# Finding an Altered Resistor in a Cubic Network

Robert A. Coury

University of Washington

**Abstract**

In this paper I will outline a method for determining the location of a resistor, or conductor, in a three-dimensional network whose conductivity has been altered from its original value.

## 1 Description of the Algorithm

The method we will use has four basic steps. First, given the base resistor network, generate the Lambda matrix for it. Second, construct the Lambda matrix for the network with the changed resistor. Third, compare the two matrices to determine the columns that exhibit the greatest change. Finally, calculate the coordinates of the changed resistor.

## 2 Constructing the Lambda Matrix

In all cases, we consider only cubic resistor networks with $n$ nodes along each edge of the cube. This means that the network has $n^3$ interior nodes, $6n^2$ boundary nodes, and a total of $3(n+1)n^2$ resistors.

## 2.1 The Numbering Scheme

We number the boundary nodes by considering one face at a time: first the bottom, then front, then left, then back, then right, and finally the top. On each face number the nodes sequentially by first going front to back, then bottom to top, then left to right, noting that only two of these directions will be used on any given face. Thus the bottom face will have nodes numbered from 1 to $n^2$, the front face from $n^2+1$ to $2n^2$, and so on. The interior nodes are numbered in a similar fashion. The bottom level will have nodes 1 to $n^2$, the next higher level $n^2+1$ to $2n^2$, etc.

The conductors are numbered in a slightly different manner. If we consider each node (both boundary and interior) to have coordinates $(x, y, z)$, where $x$ is the left-right distance, $y$ the front-back distance, and $z$ the top-bottom distance (all ranging from 0 to $n+1$), then the conductor between nodes $(x_1, y_1, z_1)$ and $(x_2, y_2, z_2)$ has coordinates $(x_1+x_2, y_1+y_2, z_1+z_2)$. For example, consider the front elevation of a 5x5x5 cube (Figure 1).

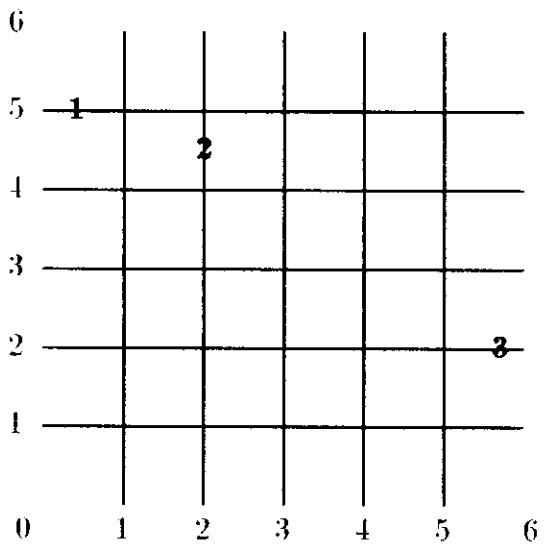


Figure 1

The conductor marked '1' has coordinates (1.2.10), while '2' has (4.2.9) and '3' has (11.2.1) for their coordinates.

It is fairly easy to see that this scheme will give us a unique representation for every resistor. It is also important to note that exactly one of the coordinate values is odd. This will important when we reconstruct the coordinates of the changed conductor.

## 2.2 Generating the Lambda Matrix

The Lambda matrix $\Lambda$ is defined to be the matrix whose $ij$-th entry $\Lambda_{i,j}$ is the current flowing into boundary node $i$ when a potential of 1 is applied at node $j$, with all other potentials at the boundary set to 0. $\Lambda$ will then be a square matrix with dimension $6n^2$.

The matrix generated in this manner will have similar properties to the Lambda matrix in the two-dimensional case: $\Lambda$ is a symmetric matrix whose rows and columns sum to zero, all entries are non-zero, all non-diagonal entries are negative, and all diagonal entries are positive.

# 3 Finding the Altered Resistor

In this section, we indicate how to determine the location of a changed resistor, given the Lambda matrices for both the base network and the changed network.

## 3.1 Comparing the Lambda Matrices

Consider each column (or row) of the Lambda matrix as a vector of dimension $6n^2$. Thus both Lambda matrices, generate a set of $6n^2$ vectors, $\{V_i\}$ and $\{W_i\}$. Calculate the angle $\theta$ between each pair of vectors $V_i$ and $W_i$ ($i = 1, 2, \ldots, 6n^2$), using the fact that $\cos\theta_i = V_i \cdot W_i / \|V_i\| \, \|W_i\|$. This yields a set of $6n^2$ angles.

## 3.2 Determining Coordinates

To start with, we separate the $\theta_i$'s into six sets of $n^2$ elements, corresponding to each of the six sides ($\theta_1$ to $\theta_{n^2}$ in one group, $\theta_{n^2+1}$ to $\theta_{2n^2}$ in another, and so on). Among each of the six sets, find the node (or nodes) with the maximum $\theta_i$-value for that side. Note that there will be six of these maximum values.

Four of the sets will have two such nodes, and the other two will have only one. The two sets with the single maximum node correspond to the faces that the changed conductor is 'in line' with. More precisely, consider Figure 2, a cross section of a cube where the resistor that has been changed is marked with an 'x'. For simplicity, we order the nodes as indicated and ignore the front and back faces, as well as travel in that direction.

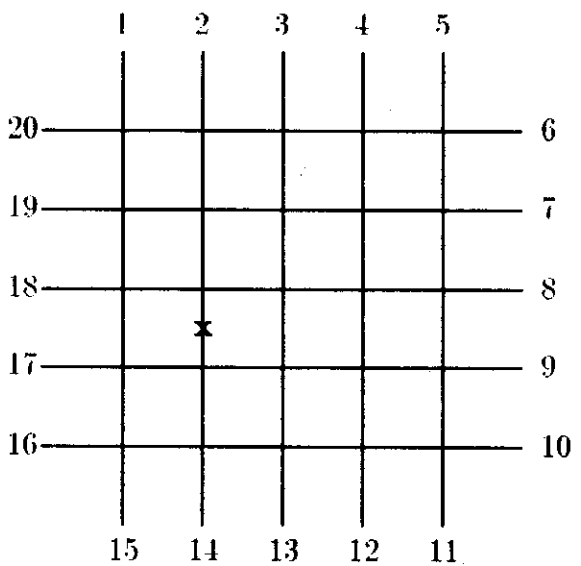


Figure 2

In this example, $\theta_i$ would attain its comparative maximum values at nodes 2, 8, 9, 11, 17, and 18. If we were to number the sides clockwise starting at the top, we would see that the marked conductor is 'in line' with sides 1 (top) and 3 (bottom), the sides with only one node attaining the maximum value for that side.

To calculate the coordinates of the changed conductor, we need to know the following: the lowest-numbered node that attains the maximum $\theta$-value on either the right or left side, and similarly for the front and back sides and the top and bottom faces. We also require the number of the side that the conductor is 'in line' with (where 1 is the bottom, 2 the front, 3 is left, 4 the back, 5 the right, and 6 the top). In Figure 3 we have the middle cross-section (three layers back) of a 5x5x5 cubic network. This time we will number the nodes properly.

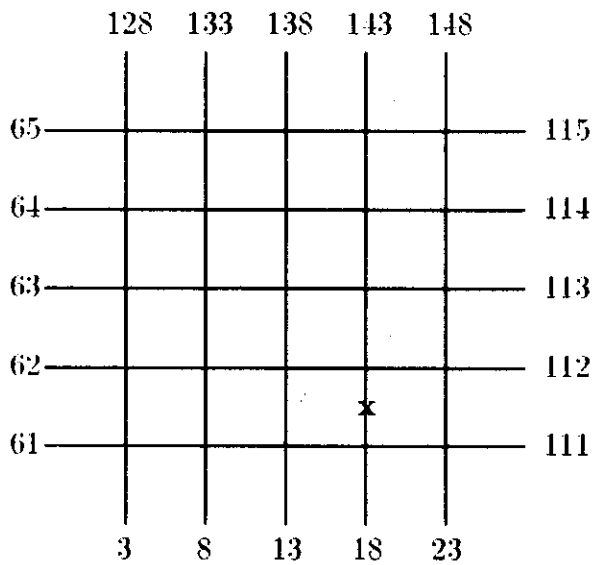


Figure 3

In this case, the conductor marked is in line with faces 1 and 6, and the nodes that attain the maximum $\theta$-value on their side are 18, 41, 42, 61, 62, 91, 92, 111, 112, and 113. (Note that nodes 41 and 42 come from the front face, and nodes 91 and 92 from the back.) From this collection of nodes we keep only nodes 18, 41, and 61. (Note that we could keep other combinations: for example, 18, 91, and 61, or 143, 91, and 111.)

We now apply these three nodes to the appropriate formulas, based on which side the conductor is 'in line' with, to determine the coordinates of the resistor. The reason that the side that the conductor is 'in line' with is so important is that it tells us which of the coordinates of the conductor must be odd.

### 3.3 The Formulas

In all of these formulas, $a$, $b$, and $c$ are the lowest- numbered nodes that attain that maximum $\theta$-value on the bottom (or top), front (or back), and left (or right) sides, respectively. The coordinates for the resistor are then $(x, y, z)$.

If the conductor is 'in line' with either the bottom or top face, then

$$\begin{aligned} x &= 2(1 + (a-1)/n) \\ y &= 2(1 - 2n + (c-1)/n) \\ z &= 2(b \bmod n) + 1 \end{aligned}$$

If the conductor is 'in line' with either the front or back face, then the coordinates are given by:

$$\begin{aligned} x &= 2(1 + (a-1)/n) \\ y &= 2(a \bmod n) + 1 \\ z &= \begin{cases} 2n, & \text{if } z \equiv 0 \pmod{n}: \\ 2(b \bmod n), & \text{otherwise.} \end{cases} \end{aligned}$$

Finally, if the conductor is 'in line' with either the right or left face, then

$$\begin{aligned} x &= 2(1 + (a-1)/n) + 1 \\ y &= 2(1 - 2n + (c-1)/n) \\ z &= \begin{cases} 2n, & \text{if } z \equiv 0 \pmod{n}: \\ 2(b \bmod n), & \text{otherwise.} \end{cases} \end{aligned}$$

# 4 The Program

The program consists of four parts, corresponding to the four basic stages in the algorithm. First, it obtains the background conductivity and calculates the Lambda matrix for the base network. It then reqests the position of the resistor to be changed and its new value, and calculates its Lambda matrix. The program now takes these two matrices and calculates the angles between pairs of columns, applying the calculated data to the appropriate system of equations to generate the location of the resistor.

The program is written in FORTRAN and is as modular as possible. Its efficiency is also boosted by the use of named COMMON blocks (which make a noticeable difference in the time needed to calculate the Lambda matrices).

## 4.1 Numerical Problems

There were some basic numerical problems encountered along the way. Some of them have been resolved, while others may or may not have a solution.

### 4.1.1 Time and Space

Time and memory space are limiting factors for this method. In order to make the program run fairly quickly, it is necessary to use a banded solver routine for generating the Lambda matrices. This gives a substantial improvement in the running times, as indicated in Table 1.

**Table 1**

| $n$ | *time* |
|---|---|
| 2 | 00:01 |
| 3 | 00:01 |
| 4 | 00:04 |
| 5 | 00:10 |
| 6 | 00:25 |
| 7 | 01:01 |
| 8 | 02:12 |

As concerns memory space, it is not possible to set aside enough memory to calculate the Lambda matrix for a 9x9x9 cubic network using the banded solver. Memory is not a problem with the general solver, but the program requires a great many hours to run.

### 4.1.2 Rounding Errors

One of the major problems encountered has to do with rounding errors. For example, if we look back to Figure 3, we see that nodes 111 and 112 should both attain the maximum $\theta$-value. However, in general, the two $\theta$-values are different due to rounding errors that occur in some of the calculations. The problem arises, for example, when $\theta_{112}$ is larger than $\theta_{111}$. This means that the lowest node that attains the maximum $\theta$-value is wrong. To compensate

for this, I introduce a tolerance level. A level of 25% of the maximum $\theta$-value on a side is enough in the 2x2x2 and 3x3x3 cases.

When there are more than 3 nodes on an edge, a tolerance level will not solve the problems. For $nxnxn$ systems with $n \geq 4$, the code needs to be rewritten to deal with the different properties that arise.

### 4.1.3 Other Problems

There are a few other incidental problems that should be noted. Most of these seem to be compiler-specific. Some examples: a GOTO statement altering calculations; variables being overwritten; and the banded solver not working when the program becomes too large. These problems, along with some of the memory restrictions, may disappear on a different machine.

## 4.2 Other Methods

Another way to compare the Lambda matrices is simply to take the difference of the diagonal entries, rather than to calculate the angle between columns; that is, instead of calculating $\theta_i$, we calculate $d_i = L_{i,i} - M_{i,i}$ where L and M are the two Lambda matrices. While the results using this method are generally correct, there are some cases where the differentiation between the $d_i$'s is not great enough, even when a tolerance level is used, to generate accurately the correct nodes.

# 5 Results

We close with a summary of the results obtained so far. Given a cubic resistor network (either 2x2x2 or 3x3x3) with a constant base conductivity of $c$ and one resistor with a conductivity of $k$, the method I have described will correctly find the coordinates of the different-valued resistor. This seems to be independent of the choice of $c$ and $k$, as long as $k - c > 10^{-6}$ and $c > 0$.

In the 4x4x4 cube, approximately a third of the resistors could be detected accurately. It is interesting to note that a certain conductor will seem always to work, or never work. In

other words, it does not appear to depend on the choice of $c$ and $k$. Another interesting fact is that 'symmetry' does not seem to predict whether or not a given resistor will be found accurately:

there are pairs of resistors which have 'symmetric' positions in the cube, but one works and the other does not.

Finally, it seems that it should be possible to calculate the change in the resistor based on the $\theta$'s generated. There appears to be a definite pattern to the size of the $\theta$'s and the magnitude of the change.

# 6 Further Ideas

The next step is to be to work on the program so that it can handle the 4x4x4 cube completely. This should require only a fairly simple reworking of the code, making it work in a manner more sophisticated than the fairly simple way it cuurently works.