Solution to Problem 1

Shanmei Liu

Department of Computer Science, MUN

Aug 17, 2016

Abstract

Among many existing algorithms for computing the connectivity of a graph, the use of eigenvalue techniques in linear algebra has given answer to the questions in graph theory. A strategy of matrix manipulation of Laplacian Matrix to find connected components in a finite, simple, undirected, unweighted graph is studied and demonstrated here.

Introduction

Finding connected components is the classic problem in graph theory, as the fundamental property of a graphs is connectivity. Connectivity plays an essential role in many network related problems that closely linked to real world applications in many disciplines [1], [2], [3].

The properties of Laplacian matrix have been studied for decades [4] since Fiedler [5] established the use of eigenvalues of Laplacian matrix for measuring algebraic connectivity. Despite of many significant insights one can gain from the Laplacian matrix, the knowledge we need here is that the number of times that 0 appears to be an eigenvalue of the Laplacian matrix equals the number of connected components in the graph. This indicates the way of finding out number of connected components in a graph is essentially to compute the eigenvalues of a matrix.

Definitions

Def.

1) Graph

A graph G = (V, E) is represented by its vertices V and edges E. If E consists ordered pairs of nodes, the graph is called directed graph as in figure 1; if unordered, then it is undirected graph. Here, we only consider about undirected graph as in figure 2.

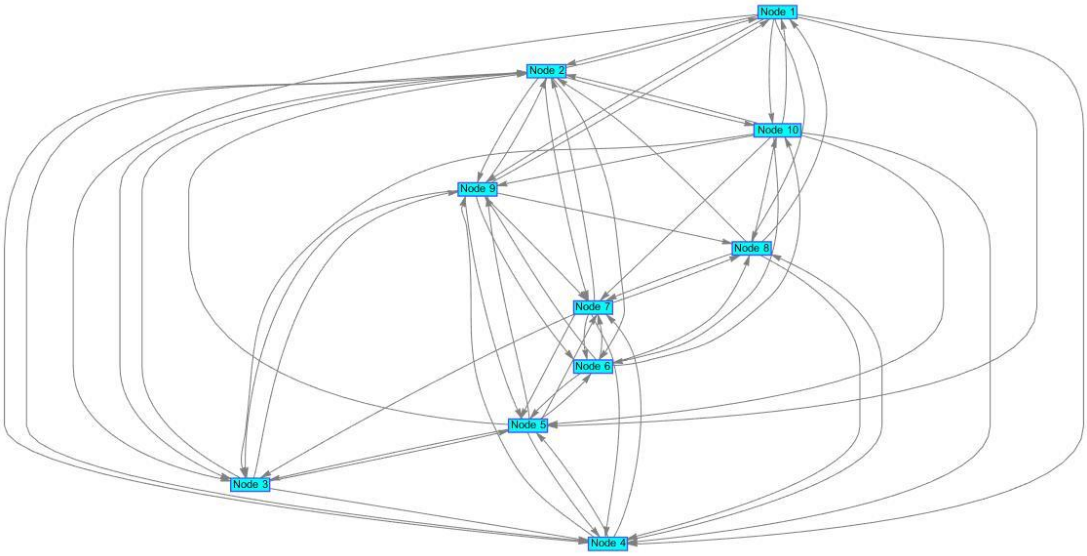


Figure 1, directed representation of data set N10.txt

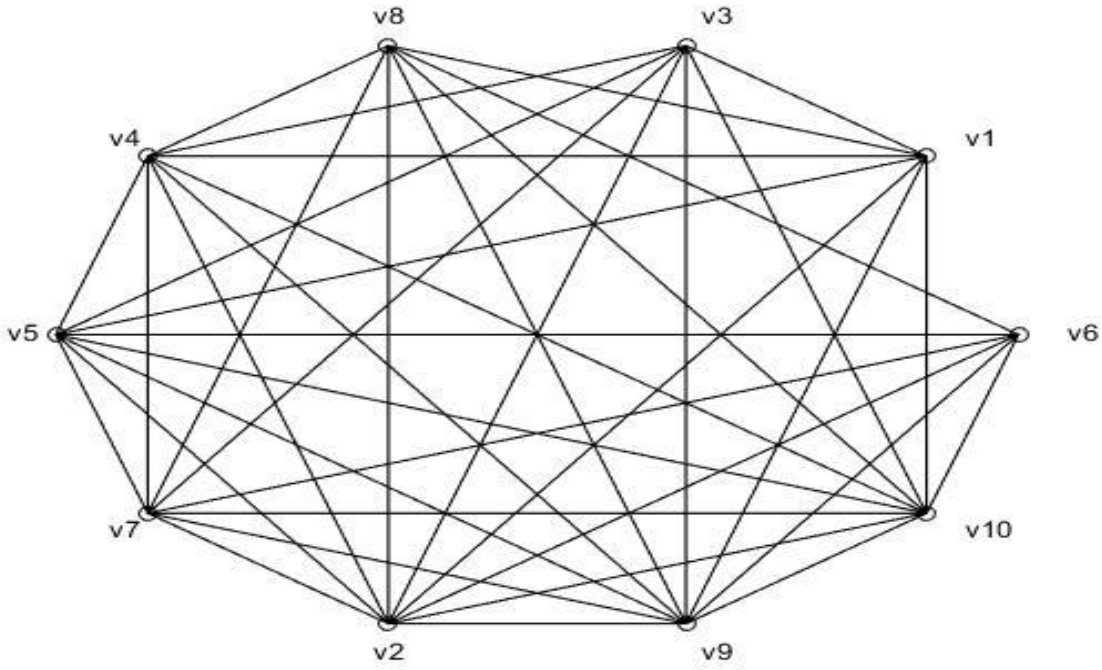


Figure 2, undirected representation of data set N10.txt

2) Connected components

Vertices *v* and *w* are connected if there is a path between them. A connected component of an undirected graph is a subgraph in which any two vertices are connected to each other by paths and

none of the vertices is connected to any vertex not in this subgraph. A vertex with no incident edges is itself a connected component.

3) Adjacency matrix

The adjacency matrix is a (0,1)- square matrix, if there is an edge between node ***i*** and ***j***, the value of ***A(i,j)*** is 1, otherwise, it is 0. For undirected graph, its adjacency matrix is symmetric with zeros on its diagonal.

4) Degree matrix

The degree matrix is a diagonal matrix, ***D(i,i)*** is the degree of node ***i*** which is the sum of row ***i*** in the corresponding adjacency matrix ***A***.

5) Laplacian matrix

The Laplacian matrix ***L*** is defined as ***L=D-A***, where ***D*** is the degree matrix and ***A*** is the adjacency matrix.

6) Incidence

If two nodes *v*, *w* are linked by edge e, nodes *v*, *w* are said to be incident to edge e, and edge e is incident to nodes *v*, *w*.

7) Degree

The number of edges incident to a given node.

8) Incidence Matrix

The unsigned incidence matrix N_{nxm} is an alternative representation of a graph G = (V, E), where

$V= \{v_1, v_2, \dots , v_n\}$

$E= \{e_1, e_2, \dots , e_m\}$

N(i, k) =1 if node $\mathbf{v_i}$ is incident to edge $\mathbf{e_k}$, and 0 otherwise, it is not difficult to prove that

$$\boldsymbol{D + A = N N^T}$$

If each edge of G is assigned an orientation, which is arbitrary but fixed, the corresponding signed incidence matrix B_{nxm} is defined as

$$\boldsymbol{B(i, j)} = \begin{cases} 1 & if\ e_j\ starts\ from\ v_i \\ -1 & if\ e_j\ terminates\ at\ v_j \\ 0 & other\ wise \end{cases}$$

Properties of associated matrices

Laplacian matrix of a graph can be decomposed into the product of the incidence matrix ***B*** and its transpose $\boldsymbol{B^T}$, i.e., $\boldsymbol{L=BB^T}$, thus L is symmetric, semidefinite with all real entries, and the rank of ***L*** is n-k, where k is the number of connected components of G [6]. More detailed proof can be found at [7], [8].

The rank of a n x n Hermitian matrix is known that equals to n – k, where k is the multiplicity of 0 as an eigenvalue of the matrix [9].

The above theorems lead to the common straight-forward method of calculating the number of connected components by computing the eigenvalues of the Laplacian matrix as presented here.

Discussion and possible improvement

From the result we see that data set N10 represents a connected graph, while the other do not. With the size of matrix increases drastically as the number of nodes increases, the eigenvalue method might not be the most efficient way with respect to storage and computation resource.

The sparsity of each data set is also different, N10 is nearly a complete graph while N10000 is relatively sparse, the adjacency matrix presentation can be impractical if the number of nodes becomes even larger

Further more, there are many conceptions and insights one can distill from a graph with a slight variation of definition, such as the spanning tree, the weighted graph and digraph, which are all worth exploring theoretically and numerically.

Another point can be the complexity for matrix computations, there exist many different algorithms for computing eigenvalues and rank of a matrix, some work better on symmetric matrix, some may not.

Reference

[1] Pavlopoulos, G. A., Secrier, M., Moschopoulos, C. N., Soldatos, T. G., Kossida, S., Aerts, J., … Bagos, P. G. (2011). Using graph theory to analyze biological networks. BioData Mining, 4, 10. http://doi.org/10.1186/1756-0381-4-10

[2] Li, Y., Li, B., Tian, B., & Yao, Q. (2013). Vehicle detection based on the AND–OR graph for congested traffic conditions. IEEE Transactions on Intelligent Transportation Systems, 14(2), 984-993.

[3] van der Schaft, A., Rao, S., & Jayawardhana, B. (2013). On the mathematical structure of balanced chemical reaction networks governed by mass action kinetics. SIAM Journal on Applied Mathematics, 73(2), 953-973.

[4] Zhang, X. D. (2011). The Laplacian eigenvalues of graphs: a survey. arXiv preprint arXiv:1111.2897

[5] Fiedler, M. (1973). Algebraic connectivity of graphs. Czechoslovak mathematical journal, 23(2), 298-305.

[6] Bapat, R. B. (1996). The Laplacian matrix of a graph. Mathematics Student-India, 65(1), 214-223.

[7] Gallier, J. (2016). Spectral Theory of Unsigned and Signed Graphs. Applications to Graph Clustering: a Survey. arXiv preprint arXiv:1601.04692.

[8] Bapat, R. B. (2010). Graphs and matrices. New York (NY): Springer.

[9] Han, Y. J., Zhang, Y. S., & Guo, G. C. (2005). Compatibility relations between the two-party reduced and global tripartite density matrices. Physical Review A, 72(5), 054302.

Appendix Code and Result

A1 Environment Setting

Hardware: Acer Aspire S7-191 / Intel Core™ i5-3337U CPU @1.80GHz 1.80GHz/ 4GB DDR3/ 128GB SSD

Operating System: Windows 8.1 64x

Programming Environment: Matlab 2014b 64x

Additional package used from "MIT Strategic Engineering": http://strategic.mit.edu/

Functions of package listed on webpage of "Matlab Tools for Network Analysis (2006-2011)"

http://strategic.mit.edu/downloads.php?page=matlab_networks

Sample data and original problem can be found at

http://www.ics.uci.edu/~wayne/research/students/

A2 Code with description

1. Overview of problem

From the conception of number of connected components, we know it equals the multiplicity of 0 as an eigenvalue of the Laplacian matrix of the graph. To compute the eigenvalues of a matrix isn't a difficult thing nowadays, though the mathematical proof requires some knowledge, the conception of matrix is much easier to understand and implement in Matlab.

2. Data structure and operations

The sample data gives the number of nodes and information of edges, we need to use this to build an adjacency matrix in Matlab , the idea is simple: first construct an N x N matrix A with all zeros, if there is an edge from Node i to j, we set the value A(i,j) to be 1,

```
M=dlmread('n10.txt')%Import Delimited Numeric Data
%M=dlmread('n100.txt');
%M=dlmread('n1000.txt');
%M=dlmread('n10000.txt');
%M=dlmread('s1');



N=M(1,1); % number of nodes
Edge=M(2:end,:);%  Create Edge table
NumOfEdge=length(Edge(:,1)) % Number of edge

B=Edge+1; % Since the indices start from 0, we assume all start from 1 here,
v1, v2, v3 ...


%Create Adjacency matrix A from Edge table B
A=zeros(N);
```

```
for i=1:NumOfEdge
    ii=B(i,1);
    jj=B(i,2);
    A(ii,jj)=1;
end
```

2.1 Number of connected components

This doesn't guarantee a symmetric matrix as it is required to treat the graph as undirected, plus the functions we need require undirected graph as input, so convert A to AA as an undirected graph

Knowing the functions below are all from the package 'Matlab Tools for Network Analysis', functions such as `num_conn_comp` are dependent of several functions in the package for computing the Laplacian and its eigenvalues

```
%convert it to the symmetric matrix for undirected graph
 AA=adj2simple(A+A');

 % plot
 % Draw a circular graph with links and nodes in order of degree
draw_circ_graph(AA)

 % Calculate the number of connected components
 s=num_conn_comp(AA)
```

2.2 Subgraph

```
%Find the connected components
 C=find_conn_comp(AA)
```

2.3 Histogram of the distribution of degrees of nodes

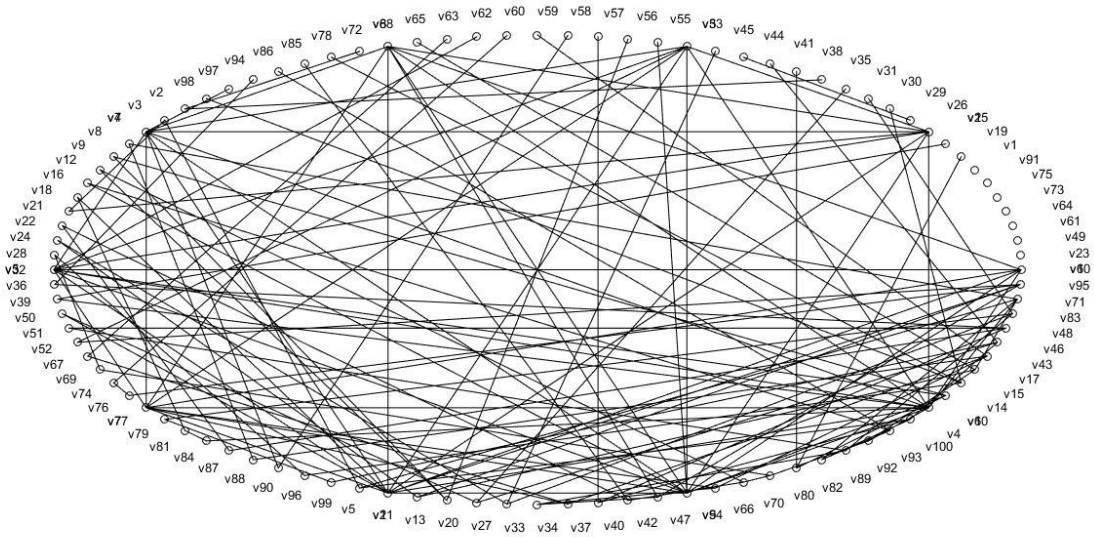
The degree of each node is simply the sum of each row, thus it gives a 1 x N vector deg

```
% Degree of each node
 deg = sum(AA);
 % Histogram
 h = histogram(deg)
```

A3 Result

Data set N100.txt is used as sample data for illustration.

1) The plot of the whole network



For N1000, it is nearly impossible to figure out the individual node, for N10000, the laptop gives error in plotting rather than an output.

The number of connected components is given as s:

N10: s=1

N100: s=12,

N1000: s=17,

N10000: s=12,

S1: s=6

The connected components are grouped in a list C of cells, each cell contains the nodes in a subgraph, for N100

C =

Columns 1 through 12

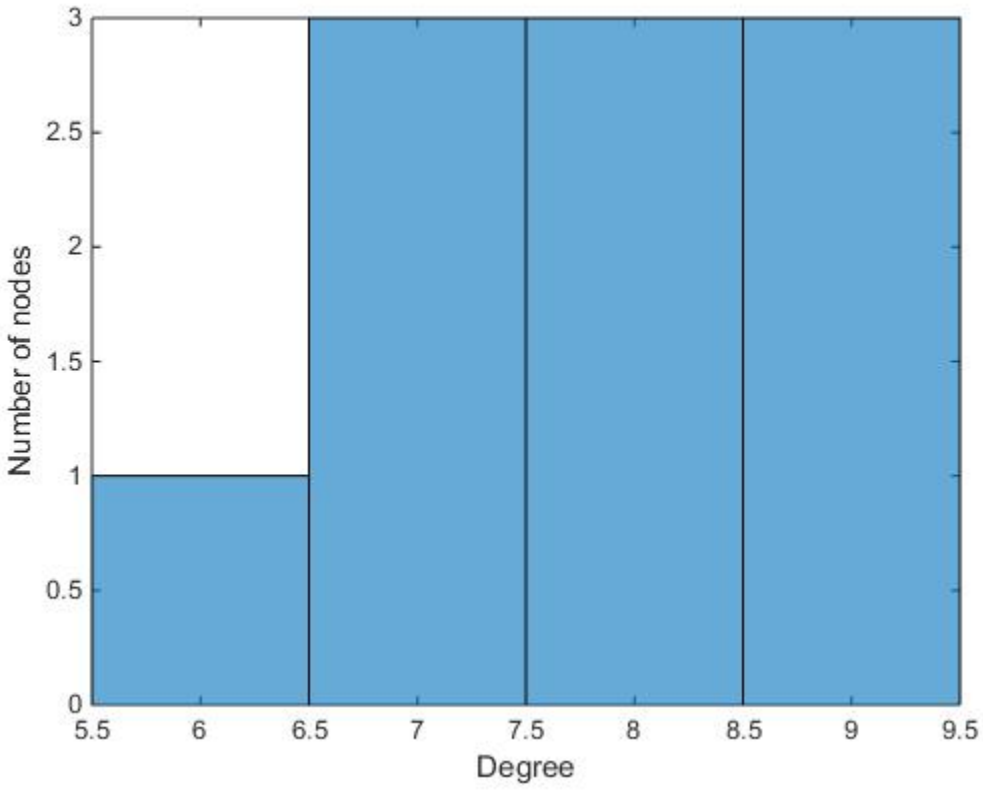
[1x83 double] [1x3 double] [1x4 double] [10] [23] [1x2 double] [49] [61] [64] [73] [75] [91]

1) Histogram for N10

Data: [7 9 7 8 8 6 8 7 9 9]

Values: [1 3 3 3]

NumBins: 4

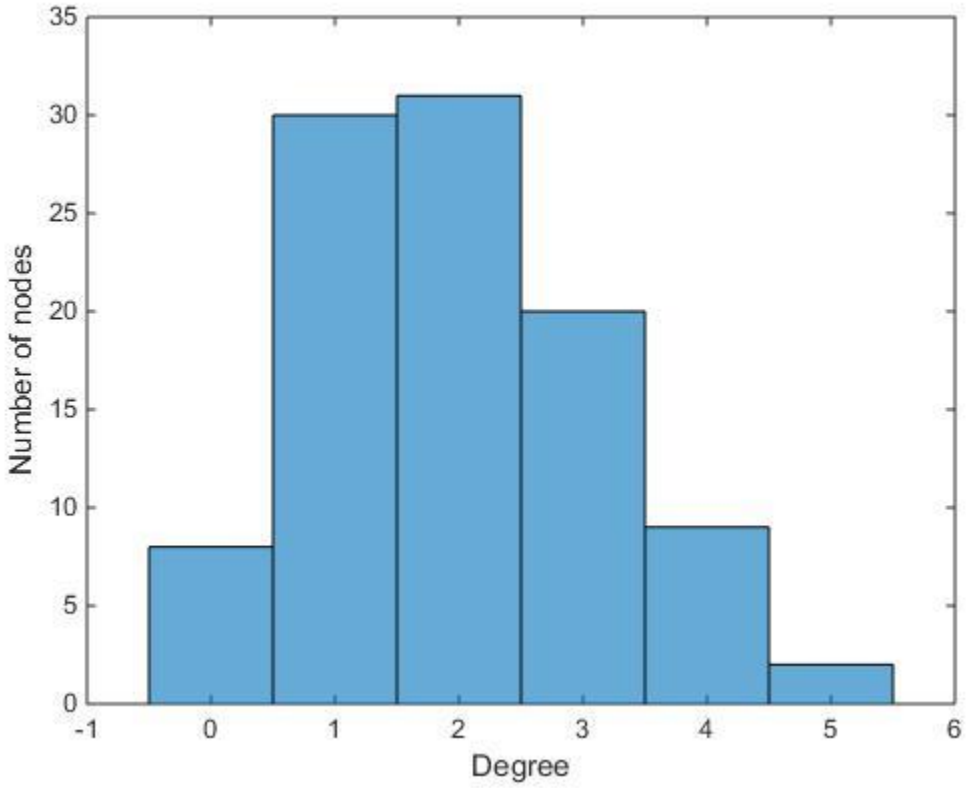


2) Histogram for N100

Data: [1x100 double]

Values: [8 30 31 20 9 2]

NumBins: 6

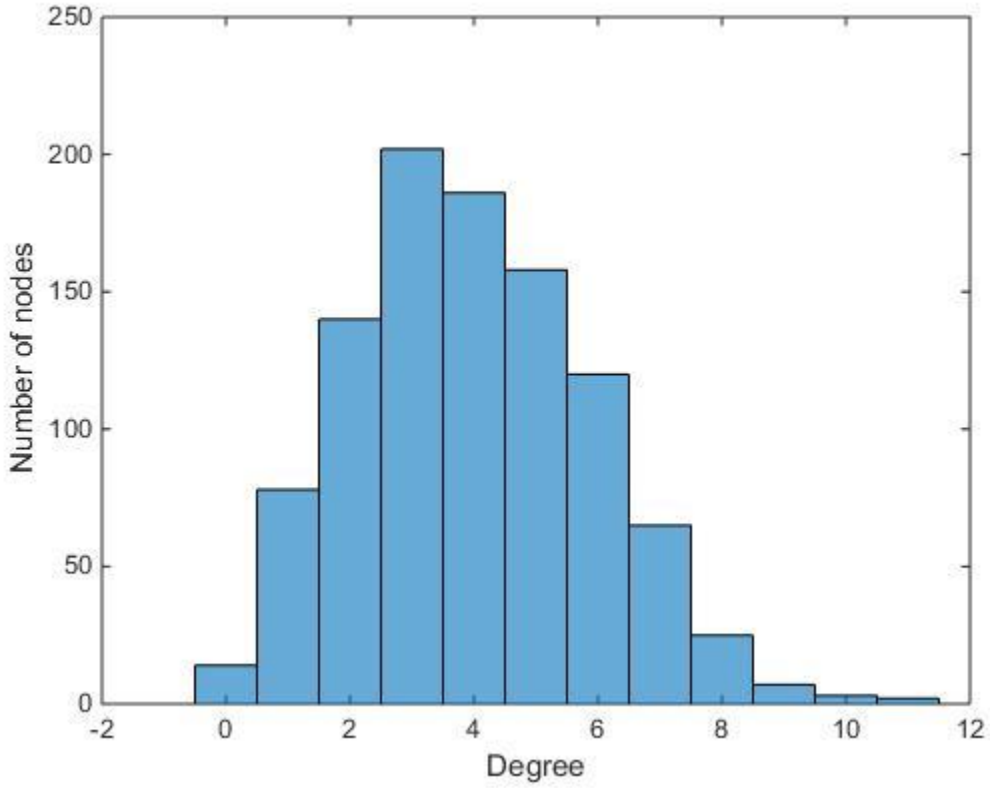


3) Histogram for N1000

Data: [1x1000 double]

Values: [14 78 140 202 186 158 120 65 25 7 3 2]

NumBins: 12

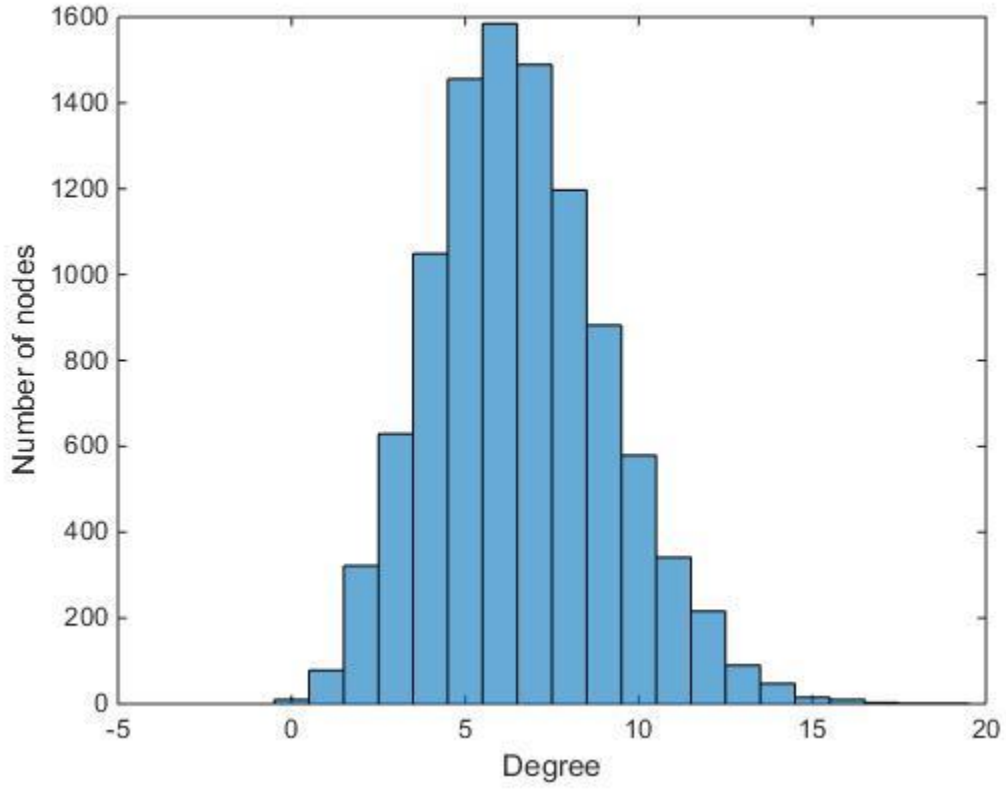


4) Histogram for N10000

Data: [1x10000 double]

Values: [10 78 322 629 1049 1456 1585 1489 1197 882 579 341 216 90 47 15 10 3 1 1]

NumBins: 20



5) Histogram for s1

Data: [1x573 double]

Values: [2 208 132 64 45 32 23 18 15 5 2 9 2 4 2 3 4 0 2 0 0 0 0 0 0 0 0 0 0 0 1]

NumBins: 31