Matrix Tree Theorem

Ahmad Sadaan

June 2026

Abstract

This paper introduces the Matrix Tree Theorem in a careful but approachable way. The theorem counts the spanning trees of a finite connected graph by taking a determinant from its Laplacian matrix. We begin with the needed ideas from graph theory and linear algebra, then state the theorem, work through a full example, and look at several different proofs, including deletion–contraction, Cauchy–Binet, random walks, generating functions, and homology. The paper ends by discussing applications to networks, circuits, Markov chains, machine learning, quantum networks, and several important extensions.

1 Introduction

Many systems in the real world naturally look like networks: communication systems, electrical circuits, quantum networks, and even social relationships. In mathematics, these systems are often modeled using **graphs**. A graph is made of vertices, joined together by edges. Inside a network like this, a **spanning tree** keeps all the vertices connected while removing every cycle. That brings us to a natural question in graph theory: how many different spanning trees can a graph have?

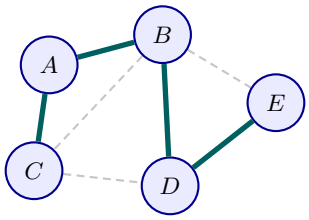


Figure 1: A simple graph with one spanning tree highlighted in teal.

At first, it may seem that counting spanning trees means checking many possible choices of edges by hand. What makes the theorem powerful is that linear algebra gives a much cleaner route. The **Matrix Tree Theorem**, first proved by Gustav Kirchhoff in 1847 while studying electrical circuits, gives the

number of spanning trees of a graph as the determinant of a matrix obtained from the graph's **Laplacian matrix** [1].

This bridge between graphs and matrices is why the theorem appears in electrical network analysis, computer science, probability theory, chemistry, network theory, and spectral graph theory.

The theorem is important because it turns a combinatorial counting problem into a linear-algebra calculation. It is practical as well: instead of listing every spanning tree one by one, which quickly becomes impossible for large graphs, we only need to compute a determinant.

Informally, the theorem says this: if G is a finite connected graph and L is its Laplacian matrix, then deleting any one row and the corresponding column of L gives a matrix whose determinant is exactly the number of spanning trees of G. A formal statement appears in Section 3.

Here is how the rest of the paper is organized. Section 2 introduces the necessary graph theory and linear algebra. Section 3 states the Matrix Tree Theorem. Section 4 gives a complete worked example. Section 5 presents several proofs of the theorem. Section 6 discusses applications, especially to networks and machine learning. Section 7 describes extensions and related directions.

2 Background

This section builds the background needed for the theorem. We start with the basic language of graphs and then move toward the matrix ideas that appear in the Matrix Tree Theorem.

2.1 Graphs

A graph is a simple way to describe how objects are connected. The objects are represented by vertices, and the connections between them are represented by edges.

Graphs are useful because many complicated situations, such as transportation systems, social networks, and computer networks, become easier to understand once they are drawn as vertices and edges.

Formally, we write a graph as $G = (V, E)$, where V is the set of vertices and E is the set of edges. Each edge in E connects two vertices from V. Unless stated otherwise, every graph in this paper is simple and undirected. A **simple graph** has no loops and no multiple edges between the same pair of vertices. An **undirected graph** has edges with no direction, so the edge joining u and v is the same as the edge joining v and u.

Figure 2 shows the reference graph that we will return to throughout this section. It consists of six vertices, labeled A, B, C, D, E, and F, together with nine edges. We will use it as a running example while introducing the basic ideas leading up to the theorem.

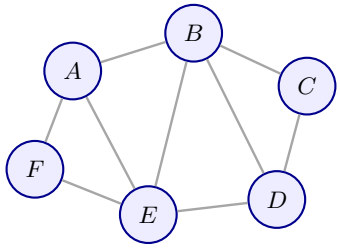


Figure 2: Reference graph used to illustrate the basic concepts in this section.

2.2 Vertices and Edges

Vertex (V) The fundamental unit of a graph, often drawn as a dot or circle. Vertices can represent people, places, objects, or any other entities in a network.

Edge (E) A connection between two vertices. In an undirected graph, edges have no direction; in a directed graph, edges have a direction.

Vertices and edges are the basic pieces from which every graph is built. If u and v are vertices in an undirected graph, the edge joining them may be written as uv or as the unordered pair $\{u, v\}$. When such an edge is present, the vertices u and v are said to be adjacent.

In the reference graph, the vertices A and B are adjacent because the edge AB belongs to the edge set. Figure 3 highlights this edge while keeping the rest of the graph visible. These simple ideas will be used again when we discuss paths, cycles, trees, spanning trees, and graph matrices.

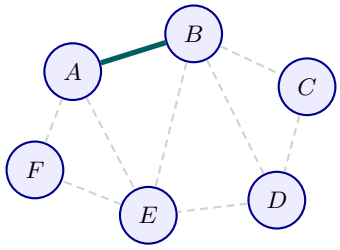


Figure 3: The labeled vertices of the reference graph, with the edge AB highlighted.

2.3 Degree

The **degree** of a vertex v, denoted by $\deg(v)$, is the number of edges incident to that vertex. In a simple undirected graph, this is equivalently the number of vertices adjacent to v.

For the reference graph, vertex B is adjacent to A, C, D, and E, and therefore $\deg(B) = 4$. Vertex C is adjacent to B and D, so $\deg(C) = 2$. The degrees of all vertices in the reference graph are shown in Table 1.

Vertex	Degree
A	3
B	4
C	2
D	3
E	4
F	2

Table 1: Degree of each vertex in the reference graph.

2.4 Paths

A **path** in a graph is a sequence of vertices in which each consecutive pair of vertices is joined by an edge. More formally, a path from a vertex u to a vertex v is a sequence

$$u = v_0, v_1, \ldots, v_k = v$$

such that $v_{i-1}v_i \in E$ for each $i = 1, 2, \ldots, k$. The length of the path is the number of edges it contains, which is k.

Paths describe how to move through a graph by following one edge after another. In the reference graph, the sequence A, B, C is a path from A to C of length 2, since both AB and BC are edges. This path is highlighted in Figure 4.

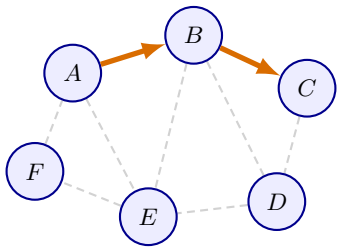


Figure 4: A path from A to C, passing through B, highlighted in the reference graph.

2.5 Cycles

A **cycle** is a path that begins and ends at the same vertex, with no repeated vertices except for the starting and ending vertex. More precisely, a cycle is a sequence of distinct vertices

$$v_0, v_1, \ldots, v_{k-1}, v_0$$

where each consecutive pair is joined by an edge and $k \geq 3$.

Cycles matter because they show where a graph has extra routes or redundant connections. A graph with no cycles is acyclic, and this idea is essential in the definition of a tree. In the reference graph, the sequence A, B, E, A forms a cycle, as shown in Figure 5.

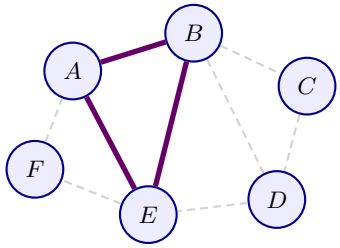


Figure 5: The cycle $A \to B \to E \to A$ highlighted in the reference graph.

2.6 Connected Graphs

A graph is called **connected** if every pair of vertices can be joined by a path. Equivalently, a graph $G = (V, E)$ is connected if, for any two vertices $u, v \in V$, there exists a sequence of vertices

$$u = v_0, v_1, \ldots, v_k = v$$

such that each consecutive pair v_{i-1} and v_i is joined by an edge. This condition means that the graph consists of one connected component.

For the Matrix Tree Theorem, we need to assume connectedness. If a graph is disconnected, then it is impossible to find a spanning tree, since no tree can connect vertices lying in different components. The reference graph in Figure 2 is connected because every vertex can be reached from every other vertex by following edges in the graph.

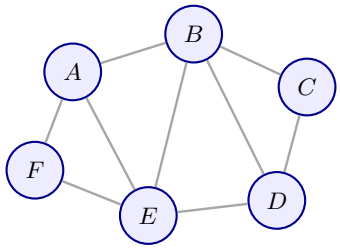


Figure 6: The reference graph is connected, since its vertices belong to a single component.

2.7 Trees

A **tree** is a connected graph that contains no cycles. In this sense, a tree has exactly enough edges to keep all of its vertices connected, but no extra edges that would create a closed path. If a tree has n vertices, then it has exactly $n - 1$ edges.

Trees are central to the Matrix Tree Theorem because they represent minimal connected structures. Removing any edge from a tree disconnects it, while

adding any new edge to a tree creates a cycle. The tree shown below uses the same labeled vertices as the reference graph and contains five edges, which is the correct number for a tree on six vertices.

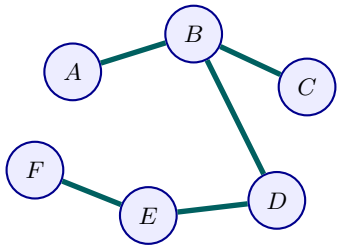


Figure 7: A tree on the same set of labeled vertices, containing no cycles.

2.8 Spanning Trees

Let $G = (V, E)$ be a connected graph. A **spanning tree** of G is a subgraph that includes every vertex of G and is itself a tree. Equivalently, it connects all vertices of the original graph without containing any cycles.

If G has n vertices, then every spanning tree of G has exactly $n - 1$ edges. The difference between a tree and a spanning tree is that a spanning tree is considered relative to a larger graph: it is formed by selecting some of the edges of G while keeping all of its vertices. The Matrix Tree Theorem provides a way to count how many such spanning trees a connected graph has.

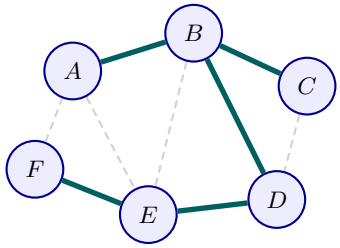


Figure 8: A spanning tree of the reference graph, highlighted in teal.

2.9 Adjacency Matrix

The **adjacency matrix** records which pairs of vertices in a graph are connected by edges. For a graph with vertices ordered as $v_1, v_2, \ldots, v_n$, the adjacency matrix is the $n \times n$ matrix $A = (a_{ij})$ defined by

$$a_{ij} = \begin{cases} 1, & \text{if } v_i \text{ is adjacent to } v_j, \\ 0, & \text{otherwise.} \end{cases}$$

For a simple undirected graph, the adjacency matrix is symmetric and has zeros on the diagonal. Symmetry reflects the fact that an edge from v_i to v_j is the same as an edge from v_j to v_i.

For the reference graph, using the vertex order A, B, C, D, E, F, the adjacency matrix is as follows.

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

Figure 9: Adjacency matrix of the reference graph, with vertices ordered as A, B, C, D, E, F.

2.10 Degree Matrix

The **degree matrix** is a diagonal matrix that records the degree of each vertex. If the vertices of a graph are ordered as $v_1, v_2, \ldots, v_n$, then the degree matrix $D = (d_{ij})$ is defined by

$$d_{ij} = \begin{cases} \deg(v_i), & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

All off-diagonal entries of D are zero, and the diagonal entries contain the vertex degrees.

For the reference graph, the vertex degrees are $3, 4, 2, 3, 4, 2$ in the order A, B, C, D, E, F. The degree matrix is therefore

$$D = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

Figure 10: Degree matrix of the reference graph, with vertex degrees placed on the diagonal.

2.11 Laplacian Matrix

The **Laplacian matrix** combines the degree and adjacency information of a graph into a single matrix. For a graph G, the Laplacian matrix is defined by

$$L = D - A,$$

where D is the degree matrix of G and A is the adjacency matrix of G.

Equivalently, if $L = (\ell_{ij})$, then

$$\ell_{ij} = \begin{cases} \deg(v_i), & \text{if } i = j, \\ -1, & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j, \\ 0, & \text{otherwise.} \end{cases}$$

The Laplacian matrix is the central matrix in the Matrix Tree Theorem. For the reference graph, it is obtained by subtracting the adjacency matrix from the degree matrix.

$$L = \begin{pmatrix} 3 & -1 & 0 & 0 & -1 & -1 \\ -1 & 4 & -1 & -1 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 3 & -1 & 0 \\ -1 & -1 & 0 & -1 & 4 & -1 \\ -1 & 0 & 0 & 0 & -1 & 2 \end{pmatrix}$$

Figure 11: Laplacian matrix of the reference graph, computed from $L = D - A$.

2.12 Determinant

The **determinant** is a scalar associated with a square matrix. If M is an $n \times n$ matrix, its determinant is denoted by $\det(M)$. Determinants play an important role in linear algebra because they encode information about invertibility, volume scaling, and systems of linear equations.

For a 2×2 matrix,

$$M = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

the determinant is

$$\det(M) = ad - bc.$$

For larger matrices, the determinant can be computed recursively using cofactors. In the Matrix Tree Theorem, the determinant of a certain matrix obtained from the Laplacian gives the number of spanning trees of the graph.

2.13 Cofactor

Let M be an $n \times n$ matrix. The minor M_{ij} is the matrix obtained by deleting row i and column j from M. The **cofactor** corresponding to the entry in row i and column j is defined by

$$C_{ij} = (-1)^{i+j} \det(M_{ij}).$$

So a cofactor is a signed determinant of a smaller matrix.

For the Matrix Tree Theorem, cofactors of the Laplacian matrix are especially important. If L is the Laplacian matrix of a connected graph G, then deleting any one row and the corresponding column from L produces a reduced Laplacian matrix. The determinant of this reduced matrix is equal to the number of spanning trees of G:

$$\tau(G) = \det(L^{(i)}),$$

where $L^{(i)}$ denotes the matrix obtained from L by deleting row i and column i. The next section states this result precisely.

3 The Matrix Tree Theorem

The Matrix Tree Theorem gives a remarkably compact way to count spanning trees using the Laplacian matrix of a graph.

Theorem. Let G be a finite connected undirected graph with Laplacian matrix L. Let $L^{(i)}$ be the matrix obtained from L by deleting the ith row and the ith column. Then the number of spanning trees of G, denoted $\tau(G)$, is

$$\tau(G) = \det\bigl(L^{(i)}\bigr).$$

This value is the same no matter which row and corresponding column are deleted.

4 Worked Example

Consider the graph G with vertex set

$$V = \{A, B, C, D, E\}$$

and edge set

$$E = \{AB, AC, AD, AE, BC, BD\}.$$

This graph is connected, so it has at least one spanning tree. We will use the Matrix Tree Theorem to count exactly how many spanning trees it has.

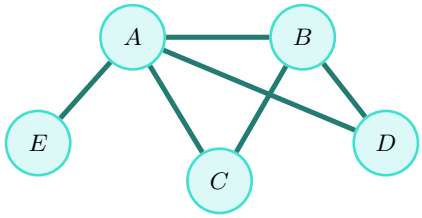


Figure 12: A connected graph with five vertices and six edges.

We order the vertices as A, B, C, D, E. The degrees are

$$\deg(A) = 4, \quad \deg(B) = 3, \quad \deg(C) = 2,$$
$$\deg(D) = 2, \quad \deg(E) = 1.$$

Thus the adjacency matrix is

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix},$$

and the degree matrix is

$$D = \begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Thus the Laplacian matrix $L = D - A$ is

$$L = \begin{pmatrix} 4 & -1 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 & 0 \\ -1 & -1 & 2 & 0 & 0 \\ -1 & -1 & 0 & 2 & 0 \\ -1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

By the Matrix Tree Theorem, we may delete any one row and the corresponding column. Delete the row and column for vertex E. The reduced Laplacian is

$$L^{(E)} = \begin{pmatrix} 4 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ -1 & -1 & 0 & 2 \end{pmatrix}.$$

Now compute its determinant. Expanding along the first row,

$$\begin{aligned}\det(L^{(E)}) &= 4\begin{vmatrix} 3 & -1 & -1 \\ -1 & 2 & 0 \\ -1 & 0 & 2 \end{vmatrix} + \begin{vmatrix} -1 & -1 & -1 \\ -1 & 2 & 0 \\ -1 & 0 & 2 \end{vmatrix} \\ &\quad - \begin{vmatrix} -1 & 3 & -1 \\ -1 & -1 & 0 \\ -1 & -1 & 2 \end{vmatrix} + \begin{vmatrix} -1 & 3 & -1 \\ -1 & -1 & 2 \\ -1 & -1 & 0 \end{vmatrix} \\ &= 4(8) + (-8) - 8 + (-8) \\ &= 8.\end{aligned}$$

Hence

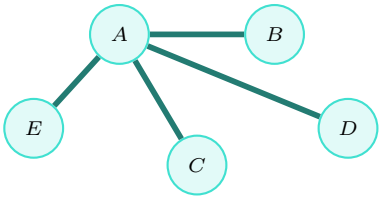
$$\tau(G) = 8.$$

So the graph has exactly eight spanning trees.

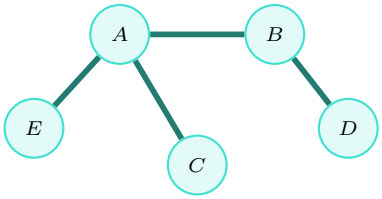
As a direct check, the eight spanning trees are obtained from the following edge sets:

$$\begin{array}{ll} \{AB, AC, AD, AE\}, & \{AB, AC, AE, BD\}, \\ \{AB, AD, AE, BC\}, & \{AB, AE, BC, BD\}, \\ \{AC, AD, AE, BC\}, & \{AC, AD, AE, BD\}, \\ \{AC, AE, BC, BD\}, & \{AD, AE, BC, BD\}. \end{array}$$

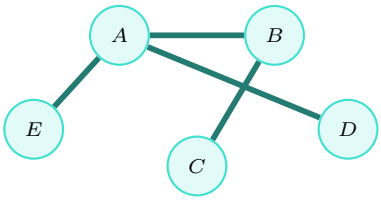
Each set contains all five vertices, has four edges, and contains no cycle; therefore each is a spanning tree. This agrees with the determinant found from the Matrix Tree Theorem.



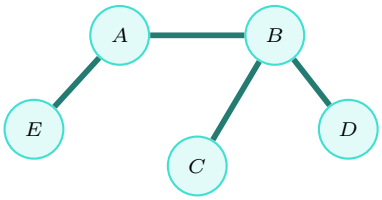
$T_1 = \{AB, AC, AD, AE\}$



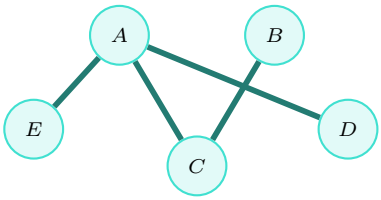
$T_2 = \{AB, AC, AE, BD\}$



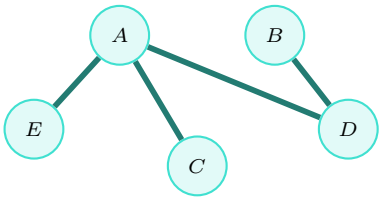
$T_3 = \{AB, AD, AE, BC\}$



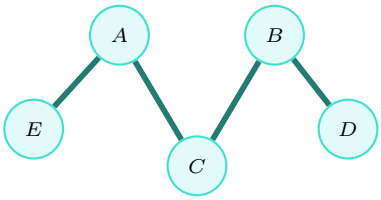
$T_4 = \{AB, AE, BC, BD\}$



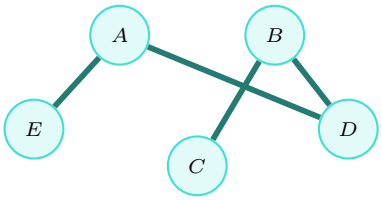
$T_5 = \{AC, AD, AE, BC\}$



$T_6 = \{AC, AD, AE, BD\}$



$T_7 = \{AC, AE, BC, BD\}$



$T_8 = \{AD, AE, BC, BD\}$

Figure 13: All eight spanning trees of the worked-example graph.

5 Proof of the Theorem

This section presents several ways to prove the Matrix Tree Theorem. We begin with an induction proof based on deletion and contraction, then give the standard algebraic proof using the Cauchy–Binet theorem. The remaining subsections explain how the same result can also be understood through random walks, generating functions, and a topological interpretation using homology.

5.1 Inductive Proof by Deletion and Contraction

We first prove the theorem by induction, using the deletion–contraction relation for spanning trees. The main idea is to show that the determinant of the reduced Laplacian satisfies the same recurrence as the number of spanning trees.

Let G be a finite undirected graph, and let $e = uv$ be an edge of G. We write $G - e$ for the graph obtained by deleting the edge e, and G/e for the graph obtained by contracting e, meaning that the two endpoints u and v are identified as a single vertex. If parallel edges are created during contraction, they are kept and counted with multiplicity.

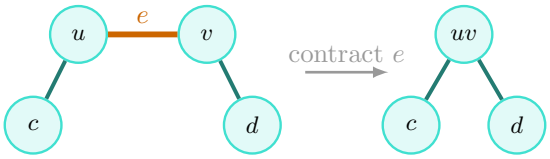


Figure 14: Contracting an edge identifies its two endpoints as one vertex.

We will prove that for every vertex r,

$$\det(L_G^{(r)}) = \tau(G),$$

where $L_G^{(r)}$ is obtained from the Laplacian of G by deleting the row and column corresponding to r.

We begin with the base cases. If G has an isolated vertex, then G has no spanning tree. If the isolated vertex is not deleted, the reduced Laplacian has a zero row, so its determinant is 0. If the isolated vertex is the deleted vertex, then the remaining graph is still disconnected, and the reduced Laplacian is singular. Thus both sides are 0. In particular, the empty graph on two vertices has

$$L_G = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix},$$

so every cofactor is 0, matching the fact that it has no spanning tree.

Now assume that G has no isolated vertices. Choose the deleted vertex to be r, and choose an edge $e = rv$ incident to r. Every spanning tree of G falls into exactly one of two classes: either it does not contain e, or it does contain e. The spanning trees not containing e are exactly the spanning trees of $G - e$. The

spanning trees containing e correspond bijectively to the spanning trees of G/e, because contracting e removes that forced edge while preserving connectivity and acyclicity. Therefore

$$\tau(G) = \tau(G - e) + \tau(G/e).$$

It remains to verify that the determinant of the reduced Laplacian satisfies the same recurrence. Since $e = rv$, deleting the row and column of r removes the off-diagonal entries involving r. The only remaining effect of the edge e is that it contributes 1 to the degree of v. Hence

$$L_G^{(r)} = L_{G-e}^{(r)} + E_{vv},$$

where E_{vv} denotes the matrix with a single 1 in the v-diagonal position and zeros elsewhere.

We use the following determinant fact. If M is a square matrix and E_{kk} has a single 1 in the (k, k) entry, then

$$\det(M + E_{kk}) = \det(M) + \det(M^{(k)}),$$

where $M^{(k)}$ is obtained from M by deleting row k and column k. This follows immediately from the cofactor expansion of the determinant along the kth row or column, because increasing the (k, k) entry by 1 increases the determinant by exactly the corresponding principal cofactor.

Applying this fact gives

$$\begin{aligned}\det(L_G^{(r)}) &= \det(L_{G-e}^{(r)} + E_{vv}) \\ &= \det(L_{G-e}^{(r)}) + \det(L_{G-e}^{(r,v)}).\end{aligned}$$

Here $L_{G-e}^{(r,v)}$ means that both the r and v rows and columns have been deleted. After both endpoints of e are removed, the edge e no longer affects the matrix, so

$$L_{G-e}^{(r,v)} = L_G^{(r,v)}.$$

On the other hand, contracting $e = rv$ identifies r and v. If the contracted vertex is denoted by v, then deleting that contracted vertex from G/e produces exactly the same reduced matrix as deleting both r and v from G. Thus

$$L_{G/e}^{(v)} = L_G^{(r,v)}.$$

Therefore

$$\det(L_G^{(r)}) = \det(L_{G-e}^{(r)}) + \det(L_{G/e}^{(v)}).$$

By the induction hypothesis applied to $G - e$ and G/e,

$$\det(L_{G-e}^{(r)}) = \tau(G - e), \qquad \det(L_{G/e}^{(v)}) = \tau(G/e).$$

As a result,

$$\det(L_G^{(r)}) = \tau(G - e) + \tau(G/e) = \tau(G).$$

This completes the induction proof of the Matrix Tree Theorem.

5.2 Algebraic Proof Using the Cauchy–Binet Theorem

Let $G = (V, E)$ be a finite connected undirected graph with

$$V = \{v_1, v_2, \ldots, v_n\}, \qquad |E| = m.$$

Choose an arbitrary orientation for each edge. This orientation is only a temporary algebraic tool; it does not change the original undirected graph. Define the **oriented incidence matrix** B to be the $n \times m$ matrix whose columns are indexed by edges and whose rows are indexed by vertices. If an oriented edge e leaves v_i and enters v_j, then the column of B corresponding to e has

$$B_{i,e} = 1, \qquad B_{j,e} = -1,$$

and all other entries in that column are 0.

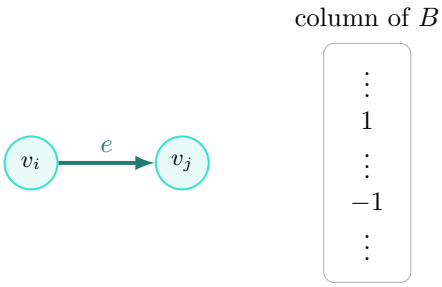


Figure 15: An oriented edge contributes one 1 and one -1 to the incidence matrix.

Lemma 1. The graph Laplacian satisfies

$$L = BB^T.$$

Proof. The diagonal entry $(BB^T)_{ii}$ is the dot product of row i with itself. Each edge incident to v_i contributes either 1^2 or $(-1)^2$, so

$$(BB^T)_{ii} = \deg(v_i).$$

If $i \neq j$, then $(BB^T)_{ij}$ is nonzero only when v_i and v_j are joined by an edge. In that case, the corresponding column contains one 1 and one -1, so the contribution is -1. Therefore

$$(BB^T)_{ij} = \begin{cases} -1, & \text{if } v_i \text{ and } v_j \text{ are adjacent,} \\ 0, & \text{otherwise.} \end{cases}$$

These are exactly the entries of the Laplacian matrix. Hence $L = BB^T$. □

Let B_i denote the matrix obtained from B by deleting the ith row. Then

$$L^{(i)} = B_i B_i^T,$$

where $L^{(i)}$ is the reduced Laplacian obtained by deleting row i and column i from L.

We now apply the following standard theorem from linear algebra.

Theorem 2 (Cauchy–Binet). Let X be an $r \times m$ matrix and let Y be an $m \times r$ matrix, where $r \leq m$. Then

$$\det(XY) = \sum_{S \subseteq \{1,2,\ldots,m\},\ |S|=r} \det(X_S) \det(Y_S),$$

where X_S is the $r \times r$ submatrix of X formed by keeping the columns indexed by S, and Y_S is the $r \times r$ submatrix of Y formed by keeping the rows indexed by S.

In our case, take $X = B_i$ and $Y = B_i^T$. Since B_i has $n-1$ rows, the Cauchy–Binet theorem gives

$$\det(L^{(i)}) = \det(B_i B_i^T) = \sum_{S \subseteq E,\ |S|=n-1} \det((B_i)_S)^2.$$

Thus each subset S of $n-1$ edges contributes either 0 or 1, as the next lemma shows.

Lemma 3. Let $S \subseteq E$ with $|S| = n-1$. Then

$$\det((B_i)_S)^2 = \begin{cases} 1, & \text{if } S \text{ is a spanning tree of } G, \\ 0, & \text{otherwise.} \end{cases}$$

Proof. First suppose that S is not a spanning tree. Since S has $n-1$ edges, it fails to be a spanning tree exactly when it is disconnected or contains a cycle. If it contains a cycle, then the columns of $(B_i)_S$ corresponding to the cycle are linearly dependent: following the oriented cycle and adding the signed incidence columns gives the zero vector. Hence the determinant is 0. If it is disconnected, then at least one component is separated from the deleted vertex after removing row i, so the rows and columns cannot have full rank. Again the determinant is 0.

Now suppose that S is a spanning tree. A tree has a leaf vertex. If the deleted row is not the leaf, the column corresponding to the leaf edge has exactly one nonzero entry in the reduced incidence matrix, equal to 1 or -1. Expanding the determinant along this column reduces the determinant to that of a smaller tree. Repeating this leaf-removal process eventually gives a 1×1 determinant equal to 1 or -1. Therefore

$$\det((B_i)_S) = \pm 1,$$

so its square is 1. □

Combining Cauchy–Binet with Lemma 3 gives

$$\det(L^{(i)}) = \sum_{S \subseteq E,\ |S|=n-1} \det((B_i)_S)^2 = \#\{\text{spanning trees of } G\}.$$

Therefore

$$\tau(G) = \det(L^{(i)}),$$

which proves the Matrix Tree Theorem.

5.3 Random Walk Proof Using Markov Chain Concepts

The theorem also has a natural interpretation in terms of random walks. Let a particle move on the graph by choosing, at each vertex, one of the incident edges uniformly at random. This defines a Markov chain whose state space is the vertex set V.

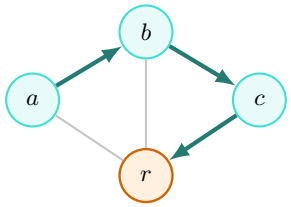


Figure 16: A random walk path toward a chosen root vertex.

Fix a root vertex r. For each vertex $v \neq r$, follow the random walk starting at v until it first reaches r. Record the first edge used whenever the walk exits a vertex that has not yet been recorded. This procedure is closely related to the loop-erased random walk construction of a spanning tree rooted at r.

The central Markov-chain fact is that hitting probabilities satisfy linear equations involving the graph Laplacian. If $h(v)$ is a harmonic function on $V \setminus \{r\}$, then

$$h(v) = \frac{1}{\deg(v)} \sum_{u \sim v} h(u).$$

Equivalently,

$$\deg(v)h(v) - \sum_{u \sim v} h(u) = 0,$$

which is precisely a Laplacian equation. Thus the reduced Laplacian $L^{(r)}$ controls the linear system for the random walk killed when it reaches r.

The determinant $\det(L^{(r)})$ is the normalizing constant for the Markov-chain weights of rooted spanning trees. In the unweighted case, every spanning tree has the same weight.

Thus the normalizing constant is exactly the number of spanning trees. Therefore

$$\det(L^{(r)}) = \tau(G).$$

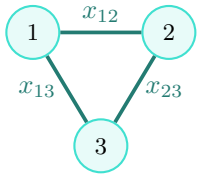
This point of view is less computational than the Cauchy–Binet proof, but it explains why the Laplacian appears naturally: it is the operator that governs harmonic functions, hitting probabilities, and flows on the graph.

5.4 Probabilistic Proof Using Generating Functions

A generating-function proof gives a useful weighted version of the Matrix Tree Theorem. Assign a variable x_e to every edge $e \in E$. The **spanning-tree generating function** is

$$T_G(x) = \sum_{T} \prod_{e \in T} x_e,$$

where the sum runs over all spanning trees T of G.



$$T_G(x) = x_{12}x_{13} + x_{12}x_{23} + x_{13}x_{23}$$

Figure 17: A weighted triangle and its spanning-tree generating function.

Define the weighted Laplacian $L(x)$ by

$$L(x)_{ij} = \begin{cases} \sum_{e \text{ incident to } v_i} x_e, & i = j, \\ -x_{ij}, & i \neq j \text{ and } v_i v_j \in E, \\ 0, & \text{otherwise.} \end{cases}$$

Here x_{ij} denotes the variable assigned to the edge joining v_i and v_j.

The weighted Matrix Tree Theorem states that

$$\det(L(x)^{(r)}) = T_G(x).$$

To prove this identity, repeat the Cauchy–Binet argument using a weighted incidence matrix. Each edge column is multiplied by a factor of $\sqrt{x_e}$. Then a set S of $n-1$ edges contributes

$$\left(\prod_{e \in S} x_e \right) \det((B_r)_S)^2.$$

By Lemma 3, this contribution is zero unless S is a spanning tree, and it is exactly $\prod_{e \in S} x_e$ when S is a spanning tree. Hence

$$\det(L(x)^{(r)}) = \sum_{T} \prod_{e \in T} x_e.$$

Setting every $x_e = 1$ gives the ordinary Matrix Tree Theorem:

$$\det(L^{(r)}) = \sum_{T} 1 = \tau(G).$$

This interpretation is probabilistic because, after normalization, the polynomial $T_G(x)$ defines a probability distribution on spanning trees:

$$\mathbb{P}(T) = \frac{\prod_{e \in T} x_e}{T_G(x)}.$$

Thus the determinant is the partition function of the random spanning tree model.

5.5 Topological Proof Using Homology Groups

Finally, a graph may also be viewed as a one-dimensional topological space, or equivalently as a 1-dimensional cell complex. Its vertices are 0-cells and its edges are 1-cells. The incidence matrix is then the matrix of the boundary map

$$\partial_1 : C_1(G) \to C_0(G),$$

which sends each oriented edge to the difference of its endpoint vertices.

1-cell mapped to a difference of 0-cells

Figure 18: The boundary map sends an oriented edge to its terminal vertex minus its initial vertex.

With respect to the standard bases of vertices and edges, the matrix of ∂_1 is the incidence matrix B. Therefore the graph Laplacian is

$$L = \partial_1 \partial_1^T.$$

The first homology group $H_1(G)$ measures cycles in the graph, while the reduced zeroth homology group $\widetilde{H}_0(G)$ measures disconnected components. Since G is connected, $\widetilde{H}_0(G) = 0$, but cycles may still exist.

From this topological point of view, a spanning tree is important because it is a maximal subcomplex with no first homology. In more concrete terms, it keeps all vertices connected while removing every cycle. Thus spanning trees are precisely the subgraphs whose first homology is trivial and whose vertex set is all of V.

In this language, deleting one row of the incidence matrix corresponds to passing to reduced 0-chains. The determinant of a maximal square submatrix of the reduced boundary map detects whether the chosen $n - 1$ edges form an acyclic connected subcomplex. If they do, the determinant is ± 1; if not, the determinant is 0. This is exactly the content of Lemma 3, interpreted topologically.

The Cauchy–Binet expansion of

$$\det(B_r B_r^T)$$

counts the number of bases of the cellular boundary map. These bases are exactly the spanning trees. Hence

$$\det(L^{(r)}) = \tau(G).$$

This topological viewpoint shows that the Matrix Tree Theorem is not only a statement about matrices. It is also a statement about how cycles and connectivity are encoded by the homology of a graph.

6 Applications

One reason the Matrix Tree Theorem is so useful is that it changes a difficult counting problem into a determinant calculation. Instead of trying to list every possible spanning tree by hand, we can use the Laplacian matrix and compute one cofactor. This makes the theorem valuable in many areas where networks appear naturally, including engineering, computer science, probability, and artificial intelligence.

Some important places where the theorem appears are:

- **Network reliability analysis:** estimating how many ways a network can remain connected if links fail.
- **Electrical circuit analysis and Kirchhoff circuit theory:** studying current flow, effective resistance, and circuit connectivity.
- **Communication and computer network design:** comparing possible network topologies and measuring redundancy.
- **Graph algorithm design:** using Laplacian matrices to count, sample, or optimize tree-like structures.
- **Algebraic and spectral graph theory:** relating spanning trees to eigenvalues of the Laplacian.
- **Monte Carlo methods and Markov chains:** sampling random spanning trees and studying Markov-chain stationary behavior.
- **Markov Chain Tree Theorem:** expressing stationary probabilities of a Markov chain through directed spanning trees.
- **Quantum networks:** studying possible connected substructures in networks used for quantum communication.
- **Machine learning and artificial intelligence:** building graph-based models for clustering, prediction, and vision tasks.

The examples below focus on a few of the most interesting uses in more detail.

6.1 Network Reliability and Communication Design

In a communication network, the vertices might represent routers, servers, or buildings, and the edges might represent physical cables or wireless links. A spanning tree gives a minimal connected backbone: it connects every location, but uses no unnecessary cycles. If the network has many spanning trees, then it has many different ways to remain connected if some links fail. For that reason, the number of spanning trees can be viewed as a measure of reliability.

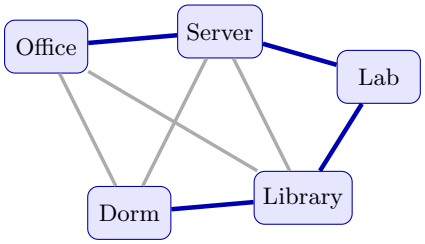


Figure 19: A communication network where a spanning tree gives a minimal connected backbone.

For example, a university network may need to connect offices, servers, laboratories, libraries, and dorms. One spanning tree gives one possible backbone for the whole system. Counting all spanning trees tells us how many alternative backbones are available, which is helpful when planning backup links and avoiding single points of failure.

6.2 Electrical Circuits and Kirchhoff's Laws

The theorem first arose from Kirchhoff's study of electrical circuits. In a circuit graph, vertices represent junctions, and edges represent wires or resistors. The Laplacian matrix records how the junctions are connected and how current can move through the circuit. Spanning trees appear naturally because they describe connected, cycle-free ways for the circuit to link all junctions.

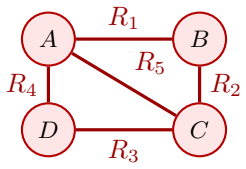


Figure 20: An electrical circuit can be modeled as a graph with junctions as vertices and resistors as edges.

In this setting, the Matrix Tree Theorem is connected to conductance, effective resistance, and the structure of current paths. The determinant of a reduced Laplacian gives information about how strongly the circuit is tied together through all of its possible tree-like connections.

6.3 Markov Chains and Random Processes

The Markov Chain Tree Theorem is a directed version of this same idea. For a finite Markov chain, each state can be drawn as a vertex, and each possible transition is a directed edge with a probability or rate. The long-term probability of being in a state can be described using directed spanning trees that point toward that state.

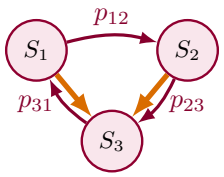


Figure 21: In a Markov chain, directed spanning trees can describe stationary probabilities.

This is useful in probability theory, Monte Carlo methods, and stochastic modeling because it gives a graph-based explanation of long-term behavior. Roughly speaking, a state tends to have a larger stationary probability when there are many strong directed tree structures leading into it.

6.4 Machine Learning and Artificial Intelligence

One of the most modern application areas is machine learning. Many machine-learning problems involve data that is naturally graph-shaped. Images, documents, social networks, molecules, recommendation systems, and knowledge graphs can all be represented using vertices and edges. The vertices may represent data points or features, while the edges represent similarity, dependence, or interaction.

The Matrix Tree Theorem is useful in machine learning because it can normalize probability distributions over tree structures. This matters when a model needs to consider many possible dependency trees at once. Listing all of them would usually be impossible, but the theorem turns the total sum over trees into a determinant that can be computed much more efficiently.

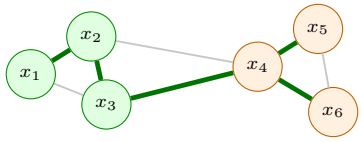


Figure 22: A graph-based machine-learning model built from similarities between data points.

For example, in **semi-supervised learning**, only a small number of data points may be labeled. A graph is built by connecting similar data points, and the labels are encouraged to spread smoothly across strong edges. The Laplacian matrix is central because it measures how much a function changes from one vertex to another. If two neighboring points are very similar, then a good prediction function should usually assign them similar labels.

In **graph-based clustering**, spanning trees can help identify the strongest connections among data points. A minimum or maximum spanning tree can reveal the main structure of the data, while random spanning trees can be used to study uncertainty in the clustering. The Matrix Tree Theorem supports this viewpoint because it gives an efficient way to count or weight tree structures.

In **structured prediction**, the output is not just one label. It may be a parse tree, a matching, a segmentation, or a dependency graph. Tree-based probability models often need a normalization constant over all possible trees. The Matrix Tree Theorem gives this constant as a determinant, which makes training and inference more practical.

In **computer vision**, an image can be represented as a graph whose vertices are pixels or regions. Edges can measure similarity in color, texture, or location. Spanning trees and Laplacian methods appear in image segmentation, object matching, and pattern recognition. A segmentation method may cut weak edges between different objects while keeping strong connections inside each object.

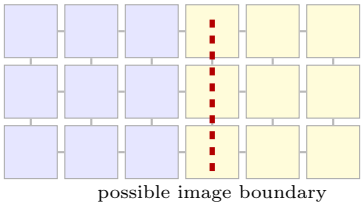


Figure 23: Image segmentation can be modeled by a graph whose vertices are pixels or regions.

In **graph neural networks**, the Laplacian and related graph operators describe how information moves between neighboring vertices. Graph neural networks do not always count spanning trees directly, but the same Laplacian

structure behind the Matrix Tree Theorem appears in message passing, smoothing, diffusion, and spectral graph methods.

In machine learning, then, the theorem is more than a counting formula. It is part of a broader set of tools for using graph structure to model dependence, similarity, uncertainty, and information flow.

6.5 Quantum Networks

Quantum networks are another modern setting where graph theory is becoming increasingly important. In a quantum communication network, vertices can represent quantum processors, laboratories, satellites, or repeater stations. Edges represent possible quantum channels between them, such as optical fibers, free-space links, or shared entanglement links.

A major goal in a quantum network is to distribute quantum information reliably between distant nodes. Quantum states are fragile, so links may fail because of noise, loss, decoherence, or limited entanglement generation rates. For this reason, the structure of the network is very important. A network with many different spanning trees has many different ways to keep all stations connected, even if some channels are unavailable.

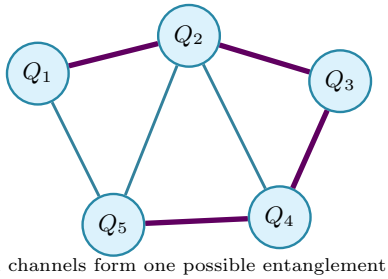


Figure 24: A quantum network can be modeled as a graph whose edges represent possible quantum channels.

In this picture, a spanning tree represents one possible entanglement backbone. It uses just enough quantum channels to connect every station, without adding redundant cycles. Different spanning trees represent different ways to route entanglement through the network. If one channel is noisy or unavailable, another spanning tree may still connect the same stations.

The Matrix Tree Theorem is useful here because it counts all such backbones through the Laplacian determinant. In a weighted version of the theorem, each edge can be assigned a weight representing channel quality, such as transmission probability, entanglement generation rate, or reliability. The weighted spanning-tree sum then measures not only how many connected backbones exist, but also how strong they are collectively.

For example, suppose high-quality quantum links receive larger edge weights. A spanning tree made from reliable links receives a larger product of weights

than a spanning tree containing weak links. The weighted Matrix Tree Theorem gives

$$\det(L(x)^{(r)}) = \sum_{T} \prod_{e \in T} x_e,$$

so the determinant summarizes the total strength of all possible entanglement backbones. This can help compare two possible quantum network designs: the better design is not necessarily the one with the most edges, but the one with many high-quality ways to keep the network connected.

This application is especially relevant for future quantum internet architectures. Such networks will need robust routing, entanglement distribution, and tolerance against link failure. The Matrix Tree Theorem gives a mathematical way to evaluate how well a proposed quantum network can maintain global connectivity through many possible tree-like communication structures.

6.6 Other Applications

Beyond the examples discussed above, the Matrix Tree Theorem also appears in several other areas:

- **Computer network topology:** comparing network layouts by how many redundant connected backbones they contain.
- **Graph algorithm design:** designing efficient algorithms for counting, sampling, and optimizing tree structures.
- **Algebraic graph theory:** studying how graph structure is encoded in matrices such as the adjacency matrix and Laplacian.
- **Spectral graph theory:** relating the number of spanning trees to Laplacian eigenvalues.
- **Monte Carlo methods:** sampling random spanning trees when exact enumeration is too expensive.
- **Diversity sampling:** selecting varied subsets or structures by using spanning trees to encourage diversity.
- **Object matching and pattern recognition:** representing possible correspondences as graph structures and using tree-based constraints.
- **Graphical models:** normalizing distributions over tree-shaped dependency structures.

7 Extensions of the Theorem

The Matrix Tree Theorem has many important extensions. Each one keeps the same central idea: a determinant of a Laplacian-type matrix counts tree-like structures. The difference is that the graph may be weighted, directed, rooted, partially rooted, or allowed to have forests instead of only spanning trees.

7.1 Weighted Matrix Tree Theorem

In a weighted graph, each edge e is assigned a weight w_e. Instead of counting every spanning tree equally, the weighted theorem counts a tree by multiplying the weights of its edges. The weighted spanning-tree sum is

$$\sum_{T} \prod_{e \in T} w_e,$$

where the sum is over all spanning trees T of the graph. If all weights are equal to 1, this reduces to the ordinary Matrix Tree Theorem.

To define the weighted Laplacian, the diagonal entry records the sum of weights incident to a vertex, and the off-diagonal entry is $-w_{ij}$ when vertices i and j are joined by an edge. Then any cofactor of this weighted Laplacian gives the weighted spanning-tree sum.

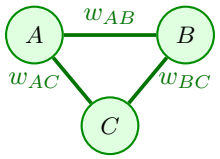


Figure 25: A weighted graph, where each edge contributes a weight to every spanning tree containing it.

This version is useful when edges do not have equal importance. For example, a reliable network link may be assigned a larger weight than an unreliable one, so spanning trees using reliable links contribute more to the total.

7.2 Directed Matrix Tree Theorem (Tutte's Theorem)

The directed Matrix Tree Theorem, also called Tutte's Theorem, applies when the graph has directed edges. In this setting, the objects being counted are directed spanning trees called **arborescences**. An arborescence is a directed tree in which every vertex is connected to a chosen root by directed paths, either toward the root or away from the root depending on the convention.

For a directed graph, the corresponding matrix is a directed Laplacian. After deleting the row and column for a chosen root, the resulting determinant counts the directed spanning trees rooted at that vertex.

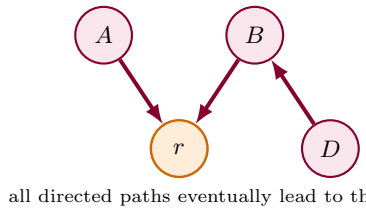


Figure 26: A directed spanning tree rooted at r.

This version matters because many real networks are directed: web links, traffic flow, Markov chains, and information flow all have natural directed edges.

7.3 All-Minors Matrix Tree Theorem

The All-Minors Matrix Tree Theorem is a broader version of the same idea. Instead of deleting just one row and one column from the Laplacian, it deletes several rows and several columns. The resulting determinant counts certain rooted forests rather than a single spanning tree.

A forest is a graph with no cycles, and it may have several connected components. In the all-minors theorem, the deleted rows and columns determine how the components are rooted and matched. The ordinary Matrix Tree Theorem is the special case where exactly one row and one column are deleted.

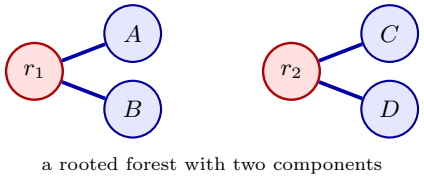


Figure 27: Deleting several rows and columns leads naturally to rooted forests.

This extension is useful when the goal is not just to count connected spanning trees, but also to count ways of splitting a graph into several rooted tree components.

7.4 Matrix Forest Theorem

The Matrix Forest Theorem counts spanning forests using a determinant involving $I + L$, where I is the identity matrix and L is the Laplacian. While the Matrix Tree Theorem focuses on connected cycle-free subgraphs, the Matrix Forest Theorem allows several tree components.

One common form says that determinants involving $I + L$ encode sums over rooted forests. This is useful since many networks are not naturally forced to be connected by one global tree. In many situations, it is more natural to allow several local tree components, each with its own root.

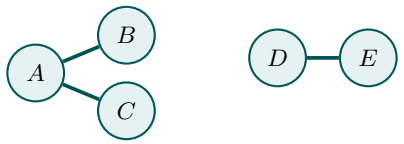


Figure 28: A spanning forest covers all vertices but may have several components.

The result appears in probability, network analysis, and graph-based learning, especially when disconnected or partly connected structures matter.

7.5 Markov Chain Tree Theorem

The Markov Chain Tree Theorem applies the directed version of the Matrix Tree Theorem to finite Markov chains. Suppose a Markov chain has states $1, 2, \ldots, n$ and transition rates between states. For each state i, the theorem considers all directed spanning trees pointing toward i. The stationary probability of state i is proportional to the total weight of those trees.

A state therefore tends to have high stationary probability when many high-weight directed trees flow into it. The result gives a clear graph-theoretic explanation of long-term behavior in Markov chains.

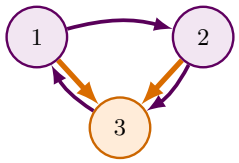


Figure 29: The stationary weight of a state comes from directed trees pointing toward it.

Here graph theory and probability meet in a very concrete way: long-term Markov-chain behavior can be understood through counting weighted directed trees.

7.6 Spectral Form

For a connected graph with Laplacian eigenvalues

$$0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_n,$$

there is a spectral form of the theorem:

$$\tau(G) = \frac{1}{n}\lambda_2\lambda_3 \cdots \lambda_n.$$

This form makes the link between spanning trees and the Laplacian spectrum especially clear, which is one reason the theorem is central in spectral graph theory.

Acknowledgments

I would like to thank Rajan Foster and Simon Rubinstein-Salzedo for their guidance and helpful feedback. I would also like to thank my classmates for their useful questions and discussions.

References

[1] G. Kirchhoff, *Ueber die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird*, Annalen der Physik und Chemie, 72 (1847), 497–508.

[2] N. Biggs, *Algebraic Graph Theory*, second edition, Cambridge University Press, 1993.

[3] C. Godsil and G. Royle, *Algebraic Graph Theory*, Springer, 2001.

[4] B. Bollobás, *Modern Graph Theory*, Springer, 1998.

[5] MIT OpenCourseWare, *The Matrix Tree Theorem*, 18.314 Combinatorial Analysis, Fall 2014. Available at `https://ocw.mit.edu/courses/18-314-combinatorial-analysis-fall-2014/resources/mit18_314f14_mt/`.

[6] Cornell University, *Matrix Tree Theorem*. Available at `https://www.cs.cornell.edu/courses/cs485/2006sp/matrix-tree-theorem.pdf`.

[7] S. Rubinstein-Salzedo, *The Matrix-Tree Theorem*, Euler Circle. Available at `https://simonrs.com/eulercircle/matrixtree.pdf`.