Documentation

MATLAB code for Learning Network Topology

Written by: Satya Jayadev P

ps_jayadev@yahoo.com

Last updated on 18-Mar-2020

Table of Contents

SECTION 1

Introduction

This document is a comprehensive guide to the MATLAB code to identify the topology of networks as per the methodology proposed in the paper **"Learning Network Topology from Flows and Partial Information"** authored by **Satya Jayadev P**, **Ramkrishna Pasumarthy** and **Nirav Bhatt**. The code requires **MATLAB version R2018a** for all the features to be functional. Firstly, the problem of learning network topology is introduced with some related graph theoretic concepts and then the code is discussed in the later chapters.

1.1 Learning Network Topology

The topology of a complex network with flows along the lines can be represented as a graph. Finding this graph from data is referred to as Learning Network Topology. The topology of a network may be required in various applications depending on the type of the network. In this work, we deal with networks which have certain flow attributes with conservation properties such as power distribution networks, water distribution networks, biological networks, financial networks and so on. The topology of these networks can be represented as directed graphs with the edge directions indicating the direction of flow along that edge. The concepts of spanning trees, cutsets, circuits, incidence matrix, f-cutset matrix, f-circuit matrix, arborescence, spanning arborescence, parent nodes, child nodes and leaf nodes for directed graphs are in consistence with the ones defined in references [2],[3] and [4]. Further, we define certain concepts in relation to the topology of networks with flow conservation properties, which are of interest in this work. These concepts are discussed in the rest of this chapter.

1.2 Types of Nodes

We define three types of nodes which exist in the topology graph of networks of interest:

1. **Source nodes**: The node with only outward oriented edges is called a source node.

2. **Sink Nodes & sink flows**: The node with only inward oriented edges is called a sink node and the flows associated with sink nodes are referred to as sink flows.

3. **Intermediate Nodes**: The node with both inward and outward oriented edges is called an intermediate node.

1.3 Types of Networks

Based on the structure of the topology graph, the networks can be classified as:

1. **Looped Networks:** Networks whose topology graph contain loops or circuits

2. **Non-Looped Networks:** Networks whose topology graph do not contain loops or circuits

3. **Arborescence Networks:** These are special class of non-looped networks whose topology graph has only single source node

In our previous work [1], we showed that it is possible to recover the topology of an arborescence network from flow data. This work focuses on recovering the topology of both looped and non-looped networks which are having non-arborescence structure. The methodology proposed in the paper is discussed in the next chapter.

1.4 Conservation Graph

For certain mathematical convenience, a new graph theoretic concept called conservation graph G_c is defined to be used in the proposed methodology. It is a concept which can be applied to any conserved network in general. Conservation graph is a topological representation of the network in which conservation equations can be written around every node. In a conserved network, it can be observed that conservation equations cannot be written around source nodes and sink nodes. Therefore, all the source and sink nodes are merged into a single node called the environment node, to get the conservation graph G_c of the network with n_c nodes and e edges.

For example, consider a network with its topology and conservation graph shown in Fig. 1.

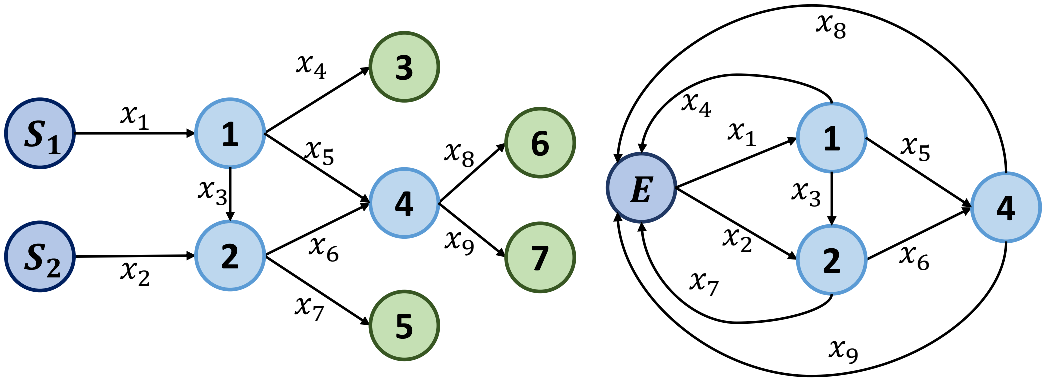


Figure 1: Topology and Conservation graph of a network

In the conservation graph shown in Fig. reffig:arb, the following conservation equations can be written around every node:

$$\textbf{At E:} x_1 + x_2 - x_4 - x_7 - x_8 - x_9 = 0$$

$$\textbf{At 1:} - x_1 + x_3 + x_5 + x_4 = 0$$

$$\textbf{At 2:} - x_2 - x_3 + x_6 + x_7 = 0$$

$$\textbf{At 4:} - x_5 - x_6 + x_8 + x_9 = 0$$

2

SECTION

Methodology

In this chapter, we discuss the methodology presented in the paper, without little explanation. For details about the correctness of the methodology, one may refer the paper. The input to the problem is steady state flow data pertaining to all the edges in the network and incidence information of certain edges. Let there be e edges in the network and n_s samples of flows along the edges are collected and stacked into a matrix $\mathbf{X}$ of size $(e \times n_s)$. Let the edge incidence data be given in the form of a matrix $\mathbf{A}_g$ with each column of $\mathbf{A}_g$ corresponding to an edge. It is shown that $\mathbf{X}$ and $\mathbf{A}_g$ are sufficient to recover the conservation graph of the network provided $\mathbf{A}_g$ has rank of atleast $n_c - 1$.
The following the major steps in the proposed methodology:

1. Learning a model matrix $\mathbf{A}$ from data giving relations between the variables (flows) using Singular Value Decomposition (SVD) or Principal Component Analysis (PCA)

2. Getting the incidence information of a spanning tree, $\mathbf{A}_b$ from $\mathbf{A}_g$, if rank condition is satisfied

3. Transformation $\mathbf{A}$ into an f-cutset matrix $\mathbf{C}_f$ of the conservation graph corresponding to the spanning tree formed by edges corresponding to columns of $\mathbf{A}_b$

4. Finding the incidence matrix of conservation graph from $\mathbf{C}_f$ and $\mathbf{A}_b$

Here, we consider finding the incidence matrix being equivalent to finding the graph because the correspondence is one-to-one.

2.1 Learning a Model (A)

If the data is noisefree, then SVD can be applied to identify a model describing the relations between variables in the data. If data is noisy, then SVD can be extended to PCA to identify the best fit model. The code is equipped to handle noisefree data as well as noisy data. So we will see how a model can be obtained in both the cases.

2.1.1 Noisefree Case

In this case, when SVD is applied on **X**, the last m singular values will be equal to zero where m is the number of linearly independent equations in the model. Therefore, SVD can be written as:

$$\mathbf{SVD(X)} = \mathbf{USV} = \mathbf{U}_1\mathbf{S}_1\mathbf{V}_1^T + \mathbf{U}_2\mathbf{S}_2\mathbf{V}_2^T$$

where $\mathbf{U}_1$ and $\mathbf{V}_1$ are the singular vectors corresponding to the non-zero singular values of **X**, and $\mathbf{U}_2$ and $\mathbf{V}_2$ are the singular vectors corresponding to the *zero* singular values of **X**. The model matrix is given by:

$$\mathbf{A} = \mathbf{U}_2^T$$

2.1.2 Noisy Case

Let the data with noisy samples be denoted by **Y**. In this case, we assume that the noise in all the variables is uncorrelated. It could be identically distributed i.e., homoscedastic noise or non-identically distributed i.e., heteroscedastic noise. In both these cases, we apply different variants of PCA to identify the best fit model.

1. **Homoscedastic case:** In this case, the smallest m singular values of **Y** are not equal to zero but they are small in comparison to other singular values and are equal. Since we are dealing with limited samples, the equality need to be established using hypothesis testing. Therefore, we propose a repetitive hypothesis test to determine the value of m. One may refer to the paper for more details on the hypothesis test. Then again the left singular vectors corresponding to smallest m singular values give the best fit model:

$$\hat{\mathbf{A}} = \mathbf{U}_2^T.$$

2. **Heteroscedastic case:** In this case, the smallest m singular values of **Y** are not equal as noise is not identically distributed in all variables. Therefore, **Y** is transformed to ensure that the noise in the samples is identically distributed. For this transformation, we assume that the covariance matrix of noise $\mathbf{\Sigma}_e$ is known. In this case also, hypothesis test is applied to determine the value of m after trans-

forming the data matrix. The best fit model is obtained as follows:

$$\begin{aligned}\mathbf{\Sigma}_e &= \mathbf{L}\mathbf{L}^T \\ \mathbf{Y}_s &= \mathbf{L}^{-1}\mathbf{Y} \\ \mathbf{SVD}(\mathbf{Y}_s) &= \mathbf{U}_{1s}\mathbf{S}_{1s}\mathbf{V}_{1s}^T + \mathbf{U}_{2s}\mathbf{S}_{2s}\mathbf{V}_{2s}^T \\ \hat{\mathbf{A}} &= \mathbf{L}^{-1}\mathbf{U}_{2s}^T,\end{aligned}$$

where **L** is the Cholesky factor of $\mathbf{\Sigma}_e$.

2.2 Getting $\mathbf{A}_b$

$\mathbf{A}_b$ is the matrix which contains just incidence information of a spanning tree of the conservation graph required to get its complete incidence matrix. $\mathbf{A}_b$ is a square matrix with m rows and columns and is of full rank. Given $\mathbf{A}_g$, to obtain $\mathbf{A}_b$, take the following steps:

1. If $\text{rank}\left(\mathbf{A}_g\right) < m$, conclude that the connectivity information is not sufficient to find the incidence matrix.

2. If $\text{rank}\left(\mathbf{A}_g\right) \geq m$, then select m linearly independent rows and columns of $\mathbf{A}_g$ to form $\mathbf{A}_b$.

2.3 Getting f-cutset matrix of Conservation Graph

Now, we determine $\mathbf{C}_f$ of the conservation graph corresponding the spanning tree formed by edge corresponding to columns of $\mathbf{A}_b$. If the edges forming $\mathbf{A}_b$ are given by $\mathbf{x}_d$ and rest by $\mathbf{x}_i$, we get $\mathbf{C}_f$ by the following transformation:

$$\begin{aligned}\mathbf{A}\mathbf{x} &= \mathbf{A}_d\mathbf{x}_d + \mathbf{A}_i\mathbf{x}_i = \mathbf{0} \\ \mathbf{x}_d &= -\mathbf{A}_d^{-1}\mathbf{A}_i\mathbf{x}_i \\ \begin{bmatrix}\mathbf{I}_m & \mathbf{A}_d^{-1}\mathbf{A}_d\end{bmatrix}\begin{bmatrix}\mathbf{x}_d \\ \mathbf{x}_i\end{bmatrix} &= \mathbf{C}_f\mathbf{x} = \mathbf{0}.\end{aligned}$$

2.4 Finding Incidence matrix of G_c

Firstly we find a reduced incidence matrix using the following formula derived in the paper:

$$\mathbf{A}_r = \mathbf{A}_b\mathbf{c}_f \tag{1}$$

Finally, the incidence matrix $\mathbf{A}_{in}$ of G_c is obtained from $\mathbf{A}_r$ by adding a row with ±1 elements in columns having only one non-zero element. The sign should be opposite to that of the non-zero element in each column since we are dealing with a directed graph.

Based on this methodology, a MATLAB code has been written to identify the topology of any network from steady state flow data. The functionalities of the code are detained in the following chapter.

3

SECTION

Functionalities of the Code

The MATLAB code is an interactive one in which the user inputs determine what sequence of actions that the program performs. The following are the two major functionalities of the code:

3.1 Generating Random Networks & Flow Data

The code has functions to generate networks randomly based on certain inputs from the user and also generate data which could be either noisefree or noisy. Following are more details about these generators:

1. **Generating a Random Network:** The user can choose to generate a random network based on one of the three network models viz. Erdos-Renyi model [10], Small world network (Watts–Strogatz model) [11] or Scale free networks(Barabasi–Albert model) [12]. The user also has the option to choose the number of nodes in the network. The parameters of the network models are fixed in the code assuming the user might not be familiar with all these models. However, users familiar with these models may alter the parameters in the source code. While converting the topology graph to conservation graph, three could be edges forming self loops at the environment. These are edges with terminal nodes being source and sink nodes. Any such edges generated by the models are removed while constructing the conservation graph of the network.

2. **Generating Flow Data:** Flow data is generated randomly by choosing certain flows to be independent and drawing samples from normal distributions. Three nor-

mal distribution with different means and different variances are chosen and randomly assigned to each of the independent flows. The data for the rest of the flows (dependent ones) is determined based on the nodal conservation equations.

The user also has the option to choose to generate noisefree or noisy data. In noisy case, the user an choose to add homoscedastic noise (independent and identically distributed) or heteroscedastic noise (only independent but not identical). For homoscedastic noise, the user has to provide an SNR value based on which variance of the noise to be added is determine while for heteroscedastic case, the user is expected to provide an array (size e) of SNR values. Accordingly, noise samples are drawn from normal distributions of zero mean and added to the flow samples.

Regarding the number of samples to be generated (n_s), they are generated in multiples of e since e is the minimum number required to apply PCA/SVD to get a model. The users have to choose a multiple z, in a given range, and the problem generates $n_s = ze$ samples.

3.2 Finding Incidence Matrix from Flow Data & Partial Information

This functionality of the code is based on the methodology described in Chapter 2. The user can choose between feeding input flow data or using the network & data generated by the program. The code has the capability to finding the incidence matrix of the G_c both with noisefree and noisy data, when provided with sufficient number of samples and partial incidence information.

SECTION 4

Functions and Variables

In this chapter, we discuss the different functions and variables that were used in the code along with their details.

4.1 Functions

The following are the different functions defined in the code, each in a different .m file:

4.1.1 Main

This is the main function of the code which is to be run to begin the program. All the user interactions and defaults are coded in this function right from taking input data to generating networks to generating data. It also comprises the code which finds the incidence matrix of the graph using the formula after obtaining a linear model.

4.1.2 Network_Generation

This function generates a network based on the choice of the user. The user also has a choice to choose the number of nodes in the network generated. It converts the network into a conservation graph and any edges which form self loops are the environment node are removed. Since there is merger of nodes, the number of nodes in the conservation graph is less than the nodes in the network generated. Then all the edges in the conservation graph are randomly indexed. The code also identifies a spanning tree of the conservation graph. It returns the incidence matrix of G_c, incidence of identified spanning tree along with branch & chord information and number of nodes and edges in G_c.

4.1.3 erdosRenyi

This function is used to generate an Erdos-Renyi network given the number of nodes and other parameters based on the Erdos Renyi model. This is function taken from the MATLAB file exchange [13].

4.1.4 smallw

This function is used to generate an Small World network given the number of nodes and other parameters based on the Watts-Strogatz model. This is a function provided by Mathworks as an example.

4.1.5 BAgraph_dir

This function is used to generate an Scale Free network given the number of nodes and other parameters based on the Barabasi-Albert model. This is function taken from the MATLAB file exchange [14].

4.1.6 Data_Generation

This function generates flow data pertaining to the generated network. Depending on the users choice, it also adds Gaussian noise, i.d. or i.i.d., to the dataset. It returns data matrix along with noise covariance matrix to the main function.

4.1.7 Linear_Model

This function identifies a linear model from flow data, either generated or user provided. If the data is noisefree, it checks for zero singular values to find m else if the data is noisy, it performs hypothesis testing to find m, the order of the linear model. It returns the model matrix $\mathbf{A}$ along with the singular values of data matrix $\mathbf{X}$.

4.1.8 Plot_Graph

This function plots the conservation graph of a network given the incidence matrix as argument. This is not called from the main function because it has a drawback. Since MATLAB does not allow plotting of parallel edges in directed graphs, any parallel edges in the conservation graph are merged into one edge while plotting the graph. Due to drawback, this function is only provided as a function which user needs to call separately by giving the Incidence matrix as input.

4.2 Variables

The following are the most important variables defined and used in the code. The ranges and default values of some of these variables, as applicable, are provided in Table 1.

1. **X :** Indicates the dataset(s) provided or generated

2. **SNR :** Indicates SNR value(s) in case data is noisy

3. **data_flag :** Indicates whether data is fed (1) or generated (2)

4. **noise_flag :** Indicates whether data is noisy (1) or noisefree (0)

5. **network_flag :** Indicates the type of network generated viz. Erdos-Renyi (1), Small world (2), Scale-Free (3)

6. **n :** Indicates the user choice of number of the nodes to be present in the network

7. **Adj :** Indicates the adjacency matrix of the undirected network graph generated by the chosen network model

8. **Adj_dir :** Indicates the adjacency matrix of the network graph generated by the chosen network model after assigning directions randomly

9. **Inc_dir :** Indicates the incidence matrix of the network graph generated by the chosen network model after assigning directions randomly

10. **Inc_Con :** Indicates the incidence matrix of G_c after merging the source and sink nodes

11. **Adj_Con :** Indicates the adjacency matrix of G_c

12. **e_c :** Indicates the number of edges in G_c

13. **n_c :** Indicates the number of nodes in G_c

14. **Inc_Con :** Indicates the incidence matrix of the G_c of generated network

15. **Cc_Con :** Indicates the f-cutset matrix pertaining to a spanning tree of G_c identified by the program

16. **Ab :** Indicates the incidence sub-matrix provided or pertaining to the identified spanning tree

17. **Ab_edges :** Indicates the edges corresponding to the columns of incidence submatrix provided

18. **branch :** Indicates the edges corresponding to the branches of the identified spanning tree

19. **chord :** Indicates the edges corresponding to the chords/links of the identified spanning tree

20. **b :** Indicates the number of branches in the identified spanning tree

21. **c :** Indicates the number of chords in the identified spanning tree

22. **NSamples :** Indicates multiple by which e is to be multiplied to get the number of samples in the dataset

23. **NSamples :** Indicates number of samples in the dataset

24. **Nrepeats :** Indicates the number of datasets generated

25. **Sigma_e :** Indicates the error covariance matrix of data

26. **Ahat :** Indicates the model explaning the data, given by PCA

27. **sin_vals :** Indicates the singular values of **X**

28. **pred_index :** Indicates the connectivity of the network predicted by the program, in vector form

29. **Cf_desired :** Indicates the desired f-cutset matrix of the arborescence network

30. **avg_time :** Indicates the time taken to find the network; average time in case of multiple datasets

31. **avg_test :** Indicates the success in finding the network; average success in case of multiple datasets

Table 1: Applicable Values & Defaults of Variables

S.No.	Variable	Applicable Values/Range	Default Value
1	data_flag	1,2	1
2	network_flag	1,2,3	1
3	noise_flag	1,0	0
4	n	10 to 100	10
5	Nmultiple	1 to 100	1
6	Nrepeats	1 to 100	10
7	SNR	5 to 500	50

5

SECTION

Steps to Use the Program

In this chapter we discuss the exact steps to be followed by the user while interacting with the program, to achieve a particular objective. As discussed in Chapter 3, the user can use the problem for the following objectives:

1. To find the topology of an arborescence network by feeding in flow data.

2. To generate a the graph of a network based on a particular network model and pertaining flow data, and then let the code find its incidence matrix from that data.

In both the objectives, the data could be noisefree or noisy with noise characteristics as discussed in Chapter 2.

5.1 Feed Data → Find Topology

In this objective, the user is expected to provide the following:

1. Data Matrix in variable named 'X' with exactly e rows and any number of columns

2. Partial incidence information in variable named 'Ab' and corresponding edges as a row vector in variable named 'Ab_edges'. Note that 'Ab' and 'Ab_edges' should have the same number of columns.

3. In case of noisy data, if it is homoscedastic or heteroscedastic

4. In case of heteroscedastic noise, a column vector of SNR values (size e) corresponding to each flow in the network

The program terminates if any of this information is not provided or provided with incorrect dimensions. The following are the steps to be followed when interacting with the program to achieve this objective:

1. Ensure that 'X', 'Ab', 'Ab_edges' and 'SNR' are provided in the workspace
2. Run the main function
3. Choose the option '1 - Feed Data', when asked
4. Program checks for variable 'X' in the workspace and terminates the program if not available
5. Program checks for variable 'Ab' in the workspace and terminates the program if not available
6. Program checks for variable 'Ab_edges' in the workspace and terminates the program if not available
7. Program checks if the columns of 'Ab' and 'Ab_edges' are equal and terminate the program if not
8. Choose whether the data is 'noisy' or 'noisefree', when asked
9. In noisy case, choose whether the noise is 'homoscedastic' or 'heteroscedastic', when asked
10. In heteroscedastic case, program checks for variable 'SNR' in the workspace and terminates the program if not available
11. Finally, the code runs and gives an output whether an incidence matrix corresponding to a conservation graph could be identified or not including the time taken

5.2 Randomly Generate Network & Data → Find Topology

In this objective, the user is has an option to choose the following or let the program run with default values [1]:

1. Type of Network to be generated
2. No. of nodes in the network to be generated
3. No. of samples in the dataset to be generated
4. Noisy or noisefree data
5. If noisy, homoscedastic or heteroscedastic noise
6. SNR values to be provided

[1] To run the code by default at any step, the user has to just press enter at every step.

7. If noisy, choose the number of datasets to be generated for a network

The following are the steps to be followed when interacting with the program to achieve this objective:

1. Run the main function
2. Choose the option '2 - Random network and Data', when asked
3. Choose the type of network to be generated: 1 - Erdos-Renyi, 2 - Small World or 3 - Scale Free, when asked or Erdos-Renyi network is taken by default
4. Choose the number of nodes in the network in the given range or default value of 10 is taken
5. Choose the number of samples in the dataset as a multiple of e or a default value is taken
6. Choose whether the data is 'noisy' or 'noisefree', when asked or a noisefree case is taken by default
7. In noisy case, choose whether the noise is 'homoscedastic' or 'heteroscedastic', when asked or homoscedastic case is chosen by default
8. In homoscedastic case, choose an SNR value for noise to be added or a default value of 10 is taken
9. In heteroscedastic case, program checks for variable 'SNR', a column vector of size e, in the workspace and terminates the program if not available
10. In noisy case, choose the number of datasets to be generated for a network or a default value of 10 is taken
11. Finally, the code runs and gives an output whether the topology could be identified or not including the average time taken

6

SECTION

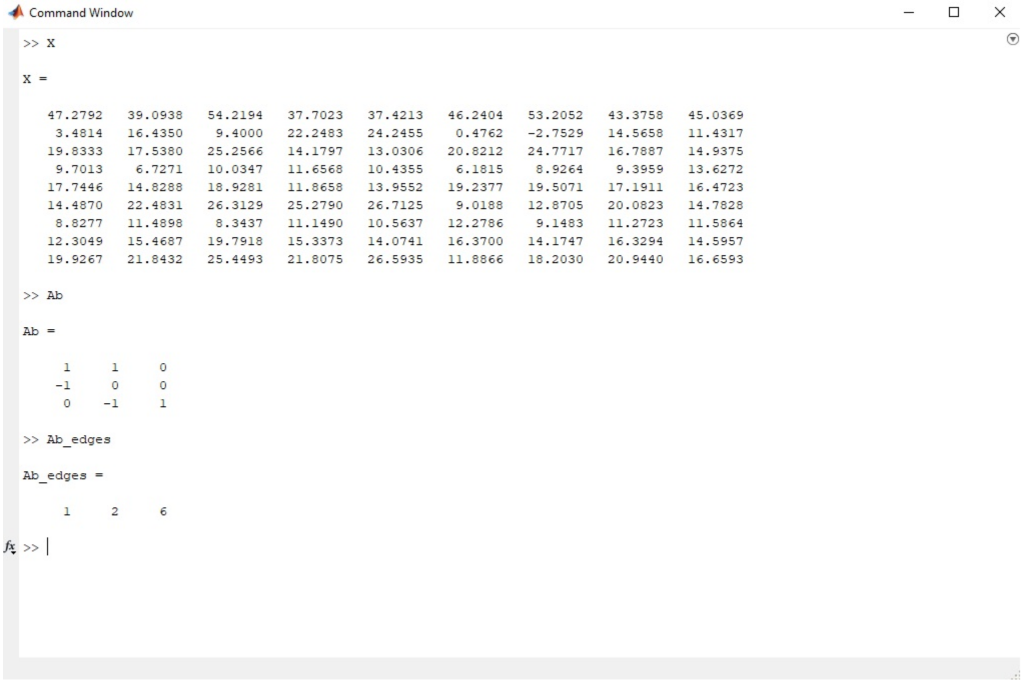
Examples

In this chapter, we discuss some specific examples of running the code to achieve specific objectives.

6.1 Example 1

This example shows how once can feed noissefree flow data to the program and get the topology of the network from which the data was collected. We take the network which was used as an example in the paper. Flow data of 9 samples pertaining to that network, is fed to the program to see if the network could be identified. The following are the steps taken and a snapshot of the MATLAB command line after each step is shown:

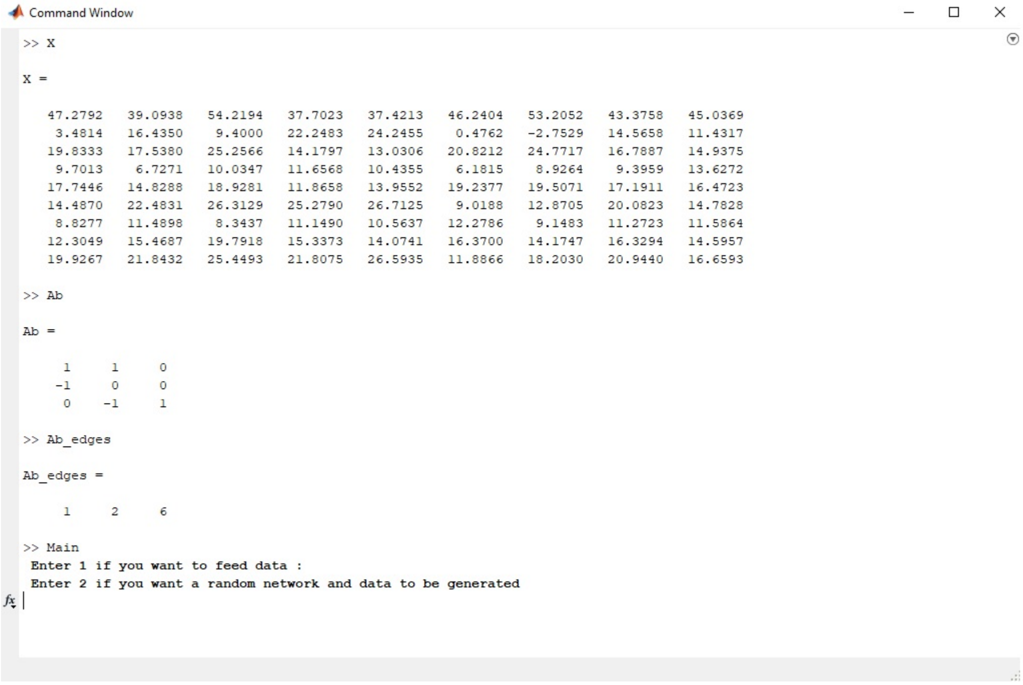
1. **Step 1:** Ensure that the data and partial incidence is provided in the workspace named as variables X, Ab and Ab_edges, having the right dimensions



```
>> X

X =

   47.2792   39.0938   54.2194   37.7023   37.4213   46.2404   53.2052   43.3758   45.0369
    3.4814   16.4350    9.4000   22.2483   24.2455    0.4762   -2.7529   14.5658   11.4317
   19.8333   17.5380   25.2566   14.1797   13.0306   20.8212   24.7717   16.7887   14.9375
    9.7013    6.7271   10.0347   11.6568   10.4355    6.1815    8.9264    9.3959   13.6272
   17.7446   14.8288   18.9281   11.8658   13.9552   19.2377   19.5071   17.1911   16.4723
   14.4870   22.4831   26.3129   25.2790   26.7125    9.0188   12.8705   20.0823   14.7828
    8.8277   11.4898    8.3437   11.1490   10.5637   12.2786    9.1483   11.2723   11.5864
   12.3049   15.4687   19.7918   15.3373   14.0741   16.3700   14.1747   16.3294   14.5957
   19.9267   21.8432   25.4493   21.8075   26.5935   11.8866   18.2030   20.9440   16.6593

>> Ab

Ab =

     1     1     0
    -1     0     0
     0    -1     1

>> Ab_edges

Ab_edges =

     1     2     6

>> |
```

Figure 2: Example 1 - Step 1

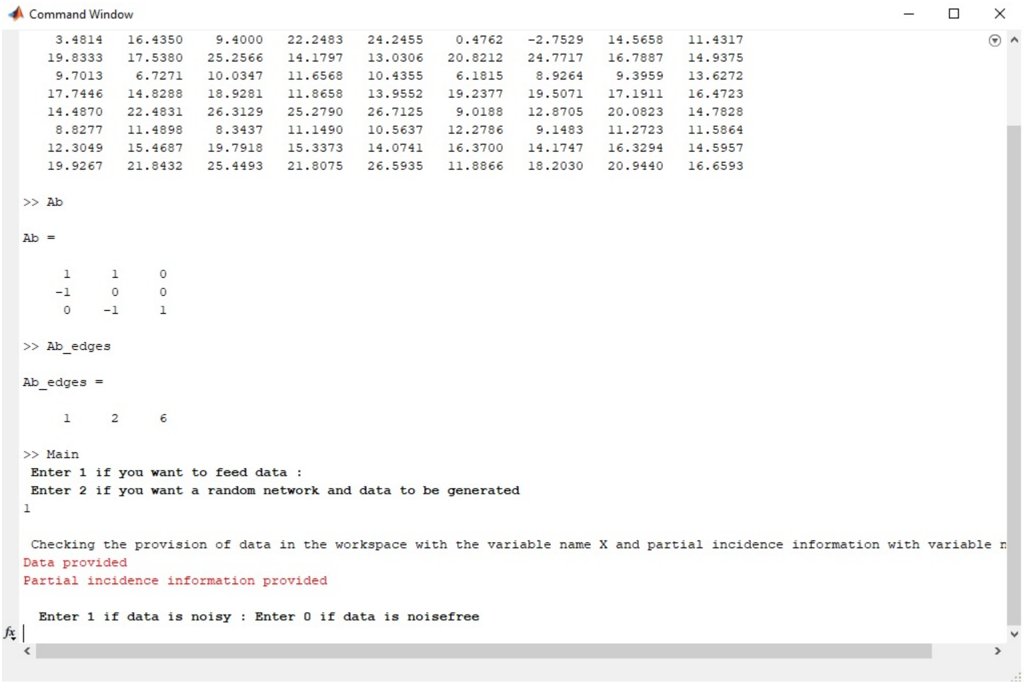
2. **Step 2:** Run the ‘Main’ function



```
>> X

X =

   47.2792   39.0938   54.2194   37.7023   37.4213   46.2404   53.2052   43.3758   45.0369
    3.4814   16.4350    9.4000   22.2483   24.2455    0.4762   -2.7529   14.5658   11.4317
   19.8333   17.5380   25.2566   14.1797   13.0306   20.8212   24.7717   16.7887   14.9375
    9.7013    6.7271   10.0347   11.6568   10.4355    6.1815    8.9264    9.3959   13.6272
   17.7446   14.8288   18.9281   11.8658   13.9552   19.2377   19.5071   17.1911   16.4723
   14.4870   22.4831   26.3129   25.2790   26.7125    9.0188   12.8705   20.0823   14.7828
    8.8277   11.4898    8.3437   11.1490   10.5637   12.2786    9.1483   11.2723   11.5864
   12.3049   15.4687   19.7918   15.3373   14.0741   16.3700   14.1747   16.3294   14.5957
   19.9267   21.8432   25.4493   21.8075   26.5935   11.8866   18.2030   20.9440   16.6593

>> Ab

Ab =

     1     1     0
    -1     0     0
     0    -1     1

>> Ab_edges

Ab_edges =

     1     2     6

>> Main
 Enter 1 if you want to feed data :
 Enter 2 if you want a random network and data to be generated
|
```

Figure 3: Example 1 - Step 2

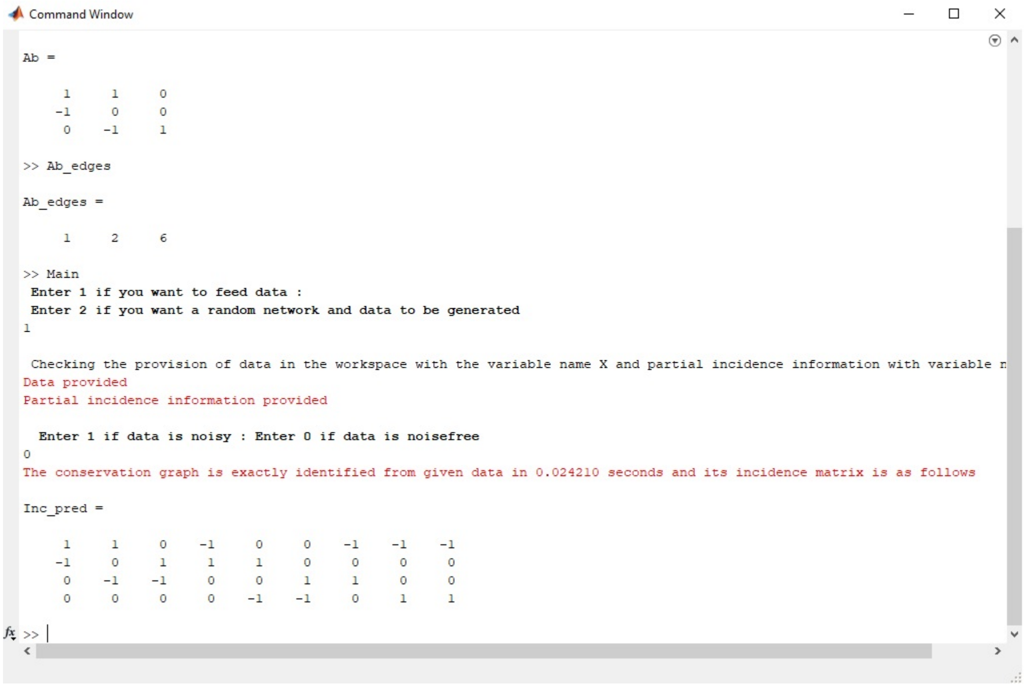
3. **Step 3:** Enter 1 to choose the ‘Feed data’ option. Programs checks for provision of data and confirms



```
     3.4814   16.4350    9.4000   22.2483   24.2455    0.4762   -2.7529   14.5658   11.4317
    19.8333   17.5380   25.2566   14.1797   13.0306   20.8212   24.7717   16.7887   14.9375
     9.7013    6.7271   10.0347   11.6568   10.4355    6.1815    8.9264    9.3959   13.6272
    17.7446   14.8288   18.9281   11.8658   13.9552   19.2377   19.5071   17.1911   16.4723
    14.4870   22.4831   26.3129   25.2790   26.7125    9.0188   12.8705   20.0823   14.7828
     8.8277   11.4898    8.3437   11.1490   10.5637   12.2786    9.1483   11.2723   11.5864
    12.3049   15.4687   19.7918   15.3373   14.0741   16.3700   14.1747   16.3294   14.5957
    19.9267   21.8432   25.4493   21.8075   26.5935   11.8866   18.2030   20.9440   16.6593

>> Ab

Ab =

     1     1     0
    -1     0     0
     0    -1     1

>> Ab_edges

Ab_edges =

     1     2     6

>> Main
 Enter 1 if you want to feed data :
 Enter 2 if you want a random network and data to be generated
1

 Checking the provision of data in the workspace with the variable name X and partial incidence information with variable n
Data provided
Partial incidence information provided

  Enter 1 if data is noisy : Enter 0 if data is noisefree
```

Figure 4: Example 1 - Step 3

4. **Step 4:** Enter 0 to inform that the data is noisefree. Then the rest of the code runs and the user is informed that the network is exactly identified from given data. The identified incidence matrix is displayed.



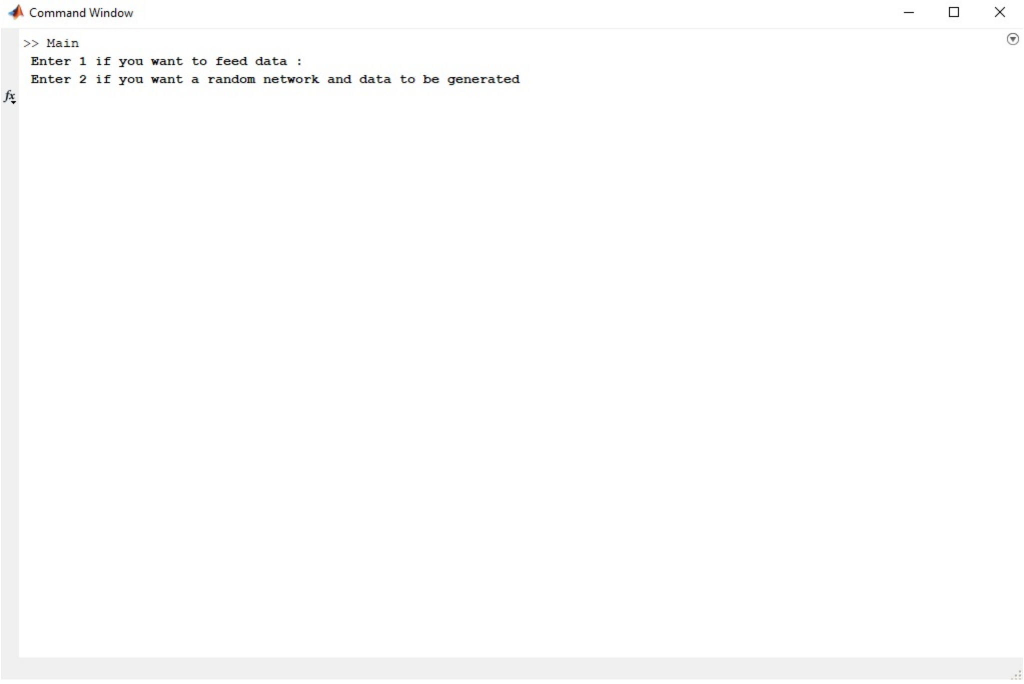
```
Ab =

     1     1     0
    -1     0     0
     0    -1     1

>> Ab_edges

Ab_edges =

     1     2     6

>> Main
 Enter 1 if you want to feed data :
 Enter 2 if you want a random network and data to be generated
1

 Checking the provision of data in the workspace with the variable name X and partial incidence information with variable n
Data provided
Partial incidence information provided

  Enter 1 if data is noisy : Enter 0 if data is noisefree
0
The conservation graph is exactly identified from given data in 0.024210 seconds and its incidence matrix is as follows

Inc_pred =

     1     1     0    -1     0     0    -1    -1    -1
    -1     0     1     1     1     0     0     0     0
     0    -1    -1     0     0     1     1     0     0
     0     0     0     0    -1    -1     0     1     1

>>
```

Figure 5: Example 1 - Step 4

6.2 Example 2

This example shows how once can generate an Erdos-Renyi network of 20 nodes and noisefree data with $2e$ number of samples pertaining to that network and then let the program identify topology from the generated data. The following are the steps taken and a snapshot of the MATLAB command line after each step is shown:

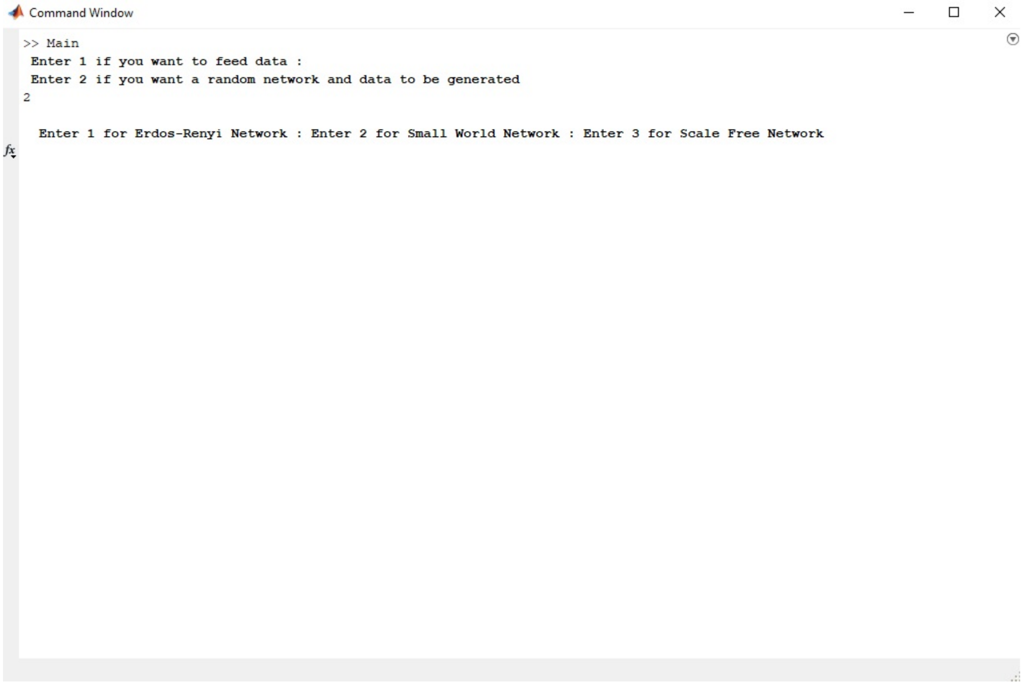
1. **Step 1:** Run the 'Main' function



```
>> Main
 Enter 1 if you want to feed data :
 Enter 2 if you want a random network and data to be generated
```

Figure 6: Example 2 - Step 1

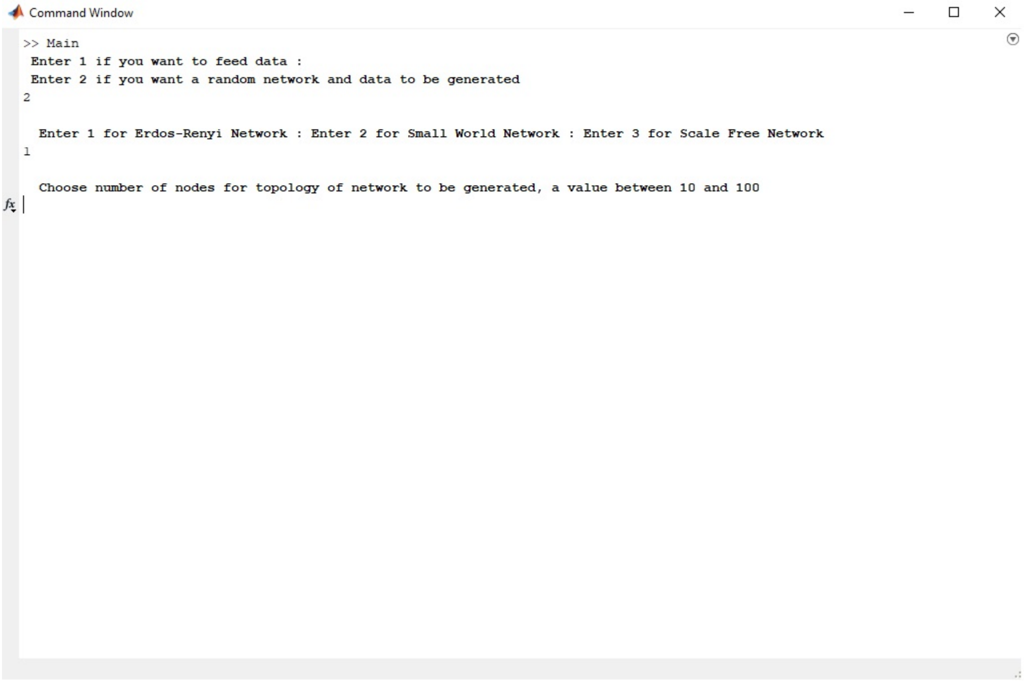
2. **Step 2:** Enter 2 to choose the 'Generate network and data' option.



```
>> Main
 Enter 1 if you want to feed data :
 Enter 2 if you want a random network and data to be generated
2

   Enter 1 for Erdos-Renyi Network : Enter 2 for Small World Network : Enter 3 for Scale Free Network
```

Figure 7: Example 2 - Step 2

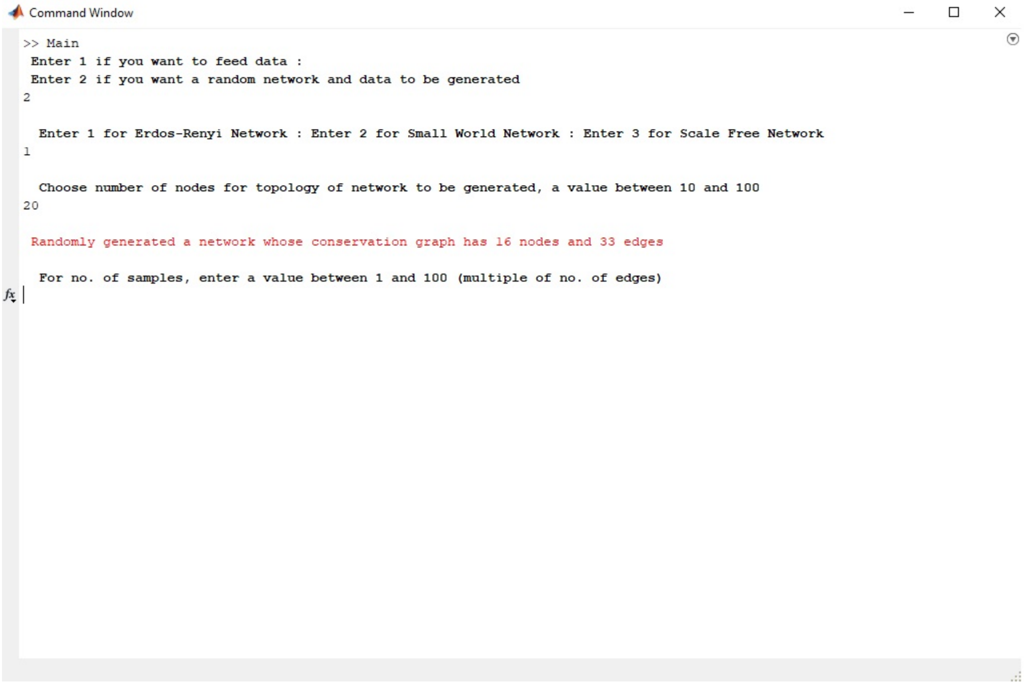
3. **Step 3:** Enter 1 to choose the ‘Erdos-Renyi Network’ option



```
>> Main
 Enter 1 if you want to feed data :
 Enter 2 if you want a random network and data to be generated
2

  Enter 1 for Erdos-Renyi Network : Enter 2 for Small World Network : Enter 3 for Scale Free Network
1

  Choose number of nodes for topology of network to be generated, a value between 10 and 100
fx |
```

Figure 8: Example 2 - Step 3

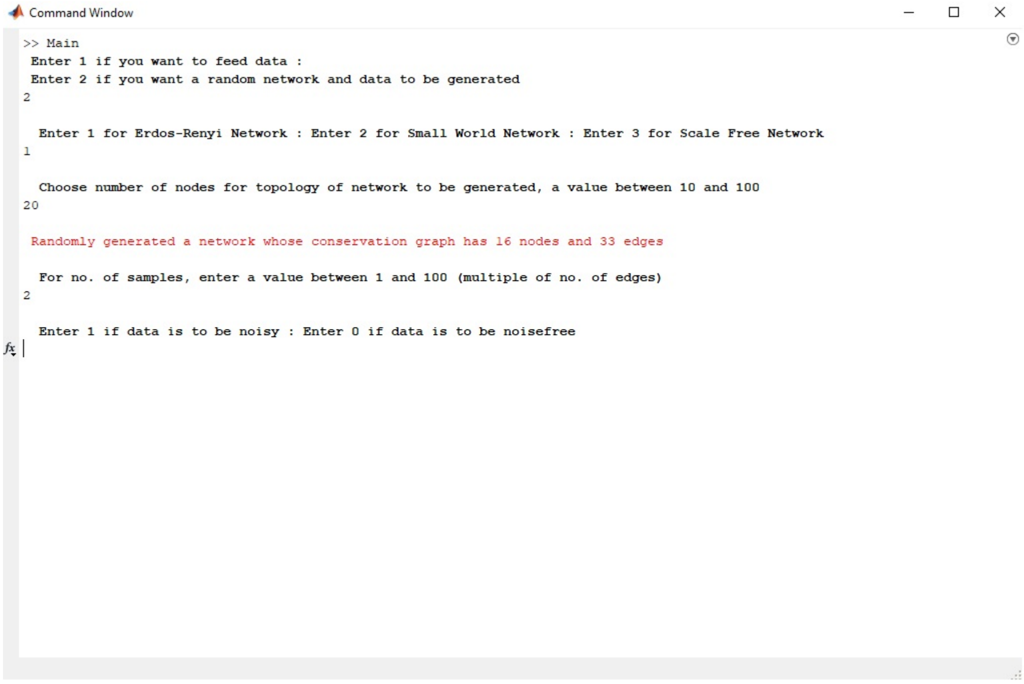
4. **Step 4:** Enter 20 to inform that the network should have 20 nodes. The program informs the a network is generated and its conservation graph is constructed with certain number of nodes and edges. It can be observed that the number of edges will be less in the conservation graph because of the merger of source and sink nodes.



```
>> Main
 Enter 1 if you want to feed data :
 Enter 2 if you want a random network and data to be generated
2

  Enter 1 for Erdos-Renyi Network : Enter 2 for Small World Network : Enter 3 for Scale Free Network
1

  Choose number of nodes for topology of network to be generated, a value between 10 and 100
20

 Randomly generated a network whose conservation graph has 16 nodes and 33 edges

  For no. of samples, enter a value between 1 and 100 (multiple of no. of edges)
fx |
```

Figure 9: Example 2 - Step 4

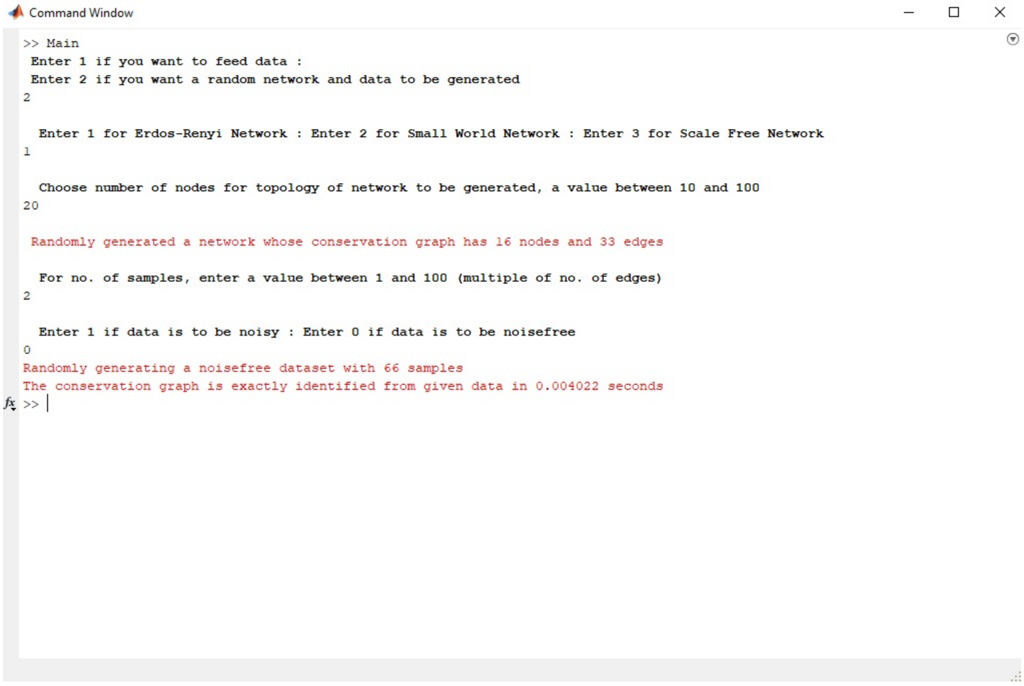
5. **Step 5:** Enter 2 to inform the code to generate $2e$ samples



```
>> Main
 Enter 1 if you want to feed data :
 Enter 2 if you want a random network and data to be generated
2

  Enter 1 for Erdos-Renyi Network : Enter 2 for Small World Network : Enter 3 for Scale Free Network
1

  Choose number of nodes for topology of network to be generated, a value between 10 and 100
20

 Randomly generated a network whose conservation graph has 16 nodes and 33 edges

  For no. of samples, enter a value between 1 and 100 (multiple of no. of edges)
2

  Enter 1 if data is to be noisy : Enter 0 if data is to be noisefree
```

Figure 10: Example 2 - Step 5

6. **Step 6:** Enter 0 to inform that the samples should be noisefree. The program generates a dataset with 66 samples. Then the program users this data to identify the topology and informs that the network identified matches the one which was generated.



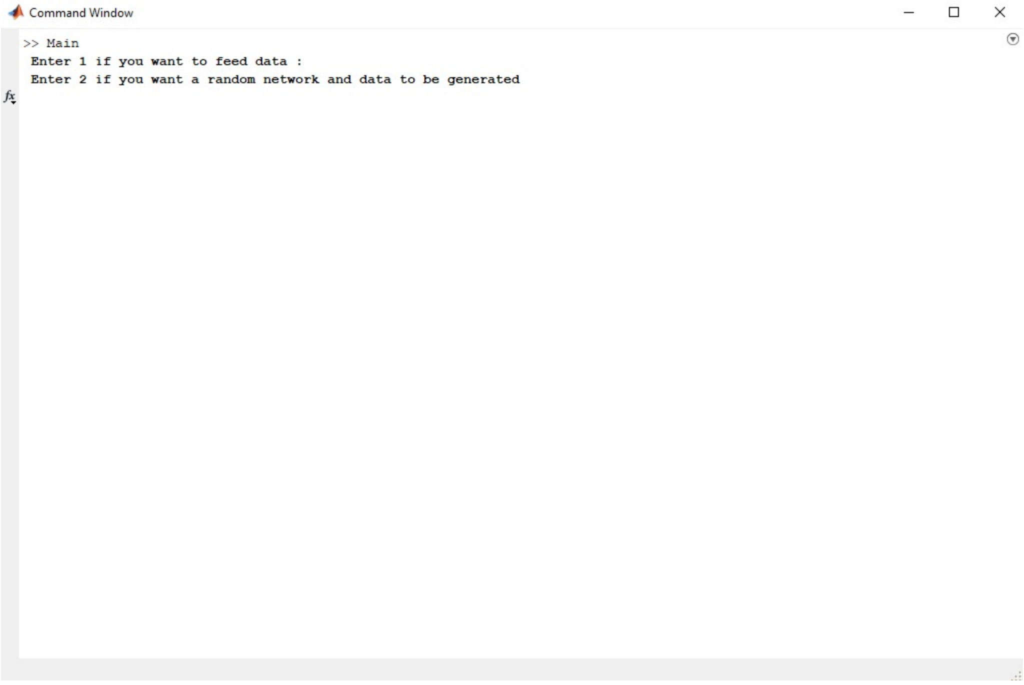
```
>> Main
 Enter 1 if you want to feed data :
 Enter 2 if you want a random network and data to be generated
2

  Enter 1 for Erdos-Renyi Network : Enter 2 for Small World Network : Enter 3 for Scale Free Network
1

  Choose number of nodes for topology of network to be generated, a value between 10 and 100
20

 Randomly generated a network whose conservation graph has 16 nodes and 33 edges

  For no. of samples, enter a value between 1 and 100 (multiple of no. of edges)
2

  Enter 1 if data is to be noisy : Enter 0 if data is to be noisefree
0
Randomly generating a noisefree dataset with 66 samples
The conservation graph is exactly identified from given data in 0.004022 seconds
>>
```

Figure 11: Example 2 - Step 6

6.3 Example 3

This example shows how once can generate a Small World network of 50 nodes. The code is asked to generate 10 datasets each of $4e$ samples with homoscedastic noise of SNR 5 pertaining to the network. Then the program goes on to identify topology from each of the generated datasets. The following are the steps taken and a snapshot of the MATLAB command line after each step is shown:

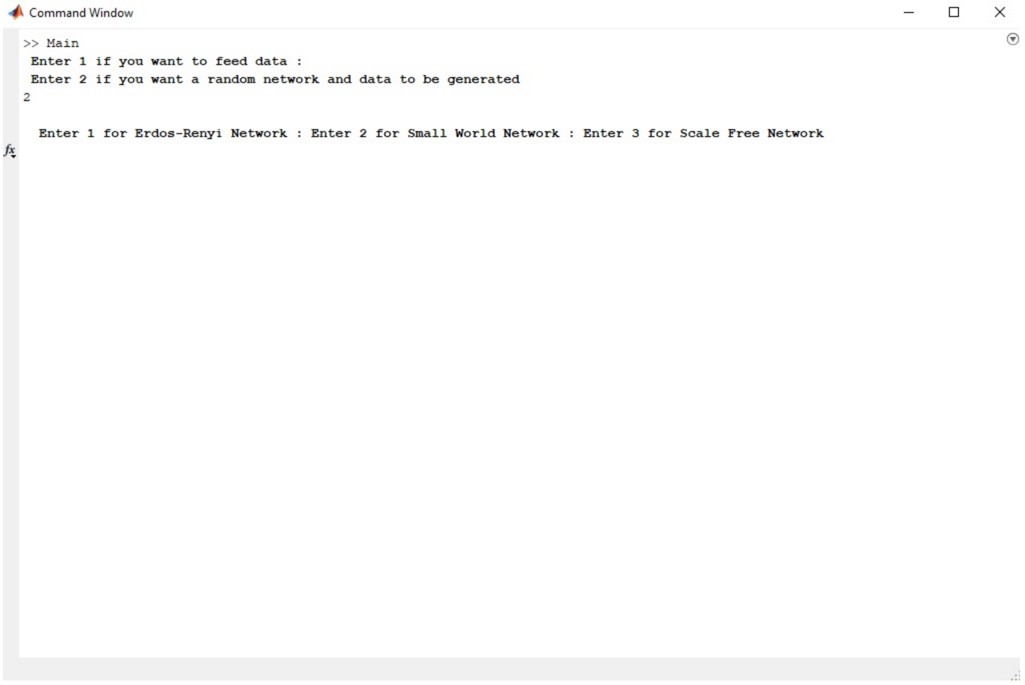
1. **Step 1:** Run the 'Main' function



```
>> Main
 Enter 1 if you want to feed data :
 Enter 2 if you want a random network and data to be generated
```

Figure 12: Example 3 - Step 1

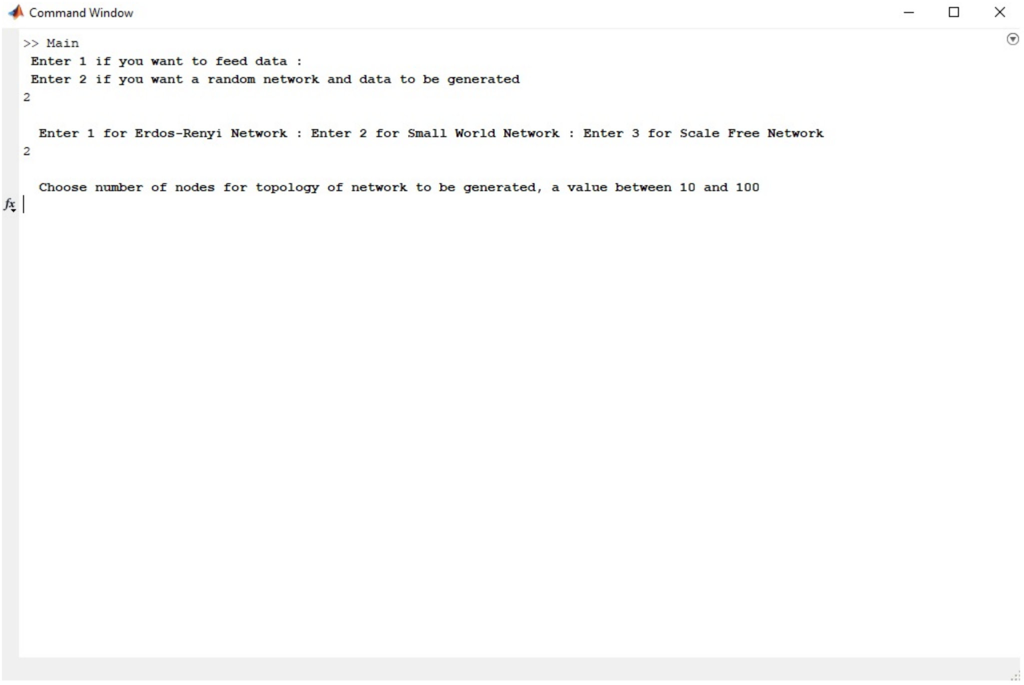
2. **Step 2:** Enter 2 to choose the 'Generate network and data' option.



```
>> Main
 Enter 1 if you want to feed data :
 Enter 2 if you want a random network and data to be generated
2

  Enter 1 for Erdos-Renyi Network : Enter 2 for Small World Network : Enter 3 for Scale Free Network
```

Figure 13: Example 3 - Step 2

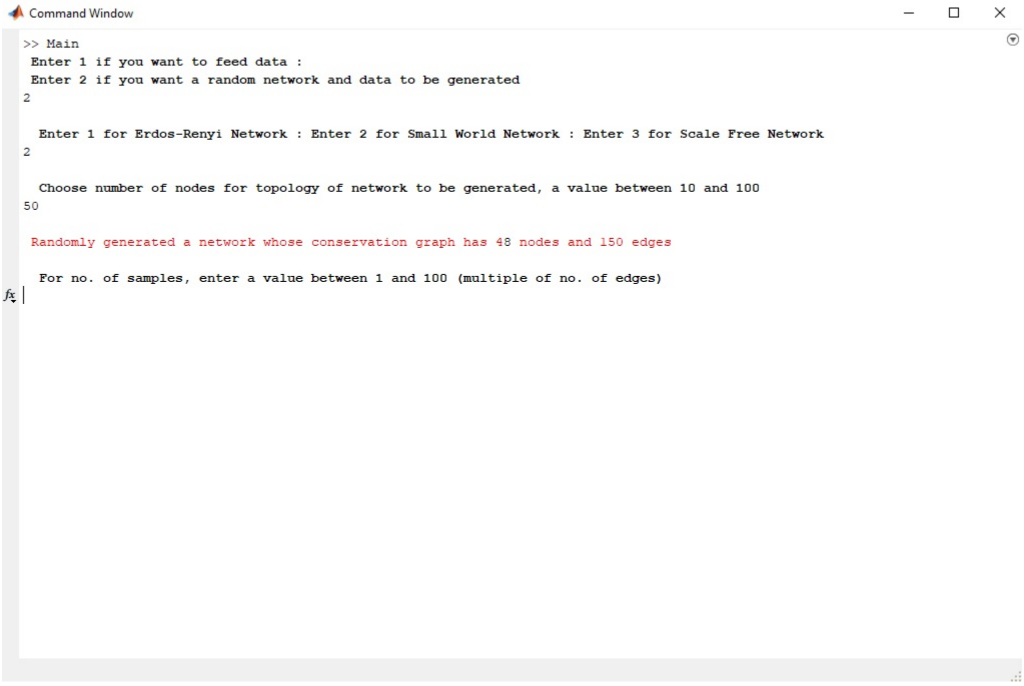
3. **Step 3:** Enter 2 to choose the ‘Small World network’ option



```
>> Main
 Enter 1 if you want to feed data :
 Enter 2 if you want a random network and data to be generated
2

  Enter 1 for Erdos-Renyi Network : Enter 2 for Small World Network : Enter 3 for Scale Free Network
2

  Choose number of nodes for topology of network to be generated, a value between 10 and 100
```

Figure 14: Example 3 - Step 3

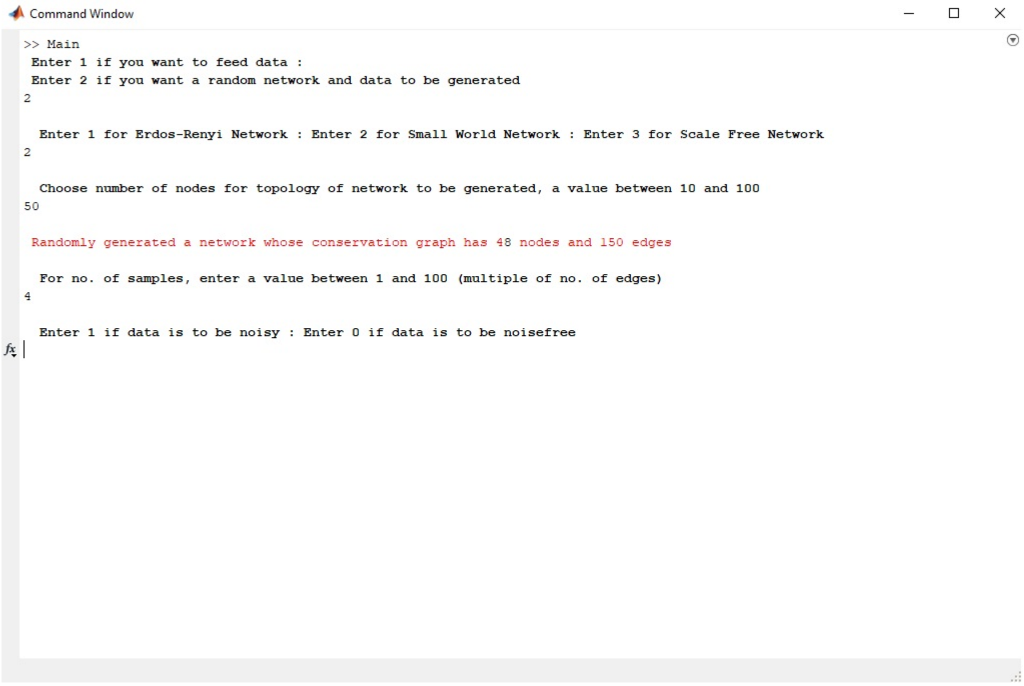
4. **Step 4:** Enter 50 to inform that the network should have 50 nodes. The program informs the a network is generated and its conservation graph is constructed with certain number of nodes and edges. It can be observed that the number of edges will be less in the conservation graph because of the merger of source and sink nodes.



```
>> Main
 Enter 1 if you want to feed data :
 Enter 2 if you want a random network and data to be generated
2

  Enter 1 for Erdos-Renyi Network : Enter 2 for Small World Network : Enter 3 for Scale Free Network
2

  Choose number of nodes for topology of network to be generated, a value between 10 and 100
50

 Randomly generated a network whose conservation graph has 48 nodes and 150 edges

  For no. of samples, enter a value between 1 and 100 (multiple of no. of edges)
fx |
```

Figure 15: Example 3 - Step 4

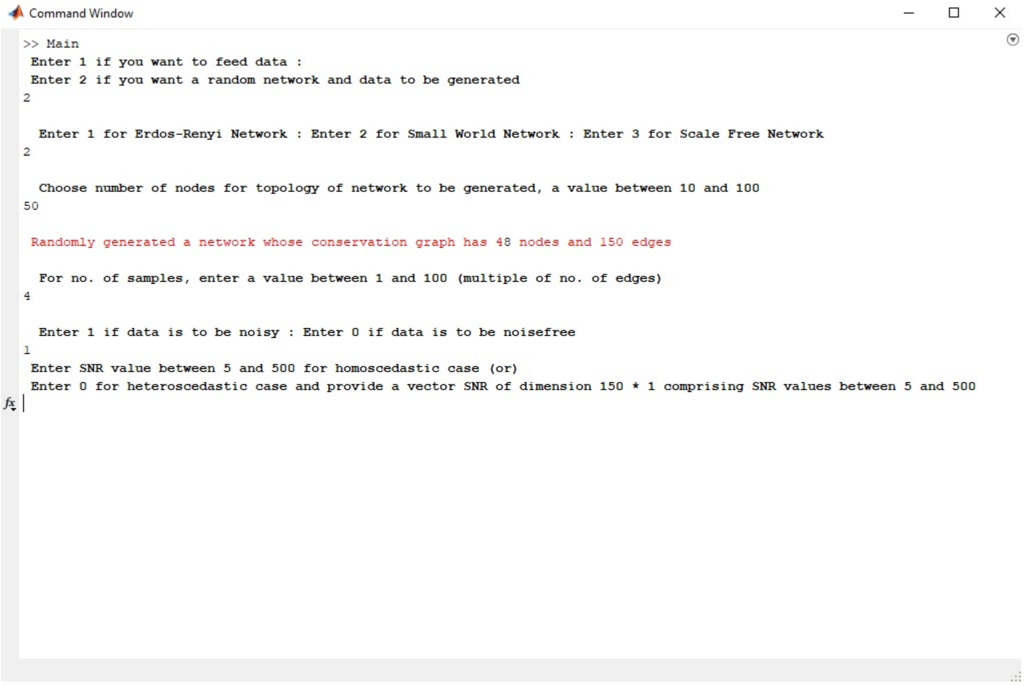
5. **Step 5:** Enter 4 to inform the code to generate $4e$ samples



```
>> Main
 Enter 1 if you want to feed data :
 Enter 2 if you want a random network and data to be generated
2

  Enter 1 for Erdos-Renyi Network : Enter 2 for Small World Network : Enter 3 for Scale Free Network
2

  Choose number of nodes for topology of network to be generated, a value between 10 and 100
50

 Randomly generated a network whose conservation graph has 48 nodes and 150 edges

  For no. of samples, enter a value between 1 and 100 (multiple of no. of edges)
4

  Enter 1 if data is to be noisy : Enter 0 if data is to be noisefree
fx |
```

Figure 16: Example 3 - Step 5

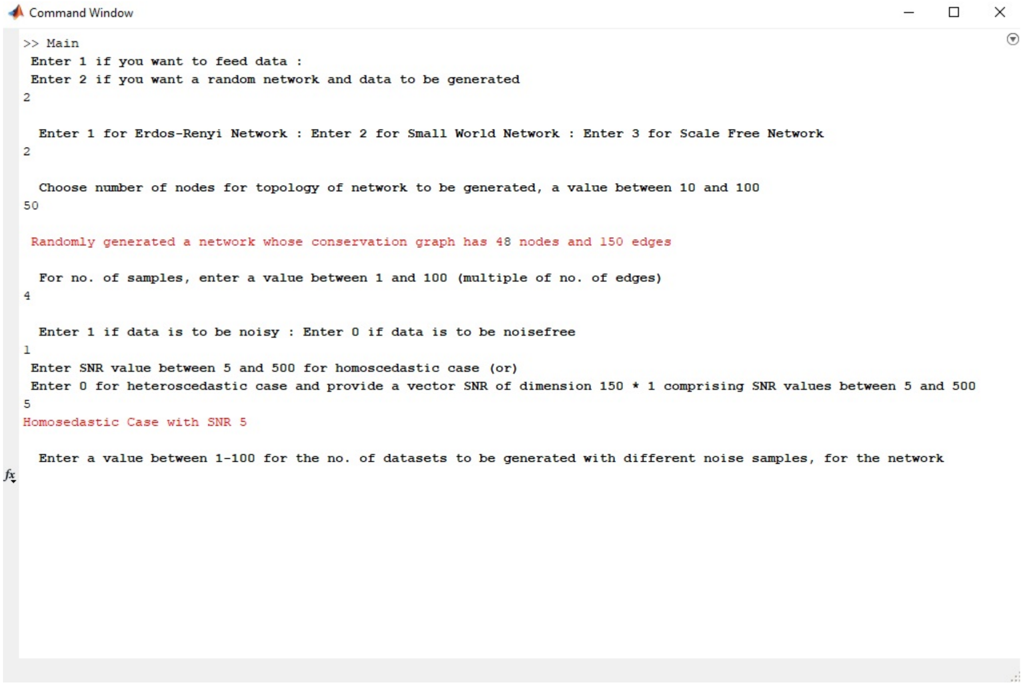
6. **Step 6:** Enter 1 to inform that the samples should be noisy



```
>> Main
 Enter 1 if you want to feed data :
 Enter 2 if you want a random network and data to be generated
2

  Enter 1 for Erdos-Renyi Network : Enter 2 for Small World Network : Enter 3 for Scale Free Network
2

  Choose number of nodes for topology of network to be generated, a value between 10 and 100
50

 Randomly generated a network whose conservation graph has 48 nodes and 150 edges

  For no. of samples, enter a value between 1 and 100 (multiple of no. of edges)
4

  Enter 1 if data is to be noisy : Enter 0 if data is to be noisefree
1
 Enter SNR value between 5 and 500 for homoscedastic case (or)
 Enter 0 for heteroscedastic case and provide a vector SNR of dimension 150 * 1 comprising SNR values between 5 and 500
fx |
```

Figure 17: Example 3 - Step 6

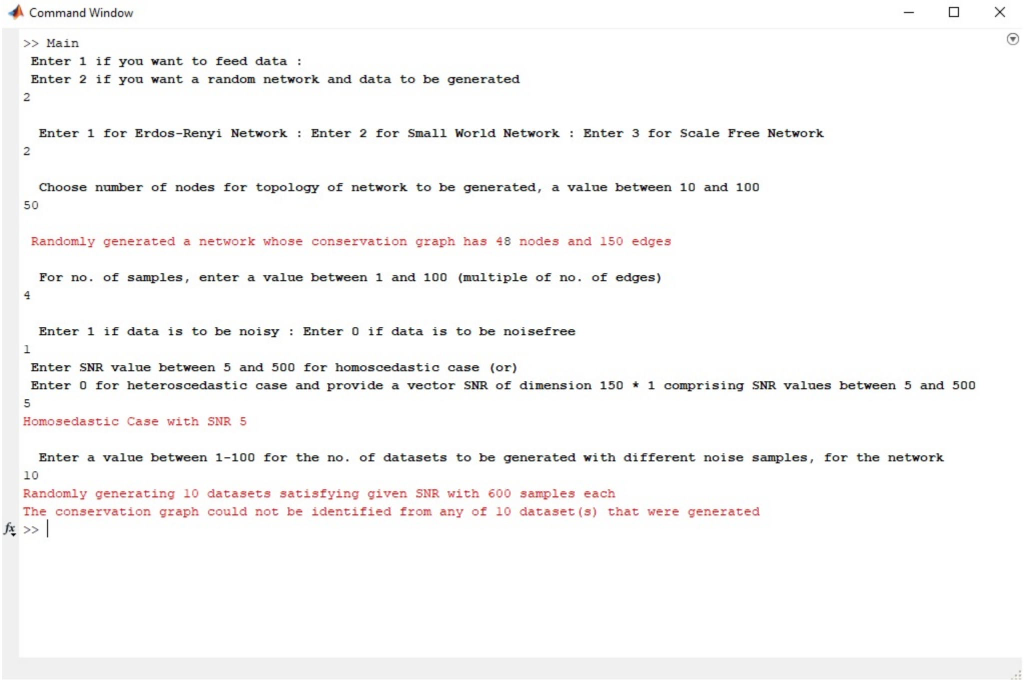
7. **Step 7:** Enter 5 to inform that homoscedastic noise of SNR 5 is to be added to the samples



```
>> Main
 Enter 1 if you want to feed data :
 Enter 2 if you want a random network and data to be generated
2

  Enter 1 for Erdos-Renyi Network : Enter 2 for Small World Network : Enter 3 for Scale Free Network
2

  Choose number of nodes for topology of network to be generated, a value between 10 and 100
50

 Randomly generated a network whose conservation graph has 48 nodes and 150 edges

  For no. of samples, enter a value between 1 and 100 (multiple of no. of edges)
4

  Enter 1 if data is to be noisy : Enter 0 if data is to be noisefree
1
 Enter SNR value between 5 and 500 for homoscedastic case (or)
 Enter 0 for heteroscedastic case and provide a vector SNR of dimension 150 * 1 comprising SNR values between 5 and 500
5
Homosedastic Case with SNR 5

  Enter a value between 1-100 for the no. of datasets to be generated with different noise samples, for the network
fx
```

Figure 18: Example 3 - Step 7

8. **Step 8:** Enter 10 to inform that that 10 datasets are to be generated. The program generates 10 datasets each with 600 samples each. Then the program tries to identify the topology from each of the datasets independently. It concludes that the network could not be identified correctly due to insufficient samples for the chosen SNR.



```
>> Main
 Enter 1 if you want to feed data :
 Enter 2 if you want a random network and data to be generated
2

  Enter 1 for Erdos-Renyi Network : Enter 2 for Small World Network : Enter 3 for Scale Free Network
2

  Choose number of nodes for topology of network to be generated, a value between 10 and 100
50

 Randomly generated a network whose conservation graph has 48 nodes and 150 edges

  For no. of samples, enter a value between 1 and 100 (multiple of no. of edges)
4

  Enter 1 if data is to be noisy : Enter 0 if data is to be noisefree
1
 Enter SNR value between 5 and 500 for homoscedastic case (or)
 Enter 0 for heteroscedastic case and provide a vector SNR of dimension 150 * 1 comprising SNR values between 5 and 500
5
Homosedastic Case with SNR 5

  Enter a value between 1-100 for the no. of datasets to be generated with different noise samples, for the network
10
Randomly generating 10 datasets satisfying given SNR with 600 samples each
The conservation graph could not be identified from any of 10 dataset(s) that were generated
>> 
```

Figure 19: Example 3 - Step 8

6

SECTION

References

[1] Satya Jayadev, P., Narasimhan, S., and Bhatt, N. (2019). Learning Conserved Networks from Flows. arXiv preprint arXiv:1905.08716.

[2] Bondy, J.A. and Murty, U.S.R., 1976. Graph theory with applications (Vol. 290). London: Macmillan.

[3] Deo, N., 2017. Graph theory with applications to engineering and computer science. Courier Dover Publications.

[4] Chen, W.K., 1997. Graph theory and its engineering applications (Vol. 5). World Scientific Publishing Company.

[5] Fujishige, S., 1980. An efficient PQ-graph algorithm for solving the graph-realization problem. Journal of Computer and System Sciences, 21(1), pp.63-86.

[6] Bixby, R.E. and Wagner, D.K., 1988. An almost linear-time algorithm for graph realization. Mathematics of operations research, 13(1), pp.99-123.

[7] Whitney, H., 1933. 2-isomorphic graphs. American Journal of Mathematics, 55(1), pp.245-254.

[8] Narasimhan, S. and Bhatt, N., 2015. Deconstructing principal component analysis using a data reconciliation perspective. Computers & Chemical Engineering, 77, pp.74-84.

[9] Jolliffe, I.T., 1986. Principal components in regression analysis. In Principal component analysis (pp. 129-155). Springer, New York, NY.

[10] Erdös, P and Rényi, A (1959). On random graphs i. Publ. Math. Debrecen, 6(290-297), 18.

[11] Watts, D. J., and Strogatz, S. H. (1998). Collective dynamics of 'small-world'networks. nature, 393(6684), 440.

[12] Albert, R., and Barabási, A. L. (2002). Statistical mechanics of complex networks. Reviews of modern physics, 74(1), 47.

[13] Pablo Blinder (2020). Erdos-Renyi Random Graph (https://www.mathworks.com/matlabcentral/fileexchange/4206-erdos-renyi-random-graph), MATLAB Central File Exchange. Retrieved March 3, 2020.

[14] Tapan (2020). Scale free network using B-A algorithm (https://www.mathworks.com/matlabcentral/fileexchange/49356-scale-free-network-using-b-a-algorithm), MATLAB Central File Exchange. Retrieved March 3, 2020.