USING NETWORK AMALGAMATION AND SEPARATION TO SOLVE THE INVERSE PROBLEM

Ryan K. Card [1]   Brandon I. Muranaka [2]

June 18, 2003

[1]University of Washington Seattle
[2]University of California Davis

Contents

1 Introduction and Notation **4**
1.1 Forward Problem 4
1.2 Connections 5
1.3 New Notation 5

2 Amalgamating Networks **7**
2.1 Amalgamating Networks 7
2.2 Identifying Boundary Nodes in the Same Network 11
2.3 Modified Network 12

3 Separating Two Networks **15**
3.1 Shell Graph and Inner Graph Separation 15
3.2 Amalgamating Networks With Negative Conductances 18
3.2.1 Spike Removal 19
3.2.2 Boundary to Boundary Edge Removal 20
3.3 Separation Across Boundary Nodes 21
3.4 Separation Across One Interior Node 24

4 Three Examples **28**
4.1 The Towers of Hanoi Networks 28
4.1.1 Calculating the Boundary Conductances 29
4.1.2 The Recursive Step 37
4.1.3 Calculations and Results 38
4.2 Rectangular Networks 39
4.2.1 Base cases 40
4.2.2 Spiked case 42
4.2.3 Square case 44
4.2.4 Summary 46
4.2.5 Results and Errors 47

4.3 The Paper Doll Networks 47

5 Conclusion **51**

A Appendix **53**

A.1 Derivation of Formulae 53

A.1.1 The Towers of Hanoi Network 53

A.1.2 The Rectangular Network 59

A.2 Matlab Code 59

A.2.1 Tower Code 59

A.2.2 Rectangular Code 69

Abstract In this paper we consider the use of network amalgamation and separation to solve the inverse problem for networks. The amalgamation process can be used in some cases to determine the recoverability of some networks. The separation of networks into simpler networks has applications in the recovery of some networks including; the rectangular, Tower of Hanoi, and paper doll networks. In general, the use of network amalgamation and separation is to simplify the recovery process and provide insight in the recoverability of some networks.

A mathematician is a machine for turning coffee into theorems.

-Paul Erdös

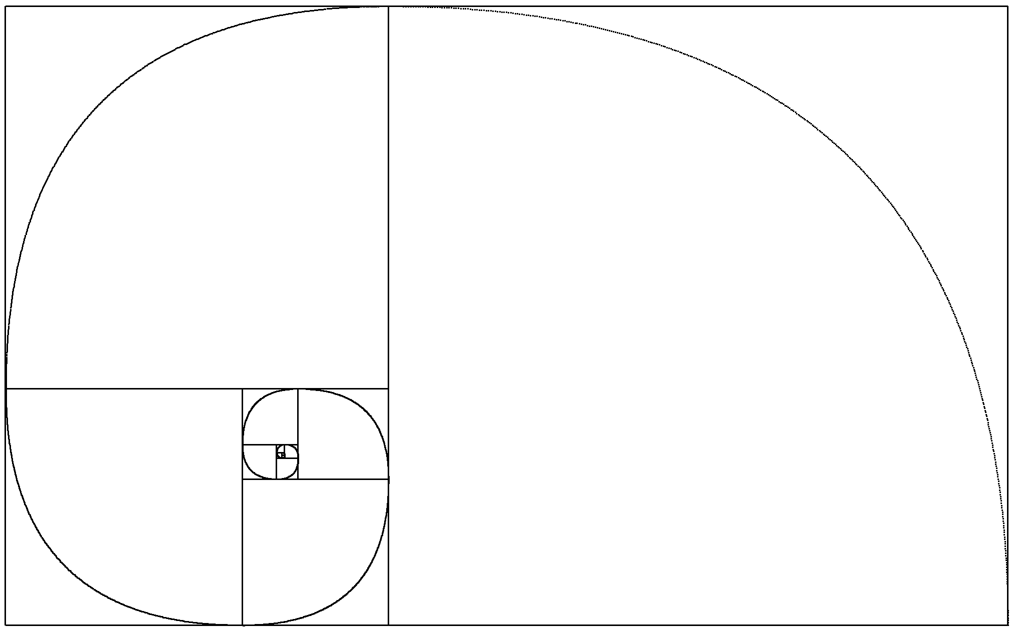


Figure 1 *The Golden Spiral*

Chapter 1

Introduction and Notation

This chapter describes the forward problem for discrete networks and introduces new notation that will be used in this paper. This chapter will also briefly describe the inverse problem and how it will be solved.

For notation denote a graph G to have vertices V and edges E, written $G = (V, E)$. Also, denote the interior nodes of G to be *int* V and the boundary nodes of G to be ∂V. For the response matrix of a graph G the notation Λ and for the Kirchhoff matrix the notation K are used. Given an edge e_{ij} denote the conductance of this edge to be γ_{ij}. For a network with conductances γ and a graph G the network is denoted by $\Gamma = (G, \gamma)$.

1.1 Forward Problem

Suppose that $\Gamma = (G, \gamma)$ is a network with graph G and Kirchhoff matrix K. Let ∂V be the set of boundary nodes with indices $H = \{1, 2..., l\}$ and let *int* V be the set of interior nodes with indices $I = \{l+1, ..., n\}$. Now, let

$$K = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix}$$

be an $n \times n$ positive semidefinite matrix, where A is $l \times l$. To find the Response matrix Λ of the network Γ we take the Schur Complement of K with repsect to C. Thus,

$$\Lambda = A - B(C)^{-1}B^T$$

is the Response matrix for the network Γ.

The Inverse Problem
The inverse problem can be paraphrased as: Given the graph G and its response matrix Λ can one calculate the Kirchhoff matrix K? This paper describes a new techniques for constructing new networks from two or more known networks, and for solving the inverse problem using a recursive "layer stripping" process.

1.2 Connections

Connections through graphs play an important role in the amalgamation and separation of graphs. It is therefore important to include some brief background about connections from the book by Curtis and Morrow (chapter 2, pp.12-13) [1].

Let $P = (p_1, \ldots, p_k)$ and $Q = (q_1, \ldots, q_k)$ be two sequences of boundary nodes of a graph, τ be a permutation of $(1, \ldots, k)$, and $\alpha_1, \ldots, \alpha_k$ be k disjoint paths. The paths α_i are such that for each i, α_i starts at p_i and ends at $q_{\tau(i)}$ and passes through no other boundary nodes. When such paths occuur it is said that there is a connection betweek P and Q.

This concept of connection has an impact on the response matrix and the subdeterminants when certain connections are broken.

1.3 New Notation

For this paper several new definitions are introduced to simplify the calculations and the description of the processes.

Definition 1.1 *Suppose $G = (E, V)$ is a graph with edges E and vertices, or nodes, V. The **shell edges** of the graph, denoted $\mathcal{SE}$, is the set of edges e_{pq} such that*

$$p \in \partial V \quad or \quad q \in \partial V.$$

Note that $\mathcal{SE} \subseteq E$.

Definition 1.2 *The degree function, $d_G : V \longrightarrow \mathbb{N}$, maps a node to the number of edges touching that node in the graph G.*

Definition 1.3 *The **inner graph** of $G = (E, V)$ is defined to be:*

$$\mathbb{I}(G) = (E - \mathcal{SE}, int\ V) = (E', V')$$

The boundary of the inner graph $\partial V'$ is the set of vertices p such that $d_{\mathbb{I}(G)}(p) < d_G(p)$, that is the number of edges touching p in G is greater than the number of edges touching p in $\mathbb{I}(G)$.

Definition 1.4 *The* ***shell graph*** *of a graph $G = (E, V)$ with inner graph $\mathbb{I}(G) = (E', V')$, denoted $\mathbb{S}(G)$ is defined as:*

$$\mathbb{S}(G) = (\mathcal{S}\mathcal{E}, \partial V \cup \partial V').$$

The conductances of the shell edges will be called the shell conductances of the graph G. The conductances of the inner graph are called the inner conductances.

Chapter 2

Amalgamating Networks

2.1 Amalgamating Networks

Suppose that there are two networks with known response matrices. This section will describe a method to amalgamate two networks through n boundary nodes, and obtain the corresponding response matrix for the new network.

Let $\Gamma_1 = (G_1, \gamma_1)$ and $\Gamma_2 = (G_2, \gamma_2)$ be two networks with known response matrices Λ_1 and Λ_2, respectively. Let L_1 be the set of boundary nodes to be connected in G_1, and L_2 to be the set of boundary nodes to be connected in G_2. Also let H be the set of boundary nodes not to be connected in G_1, and J be the set of boundary nodes not to be connected in G_2.

The set of n pairs of boundary nodes to be identified is:

$$L = \{(l_i, l'_i) : l_i \in L_1, l'_i \in L_2\}.$$

By ordering the boundary nodes so that H precedes L_1 and J precedes L_2, response matrices can be written in this form:

$$\Lambda_1 = \begin{pmatrix} A_1 & B_1 \\ B_1^T & C_1 \end{pmatrix} \qquad \text{and} \qquad \Lambda_2 = \begin{pmatrix} A_2 & B_2 \\ B_2^T & C_2 \end{pmatrix}.$$

Let $G^\star$ be the graph with pairs of boundary nodes L identified in the combined graphs of G_1 and G_2.

After identifying the boundary nodes in the pairs L, the corresponding

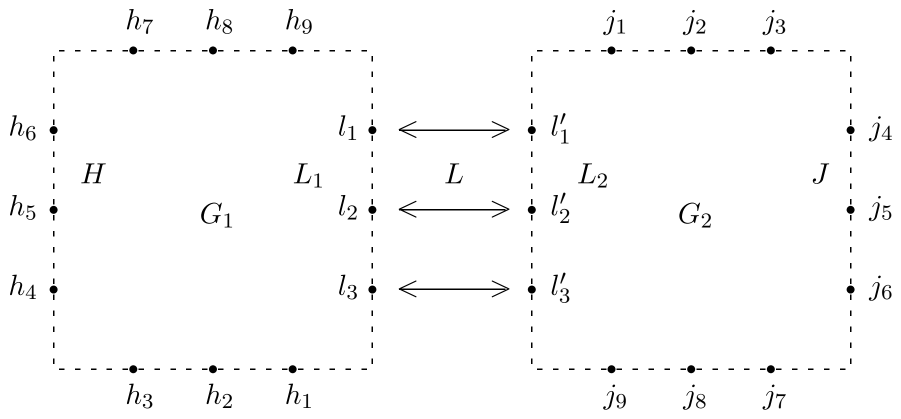


Figure 2.1 *Amalgamating Networks*

response matrix is:

$$\Lambda^\star = \begin{pmatrix} A_1 & 0 & B_1 \\ 0 & A_2 & B_2 \\ B_1^T & B_2^T & C_1 + C_2 \end{pmatrix}.$$

Where boundary nodes are ordered H first, J next, and L last.

Since the graphs were combined only through boundary nodes, there's no connection between H and J, hence the blocks of zeros in the above matrix.

To internalize the boundary nodes just connected, take the *Schur complement* of $\Lambda^\star$ with respect to the block $C_1 + C_2$:

$$\begin{aligned} \Lambda &= \Lambda^\star/(C_1 + C_2) \\ &= \begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} - \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} (C_1 + C_2)^{-1} \begin{pmatrix} B_1^T & B_2^T \end{pmatrix} \end{aligned}$$

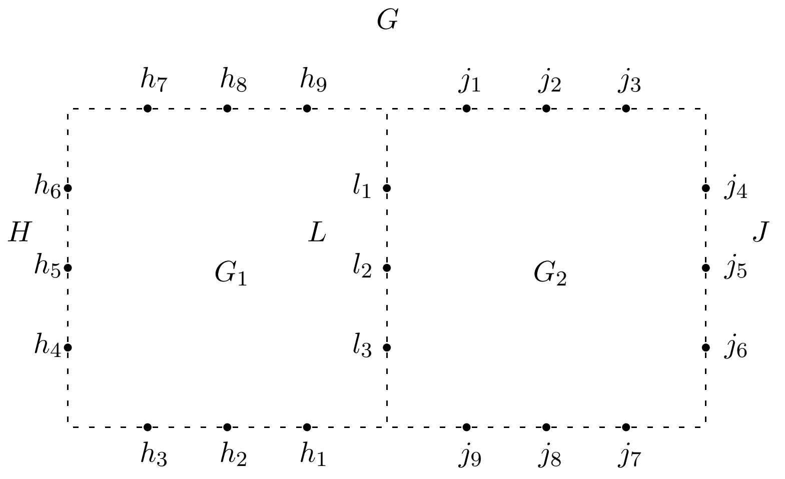


Figure 2.2 *Amalgamated Network*

Definition 2.3 *The response matrix for the combined network is denoted:*

$$\Lambda_1 \bowtie \Lambda_2$$

This is called $\mathbf{\Lambda_1}$ ***amalgamated to*** $\mathbf{\Lambda_2}$ ***through*** $\mathbf{L}$. *The set of boundary nodes to be identified, L, must be made clear from the context.*

Proposition 2.4 (Limitation of Connections Through Amalgamated Network) *Let $\Gamma_1 = (G_1, \gamma_1)$ and $\Gamma_2 = (G_2, \gamma_2)$ be two networks with known response matrices Λ_1 and Λ_2, respectively. Let L_1 be the set of boundary nodes to be identified in G_1, and L_2 to be the set of boundary nodes to be identified in G_2. The pairs of boundary nodes to be identified is L. Also let H be the set of boundary nodes not to be connected in G_1, and J be the set of boundary nodes not to be connected in G_2.*

Suppose

$$P = (p_1, p_2, \ldots, p_{n+1}) \qquad \textit{such that} \qquad p_i \in H$$

and

$$Q = (q_1, q_2, \ldots, q_{n+1}) \qquad \textit{such that} \qquad q_i \in J$$

then

$$det(\Lambda(P;Q)) = 0.$$

Proof: The graphs G_1 and G_2 were adjoined through the pairs L, so any connection from H to J must pass through at least one of the nodes in

L. Therefore, at least a pair in any set of $n+1$ paths from H to J have a common vertex in L. There is no connection in $(P;Q)$ because the paths can not be disjoint. No connection implies $det(\Lambda(P;Q)) = 0$ (for more details refer [1]). ■

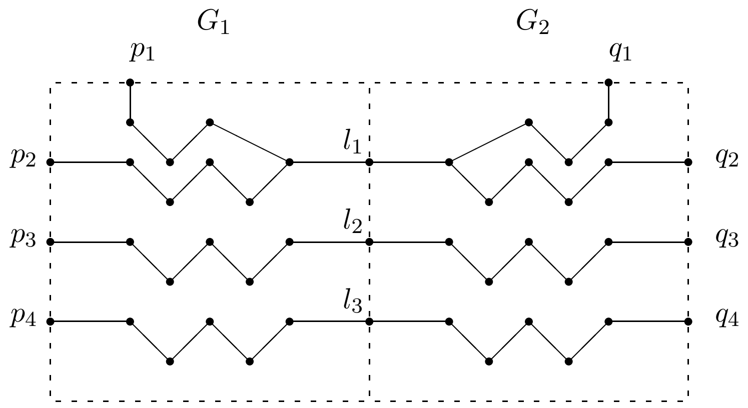


Figure 2.5 *Limitation of Connections Through Amalgamated Network*

Theorem 2.6 (Circular Planarity of the Amalgamated Network) *Let G_1 and G_2 be circular planar graphs with Λ_1 and Λ_2 as their response matrices respectively. If these networks were amalgamated through consecutive node pairs L, then the resulting graph G with response matrix $\Lambda_1 \bowtie \Lambda_2$ will be a circular planar network.*

Proof: Circular planar graphs are by definition a graph that can be drawn so that all boundary node could lie on a closed curve on a plane that contains the rest of the graph. Set up G_1 and G_2 as it appears in the Figure 2.7a. All boundary nodes named l_i and l'_i are the consecutive nodes to be connected. Notice that even if consecutive nodes does not line up between G_1 and G_2 (i.e. both ordered clockwise or counter-clockwise), one of the graphs can be flipped over so it does. Now amalgamate networks with G_1 and G_2 through $\{(l_i, l'_i)\}$ as it appears in Figure 2.7b. Since all l_i nodes are interiorized, resulting graph has all boundary nodes lying on a closed curve that contains the rest of the graph. Therefore, resulting graph is a circular planar network. ■

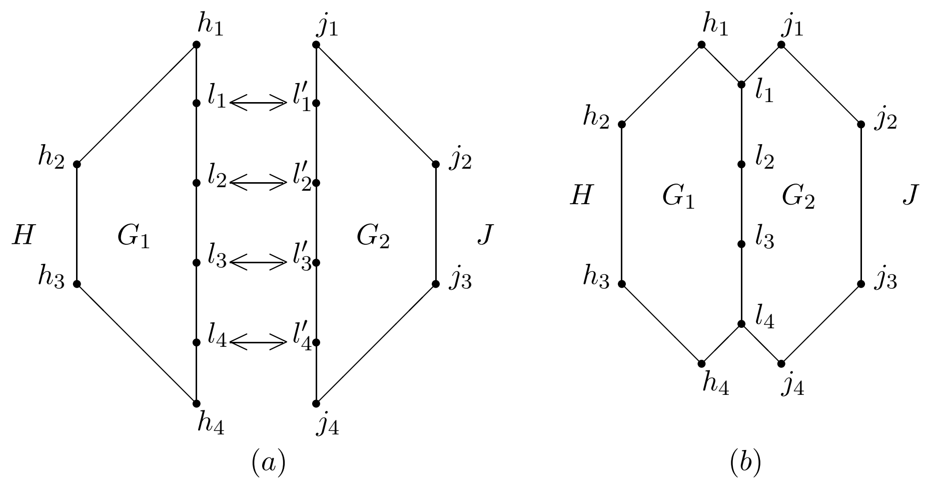


Figure 2.7 *Circular Planarity of the Amalgamated Network*

This theorem describes a sufficient condition for the amalgamated network to be a circular planar network.

2.2 Identifying Boundary Nodes in the Same Network

Let Λ be the response matrix for the network $\Gamma = (G, \gamma)$. This section describes a procedure to modify Λ so that it would correspond to a new network, Γ', with two boundary nodes in G identified together.

If l_1 and l_2 are the nodes to be identified, order these nodes last so that the response matrix can be written in this form:

$$\Lambda = \begin{pmatrix} \lambda_{1,1} & \lambda_{1,2} & \dots & \lambda_{1,l_1} & \lambda_{1,l_2} \\ \lambda_{2,1} & \lambda_{2,2} & \dots & \lambda_{2,l_1} & \lambda_{2,l_2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \lambda_{l_1,1} & \lambda_{l_1,2} & \dots & \lambda_{l_1,l_1} & \lambda_{l_1,l_2} \\ \lambda_{l_2,1} & \lambda_{l_2,2} & \dots & \lambda_{l_2,l_1} & \lambda_{l_2,l_2} \end{pmatrix}$$

The net current through the identified node would be the sum of the current through the nodes l_1 and l_2, so the response matrix for the modified network is:

$$\Lambda^\star = \begin{pmatrix} \lambda_{1,1} & \lambda_{1,2} & \dots & \lambda_{1,l_1} + \lambda_{1,l_2} \\ \lambda_{2,1} & \lambda_{2,2} & \dots & \lambda_{2,l_1} + \lambda_{2,l_2} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{l_1,1} + \lambda_{l_2,1} & \lambda_{l_1,2} + \lambda_{l_2,2} & \dots & \lambda_{l_1,l_1} + 2\lambda_{l_1,l_2} + \lambda_{l_2,l_2} \end{pmatrix}$$

2.3 Modified Network

This section is about some of the direct applications of amalgamating networks and identifying boundary nodes to analyze the recoverability of networks.

Definition 2.8 *A series of amalgamations, identifying boundary nodes, and interiorizing boundary nodes is called* ***modification*** *on networks. If modification M is operated on networks with response matrices $\Lambda_1, \Lambda_2, \dots, \Lambda_n$ then the response matrix of the resulting network is denoted $M(\Lambda_1, \Lambda_2, \dots, \Lambda_n)$.*

Theorem 2.9 (Recoverability of Modified Networks) *Let M be a modification operated on networks with graphs $G_1, G_2, \dots, G_n$ and response matrices $\Lambda_1, \Lambda_2, \dots, \Lambda_n$, respectively. Let G be the graph of the modified network. If G is a recoverable graph given the response matrix, $M(\Lambda_1, \Lambda_2, \dots, \Lambda_n)$, then all of the graphs G_i involved in the modification M are recoverable given their response matrices, Λ_i.*

Proof: Given a graph G_i, construct a set of graphs

$$A = \{G_1, \dots, G_{i-1}, G_{i+1}, \dots G_n\}$$

so that the modification M operated on networks with the graphs A and G_i will result in a network with the graph G. Then let $K_1, \dots, K_{i-1}, K_{i+1}, \dots K_n$ be the Kirchhoff matrices for the graphs A so that all couductances for the edges are 1. Solve for the response matrix $\Lambda_1, \dots, \Lambda_{i-1}, \Lambda_{i+1}, \dots, \Lambda_n$ from the Kirchhoff matrices. Recover the Kirchhoff matrix K for the graph G using response matrix $M(\Lambda_1, \Lambda_2, \dots, \Lambda_n)$. Conductances for edges in G_i are all contained in K. ■

Example 2.10 Suppose G_1 is the graph of Figure 2.11a such that all named verteces are boundary nodes. It is a non-circular planar network, so the recoverability of this network is not obvious at first. But identifying nodes l_5 and $l_5^\star$ and amalgamating a network with graph G_2 of Figure 2.11b through nodes $L = \{(l_i, l_i')\}$ will result in well-studied rectangular network, which is known to be recoverable (see section 4.1.3). Therefore, G_1 is a recoverable given its response matrix by Recoverability of Modified Networks.

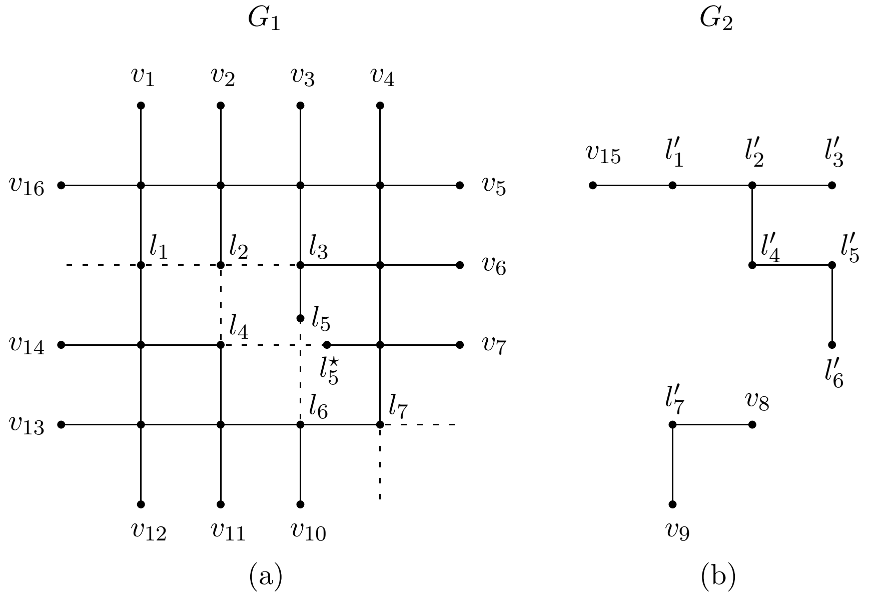


Figure 2.11 *Almost Rectangular Network*

Note that amalgamations and identifying nodes can be done only through boundary nodes, not interior nodes. There are many networks that may look like it can be modified into rectangular network, when it can not be.

Corollary 2.12 *If graph G with response matrix Λ is not recoverable, then for any modification M involving this network will result in a irrecoverable network.*

Proof: Assume the resulting network from modification M with graph G' is recoverable. Then by Recoverability of Modified Networks, graph

G with response matrix Λ is also recoverable. A contradiction is derived. Therefore, graph G can not be recoverable. ■

Example 2.13 Suppose G_1 is the graph of Figure 2.14a so that all of the nodes are indicated with dots. Boundary nodes of G are those indexed from 1 through 6. Notice that G_2 of Figure 2.14b is completely contained in G_1. In fact, identifying correct nodes between two of the graph G_2 and a piece of conductor is a modification that results a network with graph G_1. Since G_2 is a circular planar graph, methods of *medial graphs* can be applied, and can be shown to be not recoverable [1]. Since a network with G_1 can be thought of as a modified network involving graph G_2, by the above corollary, network with graph G_1 is not recoverable.

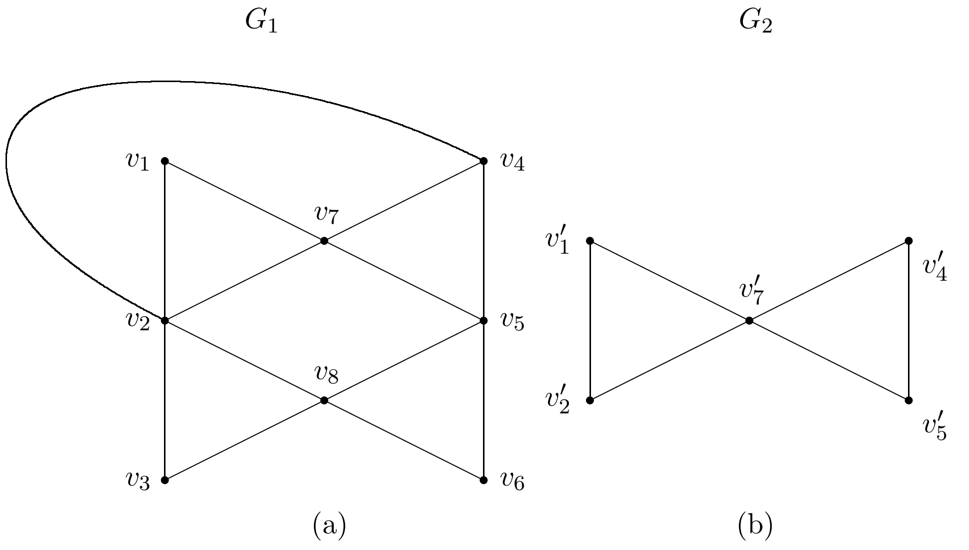


Figure 2.14 *An Irrecoverable Network*

As previous two examples have shown, modifying networks can help analyze graphs that are not circular planar by simplifying the problem. Simplifying the problem into a circular planar one is often useful because the theory involving circular planar networks is much more developed. In general, it helps to know large recoverable networks and small irrecoverable networks.

Chapter 3

Separating Two Networks

For separating two networks first recall the amalgamation. Suppose two networks Γ_1 and Γ_2 have the following response matrices:

$$\Lambda_1 = \begin{pmatrix} A_1 & B_1 \\ B_1^T & C_1 \end{pmatrix} \qquad \Lambda_2 = \begin{pmatrix} A_2 & B_2 \\ B_2^T & C_2 \end{pmatrix}$$

Amalgamate the two networks to get the following:

$$\Lambda^\star = \begin{pmatrix} A_1 & 0 & B_1 \\ 0 & A_2 & B_2 \\ B_1^T & B_2^T & C_1 + C_2 \end{pmatrix}$$

$$\Lambda = \Lambda_1 \bowtie \Lambda_2 = A - B(C^{-1})B^T$$

where,

$$A = \begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} \qquad B = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} \qquad C = C_1 + C_2.$$

The goal in separating two networks is to start with Λ and Λ_1 and recover Λ_2. However, the first question to ask is when can this process be done. One such way is *Shell Graph Removal*.

3.1 Shell Graph and Inner Graph Separation

Suppose that $G = (E, V)$ is a graph with response matrix Λ. Also, suppose that G has the shell graph, $\mathbb{S}(G)$, and inner graph, $\mathbb{I}(G)$. The goal of this

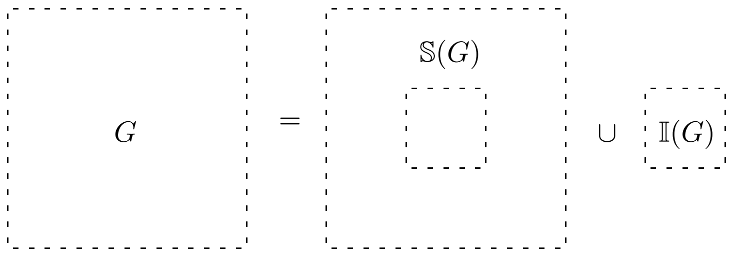


Figure 3.1 *Schematic of Shell Graph and Inner Graph Relationship*

network separation is to calculate the response matrix for the shell graph, remove the the shell graph, and recover the response matrix for the inner graph. First, it is important to state and prove a theorem about the shell of a graph.

Theorem 3.2 (Shell-to-Kirchhoff Theorem) *Suppose G is a graph and $\mathbb{S}(G)$ is the shell graph. Then the response matrix for $\mathbb{S}(G)$ is the Kirchhoff matrix for $\mathbb{S}(G)$.*

Proof: The theorem follows directly from the definition of the shell of a graph. Suppose G is a graph and $\mathbb{S}(G)$ is its shell graph. From the definition of shell, all of the nodes of $\mathbb{S}(G)$ are boundary nodes of $\mathbb{S}(G)$. Therefore, from the defintion of the Kirchhoff matrix and the response matrix the two matrices are equal. ■

With this theorem it is possible to immediately get the response matrix for a shell graph from the Kirchhoff matrix. The key assumption to make here is that the Kirchhoff matrix for the shell graph is easy to recover from response matrix of the original graph.

With this assumption we can begin to state when the shell graph and inner graph separation process if valid. Suppose $G = (E, V)$ and its shell graph, $\mathbb{S}(G)$, are given. Also, suppose that Λ, the response matrix of G, is given and a calculation can be made to determine the conductances of $\mathbb{S}(G)$.

By the Shell-to-Kirchhoff Theorem the Kirchhoff matrix for $\mathbb{S}(G)$ is Λ_1, the response matrix for $\mathbb{S}(G)$. Since every node of $\mathbb{S}(G)$ is a boundary node it is possible to amalgamate $\mathbb{S}(G)$ and $\mathbb{I}(G)$ to get G.

Let n be the number of boundary nodes of $\mathbb{S}(G)$ and j be the number of boundary nodes of $\mathbb{I}(G)$. The boundary nodes of $\mathbb{S}(G)$ need to be reordered such that the first nodes $H = (1, \dots, n - j + 1)$ are the nodes not being amalgamated with the boundary nodes of $\mathbb{I}(G)$ and the remaining nodes $J = (n - j, \dots, n)$ are the nodes being amalgamated. With this reordering the response matrix for $\mathbb{I}(G)$,

$$\Lambda_1 = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix}$$

Note that dimentions of Λ_2 and the block C of Λ_1 is both $j \times j$.

Since $\Lambda_1 \bowtie \Lambda_2 = \Lambda$ the following is true from the amalgamation,

$$\Lambda^\star = \begin{pmatrix} A & B \\ B^T & C + \Lambda_2 \end{pmatrix}$$

taking the Schur Complement yields

$$\Lambda = A - B(C + \Lambda_2)^{-1} B^T.$$

The following steps show how to solve for Λ_2

$$\begin{aligned} B(C + \Lambda_2)^{-1} B^T &= A - \Lambda \\ B^T B(C + \Lambda_2)^{-1} B^T B &= B^T (A - \Lambda) B \\ (C + \Lambda_2)^{-1} &= (B^T B)^{-1} B^T (A - \Lambda) B (B^T B)^{-1} \\ C + \Lambda_2 &= [(B^T B)^{-1} (B^T (A - \Lambda) B)(B^T B)^{-1}]^{-1}. \end{aligned}$$

The following formula is now derived

$$\Lambda_2 = [(B^T B)^{-1} (B^T (A - \Lambda) B)(B^T B)^{-1}]^{-1} - C.$$

However, this formula is only true when $B^T B$ and $\Lambda_2 + C$ are invertible. The following lemma proves the invertibility of $\Lambda_2 + C$.

Lemma 3.3 (The Fundamental Lemma of Invertibilty) *Suppose G is a graph with response matrix Λ, Λ_2 is the response matrix for $\mathbb{I}(G)$, and the response matrix for $\mathbb{S}(G)$ is*

$$\Lambda_1 = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix}$$

then $\Lambda_2 + C$ is an invertible matrix.

Proof: Suppose the hypothesis of the theorem. Let n be the number of boundary nodes of $\mathbb{S}(G)$ and j be the number of boundary nodes of $\mathbb{I}(G)$. Let $H = (1, \ldots, n-j+1)$ be the set of nodes of Λ_1 that are not being amalgamated with Λ_2 and let $J = (n-j, \ldots, n)$ be the set of boundary nodes of Λ_1 that are amalgamated with Λ_2. This implies that Λ_2 and C are both $j \times j$ matrices. Note that Λ_2 is a semi-diagonally dominant matrix by definition of a response matrix. Since the diagonal entries of C are the sum of the off diagonal entries of the corresponding row or column of the matrix Λ_1 and C is a submatrix of Λ_1. It follows that the diagonal entries of C will be larger in absolute values than the sum of the off diagonal entries of C. Thus, C is a diagonally dominant matrix. Since C is diagonally dominant and Λ_2 is semi-dominant it follows that their sum $\Lambda_2 + C$ must be diagonally dominant. Hence, $\Lambda_2 + C$ is an invertible matrix. ■

The following theorem states when shell graph and inner graph separation is possible and provides the general formula previously derived.

Theorem 3.4 (The Shell Stripping Formula) *Let G be a graph with response matrix Λ. Suppose that $\mathbb{S}(G)$ has n boundary nodes and an $n \times n$ repsonse matrix*

$$\Lambda_1 = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix}$$

such that $H = (1, \ldots, n-j+1)$ is the set of nodes not being amalgamated to $\mathbb{I}(G)$ and $J = (n-j, \ldots, n)$ is the set of nodes being amalgamated. Suppose further that $(B^T B)$ is a $j \times j$ invertible matrix. Then,

$$\Lambda_2 = [(B^T B)^{-1}(B^T(A - \Lambda)B)(B^T B)^{-1}]^{-1} - C$$

where Λ_2 is the $j \times j$ response matrix for $\mathbb{I}(G)$.

Proof: The proof for this theorem follows from the above arguement about the existance of the shell stripping formula. ■

3.2 Amalgamating Networks With Negative Conductances

This section discusses the another process for separating networks using amalgamation of networks with negative conductances. At this point, however, this method only works for very specific cases, namely the spike removal and boundary to boundary edge removal. More research is required to determine when in general this method will work (see conculsion). Despite this restriction there are two main advantages in choosing this method. The first being that only one matrix inverse is taken in the calculation, whereas in the shell removal process several need to be taken. The second advantage is that this method can be done in parts, that is one spike or edge can be removed at a time, or in groups.

3.2.1 Spike Removal

The goal of spike removal is to remove a given number of spikes with known conductances from a graph to possibly make the graph more recoverable. For example, it is possible to remove the spikes from a rectangular network to arrive a network (called to square network). Chapter 4 contains an example of how spike removal can be used to recover the conductances of a rectangular network.

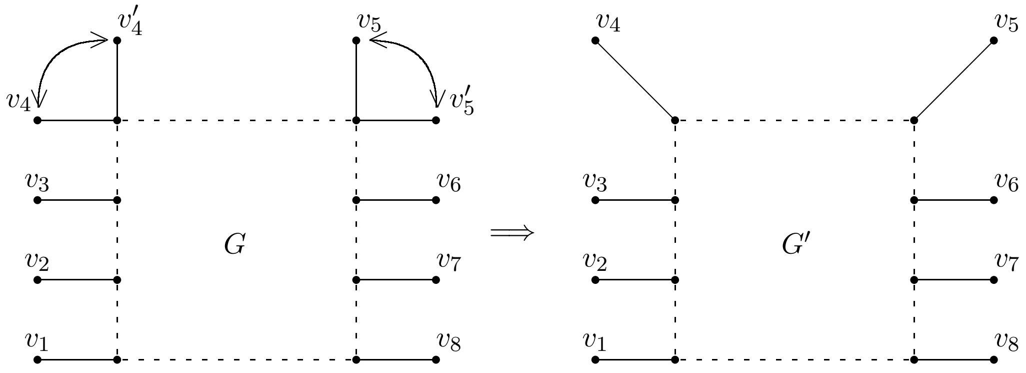


Figure 3.5 *Identifying Boundary Nodes*

To begin spike removal, suppose that $\Gamma = (G, \gamma)$ is a network with a given number of spikes with known conductances, or conductances which can be found and Λ is its response matrix. The first step in spike removal is to look for the spikes with a common interior node. In Figure 3.5 the spikes with boundary nodes v_4 and v_4' have a common interior node, as do the spikes with boundary nodes v_5, and v_5'. Using § 2.2, the boundary nodes v_4 and v_4' and v_5, and v_5' may be identified into just the nodes v_4 and v_5. This creates a new graph, call it G'. By § 2.2, call the response matrix for G' Λ'. Also, because all of the conductances of the spikes are known, the new conductances of the newly identified spikes in G' are also known.

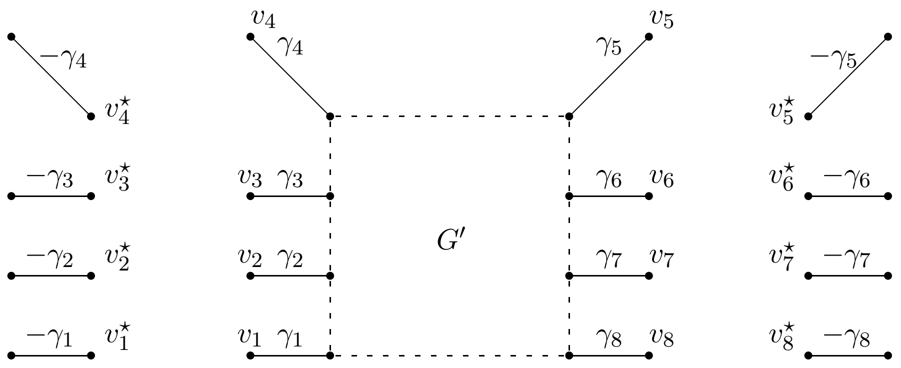


Figure 3.6 *Removing the Spikes*

The next step in the process is to amgalgamate at each spike in G', a graph which consists of only an edge and two boundary nodes. Each of these graphs must have conductance which is the negative of the conductance of the spike which they are being amalgamated to. By the Shell-to-Kirchhoff Theorem, the response matrices for these graphs is their Kirchhoff matrices. In figure 3.6 node v_1 is being amagamated to $v_1^\star$, the conductance of the spike is γ_1, and the conductance of the graph with $v_1^\star$ as a boundary node is $-\gamma_1$. Once these new graphs are amalgamated to G' the process is complete and the new graph is G without any spikes and a new response matrix Λ''.

3.2.2 Boundary to Boundary Edge Removal

Boundary to boundary edge removal, is in effect just the amalgamaiton of a graph with negative conductance to the boundary to boundary edge of a give graph.

Suppose that $\Gamma = (G, \gamma)$ is a network with a response matrix Λ. Let p

and q be two boundary nodes with an edge e_{pq} between them. Also, let this edge, e_{pq} be the edge that is to be removed and have a conductance ξ.

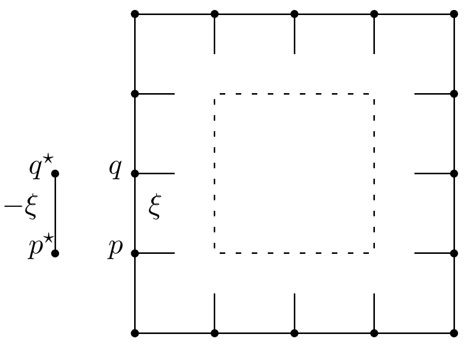


Figure 3.7 *Boundary to Boundary Edge Removal*

To remove the edge, another graph which consists of two boundary nodes an the edge between them must be amalgamated to the graph G at nodes p and q. Let this graph be an edge e' with boundary nodes $p^\star$ and $q^\star$. When the e' is amalgamated to G, amalgamate p to $p^\star$ and q to $q^\star$. Figure 3.7 shows G and e'. When the process is complete it is possible to remove other boundary to boundary edges and arrive at spikes. If the conductances of these spikes can be found it is possible to remove them using the spike removal process previously described.

3.3 Separation Across Boundary Nodes

For separation across boundary nodes the goal is to separate a graph at a set of designated boundary nodes as in Figure 3.8 and recover the response matrices of its parts. Suppose G is a graph as in Figure 3.8 with response matrix Λ. Let the set of boundary nodes where the separation is to occur be $(l_1, \ldots, l_n)$ and let the remaining boundary nodes be $(j_1, \ldots, j_s)$ and $(h_1, \ldots, h_t)$. Call G_1 and G_2 the two graphs that G will be separated into. Also, let Λ_1 and Λ_2 be the response matrices for G_1 and G_2, respectively. The goal of the separation is to calculate the entries of Λ_1 and Λ_2 from Λ.

Since $(l_1, \ldots, l_n)$ are the boundary nodes where G_1 and G_2 would amal-

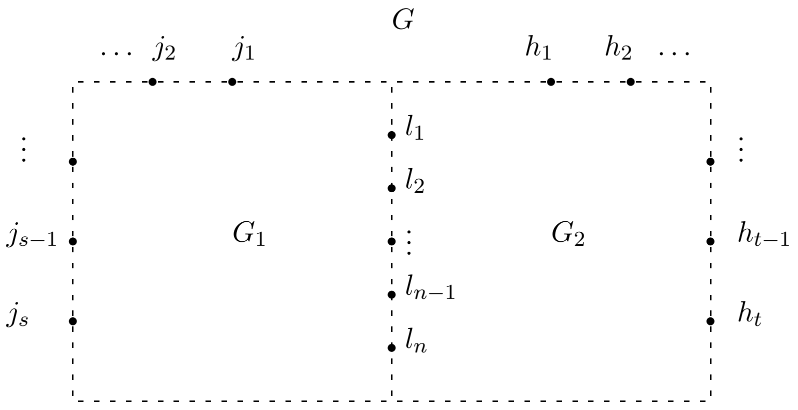


Figure 3.8 *Separation Across Boundary Nodes*

gamate at to get G,

$$\Lambda_1 = \begin{pmatrix} A_1 & B_1 \\ B_1^T & C \end{pmatrix} \qquad \Lambda_2 = \begin{pmatrix} A_2 & B_2 \\ B_2^T & D \end{pmatrix}$$

where B_1 is $s \times n$, B_2 is $t \times n$, A_1 is $s \times s$, A_2 is $t \times t$, and C and D are $n \times n$.

Also, since $(l_1, \dots, l_n)$ remain boundary nodes after the amalgamtion the response matrix for G, Λ is found without taking a Schur Complement and is written:

$$\Lambda = \begin{pmatrix} A_1 & 0 & B_1 \\ 0 & A_2 & B_2 \\ B_1^T & B_2^T & C+D \end{pmatrix}$$

From Λ the entries for B_1, B_2, A_1, and A_2 can be found. The only unknown remaining entries of Λ_1 and Λ_2 are C and D. Since Λ is a symmetric matrix its submatrix, $C+D$, is also symmetric. Thus, $C+D$ can be written as:

$$C+D = \begin{pmatrix} c_{11}+d_{11} & c_{12}+d_{12} & \dots & c_{1n}+d_{1n} \\ c_{12}+d_{12} & c_{22}+d_{22} & \dots & c_{2n}+d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{1n}+d_{1n} & c_{2n}+d_{2n} & \dots & c_{nn}+d_{nn} \end{pmatrix}$$

Since $C+D$ is known, knowing entries of C determines the entries of D. Since C is symmetric, the only unknowns of C are

$$c_{11}, c_{12}, \dots, c_{1n}, c_{22}, \dots, c_{2n}, \dots, c_{nn},$$

by a simple counting arguement there are $\dfrac{n(n+1)}{2}$ unknowns. From the definition of the response matrix, the values of c_{ii} are the negative of the sum of the rows. Thus,

$$c_{ii} = \sum_{r=1}^{s} b_{ir} + \sum_{r \neq i} c_{ir}$$

for $i = 1$ to $i = n$. Therefore, there are n equations and $\dfrac{n(n+1)}{2}$ unknowns. This leaves $\dfrac{n(n-1)}{2}$ free variables.

Plugging in arbitary values for the free variables does produce a pair of networks that amalgamates into the original graph, but seperated graphs may contain negetive conductors. For example, consider separation along 3 boundary nodes as shown in Figure 3.9a. Say that c_{12}, c_{13}, and c_{23} are chosen to be the 3 free variables. The value chosen for c_{12} effects the conductances of $\gamma^{\star}_{12}$ and γ'_{12} (see figure 3.10b), but they always maintain the relationship $\gamma_{12} = \gamma^{\star}_{12} + \gamma'_{12}$. Note that even if $\gamma_{12} = 0$ (there is no conductor between nodes l_1 and l_2 in the original graph), $\gamma^{\star}_{12}$ and γ'_{12} may still be nonzero. In this case, they will maintain the relationship $\gamma^{\star}_{12} = -\gamma'_{12}$.

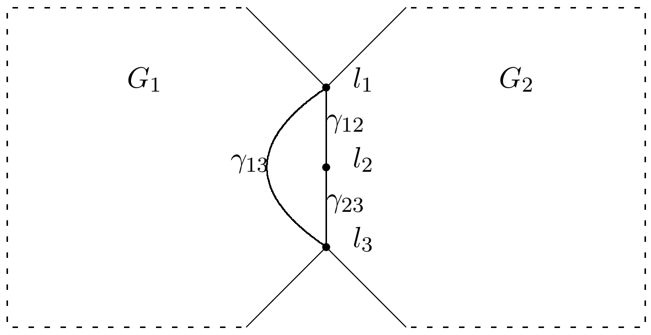


Figure 3.9 *Limitations of Separation Along Boundary Nodes (a)*

Negetive conductors has exactly the opposite effect of positive conductors; they push the current up to nodes with higher voltages instead of nodes

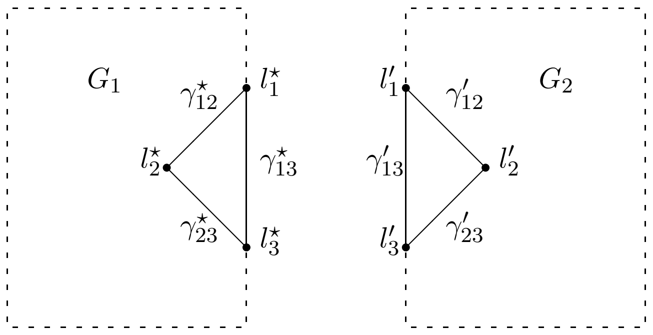


Figure 3.10 *Limitations of Separation Along Boundary Nodes (b)*

with lower voltages. When calculations are being made around negetive conductors, care should be taken not to use any theorems or results obtained by the assumption of all conductors having positive values. Finding the correct range of c_{ij} so that both $\gamma_{ij}^{\star}$ and γ_{ij}' will take on positive values (perhaps by a numerical method) can be a topic for further research.

A necessary condition for c_{ij} to have both $\gamma_{ij}^{\star}$ and γ_{ij}' positive is that:

$$\text{If} \quad i = j, \quad \text{then} \quad 0 \leq c_{ij} \leq (C + D)_{ij}$$

and

$$\text{If} \quad i \neq j, \quad \text{then} \quad (C + D)_{ij} \leq c_{ij} \leq 0$$

but this is not sufficient to gurantee positive conductances.

3.4 Separation Across One Interior Node

This section deals with a theorem about separating a graph into two graphs at an interior node. Some possible applications are given as pictures in the end of this section.

Theorem 3.11 (Separation at One Interior Node) *Let G be a graph with response matrix Λ. Suppose that $\Lambda = \Lambda_1 \bowtie \Lambda_2$ where Λ_1 and Λ_2 are response matrices for G_1 and G_2. Also, suppose that G_1 and G_2 are* *amalgamated to produce G at a single interior node of G called p. Finally, suppose that when G is separated into G_1 and G_2 at least one 1-connection in either G_1 or G_2 is broken. Then, Λ_1 and Λ_2 can be recovered from Λ.*

Proof: Let p be the interior node of G where the separation takes place. Let H be the set of boundary nodes of G_1 that are boundary nodes of G and J be the set of boundary nodes of G_2 that are boundary nodes of G. Say that H has m elements and J has n elements. Thus, Λ is a $(m+n)\times(m+n)$ matrix. From the amalgamation of networks and $\Lambda_1 \bowtie \Lambda_2 = \Lambda$

$$\Lambda^\star = \begin{pmatrix} A_1 & 0 & B_1 \\ 0 & A_2 & B_2 \\ B_1^T & B_2^T & -\sum_{i=1}^{m+n} b_i \end{pmatrix}$$

where

$$\Lambda_1 = \begin{pmatrix} A_1 & B_1 \\ B_1^T & -\sum_{i=1}^{m} b_i \end{pmatrix} \quad \text{and} \quad \Lambda_2 = \begin{pmatrix} A_2 & B_2 \\ B_2^T & -\sum_{i=m+1}^{m+n} b_i \end{pmatrix}$$

and B_1 and B_2 are column vectors. Completing the amalgamation process yields,

$$\Lambda = \quad A \quad + \quad \frac{1}{\sum_{i=1}^{m+n} b_i} B^T B$$

where $B = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix}$ and $B^T = \begin{pmatrix} B_1^T & B_2^T \end{pmatrix}$.
With this relationship between Λ and A_1, A_2, B, and B^T the following equations are established:

$$a_{\tau\epsilon} + \frac{b_\tau b_\epsilon}{\sum_{i=1}^{m+n} b_i} = \lambda_{\tau\epsilon} \quad \text{for} \quad \tau, \epsilon \in H \tag{3.1}$$

$$\frac{b_\tau b_\xi}{\sum_{i=1}^{m+n} b_i} = \lambda_{\tau\xi} \quad \text{for} \quad \tau \in H, \xi \in J \tag{3.2}$$

$$a_{\xi\nu} + \frac{b_\xi b_\nu}{\sum_{i=1}^{m+n} b_i} = \lambda_{\xi\nu} \quad \text{for} \quad \xi, \nu \in J \tag{3.3}$$

where a_{ij} is the ij^{th} entry of the matrix A, and λ_{ij} is the ij^{th} entry of Λ.

In equation 3.2 if all $\lambda_{ij} = 0$, then this case can be disregarded because this means that there is not even a 1-connection between H nodes and J

nodes.

Again, in equation 3.2 suppose that there exist ρ or ζ such that $\lambda_{\rho\zeta} = 0$ then one of the following must be true: (i) $b_\rho = 0$, or (ii) $b_\zeta = 0$. If (i) is true then $\lambda_{\rho i} = 0$ for all i. This implies that $a_{\rho i} = \lambda_{\rho i} = 0$ by equation 3.1. If (ii) is true then $\lambda_{j\zeta} = 0$ for all j. Which implies that $a_{j\zeta} = \lambda_{j\zeta} = 0$ by equation 3.3.

Now take $b_i \neq 0$ for all $1 \leq i \leq m+n$. For fixed $\zeta, \rho \in H$ and any $i \in J$ $\frac{b_\zeta b_i}{\sum_{j=1}^{m+n} b_j} = \lambda_{\zeta i}$ and $\frac{b_\rho b_i}{\sum_{j=1}^{m+n} b_j} = \lambda_{\rho i}$ by equation 3.2. Taking the ratio of the first equality to the second yields the following for fixed $\zeta, \rho \in H$ and any $i \in J$:

$$\frac{b_\zeta}{b_\rho} = \frac{\lambda_{\zeta i}}{\lambda_{\rho i}}. \tag{3.4}$$

Apply the same technique for fixed $\beta, \delta \in J$ and any $j \in H$ $\frac{b_j b_\beta}{\sum_{i=1}^{m+n} b_i} = \lambda_{j\beta}$ and $\frac{b_j b_\delta}{\sum_{i=1}^{m+n} b_i} = \lambda_{j\delta}$ from equation 3.2. Again, taking the ratios of these two equalities yields for fixed $\beta, \delta \in J$ and any $j \in H$

$$\frac{b_\delta}{b_\beta} = \frac{\lambda_{\delta i}}{\lambda_{\beta i}}. \tag{3.5}$$

Now, suppose that the connection through p that must be broken during separation is the connection in H between μ and $\mu^\star$. It follows that, $a_{\mu\mu^\star} = 0$ because the connection is broken. From equation 3.1 $a_{\mu\mu^\star} = 0$ implies

$$\frac{b_\mu b_{\mu^\star}}{\sum_{i=1}^{m+n} b_i} = \lambda_{\mu\mu^\star}. \tag{3.6}$$

Now dividing equation 3.6 by equation 3.2 with

$$\sigma \in J \quad \text{and} \quad \mu \in H \qquad \frac{b_\mu b_\sigma}{\sum_{i=1}^{m+n} b_i} = \lambda_{\mu\sigma}$$

Yields,

$$\frac{b_{\mu^\star}}{b_\sigma} = \frac{\lambda_{\mu\mu^\star}}{\lambda_{\mu\sigma}} \tag{3.7}$$

which will be called the *Key Ratio*.

The Key Ratio combined with equation 3.4 and 3.5 will give a ratio between b_i and b_j. Now, to find any b_k, simply start with equation 3.2, $\frac{b_k b_\sigma}{\sum_{i=1}^{m+n} b_i} = \lambda_{k\sigma}$. Using the Key Ratio, all of the terms of the summation can be put in terms of b_σ. Factoring out the b_σ and canceling yields

$$\frac{b_k}{\psi_\sigma} = \lambda_{k\sigma}$$

where ψ_σ is an expression composed entirely of known λ_{ij}. Therefore, $b_k = \lambda_{k\sigma}\psi_\sigma$.

Since every b_k can be found using equations 3.1 and 3.3 all of the a_{ij} can also be found. Therefore, Λ_1 and Λ_2 can be recovered. ■

Chapter 4

Three Examples

These three examples utilize several different aspects of network amalgamation, shell graph removal, network separation, and spike and boundary to boundary edge removals to recover the Kirchhoff matrix from the response matrix.

4.1 The Towers of Hanoi Networks

The Tower of Hanoi Networks are defined in the paper by Mike Usher in terms of lattice points [2]. However, the networks are simple enough to be defined in terms of their pictures. Figure 4.1 is an example of a tower network with 8 boundary nodes and Figure 4.2 is an example of a tower network with 7 boundary nodes. The boundary nodes of the tower networks are always outer most nodes on the top right and left sides of the graph, in Figures 4.1 and 4.2 the boundary nodes are labeled v_i.

When a shell graph is removed from a Tower of Hanoi Network another tower of smaller size remains. It is through this property that the recovery of the Kirchhoff Matrix for a Tower of Hanoi Network is possible. The first step in this recovery process is to calculate the conductances of the shell edges of the tower. The next step is to remove the shell graph and calculate the response matrix for the smaller tower network. Once enough shell graphs are removed a simple recoverable network remains.

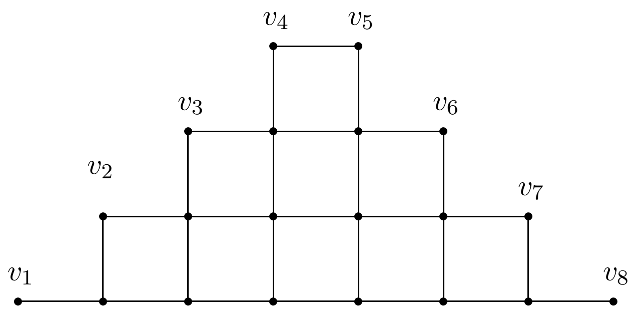


Figure 4.1 *8 Boundary Node Even Case*

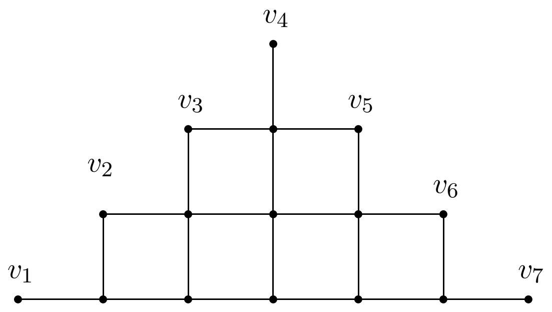


Figure 4.2 *7 Boundary Node Odd Case*

4.1.1 Calculating the Boundary Conductances

The Even Case

Assume that there are an even number of boundary nodes. We first calculate the boundary conductances for the rightside of the boundary and then the leftside of the boundary.

The Rightside

For the rightside we will calculate the conductances between nodes $\frac{n}{2}$ and n.

Boundary Conditions

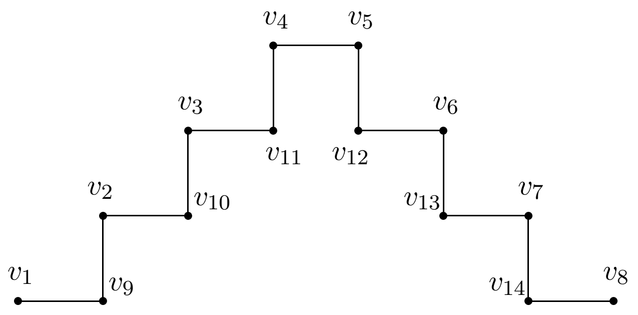


Figure 4.3 *Even Shell of the 8 Boundary Node Case*

For simplicity let $n = 2m$. Now, impose currents of zero at boundary nodes 1 through $m - 1$ and impose the following voltages at all other nodes.

$$\alpha_i = \begin{cases} 0 & \text{if} \quad 1 \leq i \leq m \\ 1 & \text{if} \quad i = m+1 \\ \text{unknown} & \text{if} \quad m+2 \leq i \leq n \end{cases}$$

Setup and Solve Linear System
By imposing currents of zero at nodes 1 to $m - 1$ we have the following equations:

$$\begin{aligned} 0 &= \sum_{j=m+1}^{2m} \alpha_j \lambda_{1,j} \\ 0 &= \sum_{j=m+1}^{2m} \alpha_j \lambda_{2,j} \\ 0 &= \sum_{j=m+1}^{2m} \alpha_j \lambda_{3,j} \\ \vdots & \quad \vdots \\ 0 &= \sum_{j=m+1}^{2m} \alpha_j \lambda_{2m,j} \end{aligned}$$

In order to solve this system, let

$$\mathbf{A} = \begin{pmatrix} \lambda_{1,m+2} & \lambda_{1,m+3} & \cdots & \lambda_{1,2m} \\ \lambda_{2,m+2} & \lambda_{2,m+3} & \cdots & \lambda_{2,2m} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{m-1,m+2} & \lambda_{m-1,m+3} & \cdots & \lambda_{m-1,2m} \end{pmatrix}$$

$$\mathbf{S} = \begin{pmatrix} -\lambda_{1,m+1} \\ -\lambda_{2,m+1} \\ \vdots \\ -\lambda_{m-1,m+1} \end{pmatrix} \qquad \mathbf{X} = \begin{pmatrix} \alpha_{m+2} \\ \vdots \\ \alpha_{2m} \end{pmatrix}$$

We now have the following:

$$\mathbf{AX} = \mathbf{S}$$

thus,

$$\mathbf{X} = \mathbf{A}^{-1}\mathbf{S} = \begin{pmatrix} \alpha_{m+2} \\ \vdots \\ \alpha_{2m} \end{pmatrix}$$

We now have the values for α_i for $i = 1$ to $i = 2m$.

Calculate γ_{ij} on the Shell
First we need to establish a vector of currents, call it $\mathbf{P}$.

$$\mathbf{P} = \mathbf{\Lambda} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{2m} \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{2m} \alpha_j \lambda_{1,j} \\ \sum_{j=1}^{2m} \alpha_j \lambda_{2,j} \\ \vdots \\ \sum_{j=1}^{2m} \alpha_j \lambda_{2m,j} \end{pmatrix}$$

Note that the entry $\mathbf{P_i}$ is the current at node i due to the voltages α_j. Using this vector $\mathbf{P}$ we can calculate the values of the conductance γ_{ij} using the following formulae [1].

$$\gamma_{i,i+2m-1} = \frac{\sum_{j=m}^{i} \mathbf{P}_j}{\alpha_i} \qquad \gamma_{i+1,i+2m-1} = \frac{\sum_{j=m}^{i} \mathbf{P}_j}{-\alpha_{i+1}}$$

for $i = m$ to $i = 2m - 1$.

[1] see appendix for details of derivation

The Leftside

For the leftside we will calculate the conductances for the remaining boundary edges between nodes 1 and $\frac{n}{2}$.

Boundary Conditions

Note $n = 2m$. Again, impose currents of zero at boundary nodes $m + 2$ through $2m$ and impose the following voltages at all other nodes.

$$\alpha_i = \begin{cases} \text{unknown} & \text{if} \quad 1 \le i \le m-1 \\ 1 & \text{if} \quad i = m \\ 0 & \text{if} \quad m+1 \le i \le 2m \end{cases}$$

Setup and Solve Linear System

By imposing currents of zero at nodes $m + 2$ to $2m$ we have the following equations:

$$\begin{aligned} 0 &= \sum_{j=1}^{m} \alpha_j \lambda_{m+2,j} \\ 0 &= \sum_{j=1}^{m} \alpha_j \lambda_{m+3,j} \\ 0 &= \sum_{j=1}^{m} \alpha_j \lambda_{m+4,j} \\ \vdots & \quad \vdots \\ 0 &= \sum_{j=1}^{m} \alpha_j \lambda_{2m,j} \end{aligned}$$

In order to solve this system, let

$$\mathbf{A} = \begin{pmatrix} \lambda_{m+2,1} & \lambda_{m+2,2} & \cdots & \lambda_{m+2,m-1} \\ \lambda_{m+3,1} & \lambda_{m+3,2} & \cdots & \lambda_{m+3,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{2m,1} & \lambda_{2m,2} & \cdots & \lambda_{2m,m-1} \end{pmatrix}$$

$$\mathbf{S} = \begin{pmatrix} -\lambda_{m+2,m} \\ -\lambda_{m+3,m} \\ \vdots \\ -\lambda_{2m,m} \end{pmatrix} \qquad \mathbf{X} = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{m-1} \end{pmatrix}$$

We now have the following:

$$\mathbf{AX} = \mathbf{S}$$

thus,

$$\mathbf{X} = \mathbf{A}^{-1}\mathbf{S} = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{m-1} \end{pmatrix}$$

We now have the values for α_i for $i = 1$ to $i = 2m$.

Calculate γ_{ij} on the Shell
First, we need to establish a vector of currents, call it $\mathbf{P}$.

$$\mathbf{P} = \mathbf{\Lambda} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{2m} \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{2m} \alpha_j \lambda_{1,j} \\ \sum_{j=1}^{2m} \alpha_j \lambda_{2,j} \\ \vdots \\ \sum_{j=1}^{2m} \alpha_j \lambda_{2m,j} \end{pmatrix}$$

Note that the entry $\mathbf{P_i}$ is the current at node i due to the voltages α_j. Using the vector $\mathbf{P}$ we can calculate the values of the conductance γ_{ij} using the following formulae [2].

$$\gamma_{i,i+2m} = \frac{\sum_{j=1}^{i} \mathbf{P}_j}{\alpha_i} \qquad \gamma_{i+1,i+2m} = \frac{\sum_{j=1}^{i} \mathbf{P}_j}{-\alpha_{i+1}}$$

for $i = 1$ to $i = m - 1$.

The Odd Case

Suppose there are an odd number of boudary nodes. We first calculate the rightside conductances and then the leftside conductances.

The Rightside

For the rightside we calculate the conductances between nodes $\dfrac{n+3}{2}$ and n.

Boundary Conditions
Let $n = 2l - 1$. Now, impose currents of zero at boundary nodes $l + 1$ through $2l - 1$ and impose the following voltages at all other nodes.

[2] see appendix for details of derivation

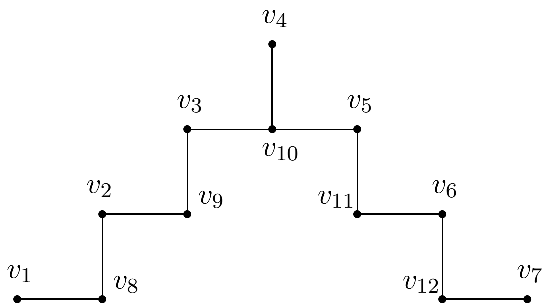


Figure 4.4 *Odd Shell of 7 Boundary Node Case*

$$\alpha_i = \begin{cases} 0 & \text{if} \quad 1 \le i \le l-1 \\ 1 & \text{if} \quad i = l \\ \text{unknown} & \text{if} \quad l+1 \le i \le 2l-1 \end{cases}$$

Setup and Solve Linear System
By imposing currents of zero at nodes $l+1$ to $2l-1$ we have the following equations:

$$\begin{aligned} 0 &= \sum_{j=l+1}^{2l-1} \alpha_j \lambda_{1,j} \\ 0 &= \sum_{j=l+1}^{2l-1} \alpha_j \lambda_{2,j} \\ 0 &= \sum_{j=l+1}^{2l-1} \alpha_j \lambda_{3,j} \\ \vdots & \quad \vdots \\ 0 &= \sum_{j=l+1}^{2l-1} \alpha_j \lambda_{l-1,j} \end{aligned}$$

In order to solve the system, let

$$\mathbf{A} = \begin{pmatrix} \lambda_{1,l+1} & \lambda_{1,l+2} & \cdots & \lambda_{1,2l-1} \\ \lambda_{2,l+1} & \lambda_{2,l+2} & \cdots & \lambda_{2,2l-1} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{l-1,l+1} & \lambda_{l-1,l+2} & \cdots & \lambda_{l-1,2l-1} \end{pmatrix}$$

$$\mathbf{S} = \begin{pmatrix} -\lambda_{1,l} \\ -\lambda_{2,l} \\ \vdots \\ -\lambda_{l-1,l} \end{pmatrix} \qquad \mathbf{X} = \begin{pmatrix} \alpha_{l+1} \\ \vdots \\ \alpha_{2l-1} \end{pmatrix}$$

We now have the following:

$$\mathbf{AX} = \mathbf{S}$$

thus,

$$\mathbf{X} = \mathbf{A}^{-1}\mathbf{S} = \begin{pmatrix} \alpha_{l+1} \\ \vdots \\ \alpha_{2l-1} \end{pmatrix}$$

Thus, we now have the values for α_i for $i = 1$ to $i = 2l - 1$.

Calculate γ_{ij} on the Shell
First we need to establish a vector of currents, call it $\mathbf{P}$.

$$\mathbf{P} = \mathbf{\Lambda} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{2l-1} \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{2l-1} \alpha_j \lambda_{1,j} \\ \sum_{j=1}^{2l-1} \alpha_j \lambda_{2,j} \\ \vdots \\ \sum_{j=1}^{2l-1} \alpha_j \lambda_{2l-1,j} \end{pmatrix}$$

Note that the entry $\mathbf{P_i}$ is the current at node i due to the voltages α_j. Using this vector $\mathbf{P}$ we can calculate the values of the conductance γ_{ij} using the following formulae [3].

$$\gamma_{i,i+2l-2} = \frac{\sum_{j=l}^{i} \mathbf{P}_j}{\alpha_i} \qquad \gamma_{i+1,i+2l-2} = \frac{\sum_{j=l}^{i} \mathbf{P}_j}{-\alpha_{i+1}}$$

[3]see appendix for details of derivation

for $i = l$ to $i = 2l - 2$.

The Leftside

For the leftside we calculate the conductances between nodes 1 and $\dfrac{n-1}{2}$.

Boundary Conditions

Note that $n = 2l - 1$. Impose currents of zero at boundary nodes 1 through $l - 1$ and impose the following voltages at all other nodes.

$$\alpha_i = \begin{cases} \text{unknown} & \text{if} \quad 1 \le i \le l-1 \\ 1 & \text{if} \quad i = l \\ 0 & \text{if} \quad l+1 \le i \le 2l-1 \end{cases}$$

Setup and Solve Linear System

By imposing currents of zero at nodes 1 to $l - 1$ we have the following equations:

$$\begin{aligned} 0 &= \sum_{j=1}^{l-1} \alpha_j \lambda_{l+1,j} \\ 0 &= \sum_{j=1}^{l-1} \alpha_j \lambda_{l+2,j} \\ 0 &= \sum_{j=1}^{l-1} \alpha_j \lambda_{l+3,j} \\ \vdots & \quad \vdots \\ 0 &= \sum_{j=1}^{l-1} \alpha_j \lambda_{2l-1,j} \end{aligned}$$

In order to solve the system, let

$$\mathbf{A} = \begin{pmatrix} \lambda_{l+1,1} & \lambda_{l+1,2} & \cdots & \lambda_{l+1,l-1} \\ \lambda_{l+2,1} & \lambda_{l+2,2} & \cdots & \lambda_{l+2,l-1} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{2l-1,1} & \lambda_{2l-1,2} & \cdots & \lambda_{2l-1,l-1} \end{pmatrix}$$

$$\mathbf{S} = \begin{pmatrix} -\lambda_{1,l} \\ -\lambda_{2,l} \\ \vdots \\ -\lambda_{l-1,l} \end{pmatrix} \qquad \mathbf{X} = \begin{pmatrix} \alpha_{l+1} \\ \vdots \\ \alpha_{2l-1} \end{pmatrix}$$

We now have the following:

$$\mathbf{AX} = \mathbf{S}$$

thus,

$$\mathbf{X} = \mathbf{A}^{-1}\mathbf{S} = \begin{pmatrix} \alpha_{l+1} \\ \vdots \\ \alpha_{2l-1} \end{pmatrix}$$

Thus, we now have the values for α_i for $i = 1$ to $i = 2l - 1$.

Calculate γ_{ij} on the Shell
First we need to establish a vector of currents, call it $\mathbf{P}$.

$$\mathbf{P} = \mathbf{\Lambda} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{2l-1} \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{2l-1} \alpha_j \lambda_{1,j} \\ \sum_{j=1}^{2l-1} \alpha_j \lambda_{2,j} \\ \vdots \\ \sum_{j=1}^{2l-1} \alpha_j \lambda_{2l-1,j} \end{pmatrix}$$

Note that the entry $\mathbf{P_i}$ is the current at node i due to the voltages α_j. Using the vector $\mathbf{P}$ we can calculate the values of the conductance γ_{ij} using the following formulae [4].

$$\gamma_{i,i+2l-1} = \frac{\sum_{j=1}^{i} \mathbf{P}_j}{\alpha_i} \qquad \gamma_{i+1,i+2l-1} = \frac{\sum_{j=1}^{i} \mathbf{P}_j}{-\alpha_{i+1}}$$

for $i = 1$ to $i = l - 1$.

4.1.2 The Recursive Step

The first step in the recursive recovery process is to calculate the conductances of the shell edges of the tower network. Once this is done the values

[4]see appendix for details of derivation

of these conductances can be stored into a matrix which will eventually become the Kirchhoff matrix. Since the conductances of the shell edges will create the Kirchhoff matrix for the shell graph of the tower, applying the Shell-to-Kirchhoff Theorem yields the response matrix for the shell graph.

From the shell graph removal section if Λ is the response matrix for the original tower and Λ_1 is the shell graph response matrix, then Λ_2, the response matrix for the inner graph of the original tower, can be found. This is accomplished with the Shell Stripping Formula.

Since the inner graph of a Tower of Hanoi Network is another Tower of Hanoi when the shell graph is removed the process can be repeated until the base are reached. Figure 4.5 (a) is the base case for the even towers and Figure 4.5 (b) is the base case for the odd towers.

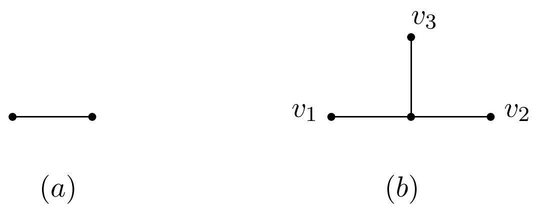


Figure 4.5 *Even and Odd Base Cases*

In the even base case every node is a boundary node, so the Kirchhoff Matrix is the response matrix. In the odd base case the boundary nodes are nodes v_1, v_2, and v_3 in Figure 4.5 (b). By imposing a current and voltage of 0 at node v_1 and by imposing a voltage of 1 at node v_3 it is possible to calculate the potential at node v_2. Once all of these conditions are met it is possible to calcuate the conductance at two of the three edges. To calculate the conductance of the third edge the impositions are reversed and the remaining conductance is read off.

4.1.3 Calculations and Results

MATLAB [5] was used to test the shell graph removal and recursive processes for the calculation of the Kirchhoff matrix for a Tower of Hanoi Network with various numbers of boundary nodes. According to the results with each increase in size approximately one decimal of accuracy is lost to rounding

[5]see appendix for MATLAB code

error. The error was measured at the by summing the absolutle values of the entries of the last column of the difference matrix between the original Kirchhoff Matrix and the calculated Kirchhoff Matrix. The output is listed as follows:

```
1                   0
2                   0
3                   0
4                   0
5    0.00000000000000
6    0.00000000000001
7    0.00000000000009
8    0.00000000000016
9    0.00000000000109
10   0.00000000000239
11   0.00000000004420
12   0.00000000011421
13   0.00000000102036
14   0.00000000518289
15   0.00000001792795
16   0.00000042297024
17   0.00000058812932
18   0.00000528858973
19   0.00002451756083
20   0.00042311265449
21   0.00296880884973
22   0.00023047406484
23   0.03022278375924
24   1.35906795181771
```

The first column is the size of the nodes and the second column is the error.

4.2 Rectangular Networks

m by n rectangular networks are grid-patterned networks with outer nodes designated as boundary nodes (see Figure 4.6). There are two cases: a spiked case and a square case. Spiked case does not have edges connecting adjacent boundary nodes, and m and n are the number of the boundary nodes on the sides. Adjacent boundary nodes in the square cases are connected with an edge, and m and n refers to the number of edges on the sides.

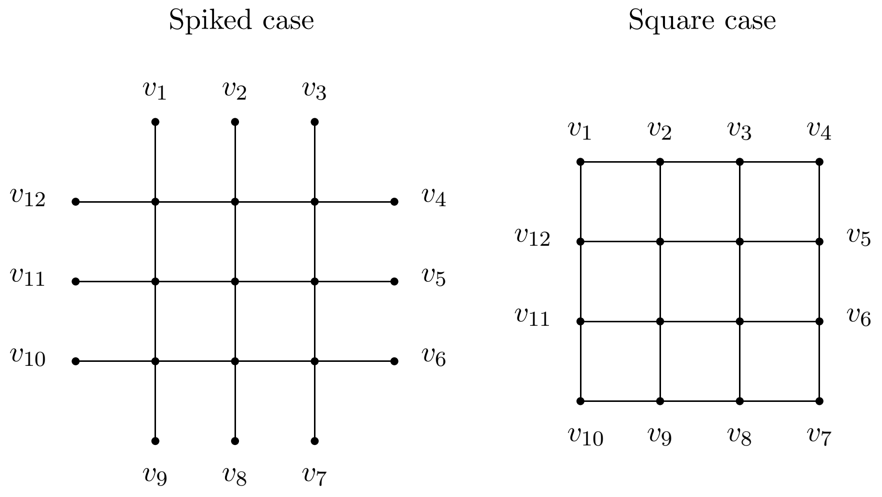


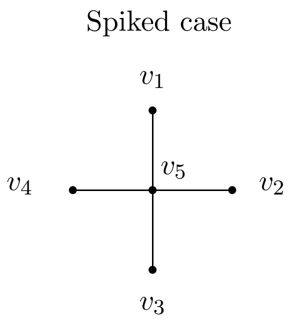
Figure 4.6 3 *by* 3 *rectangular network*

Note that any m by n rectangular network where $m \neq n$ can be modified into either m by m or n by n rectangular network, whichever one that is the larger of the two by simply amalgamting an appropriate strip of network with known conductance (see § 2.3). Therefore, without loss of generality, this section provides a method to solve for the Kirchhoff matrix K of the general n by n rectangular network with response matrix Λ.

4.2.1 Base cases

This section will describe how to get the conductances from the Response matrices for 1 by 1 spiked rectangular network (see Figure 4.7a) and 1 by 1 square rectangular network (see Figure 4.7b).

For 1 by 1 spiked case, let $\Lambda = (\lambda_{ij})$ be the response matrix for the network with the graph of Figure 4.7a. Impose voltage of 1 at node v_1 and voltages of 0 at nodes v_3 and v_4. To find voltage α to place on v_2 so that



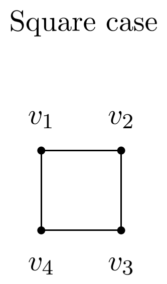


Figure 4.7 *Base cases*

the current at node v_3 will be 0, solve the equation

$$\alpha\lambda_{32} + \lambda_{31} = 0$$

for α. So

$$\alpha = -\frac{\lambda_{31}}{\lambda_{32}}$$

The current flowing out of node v_1 is

$$\begin{aligned} I &= \Delta V \gamma_{15} \\ &= (1-0)\gamma_{15} \\ &= \gamma_{15} \\ &= \alpha\lambda_{12} + \lambda_{11} \\ &= -\frac{\lambda_{31}}{\lambda_{32}}\lambda_{12} + \lambda_{11} \end{aligned}$$

Therefore, $\gamma_{15} = -\frac{\lambda_{31}}{\lambda_{32}}\lambda_{12} + \lambda_{11}$. By symmetry, other conductances can be found in a similar fashion.

No calculations are needed for 1 by 1 square case. Since all of the nodes in the graph are boundary nodes, response matrix is the same thing as the Kirchhoff matrix. Conductances are the entries of the Kirchhoff matrix by definition.

4.2.2 Spiked case

This section will describe how to solve for the conductances for the shell edges of n by n spiked case rectangular network and reduce the problem into $n-1$ by $n-1$ square case rectangular network.

Let $\Lambda = (\lambda_{ij})$ be the response matrix for n by n spiked case rectangular network. Consider a boundary node p on the northern face (let it be the kth node counted from the eastern face). Let ξ be the conductance of the spike edge by p. Let $W = (w_1, w_2, \ldots, w_k)$ be the first k indexes of boundary nodes on the western face counted from the northern face. Let $E = (e_1, e_2, \ldots, e_k)$ be the first k indexes of boundary nodes on the eastern face counted from the northern face. Let voltage at node p be 1, and impose 0 voltages at all other boundary nodes in northern, western, and sounthern faces. Also let currents at all boundary nodes in the western wall be 0. Voltages of E are uniquely determined from these conditions (see Figure 4.8), call it vector $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_k)$. Notice that all other voltages of eastern face is already determined to be 0.

Since all of the currents at W is set to be 0,

$$\Lambda(W;E)\vec{\alpha} + \Lambda(W;p) = 0$$

Therefore,

$$\vec{\alpha} = -\Lambda(W;E)^{-1}\Lambda(W;p)$$

The current flowing out of node p is

$$\begin{aligned} I &= \Delta V \xi \\ &= (1-0)\xi \\ &= \xi \\ &= \Lambda(p;E)\vec{\alpha} + \lambda_{pp} \\ &= -\Lambda(p;E)\Lambda(W;E)^{-1}\Lambda(W;p) + \lambda_{pp} \end{aligned}$$

Therefore, $\xi = -\Lambda(p;E)\Lambda(W;E)^{-1}\Lambda(W;p) + \lambda_{pp}$. Since node p is arbitary (and the definition of northern face can be rotated), conductances for other spikes can be solved in a similar fashion. Let Λ_{out} be the response matrix for the shell network.

To obtain the response matrix Λ_{in} for $n-1$ by $n-1$ square case rectangular network, one of the two methods can be used:

Spiked case

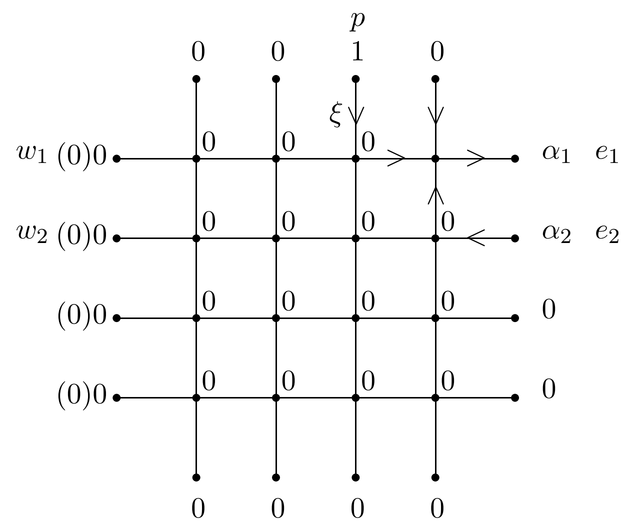


Figure 4.8 *Solving for conductance of a spike*

Spike removal. First identify boundary nodes at each of the four corners in Λ and Λ_{out} and obtain Λ' and Λ'_{out}, respectively. Now use the spike removal technique described in § **??**

$$\Lambda_{in} = \Lambda' \bowtie (-\Lambda'_{out})$$

to obtain Λ_{in}.

Shell stripping formula. Using the Shell stripping formula described in § **??**

If

$$\Lambda_{out} = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix}$$

where nodes are ordered so that boundary nodes of the original graph precedes the boundary nodes of the inner graph, then

$$\Lambda_{in} = [(B^TB)^{-1}(B^T(A - \Lambda)B)(B^TB)^{-1}]^{-1} - C$$

After obtaining Λ_{in} as the response matrix for $n-1$ by $n-1$ square case rectangular network, proceed to section 4.2.3

4.2.3 Square case

This section will describe how to solve for the conductances of the outer square of n by n square case rectangular network and reduce the problem into $n-1$ by $n-1$ spiked case rectangular network.

Conductances of the edges that has one end as the corner boundary node is easy to find. Let corner boundary node be q and the non-corner boundary node of the same edge be p. Impose voltage of -1 at q, and 0 everywhere else. Current at p is

$$\begin{aligned} I_p &= \Delta V \xi \\ &= [0-(-1)]\xi \\ &= \xi \\ &= -\lambda_{pq} \end{aligned}$$

Therefore, $\xi = -\lambda_{pq}$.

Let $\Lambda = (\lambda_{ij})$ be the response matrix for n by n square case rectangular network. Consider neighbor boundary nodes p and q on the northern face (let p and q be the $k+1$th and kth non-corner node counted from the eastern face, respectively). Let ξ be the conductance of the boundary to boundary edge that connects p and q. Let $W = (w_1, w_2, \ldots, w_k)$ be the first k indexes of non-corner boundary nodes on the western face counted from the northern face. Let $E = (e_1, e_2, \ldots, e_k)$ be the first k indexes of non-corner boundary nodes on the eastern face counted from the northern face. Let voltage at node q be 1, and impose 0 voltages at all other boundary nodes in northern, western, and sounthern faces. Also let currents at all non-corner boundary nodes in the western wall be 0. Voltages of E are uniquely determined from these conditions (see Figure 4.9), call it vector $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_k)$. Notice that all other voltages of eastern face is already determined to be 0.

Since all of the currents at W is set to be 0,

$$\Lambda(W;E)\vec{\alpha} + \Lambda(W;q) = 0$$

Therefore,

$$\vec{\alpha} = -\Lambda(W;E)^{-1}\Lambda(W;q)$$

Square case

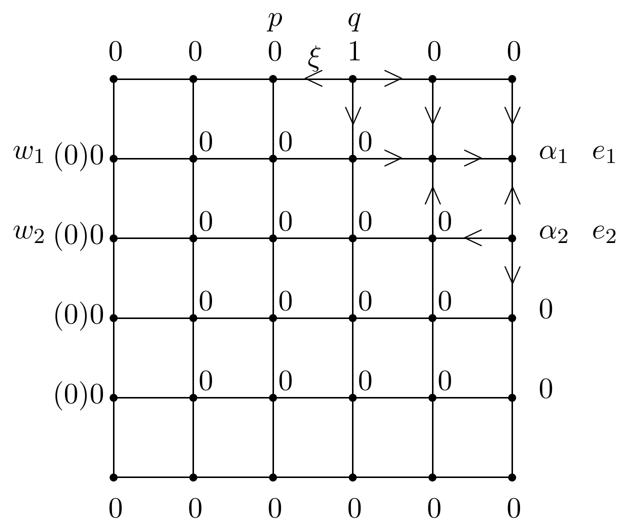


Figure 4.9 *Solving for conductance of a boundary to boundary edge*

The current flowing out of node p is

$$\begin{aligned} I_p &= \Delta V \xi \\ &= (0-1)\xi \\ &= -\xi \\ &= \Lambda(p;E)\vec{\alpha} + \lambda_{pq} \\ &= -\Lambda(p;E)\Lambda(W;E)^{-1}\Lambda(W;q) + \lambda_{pq} \end{aligned}$$

Therefore, $\xi = \Lambda(p;E)\Lambda(W;E)^{-1}\Lambda(W;q) - \lambda_{pq}$. Since nodes p and q are placed arbitary (and the definition of northern face can be rotated), conductances for other non-corner boundary to boundary spikes can be solved in a similar fashion. Let Λ_{out} be the response matrix for the outer square network.

Taking off the edges of outer square is simply the subtraction of response matrices (see § **??**).

$$\Lambda_{in} = \Lambda - \Lambda_{out}$$

Removing isolated boundary nodes in Λ_{in} (the rows and columns of zeros that correspond to the corner boundary nodes) will result in response matrix for the $n-1$ by $n-1$ spiked case rectangular network.

Repeat steps in section 4.2.3 and this section until network is reduced to the base case.

4.2.4 Summary

By induction, this algorithm solves any n by n rectangular network. The algorithm flowchart (see Figure 4.10) summarizes the steps of the algorithm graphically.

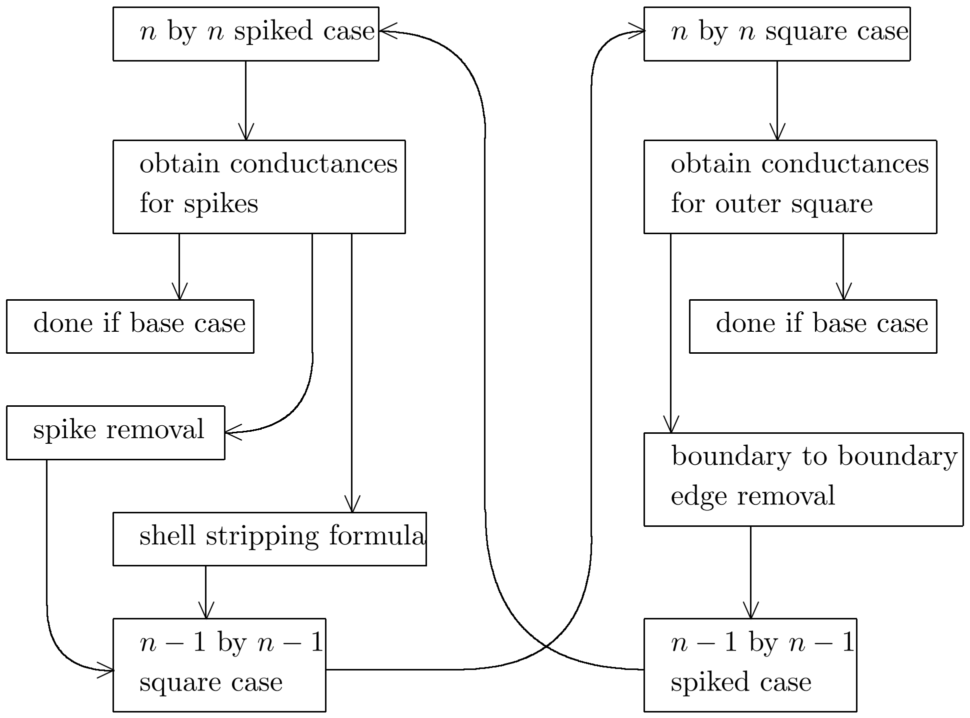

n by n spiked case
n by n square case
obtain conductances for spikes
obtain conductances for outer square
done if base case
done if base case
spike removal
boundary to boundary edge removal
shell stripping formula
$n-1$ by $n-1$ square case
$n-1$ by $n-1$ spiked case

Figure 4.10 *Algorithm flowchart*

4.2.5 Results and Errors

MATLAB was used to calculate the Kirchhoff Matrix for the rectangular networks of various sizes. Like the Tower of Hanoi Networks as the number of boundary nodes increased by one, approximately a decimal of accuracy was lost to rounding errors. The error was approximated by summing the entries of the last column of the difference matrix between the original Kirchhoff Matrix and the calculated Kirchhoff Matrix. The errors are listed as follows:

```
1                  0
2     0.00000000000000
3     0.00000000000000
4     0.00000000000000
5     0.00000000000003
6     0.00000000000007
7     0.00000000000068
8     0.00000000000482
9     0.00000000008866
10    0.00000000063123
11    0.00000003006876
12    0.00000074820295
13    0.00002143300129
14    0.00014559938951
15    0.01737611939126
16    0.08385562267891
17   30.43242863872625
```

The first column is the size of the rectangular network and the second column is the error.

4.3 The Paper Doll Networks

One example where the separation across one boundary node is used is the "paper doll" network. In this network there is a finite string of recoverable networks which look like a set of paper dolls standing in a line holding sticks (see Figure 4.11). The recoverable graph is a pentagon with spikes, and the added spike, or stick, at each interior node where the networks are connected is there to make a connection which will be broken during separation (see § **??**).

The first step in the recovery process is to apply Theorem 3.4 the separation across one interior node theorem to the string of networks in Figure 4.11

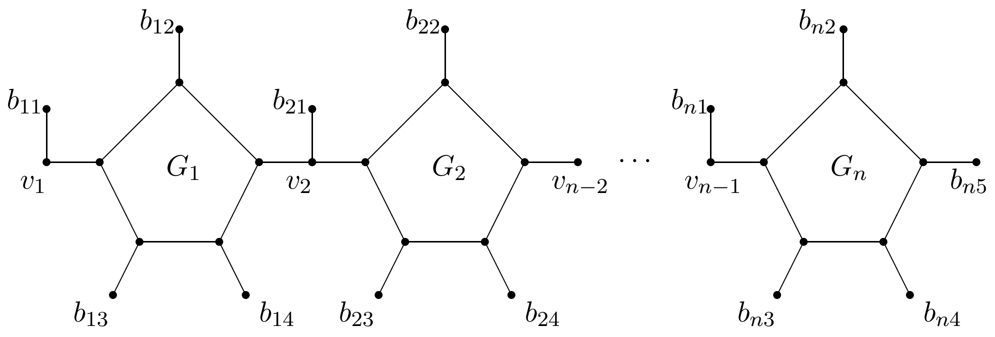


Figure 4.11 *Paper Doll Network*

at the interior nodes v_i for $i = 2$ to $i = n-1$. Note that the boundary nodes of the graph are labeled b_{jk} in Figure 4.11. The resulting networks are as in Figure 4.12. Note that there are now n graphs G_1 to G_n and in each of these graphs the node v_i for $i = 1$ to $i = n-1$ is a boundary node and the nodes b_{jk} are still boundary nodes. Now, it is easy to recover each of these graphs once the extra spikes at the $v_i's$ (the "sticks") are removed.

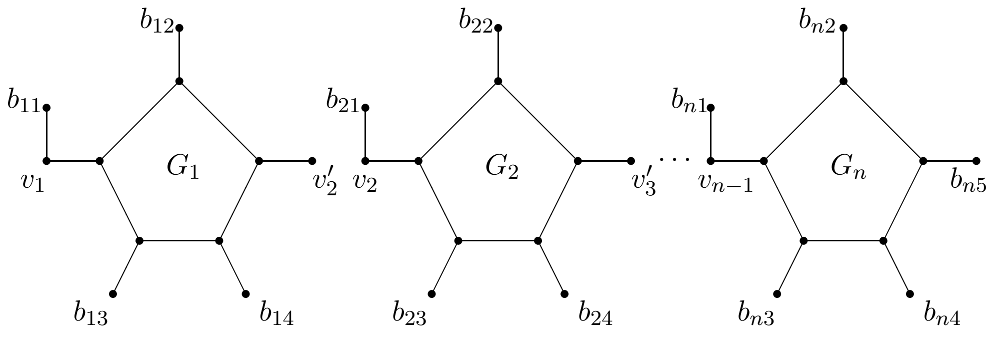


Figure 4.12 *Paper Doll Network After Separation*

Figure 4.13 shows how to remove the extra spike or "stick" from the graph G_i. Before a "stick" can be removed its conductance must be known. Since the v_i are all boundary nodes, and the b_{i1} are also boundary nodes, to find the conductance of the "stick" one just needs to read off the entry from the response matrix, by setting the voltage at node v_i to be zero and the voltage at node b_{i1} to be 1. The next step is the same process described in Chapter 3 called boundary to boundary removal. Once the "stick" is

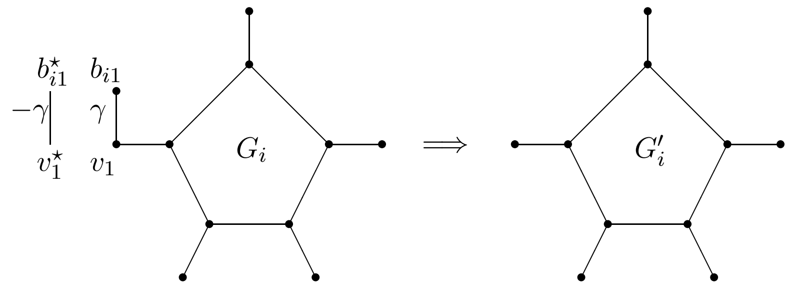


Figure 4.13 *Single Paper Doll Network*

removed the recovery process for the new graphs G_i' are very simple and can also be done with the spike removal process.

To remove the spikes from the graph as in Figure 4.14, the conductances of the spikes must be known. In Figure 4.14 the interior nodes are nodes z_k and the boundary nodes are still b_{ik}. Since the graph G_i' is symmetric it suffices to only show how to find one of the spikes. Imposing the conditions as in Figure 4.14, the unknown voltages α and β can be uniquely determined from the response matrix and the two zero current impositions at nodes b_{i1} and b_{i5}. Thus, the current at node b_{i2} is $I = \alpha\lambda_{23}+\beta\lambda_{24}$. But, the voltage at b_{i2} is 1 and the voltage at z_2 is zero so the voltage drop and the current are known. Therefore, the conductance at the edge between b_{i2} and z_2 can be calculated. Now that the conductance of every spike is known, with the spike removal process the spikes may be removed. If the spikes are removed then the remaining graph is a pentagon with all of its nodes as boundary nodes, from the Shell-to-Kirchhoff Theorem the response matrix of this graph is the Kirchhoff matrix. Thus, the graph is completely recovered.

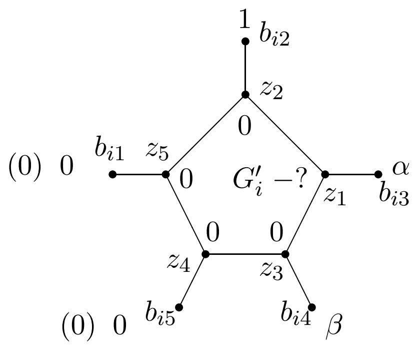


Figure 4.14 *Single Paper Doll Network With Boundary Conditions*

Chapter 5

Conclusion

This investigation has shown that through network amalagamation and separation different recovery processes are possible. The use of amalgamation to provide insight into the recoverability of a network was also revealed. In a similar way, network separation was shown to be directly useful in the recovery of the conductances of a network.

The results of the investigation reveal that it is possible to use separation to recover the conductances of a network. However, this particular investigation was unable to determine whether or not this particular method of recovery is the most accurate for circular planar networks. It is here that more research in the area of accuracy is required.

Further investigation is also needed in the area of the amalgamation operation on networks. From a theoretical stand point it is possible to ask about the operator it self. What kind of an operator is the amalgamation ($\bowtie$)? Does this operator have a direct inverse and when can the inverse be taken? Is the separation of networks the inverse operator? The amalgamation operation also requires more research in the area of its application, or more precisely how can it be used in various other problems in the area of the inverse problem.

***separation along boundary nodes, how determine the range to pick values to not get negative

In the separation of networks section, there are several places where further research is necessary. In particular, when can network separation be used in general? This investigation only covers the very specific cases of shell graph removal, spike removal, and boundary to boundary edge removal. In general, the use of the Shell Stripping Formula requires the inveritablity of the matrix $B^T B$, further research can be made to determine precisely when this matrix is invertible. Also, this investigation has presented the separation across one interior nodes, it may also be possible that separation can be made across several interior nodes.

Finally, there is one example which was left unfinished by this investigation. The recovery of the hexagonal networks is possible and it is in theory possible to use the shell graph removal and spike removal processes to recover the conductances of the graph. This example would be another way to show the usefulness of the algorithms and methods described in this investigation.

-R.K.C. & B.I.M.

Appendix A

A.1 Derivation of Formulae

A.1.1 The Towers of Hanoi Network

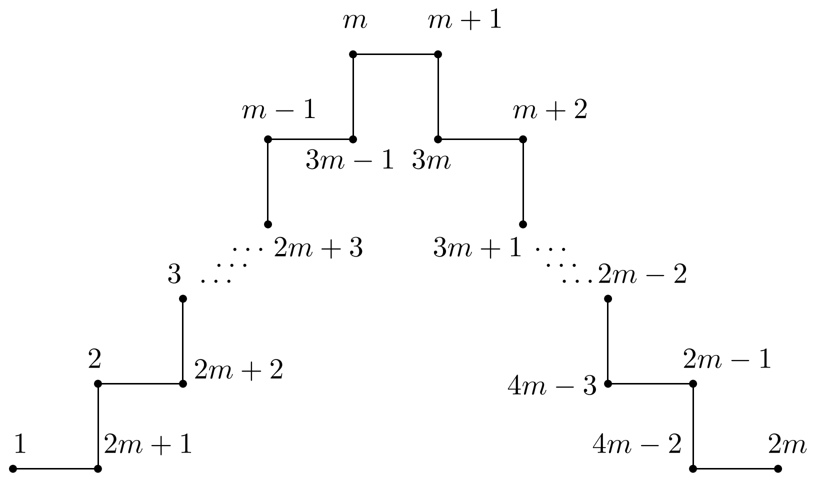


Figure A.1 *n = 2m Boundary Node Case*

The Even Case

Note for the even case, $n = 2m$ for simplicity of the calculation and notation.

The Rightside

For this calculation, currents and voltages of zero are imposed at boundary nodes 1 to m and a voltage of 1 is imposed at boundary node $m+1$. By the calculation done in chapter 4, all of the α_i are known for each $1 \leq i \leq 2m$.

The column vector P is also important here, the P_i entry is the current at node i due to voltages of α_j at every boundary node j. To start the derivation of the formula it is necessary to do several of the base cases to see how the pattern develops into the formula.

$$\begin{aligned}
P_m &= (\alpha_{m+1} - 0)\gamma_{m,m+1} \\
\text{thus,} \quad \gamma_{m,m+1} &= \frac{P_m}{-\alpha_{m+1}} \\
P_{m+1} &= (\alpha_{m+1} - 0)\gamma_{m,m+1} + (\alpha_{m+1} - 0)\gamma_{m+1,3m} \\
\text{thus,} \quad \gamma_{m+1,3m} &= \frac{P_{m+1} + P_m}{\alpha_{m+1}} \\
(\alpha_{m+1} - 0)\gamma_{m+1,3m} &= -(\alpha_{m+2} - 0)\gamma_{m+2,3m} \\
\text{thus,} \quad \gamma_{m+2,3m} &= \frac{P_{m+1} + P_m}{-\alpha_{m+2}} \\
P_{m+2} &= (\alpha_{m+2} - 0)\gamma_{m+2,3m} + (\alpha_{m+2} - 0)\gamma_{m+2,3m+1} \\
\text{thus,} \quad \gamma_{m+2,3m+1} &= \frac{P_{m+2} + P_{m+1} + P_m}{\alpha_{m+2}}
\end{aligned}$$

Following this pattern yields,

$$\begin{aligned}
\gamma_{2m-2,4m-4} &= \frac{\sum_{j=m}^{2m-3} P_j}{-\alpha_{2m-2}} \\
P_{2m-2} &= (\alpha_{2m-2} - 0)\gamma_{2m-2,4m-4} + (\alpha_{2m-2} - 0)\gamma_{2m-2,4m-3} \\
\text{thus,} \quad \gamma_{2m-2,4m-3} &= \frac{\sum_{j=m}^{2m-2} P_j}{\alpha_{2m-2}} \\
(\alpha_{2m-2} - 0)\gamma_{2m-2,4m-3} &= -(\alpha_{m-1} - 0)\gamma_{2m-1,4m-3} \\
\text{thus,} \quad \gamma_{2m-1,4m-3} &= \frac{\sum_{j=m}^{2m-2} P_j}{-\alpha_{2m-1}} \\
P_{2m-1} &= (\alpha_{2m-1} - 0)\gamma_{2m-1,4m-3} + (\alpha_{2m-1} - 0)\gamma_{2m-1,4m-2} \\
\text{thus,} \quad \gamma_{2m-1,4m-2} &= \frac{\sum_{j=m}^{2m-1} P_j}{\alpha_{2m-1}} \\
(\alpha_{2m-1} - 0)\gamma_{2m-1,4m-2} &= -(\alpha_{2m} - 0)\gamma_{2m,4m-2} \\
\text{thus,} \quad \gamma_{2m,4m-2} &= \frac{\sum_{j=m}^{2m-1} P_j}{-\alpha_{2m}}
\end{aligned}$$

Finally, using these patterns the following formulae are derived:

$$\gamma_{i,i+2m-1} = \frac{\sum_{j=m}^{i} P_j}{\alpha_i} \qquad \gamma_{i+1,i+2m-1} = \frac{\sum_{j=m}^{i} P_j}{-\alpha_{i+1}}$$

for $i = m+1$ to $i = 2m-1$.

The Leftside

For this calculation, currents and voltages of zero are imposed at boundary nodes m to $2m$ and a voltage of 1 is imposed at boundary node m. By the calculation done in chapter 4, all of the α_i are known for each $1 \leq i \leq 2m$. The column vector P is also important here, the P_i entry is the current at node i due to voltages of α_j at every boundary node j. To start the derivation of the formula it is necessary to do several of the base cases to see how the pattern develops into the formula.

$$\begin{aligned}
P_1 &= (\alpha_1 - 0)\gamma_{1,2m+1} \\
\text{thus,} \quad \gamma_{1,2m+1} &= \frac{P_1}{\alpha_1} \\
P_1 = (\alpha_1 - 0)\gamma_{1,2m+1} &= -(\alpha_2 - 0)\gamma_{2,2m+1} \\
\text{thus,} \quad \gamma_{2,2m+1} &= \frac{P_1}{-\alpha_2} \\
P_2 &= (\alpha_2 - 0)\gamma_{2,2m+1} + (\alpha_2 - 0)\gamma_{2,2m+2} \\
\text{thus,} \quad \gamma_{2,2m+2} &= \frac{P_1 + P_2}{\alpha_2} \\
(\alpha_2 - 0)\gamma_{2,2m+2} &= -(\alpha_3 - 0)\gamma_{3,2m+2} \\
\text{thus,} \quad \gamma_{3,2m+2} &= \frac{P_1 + P_2}{-\alpha_3}
\end{aligned}$$

Following this pattern yields,

$$\begin{aligned}
\gamma_{m-2,3m-2} &= \frac{\sum_{j=1}^{m-2} P_j}{\alpha_{m-2}} \\
(\alpha_{m-1} - 0)\gamma_{m-1,3m-2} &= -(\alpha_{m-2} - 0)\gamma_{m-2,3m-2} \\
\text{thus,} \quad \gamma_{m-1,3m-2} &= \frac{\sum_{j=1}^{m-2} P_j}{-\alpha_{m-1}} \\
P_{m-1} &= (\alpha_{m-1} - 0)\gamma_{m-1,3m-2} + (\alpha_{m-1} - 0)\gamma_{m-1,3m-1} \\
\text{thus,} \quad \gamma_{m-1,3m-1} &= \frac{\sum_{j=1}^{m-1} P_j}{\alpha_{m-1}} \\
(\alpha_{m-1} - 0)\gamma_{m-1,3m-1} &= -(\alpha_m - 0)\gamma_{m,3m-1} \\
\text{thus,} \quad \gamma_{m,3m-1} &= \frac{\sum_{j=1}^{m-1} P_j}{-\alpha_m}
\end{aligned}$$

Finally, using these patterns the following formulae are derived:

$$\gamma_{i,i+2m} = \frac{\sum_{j=1}^{i} P_j}{\alpha_i} \qquad \gamma_{i+1,i+2m} = \frac{\sum_{j=1}^{i} P_j}{-\alpha_{i+1}}$$

for $i = 1$ to $i = m$.

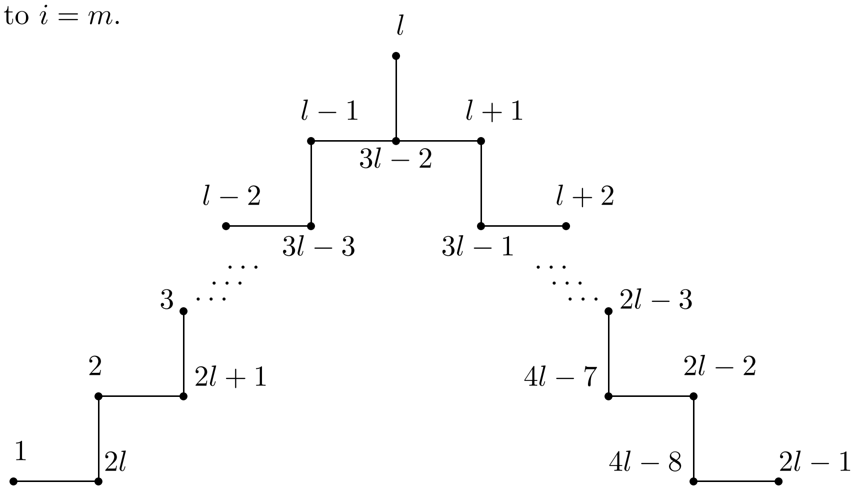


Figure A.2 *n = 2l - 1 Boundary Node Case*

The Odd Case

Note for the odd case, $n = 2l - 1$ for simplicity of the calculation and the notation.

The Rightside
For this calculation, currents and voltages of zero are imposed at boundary nodes 1 to $l-1$ and a voltage of 1 is imposed at boundary node l. By the calculation done in chapter 4, all of the α_i are known for each $1 \leq i \leq 2l-1$. The column vector P is also important here, the P_i entry is the current at node i due to voltages of α_j at every boundary node j. To start the derivation of the formula it is necessary to do several of the base cases to see how the pattern develops into the formula.

$$\begin{aligned} P_l &= (\alpha_l - 0)\gamma_{l,3l-2} \\ \text{thus,} \quad \gamma_{l,3l-2} &= \frac{P_l}{\alpha_l} \end{aligned}$$

$$\begin{aligned}
P_l = (\alpha_l - 0)\gamma_{l,3l-2} &= -(\alpha_{l+1} - 0)\gamma_{l+1,3l-2} \\
\text{thus,} \quad \gamma_{l+1,3l-2} &= \frac{P_l}{-\alpha_{l+1}} \\
P_{l+1} &= (\alpha_{l+1} - 0)\gamma_{l+1,3l-2} + (\alpha_{l+1} - 0)\gamma_{l+1,3l-1} \\
\text{thus,} \quad \gamma_{l+1,3l-1} &= \frac{P_l + P_{l+1}}{\alpha_{l+1}} \\
(\alpha_{l+1} - 0)\gamma_{l+1,3l-1} &= -(\alpha_{l+2} - 0)\gamma_{l+2,3l-1} \\
\text{thus,} \quad \gamma_{l+2,3l-1} &= \frac{P_l + P_{l+1}}{-\alpha_{l+2}}
\end{aligned}$$

Following this pattern yields,

$$\begin{aligned}
\gamma_{2l-3,4l-5} &= \frac{\sum_{j=l}^{2l-3} P_j}{\alpha_{2l-3}} \\
(\alpha_{2l-3} - 0)\gamma_{2l-3,4l-4} &= -(\alpha_{2l-2} - 0)\gamma_{2l-2,4l-5} \\
\text{thus,} \quad \gamma_{2l-2,4l-5} &= \frac{\sum_{j=l}^{2l-3} P_j}{-\alpha_{2l-2}} \\
P_{2l-2} &= (\alpha_{2l-2} - 0)\gamma_{2l-2,4l-5} + (\alpha_{2l-2} - 0)\gamma_{2l-2,4l-4} \\
\text{thus,} \quad \gamma_{2l-2,4l-4} &= \frac{\sum_{j=l}^{2l-2} P_j}{\alpha_{2l-2}} \\
(\alpha_{2l-2} - 0)\gamma_{2l-2,4l-4} &= -(\alpha_{2l-1} - 0)\gamma_{2l-1,4l-4} \\
\text{thus,} \quad \gamma_{2l-1,4l-4} &= \frac{\sum_{j=l}^{2l-2} P_j}{-\alpha_{2l-1}}
\end{aligned}$$

Finally, using these patterns the following formulae are derived:

$$\gamma_{i,i+2l-2} = \frac{\sum_{j=l}^{i} P_j}{\alpha_i} \qquad \gamma_{i+1,i+2l-2} = \frac{\sum_{j=l}^{i} P_j}{-\alpha_{i+1}}$$

for $i = l$ to $i = 2l - 2$.

The Leftside

For this calculation, currents and voltages of zero are imposed at boundary nodes $l+1$ to $2l-1$ and a voltage of 1 is imposed at boundary node l. By the calculation done in chapter 4, all of the α_i are known for each $1 \le i \le 2l-1$. The column vector P is also important here, the P_i entry is the current at node i due to voltages of α_j at every boundary node j. To start the derivation of the formula it is necessary to do several of the base cases to see how the pattern develops into the formula.

$$\begin{aligned}
P_1 &= (\alpha_1 - 0)\gamma_{1,2l} \\
\text{thus,} \quad \gamma_{1,2l} &= \frac{P_1}{\alpha_1} \\
P_1 = (\alpha_1 - 0)\gamma_{1,2l} &= -(\alpha_2 - 0)\gamma_{2,2l} \\
\text{thus,} \quad \gamma_{2,2l} &= \frac{P_1}{-\alpha_2} \\
P_2 &= (\alpha_2 - 0)\gamma_{2,2l} + (\alpha_2 - 0)\gamma_{2,2l+1} \\
\text{thus,} \quad \gamma_{2,2l+1} &= \frac{P_1 + P_2}{\alpha_2} \\
(\alpha_3 - 0)\gamma_{3,2l+1} &= -(\alpha_2 - 0)\gamma_{2,2l+1} \\
\text{thus,} \quad \gamma_{3,2l+1} &= \frac{P_1 + P_2}{-\alpha_3}
\end{aligned}$$

Following this pattern yields,

$$\begin{aligned}
\gamma_{l-2,3l-3} &= \frac{\sum_{j=1}^{l-2} P_j}{\alpha_{l-2}} \\
(\alpha_{l-2} - 0)\gamma_{l-2,3l-3} &= -(\alpha_{l-1} - 0)\gamma_{l-1,3l-3} \\
\text{thus,} \quad \gamma_{l-1,3l-3} &= \frac{\sum_{j=1}^{l-2} P_j}{-\alpha_{l-1}} \\
P_{l-1} &= (\alpha_{l-1} - 0)\gamma_{l-1,3l-3} + (\alpha_{l-1} - 0)\gamma_{l-1,3l-2} \\
\text{thus,} \quad \gamma_{l-1,3l-2} &= \frac{\sum_{j=1}^{l-1} P_j}{\alpha_{l-1}} \\
(\alpha_{l-1} - 0)\gamma_{l-1,3l-2} &= -(\alpha_l - 0)\gamma_{l,3l-2} \\
\text{thus,} \quad \gamma_{l,3l-2} &= \frac{\sum_{j=1}^{l-1} P_j}{-\alpha_l}
\end{aligned}$$

Finally, using these patterns the following formulae are derived:

$$\gamma_{i,i+2l-1} = \frac{\sum_{j=1}^{i} P_j}{\alpha_i} \qquad \gamma_{i+1,i+2l-1} = \frac{\sum_{j=1}^{i} P_j}{-\alpha_{i+1}}$$

for $i = 1$ to $i = l - 1$.

A.1.2 The Rectangular Network

A.2 Matlab Code

A.2.1 Tower Code

Shell Code: Even Case Rightside

```
function shell=shelleright(L) %RIGHTSIDE SOLUTION EVEN

%Input: Response For Even Tower
%Output: Kirchhoff For Shell Graph

n=size(L,1);
A = zeros((n-2)/2, (n-2)/2);
S = zeros( (n-2)/2, 1);
X = zeros( 1, (n-2)/2);
alpha = zeros(n, 1);
gamma = zeros(2*n-2, 2*n -2);
P = zeros(1,n);

for i=1:n
   if 1 <= i & i <= n/2
      alpha(i) = 0;
   elseif i == (n+2)/2
      alpha(i) = 1;
   elseif ((n+4)/2) <= i & i <= n
      alpha(i) = 0;
   end
end

for k=1:(n-2)/2
   for j=(n+4)/2:n
      A(k, j - (n+2)/2) = L(k, j);
   end
end

for i=1:(n-2)/2
   S(i) = -L(i, (n+2)/2);
end
```

```
X = A\S;

for i=1:(n-2)/2
   alpha(i + (n+2)/2) = X(i);
end

P = L*alpha;

gamma(n/2, (n+2)/2) =  -P(n/2) ;

for i=(n+2)/2:n-1
   gamma(i, i + n-1) =  (sum(P(n/2:i)))/alpha(i) ;
end

for i=(n+2)/2:n-1
   gamma(i +1, i + n - 1 ) =  (sum(P(n/2:i)))/-alpha(i+1);
end

shell = -gamma;
```

Shell Code: Even Case Leftside

```
function shell=shelleleft(L) %LEFTSIDE SOLUTION EVEN

%Input: Response For Even Tower
%Output: Kirchhoff For Shell Graph

n=size(L,1);
A = zeros((n-2)/2, (n-2)/2);
S = zeros( (n-2)/2, 1);
alpha = zeros(n, 1);
gamma = zeros(2*n-2, 2*n -2);
P = zeros(1,n);

for i=1:n
   if 1 <= i & i <= (n-2)/2
      alpha(i) = 0;
   elseif i == n/2
      alpha(i) = 1;
   elseif ((n+2)/2) <= i & i <= n
      alpha(i) = 0;
   end
end

for k=(n+4)/2:n
   for j=1:(n-2)/2
      A(k - (n+2)/2, j) = L(k, j);
   end
end

for i=1:(n-2)/2
   S(i) = -L(i + (n+2)/2, n/2);
end

X = A\S;

for i=1:(n-2)/2
   alpha(i) = X(i);
end

P = L*alpha;

for i=1:(n-2)/2
   gamma(i, i + n) =   ( sum(P(1:i)) ) /alpha(i) ;
end

for i=1:(n-2)/2
   gamma(i + 1, i +n) =  ( sum(P(1:i) ) ) /-alpha(i+1) ;
end

shell = -gamma;
```

Shell Code: Odd Case Rightside

```
function shell=shelloright(L) %RIGHTSIDE SOLUTION ODD

%Input: Response For Odd Tower
%Output: Kirchhoff For Shell Graph
```

```
n=size(L,1);
A = zeros((n-1)/2, (n-1)/2);
S = zeros( (n-1)/2, 1);
alpha = zeros(n, 1);
gamma = zeros(2*n-2, 2*n -2);
P = zeros(1,n);

for i=1:n
   if 1 <= i & i <= (n-1)/2
      alpha(i) = 0;
   elseif i == ((n+1)/2)
      alpha(i) = 1;
   elseif ((n+3)/2) <= i & i <= n
      alpha(i) = 0;
   end
end

for k=1:(n-1)/2
   for j=(n+3)/2:n
      A(k, j - (n+1)/2) = L(k, j);
   end
end

for i=1:(n-1)/2
   S(i) = -L(i, (n+1)/2);
end

X = A\S;

for i=1:(n-1)/2
   alpha(i + (n+1)/2) = X(i);
end

P = L*alpha;

for i=(n+1)/2:n-1
   gamma(i, n+i - 1) = sum(P((n+1)/2:i)) /alpha(i);
end

for i=(n+1)/2:n-1
```

```
   gamma(i+1,i+n-1) = sum(P((n+1)/2:i)) /-alpha(i+1);
end

shell = -gamma;
```

Shell Code: Odd Case Leftside

```
function shell=shelloleft(L) %LEFTSIDE SOLUTION ODD

%Input: Response For Odd Tower
%Output: Kirchhoff For Shell Graph

n=size(L,1);
A = zeros((n-1)/2, (n-1)/2);
S = zeros( (n-1)/2, 1);
alpha = zeros(n, 1);
gamma = zeros(2*n-2, 2*n -2);
P = zeros(1,n);


for i=1:n
   if 1 <= i & i <= (n-1)/2
      alpha(i) = 0;
   elseif i == ((n+1)/2)
      alpha(i) = 1;
   elseif ((n+3)/2) <= i & i <= n
      alpha(i) = 0;
   end
end

for k=(n+3)/2:n
   for j=1:(n-1)/2
      A(k - (n+1)/2, j) = L(k, j);
   end
end

for i=1:(n-1)/2
   S(i) = -L((i + (n+1)/2), (n+1)/2);
end
```

```
X = A\S;

for i=1:(n-1)/2
   alpha(i) = X(i);
end

P = L*alpha;

for i=1:(n-1)/2
   gamma(i, n+i) = sum(P(1:i)) /(alpha(i)) ;
end

for i=1:(n-1)/2
   gamma(i+1,n+i) = sum(P(1:i)) /-(alpha(i+1));
end

gamma((n+1)/2, (3*n - 1)/2) = 0;

shell = -gamma;
```

Recursive Code

```
function LL=complete(L)

% input:  Upper right trianguler response matrix
% output: Complete response matrix

n=size(L,1);

LL=zeros(n,n);
LL=L;
for i=1:n
  for j=1:i
    LL(i,j)=0;
  end
end
LL=LL+LL';

for i=1:n
  sum=0;
  for j=1:n
    sum=sum+LL(i,j);
  end
  LL(i,i)=-sum;
end


function L=forward(K,n)

% input:  K = Kirchhoff matrix
%         n = Number of boundary nodes
% output: L = Response matrix

m=size(K,1);

A=K(1:n,1:n);
B=K(1:n,n+1:m);
C=K(n+1:m,n+1:m);

L=A-B*inv(C)*B';
```

```
function K=onetower(n)

K=randtower(n);

for i=1:length(K)
   for j=i:length(K)
      if K(i,j)~=0
         K(i,j)=-1;
      end
   end
end

K=complete(K);

function K=randtower(n)

% input:  n = Size of the tower to be generated
% output: K = Kirchhoff matrix for the generated tower

% m = Size of the Kirchhoff matrix
if mod(n,2)==0
  m=(n*n/2+n)/2;
else
  m=(n-1)*(n-1)/4+n;
end

K=zeros(m,m);

if n>=4
  if mod(n,2)==0
    % Even towers
    for i=1:n/2-1
      K(i:i+1,i+n)=-10*rand(2,1);
    end
    K(n/2,n/2+1)=-10*rand(1,1);
    for i=n/2+1:n-1
      K(i:i+1,i+n-1)=-10*rand(2,1);
    end
  else
    % Odd towers
    for i=1:(n-1)/2
      K(i:i+1,i+n)=-10*rand(2,1);
    end
    K((n+3)/2,(3*n-1)/2)=-10*rand(1,1);
    for i=(n+3)/2:n-1
      K(i:i+1,i+n-1)=-10*rand(2,1);
    end
  end
  % Recursive step
  K(n+1:m,n+1:m)=randtower(n-2);
elseif n==2
  % Base case for even towers
  K(1,2)=-10*rand(1,1);
elseif n==3
  % Base case for odd towers
  K(1:3,4)=-10*rand(3,1);
end

K=complete(K);

function K=tower(L)

% input:  L = Response matrix for a tower
% output: K = The original Kirchhoff matrix

n=size(L,1);

% m = Size of the Kirchhoff matrix
if mod(n,2)==0
  m=(n*n/2+n)/2;
else
  m=(n-1)*(n-1)/4+n;
end

K=zeros(m,m);

% Find Lout
```

```
if mod(n,2)==1
  % Odd case
  Lout=complete(shelloright(L)+shelloleft(L));
else
  % Even case
  Lout=complete(shelleright(L)+shelleleft(L));
end

K(1:2*n-2,1:2*n-2)=Lout;

% Recursive step.

if n>=4
  A=Lout(1:n,1:n);
  C=Lout(1:n,n+1:2*n-2);
  D=Lout(n+1:2*n-2,n+1:2*n-2);
  B=inv(C'*C);

  Lin=inv(B*C'*(A-L)*C*B)-D;

  K(n+1:m,n+1:m)=tower(Lin);
end

K=complete(K);

function [K,L,KK,diff]=towertesttwo(n)

K=onetower(n);
L=forward(K,n);
KK=tower(L);
diff=K-KK;
sum(abs(diff(length(diff),1:length(diff)-1)))

% diff(length(diff),length(diff))
```

A.2.2 Rectangular Code

Spike Code

```
function K=spike(L)

% Input:  L = Response matrix for the "spiked" rectangular network
% Output: K = Kirchhoff matrix for the same network

% n = Size of the rectangular network
n = length(L)/4;

% m = Size of the Kirchhoff matrix
m = n*(4+n);

K=zeros(m,m);

for  i = 1:4*n
   % find sect, the section boundary node i belongs to
   sect = ceil(2*i/n);
   % find the following case by case:
   % len, the size of the condition that must be set
   % zerovec, vector of the indexes of the nodes to be set to zero current
   % unknvec, vector of the indexes of the nodes with unknown voltages
   switch sect
   case 1
      len = i;
      zerovec = n+1:n+len;
      unknvec = 4*n:-1:4*n-len+1;
   case 2
      len = n+1-i;
      zerovec = 4*n:-1:4*n-len+1;
      unknvec = n+1:n+len;
   case 3
      len = i-n;
      zerovec = 2*n+1:2*n+len;
      unknvec = n:-1:n-len+1;
   case 4
      len = 2*n+1-i;
      zerovec = n:-1:n-len+1;
      unknvec = 2*n+1:2*n+len;
   case 5
```

```
      len = i-2*n;
      zerovec = 3*n+1:3*n+len;
      unknvec = 2*n:-1:2*n-len+1;
   case 6
      len = 3*n+1-i;
      zerovec = 2*n:-1:2*n-len+1;
      unknvec = 3*n+1:3*n+len;
   case 7
      len = i-3*n;
      zerovec = 1:len;
      unknvec = 3*n:-1:3*n-len+1;
   case 8
      len = 4*n+1-i;
      zerovec = 3*n:-1:3*n-len+1;
      unknvec = 1:len;
   end
   % find the conductance and store it in Kirchhoff matrix
   K(i,in(n,i))=L(i,unknvec)*(L(zerovec,unknvec)\L(zerovec,i))-L(i,i);
end

K=complete(K);

% Lout = Response matrix for the outer shell
% l    = Size of Lout
l = in(n,4*n-1);
Lout = K(1:l,1:l);

% Recursive step

if n > 1
  A=Lout(1:4*n,1:4*n);
  C=Lout(1:4*n,4*n+1:l);
  D=Lout(4*n+1:l,4*n+1:l);
  B=inv(C'*C);

  Lin=inv(B*C'*(A-L)*C*B)-D;

  K(4*n+1:m,4*n+1:m)=square(Lin);
end
```

```
K=complete(K);


function K=newspike(L)

% Input:  L = Response matrix for the "spiked" rectangular network
% Output: K = Kirchhoff matrix for the same network

% n = Size of the rectangular network
n = length(L)/4;

% m = Size of the Kirchhoff matrix
m = n*(4+n);

K=zeros(m,m);

for  i = 1:4*n
   % find sect, the section boundary node i belongs to
   sect = ceil(2*i/n);
   % find the following case by case:
   % len, the size of the condition that must be set
   % zerovec, vector of the indexes of the nodes to be set to zero current
   % unknvec, vector of the indexes of the nodes with unknown voltages
   switch sect
   case 1
      len = i;
      zerovec = n+1:n+len;
      unknvec = 4*n:-1:4*n-len+1;
   case 2
      len = n+1-i;
      zerovec = 4*n:-1:4*n-len+1;
      unknvec = n+1:n+len;
   case 3
      len = i-n;
      zerovec = 2*n+1:2*n+len;
      unknvec = n:-1:n-len+1;
   case 4
      len = 2*n+1-i;
      zerovec = n:-1:n-len+1;
      unknvec = 2*n+1:2*n+len;
   case 5
      len = i-2*n;
      zerovec = 3*n+1:3*n+len;
      unknvec = 2*n:-1:2*n-len+1;
   case 6
      len = 3*n+1-i;
      zerovec = 2*n:-1:2*n-len+1;
      unknvec = 3*n+1:3*n+len;
   case 7
      len = i-3*n;
      zerovec = 1:len;
      unknvec = 3*n:-1:3*n-len+1;
   case 8
      len = 4*n+1-i;
      zerovec = 3*n:-1:3*n-len+1;
      unknvec = 1:len;
   end
   % find the conductance and store it in Kirchhoff matrix
   K(i,in(n,i))=L(i,unknvec)*(L(zerovec,unknvec)\L(zerovec,i))-L(i,i);
end

K=complete(K);

% Lout = Response matrix for the outer shell
% l    = Size of Lout
l = in(n,4*n-1);
Lout = K(1:l,1:l);

% Recursive step

if n > 1

%  Old code :)
%  A=Lout(1:4*n,1:4*n);
%  C=Lout(1:4*n,4*n+1:l);
%  D=Lout(4*n+1:l,4*n+1:l);
%  B=inv(C'*C)
%  Lin=inv(B*C'*(A-L)*C*B)-D;

   Lprime = Lout;
   Lprime(n,in(n,n))     = 0;
   Lprime(2*n,in(n,2*n)) = 0;
   Lprime(3*n,in(n,3*n)) = 0;
   Lprime(4*n,in(n,4*n)) = 0;
   Lprime = complete(Lprime);

   Lstar = [L,zeros(4*n,4*n-4);zeros(4*n-4,8*n-4)] - Lprime;
   Lstar = spforward(Lstar,[1:n-1,n+1:2*n-1,2*n+1:3*n-1,3*n+1:4*n-1]);
   Lstar(1,4+n)   = 0;
   Lstar(2,3+2*n) = 0;
   Lstar(3,2+3*n) = 0;
   Lstar(4,5)     = 0;
   Lstar = complete(Lstar);

   Lin = Lstar(5:4*n,5:4*n);

   K(4*n+1:m,4*n+1:m)=newsquare(Lin);
end

K=complete(K);


function K=onespike(n)

% Input:  n = Size of the spike matrix to be generated
% Output: K = Kirchhoff matrix for spike with all conductance 1

% m = Size of the Kirchhoff matrix
m = n*(n+4);

K = zeros(m,m);

% Add conductors of outer spike
for i = 1:4*n
   K(i,in(n,i))=-1;
end

% Recursive case
if n > 1
   K(4*n+1:m,4*n+1:m)=onesquare(n-1);
```

```
end

K=complete(K);
```

Square Code

```
function K=square(L)

% Input:  L = Response matrix for the "square" rectangular network
% Output: K = Kirchhoff matrix for the same network

% n = Size of the rectangular network
n = length(L)/4;

% m = Size of the Kirchhoff matrix
m = (n+1)*(n+1);
K=zeros(m,m);
Lout = zeros(4*n, 4*n);

if n==1
   Lout(1, 2) = L(2, 1);
   Lout(1, 4) = L(4, 1);
   Lout(2, 3) = L(2, 3);
   Lout(3, 4) = L(4, 3);
elseif n ~= 1

for  i = 2:4*n-1
   if mod(i,n) > 1
   % find sect, the section boundary node i belongs to
   sect = ceil(2*(i-1)/n);
   % find the following case by case:
   % len, the size of the condition that must be set
   % zerovec, vector of the indexes of the nodes to be set to zero current
   % unknvec, vector of the indexes of the nodes with unknown voltages
   switch sect
   case 1
      len = i-1;
      zerovec = n+2:n+1+len;
      unknvec = 4*n:-1:4*n-len+1;
      Lout(i,i+1)=-L(i+1,unknvec)*(L(zerovec,unknvec)\L(zerovec,i))+L(i,i+1);
   case 2
      len = n-i;
      zerovec = 4*n:-1:4*n-len+1;
      unknvec = n+2:n+1+len;
      Lout(i,i+1)=-L(i,unknvec)*(L(zerovec,unknvec)\L(zerovec,i+1))+L(i,i+1);
   case 3
      len = i-n-1;
      zerovec = 2*n+2:2*n+1+len;
      unknvec = n:-1:n-len+1;
      Lout(i,i+1)=-L(i+1,unknvec)*(L(zerovec,unknvec)\L(zerovec,i))+L(i,i+1);
   case 4
      len = 2*n-i;
      zerovec = n:-1:n-len+1;
      unknvec = 2*n+2:2*n+1+len;
      Lout(i,i+1)=-L(i,unknvec)*(L(zerovec,unknvec)\L(zerovec,i+1))+L(i,i+1);
   case 5
      len = i-2*n-1;
      zerovec = 3*n+2:3*n+1+len;
      unknvec = 2*n:-1:2*n-len+1;
      Lout(i,i+1)=-L(i+1,unknvec)*(L(zerovec,unknvec)\L(zerovec,i))+L(i,i+1);
   case 6
      len = 3*n-i;
      zerovec = 2*n:-1:2*n-len+1;
      unknvec = 3*n+2:3*n+1+len;
      Lout(i,i+1)=-L(i,unknvec)*(L(zerovec,unknvec)\L(zerovec,i+1))+L(i,i+1);
   case 7
      len = i-3*n-1;
      zerovec = 2:1+len;
      unknvec = 3*n:-1:3*n-len+1;
      Lout(i,i+1)=-L(i+1,unknvec)*(L(zerovec,unknvec)\L(zerovec,i))+L(i,i+1);
   case 8
      len = 4*n-i;
      zerovec = 3*n:-1:3*n-len+1;
      unknvec = 2:1+len;
      Lout(i,i+1)=-L(i,unknvec)*(L(zerovec,unknvec)\L(zerovec,i+1))+L(i,i+1);
   end
   end
end

% Read off the corners here
Lout(1, 2)         = L(2,1);
Lout(1, 4*n)       = L(4*n, 1);
Lout(n, n+1)       = L(n, n+1);
Lout(n+1, n+2)     = L(n+2, n+1);
Lout(2*n, 2*n+1)   = L(2*n, 2*n+1);
Lout(2*n+1, 2*n+2) = L(2*n+2, 2*n+1);
Lout(3*n, 3*n+1)   = L(3*n, 3*n+1);
Lout(3*n+1, 3*n+2) = L(3*n+2, 3*n+1);

end

Lout = complete(Lout);
K(1:4*n,1:4*n) = Lout;

if n > 1
   % Rip off the conductors around the edge
   temp = L-Lout;
   % Take out the extra zero rows and cols
   a = [2:n,n+2:2*n,2*n+2:3*n,3*n+2:4*n];
   Lin = temp(a,a);
   % Obtain the Kirchhoff matrix of inside and insert the zeros back
   m = length(K);
   Kin = zeros(m,m);
   Kin(2:m-3,2:m-3) = spike(Lin);
   Kin(:,n+2:m)     = Kin(:,n+1:m-1);
   Kin(:,n+1)       = zeros(m,1);
   Kin(:,2*n+2:m)   = Kin(:,2*n+1:m-1);
   Kin(:,2*n+1)     = zeros(m,1);
   Kin(:,3*n+2:m)   = Kin(:,3*n+1:m-1);
   Kin(:,3*n+1)     = zeros(m,1);
   Kin(n+2:m,:)     = Kin(n+1:m-1,:);
   Kin(n+1,:)       = zeros(1,m);
   Kin(2*n+2:m,:)   = Kin(2*n+1:m-1,:);
   Kin(2*n+1,:)     = zeros(1,m);
   Kin(3*n+2:m,:)   = Kin(3*n+1:m-1,:);
   Kin(3*n+1,:)     = zeros(1,m);

   K = K+Kin;
elseif n==1
   K = Lout;
end

K=complete(K);
```

```
function K=newsquare(L)

% Input:  L = Response matrix for the "square" rectangular network
% Output: K = Kirchhoff matrix for the same network

% n = Size of the rectangular network
n = length(L)/4;

% m = Size of the Kirchhoff matrix
m = (n+1)*(n+1);
K=zeros(m,m);
Lout = zeros(4*n, 4*n);

if n==1
   Lout(1, 2) = L(2, 1);
   Lout(1, 4) = L(4, 1);
   Lout(2, 3) = L(2, 3);
   Lout(3, 4) = L(4, 3);
elseif n ~= 1

for  i = 2:4*n-1
   if mod(i,n) > 1
   % find sect, the section boundary node i belongs to
   sect = ceil(2*(i-1)/n);
   % find the following case by case:
   % len, the size of the condition that must be set
   % zerovec, vector of the indexes of the nodes to be set to zero current
   % unknvec, vector of the indexes of the nodes with unknown voltages
   switch sect
   case 1
      len = i-1;
      zerovec = n+2:n+1+len;
      unknvec = 4*n:-1:4*n-len+1;
      Lout(i,i+1)=-L(i+1,unknvec)*(L(zerovec,unknvec)\L(zerovec,i))+L(i,i+1);
   case 2
      len = n-i;
      zerovec = 4*n:-1:4*n-len+1;
      unknvec = n+2:n+1+len;
      Lout(i,i+1)=-L(i,unknvec)*(L(zerovec,unknvec)\L(zerovec,i+1))+L(i,i+1);
   case 3
      len = i-n-1;
      zerovec = 2*n+2:2*n+1+len;
      unknvec = n:-1:n-len+1;
      Lout(i,i+1)=-L(i+1,unknvec)*(L(zerovec,unknvec)\L(zerovec,i))+L(i,i+1);
   case 4
      len = 2*n-i;
      zerovec = n:-1:n-len+1;
      unknvec = 2*n+2:2*n+1+len;
      Lout(i,i+1)=-L(i,unknvec)*(L(zerovec,unknvec)\L(zerovec,i+1))+L(i,i+1);
   case 5
      len = i-2*n-1;
      zerovec = 3*n+2:3*n+1+len;
      unknvec = 2*n:-1:2*n-len+1;
      Lout(i,i+1)=-L(i+1,unknvec)*(L(zerovec,unknvec)\L(zerovec,i))+L(i,i+1);
   case 6
      len = 3*n-i;
      zerovec = 2*n:-1:2*n-len+1;
      unknvec = 3*n+2:3*n+1+len;
      Lout(i,i+1)=-L(i,unknvec)*(L(zerovec,unknvec)\L(zerovec,i+1))+L(i,i+1);
   case 7
      len = i-3*n-1;
      zerovec = 2:1+len;
      unknvec = 3*n:-1:3*n-len+1;
      Lout(i,i+1)=-L(i+1,unknvec)*(L(zerovec,unknvec)\L(zerovec,i))+L(i,i+1);
   case 8
      len = 4*n-i;
      zerovec = 3*n:-1:3*n-len+1;
      unknvec = 2:1+len;
      Lout(i,i+1)=-L(i,unknvec)*(L(zerovec,unknvec)\L(zerovec,i+1))+L(i,i+1);
   end
   end
end

% Read off the corners here
Lout(1, 2)         = L(2,1);
Lout(1, 4*n)       = L(4*n, 1);
Lout(n, n+1)       = L(n, n+1);
Lout(n+1, n+2)     = L(n+2, n+1);
Lout(2*n, 2*n+1)   = L(2*n, 2*n+1);
Lout(2*n+1, 2*n+2) = L(2*n+2, 2*n+1);
Lout(3*n, 3*n+1)   = L(3*n, 3*n+1);
Lout(3*n+1, 3*n+2) = L(3*n+2, 3*n+1);

end

Lout = complete(Lout);
K(1:4*n,1:4*n) = Lout;

if n > 1
   % Rip off the conductors around the edge
   temp = L-Lout;
   % Take out the extra zero rows and cols
   a = [2:n,n+2:2*n,2*n+2:3*n,3*n+2:4*n];
   Lin = temp(a,a);
   % Obtain the Kirchhoff matrix of inside and insert the zeros back
   m = length(K);
   Kin = zeros(m,m);
   Kin(2:m-3,2:m-3) = newspike(Lin);
   Kin(:,n+2:m)     = Kin(:,n+1:m-1);
   Kin(:,n+1)       = zeros(m,1);
   Kin(:,2*n+2:m)   = Kin(:,2*n+1:m-1);
   Kin(:,2*n+1)     = zeros(m,1);
   Kin(:,3*n+2:m)   = Kin(:,3*n+1:m-1);
   Kin(:,3*n+1)     = zeros(m,1);
   Kin(n+2:m,:)     = Kin(n+1:m-1,:);
   Kin(n+1,:)       = zeros(1,m);
   Kin(2*n+2:m,:)   = Kin(2*n+1:m-1,:);
   Kin(2*n+1,:)     = zeros(1,m);
   Kin(3*n+2:m,:)   = Kin(3*n+1:m-1,:);
   Kin(3*n+1,:)     = zeros(1,m);

   K = K+Kin;
elseif n==1
   K = Lout;
end

K=complete(K);
```

```
function K=onesquare(n)

% Input:  n = Size of the square matrix to be generated
% Output: K = Krichhoff matrix for square with all conductance 1

% m = Size of the Kirchhoff matrix
m = (n+1)*(n+1);

K=zeros(m,m);
temp=K;

% Add conductors of outer square
for i = 1:4*n-1
   K(i,i+1)=-1;
end
K(1,4*n)=-1;

% Recursive case
if n > 1
   temp(2:m-3,2:m-3)=onespike(n-1);
   temp(:,n+2:m)=temp(:,n+1:m-1);
   temp(:,n+1)=zeros(m,1);
   temp(:,2*n+2:m)=temp(:,2*n+1:m-1);
   temp(:,2*n+1)=zeros(m,1);
   temp(:,3*n+2:m)=temp(:,3*n+1:m-1);
   temp(:,3*n+1)=zeros(m,1);
   temp(n+2:m,:)=temp(n+1:m-1,:);
   temp(n+1,:)=zeros(1,m);
   temp(2*n+2:m,:)=temp(2*n+1:m-1,:);
   temp(2*n+1,:)=zeros(1,m);
   temp(3*n+2:m,:)=temp(3*n+1:m-1,:);
   temp(3*n+1,:)=zeros(1,m);
   K=K+temp;
end

K=complete(K);
```

Recursive Code

```
function LL=complete(L)

% input:  Upper right trianguler response matrix
% output: Complete response matrix

n=size(L,1);

LL=zeros(n,n);
LL=L;
for i=1:n
  for j=1:i
    LL(i,j)=0;
  end
end
LL=LL+LL';

for i=1:n
  sum=0;
  for j=1:n
    sum=sum+LL(i,j);
  end
  LL(i,i)=-sum;
end


function L=forward(K,n)

% input:  K = Kirchhoff matrix
%         n = Number of boundary nodes
% output: L = Response matrix

m=size(K,1);

A=K(1:n,1:n);
B=K(1:n,n+1:m);
C=K(n+1:m,n+1:m);

L=A-B*inv(C)*B';
```

```
function i=in(n,b)

% Input:  n = Size of the lattice
%         b = Index of boundary node on the spike
% Output: i = Index of the interior node just inside of b

if b <= n
   i = 4*n+b;
elseif b <= 2*n
   i = 4*n+b-1;
elseif b <= 3*n
   i = 4*n+b-2;
elseif b < 4*n
   i = 4*n+b-3;
elseif b == 4*n
   i = 4*n+1;
end

function L=spforward(K,v)

% input:  K = Kirchhoff matrix
%         v = Vector of nodes to be internalized
% output: L = Response matrix

% u = Vector of boundary nodes (complement of v)
u = [];
for i = 1:length(K)
   found_in_v = 0;
   for j = 1:length(v)
      if v(j) == i
         found_in_v = 1;
      end
   end
   if found_in_v == 0;
      u(length(u)+1) = i;
   end
end
L = forward([K(u,u),K(u,v);K(v,u),K(v,v)],length(u));
```

References

[1] Edward B. Curtis and James A. Morrow, Inverse Problems for Electrical Networks, World Scientific, 13 (2000).
[2] Mike Usher, *Determining Current Sources in a Network*, unpublished.