IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

COMPLETE UNDIRECTED GRAPH IN CODING

[1]S Hemamalini [2]Dr.R.Malathi
[1]Author & BE CSE Final year
[1]SCSVMV, Enathur, Kanchipuram,
Tamilnadu, India

Abstract: The spectral graph theory studies the properties of graphs via the eigenvalues and eigenvectors of their associated graph matrices: the adjacency matrix and the graph Laplacian and its variants. Both matrices have been extremely well studied from an algebraic point of view. Using Turbo C++, we also have coded in C++ programming language and executed the Laplacian Matrix. The Laplacian allows a natural link between discrete representations, such as graphs, and continuous representations, such as vector spaces and manifolds. Given a complete undirected graph G with n vertices, its Laplacian of a diagonal matrix L_{nxn} is defined as L= D-A, Where A is the adjacency matrix of the graph and D is the diagonal matrix of A.

Keywords: Complete graph, Laplacian matrix, adjacency matrix, diagonal matrix, coding and C++.

AMS Classification : 05C50; 15A48

1. Introduction

Informally, a graph is a diagram consisting of points, called vertices, joined together by lines, called edges; each edge joins exactly two vertices. A graph G is a triple consisting of a vertex set of V(*G*), an edge set E(G), and a relation that associates with each edge two vertices (not necessarily distinct) called its endpoints.

Graph

A **graph** (sometimes called undirected graph for distinguishing from a directed graph, or simple graph for distinguishing from a multigraph) is a pair G = (V, E), where V is a set whose elements are called vertices (singular: vertex), and E is a set of two-sets (sets with two distinct elements) of vertices, whose elements are called edges (sometimes links or lines)[2].

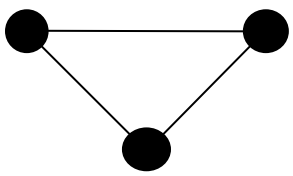


Fig: A graph with three vertices and three edges

The vertices x and y of an edge {x, y} are called the endpoints of the edge. The edge is said to join x and y and to be incident on x and y. A vertex may not belong to any edge.
A multigraph is a generalization that allows multiple edges adjacent to the same pair of vertices. In some texts, multigraphs are simply called graphs.

Sometimes, graphs are allowed to contain loops, which are edges that join a vertex to itself. For allowing loops, the above definition must be changed by defining edges as multisets of two vertices instead of two-sets. Such generalized graphs are called graphs with loops or simply graphs when it is clear from the context that loops are allowed.

Directed graph

A **directed graph** or **digraph** is a graph in which edges have orientations.

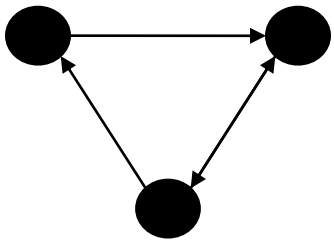


Fig: A directed graph with three vertices and four directed edges

In one restricted but very common sense of the term, a **directed graph** is an ordered pair G = (V, E) comprising:

- V a set of vertices (also called nodes or points);
- $E \subseteq \{(x, y) | (x, y) \in V^2 \wedge x \neq y\}$ a set of edges (also called directed edges, directed links, directed lines, arrows or arcs) which are ordered pairs of distinct vertices (i.e., an edge is associated with two distinct vertices).

Directed graphs as defined in the two definitions above cannot have loops, because a loop joining a vertex x is the edge (for a directed simple graph) or is incident on (for a directed multigraph) (x, x) which is not in $\{(x, y) \mid (x, y) \in V^2 \wedge x \neq y\}$. So to allow loops the definitions must be expanded. For directed simple graphs, $E \subseteq \{(x, y) \mid (x, y) \in V^2 \wedge x \neq y\}$ should become $E \subseteq V^2$. For directed multigraphs, $\phi: E \to \{(x, y) \mid (x, y) \in V^2 \wedge x \neq y\}$ should become $\phi: E \to V^2$. To avoid ambiguity, these types of objects may be called precisely a **directed simple graph permitting loops** and a **directed multigraph permitting loops** (or a quiver) respectively.

The edges of a directed simple graph permitting loops G is a homogeneous relation ~ on the vertices of G that is called the adjacency relation of G. Specifically, for each edge (x, y), its endpoints x and y are said to be adjacent to one another, which is denoted x ~ y.

Mixed graph

A mixed graph is a graph in which some edges may be directed and some may be undirected. It is an ordered triple G = (V, E, A) for a mixed simple graph and G = (V, E, A, ϕ_E, ϕ_A) for a mixed multigraph with V, E (the undirected edges), A (the directed edges), ϕ_E and ϕ_A defined as above. Directed and undirected graphs are special cases.

Complete graph

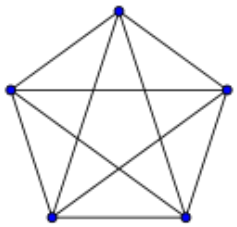


Fig: A complete graph with five vertices and ten edges.

Each vertex has an edge to every other vertex. A complete graph is a graph in which each pair of vertices is joined by an edge. A complete graph contains all possible edges.

Finite graph

A finite graph is a graph in which the vertex set and the edge set are finite sets. Otherwise, it is called an infinite graph.
Most commonly in graph theory it is implied that the graphs discussed are finite. If the graphs are infinite, that is usually specifically stated.

Bipartite graph

A bipartite graph is a simple graph in which the vertex set can be partitioned into two sets, W and X, so that no two vertices in W share a common edge and no two vertices in X share a common edge. Alternatively, it is a graph with a chromatic number of 2.
In a complete bipartite graph, the vertex set is the union of two disjoint sets, W and X, so that every vertex in W is adjacent to every vertex in X but there are no edges within W or X.

Path graph

A path graph or linear graph of order $n \geq 2$ is a graph in which the vertices can be listed in an order $v_1, v_2, ..., v_n$ such that the edges are the $\{v_i, v_{i+1}\}$ where $i = 1, 2, ..., n - 1$. Path graphs can be characterized as connected graphs in which the degree of all but two vertices is 2 and the degree of the two remaining vertices is 1. If a path graph occurs as a subgraph of another graph, it is a path in that graph.

Planar graph

A planar graph is a graph whose vertices and edges can be drawn in a plane such that no two of the edges intersect.

Adjacency matrix

In graph theory and computer science, an **adjacency matrix** is a square matrix used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices are adacent or not in the graph.

In the special case of a finite simple graph, the adjacency matrix is a (0, 1) - matrix with zeros on its diagonal. If the graph is undirected, the adjacency matrix is symmetric. The relationship between a graph and the eigen values and eigen vectors of its adjacency matrix is studied in spectral graph theory.

The adjacency matrix should be distinguished from the incidence matrix for a graph, a different matrix representation whose elements indicate whether vertex–edge pairs are incident or not, and degree matrix which contains information about the degree of each vertex.

Incidence matrix

Define an $|e| \times |v|$ oriented incidence matrix M with element M_{ev} for edge e (connecting vertex i and j, with $i > j$) and vertex v given by

$$M_{ev} = \begin{cases} 1, & \text{if } v = i \\ -1, & \text{if } v = j \\ 0, & \text{otherwise} \end{cases}$$

Diagonal matrix

In linear algebra, a **diagonal matrix** is a matrix in which the entries outside the main diagonal are all zero; the term usually refers to square matrices. An example of a 2-by-2 diagonal matrix is $\begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}$, while an example of a 3-by-3 diagonal matrix is $\begin{bmatrix} 6 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 4 \end{bmatrix}$. An identity matrix of any size, or any multiple of it (a scalar matrix), is a diagonal matrix.

A diagonal matrix is sometimes called a scaling matrix, since matrix multiplication with it results in changing scale (size). Its determinant is the product of its diagonal values.

The term *diagonal matrix* may sometimes refer to a **rectangular diagonal matrix**, which is an *m*-by-*n* matrix with all the entries not of the form $d_{i,i}$ being zero. For example:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & -3 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & -3 & 0 & 0 \end{bmatrix}$$

More often, however, *diagonal matrix* refers to square matrices, which can be specified explicitly as a **square diagonal matrix**. A square diagonal matrix is a symmetric matrix, so this can also be called a **symmetric diagonal matrix**.

Laplacian matrix for *simple graphs*

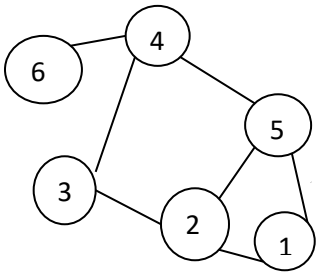
Given a simple graph G with n vertices, its Laplacian matrix L_{nxn} is defined as:

L = D – A where *D* is the degree matrix and *A* is the adjacency matrix of the graph.Since G is a simple graph, A only contains 1s or 0s and its diagonal elements are all 0s.

In the case of directed graphs, either the indegree or outdegree might be used, depending on the application.

Here is a simple example of a labeled, undirected graph and its Laplacian matrix[7].

Labelled graph:



Degree matrix: **Adjacency matrix:** **Laplacian matrix:**

$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$

2. Experimental Details

Given a completed graph G with n vertices, its Laplacian matrix L_{nxn} is defined as: L = D – A where *D* is the diagonal matrix and *A* is the adjacency matrix of the graph. Let us see the results of Laplacian through matlab outputs.

$$A=\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

```
A=[0 1;1 0]

A =

     0     1
     1     0

>>eig(A)

ans =

    -1
     1

>> [V D] = eig(A)

V =

   -0.7071    0.7071
    0.7071    0.7071
```

```
D =

    -1     0
     0     1

>> L = D-A

L =

    -1    -1
    -1     1

B= [0 1 1;1 0 1;1 1 0]

B =

     0     1     1
     1     0     1
     1     1     0

>>eig (B)

ans =

   -1.0000
   -1.0000
    2.0000

>> [V D] = eig(B)

V =

   -0.7152    0.3938    0.5774
    0.0166   -0.8163    0.5774
    0.6987    0.4225    0.5774

D =

   -1.0000         0         0
         0   -1.0000         0
         0         0    2.0000

>> L= D-B

L =

   -1.0000   -1.0000   -1.0000
   -1.0000   -1.0000   -1.0000
   -1.0000   -1.0000    2.0000

C=[0 1 1 1;1 0 1 1;1 1 0 1;1 1 1 0]
```

```
C =

     0     1     1     1
     1     0     1     1
     1     1     0     1
     1     1     1     0

>>eig(C)

ans =

   -1.0000
   -1.0000
   -1.0000
    3.0000

>> [V D] = eig(C)

V =

    0.7887   -0.2113    0.2887    0.5000
   -0.2113    0.7887    0.2887    0.5000
   -0.5774   -0.5774    0.2887    0.5000
         0         0   -0.8660    0.5000

D =

   -1.0000         0         0         0
         0   -1.0000         0         0
         0         0   -1.0000         0
         0         0         0    3.0000

>> L= D-C

L =

   -1.0000   -1.0000   -1.0000   -1.0000
   -1.0000   -1.0000   -1.0000   -1.0000
   -1.0000   -1.0000   -1.0000   -1.0000
   -1.0000   -1.0000   -1.0000    3.0000

E= [0 1 1 1 1;1 0 1 1 1;1 1 0 1 1;1 1 1 0 1;1 1 1 1 0]

E =

     0     1     1     1     1
     1     0     1     1     1
     1     1     0     1     1
     1     1     1     0     1
     1     1     1     1     0

>>eig(E)
```

```
ans =

   -1.0000
   -1.0000
   -1.0000
   -1.0000
    4.0000

>> [V D] = eig(E)


V =

    0.1119   -0.2113    0.7887    0.3476   -0.4472
    0.1119    0.7887   -0.2113    0.3476   -0.4472
    0.1119   -0.5774   -0.5774    0.3476   -0.4472
    0.5053   -0.0000   -0.0000   -0.7380   -0.4472
   -0.8409         0         0   -0.3048   -0.4472

D =

   -1.0000         0         0         0         0
         0   -1.0000         0         0         0
         0         0   -1.0000         0         0
         0         0         0   -1.0000         0
         0         0         0         0    4.0000

>> L= D-E

L =

   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000
   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000
   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000
   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000
   -1.0000   -1.0000   -1.0000   -1.0000    4.0000
```

3. Results and discussion

Let us now see theoutput for computation of Laplacian Diagonal Matrix using coding.

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
void main()
{
clrscr();
int a[3][3],d[3][3],l[3][3],s1,s2,s3,e,f,n,p,q,r;
double dis,r1,r2,r3;
float rp,ip;
cout<<"Enter the value of n in nxn matrix: ";
cin>>n;
for(int i=0;i<n;i++)
```

```
{
for(int j=0;j<n;j++)
{
if(i==j)
a[i][j]=0;
else
a[i][j]=1;
}
}
cout<<"\nThe A matrix is:"<<endl;
for(int ii=0;ii<n;ii++)
{
for(int jj=0;jj<n;jj++)
{
cout<<a[ii][jj]<<"  ";
}
cout<<endl;
cout<<" ";
}
cout<<endl;
s1=a[0][0]+a[1][1]+a[2][2];
s2=((a[1][1]*a[2][2])-(a[2][1]*a[1][2]))+((a[0][0]*a[2][2])-(a[2][0]*a[0][2]))+((a[1][1]*a[0][0])-
(a[1][0]*a[0][1]));
s3=(a[0][0]*((a[1][1]*a[2][2])-(a[2][1]*a[1][2])))-(a[0][1]*((a[1][0]*a[2][2])-
(a[2][0]*a[1][2])))+(a[0][2]*((a[1][0]*a[2][1])-(a[2][0]*a[1][1])));
f=0;
do
{
f++;
e=(f*f*f)-(s1*f*f)+(s2*f)-s3;
}while(e!=0);
p=1;
q=s1+(f*p);
r=s2+(f*q);
r3=f;
dis=(q*q)-(4*p*r);
if(dis>0)
{
r1=(-q+sqrt(dis))/(2*p);
r2=(-q-sqrt(dis))/(2*p);
cout<<"The roots are: "<<r1<<", "<<r2<<" and "<<r3;
}
else if(dis==0)
{
r1=r2=(-q+sqrt(dis))/(2*p);
cout<<"The roots are: "<<r1<<", "<<r2<<" and "<<r3;
}
else
{
float xx=-q;
float yy=2*p;
rp=xx/yy;
ip=sqrt(-dis)/(2*p);
cout<<"The roots are: "<<rp<<" + "<<ip<<"i, "<<rp<<" - "<<ip<<"i, "<<r3;
```

```
}
cout<<endl;
d[0][0]=r1;
d[1][1]=r2;
d[2][2]=r3;
d[0][1]=d[0][2]=d[1][0]=d[1][2]=d[2][0]=d[2][1]=0;
cout<<"\nThe diagonal matrix is: "<<endl;
for(int g=0;g<n;g++)
{
for(int h=0;h<n;h++)
{
cout<<d[g][h]<<"  ";
}
cout<<endl;
cout<<" ";
}
cout<<"\nThe Laplacian Matrix is: "<<endl;
for(int u=0;u<n;u++)
{
for(int v=0;v<n;v++)
{
l[u][v]=d[u][v]-a[u][v];
cout<<l[u][v]<<"  ";
}
cout<<endl;
}
getch();
}
```

Output:

```
Enter the value of n in nxn matrix: 3

The A matrix is:
0  1  1
1  0  1
1  1  0

The roots are: -1, -1 and 2

The diagonal matrix is:
-1  0  0
0  -1  0
0  0  2

The Laplacian Matrix is:
-1  -1  -1
-1  -1  -1
-1  -1  2
```

4. Conclusion

The adjacency matrix A(G) of G is annxn matrix with its rows and columns indexed by V(G) and with the (i, j) – entry equal to 1 if vertices i, j are adjacent and 0 otherwise. Thus A(G) is a symmetric matrix with its i-throw(or column) sum equal to $d_i(G)$, which by definition is the degree of the vertex i, i=1,2,…,n. The Laplacian value of nxn matrix is -1 except for (n,n) and the value of (n, n) is n-1.We also verified the same from the output we got from coding.

5. References

[1] Godsil, Chris; Royle,**2001**.Gordon Algebraic Graph Theory.*Springer*, ISBN 0-387-95241-1, pp.164
[2] Goodrich &Tamassia**2015**.There are two data structures that people often use to represent graphs, the adjacency list and the adjacency matrix.pp. 361
[3] Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L.; Stein, Clifford **2001**, Section 22.1, Representations of graphs, Introduction to Algorithms (Second ed.), *MIT Press and McGraw-Hill*,ISBN 0-262-03293-7, pp. 527–531
[4] Turán, György**1984**. On the succinct representation of graphs, *Discrete Applied Mathematics*, 8 (3), doi:10.1016/0166-218X(84)90126-4, MR 0749658, pp.289–294
[5] McKay, Brendan, Description of graph6 and sparse6 encodings.
[6] Goodrich, Michael T, Tamassia, Roberto **2015**, Algorithm Design and Applications, Wiley, pp. 363.
[7] Weisstein, Eric W. Laplacian Matrix. Math World.
[8] Godsil, C.; Royle, G. **2001**. Algebraic Graph Theory, Graduate Texts in Mathematics. *Springer-Verlag.*
[9] Morbidi, F. **2013.** The Deformed Consensus Protocol. Automatica. 49 (10), doi: 10.1016/j.automatica.2013.07.006. pp.3049–3055
[10] Cvetkovic, Dragos, Simic, Slobodan K. **2010**. Towards A Spectral Theory Of Graphs Based On The Signless Laplacian, III. *Applicable Analysis and Discrete Mathematics*. 4 (1),ISSN 1452-8630.pp.156–166.