            adj_mat [y][i] = weight_pages
        print()
    vec = np . zeros([n , 1])
    for i in range(n) :
        vec [i][0] = round ( 1/n,4)
    print( "Vector : " )
    print_mat( vec , n , 1 )
    print("Transition  Matrix :")
    print_mat(adj_mat,n,n)

    for i in range(100):
        vec1 = np.dot(adj_mat,vec)
        for j in range(n-1):
            if ( vec [ j ] == vec1 [ j ] ) :
                break
            vec = vec1
    print("Final weights of the pages:")
    print(vec)

if __name__ =="__main__":
    main()
```

output



```
D:\PES\sem4\LA\Assignments\5>python pageRank.py
Enter the number of pages :4

For page 0  :
Enter the number of links in the page :1
Page referenced :3

For page 1  :
Enter the number of links in the page :2
Page referenced :1
Page referenced :0

For page 2  :
Enter the number of links in the page :3
Page referenced :0
Page referenced :1
Page referenced :3

For page 3  :
Enter the number of links in the page :4
Page referenced :0
Page referenced :1
Page referenced :2
Page referenced :3

Vector :
0.25
0.25
0.25
0.25

Transition  Matrix :
0.0     0.5     0.3333  0.25
0.0     0.5     0.3333  0.25
0.0     0.0     0.0     0.25
1.0     0.0     0.3333  0.25

Final weights of the pages:
[[0.25781111]
 [0.25781111]
 [0.09668041]
 [0.38671791]]

D:\PES\sem4\LA\Assignments\5>
```

****