DISTRUBUTED AND LUMPED NETWORKS WITH PIECEWISE CONSTANT CONDUCTIVITIES

MARC PICKETT I

Abstract. The purpose of this project is to investigate approximations for the Dirichlet norm for distributed networks piecewise constant conductivities. This paper follows the methods developed by Duffin [2], and uses "distributed" and "lumped" are as defined by [2].

Contents

1. Piecewise Constant Conductivity

Consider a distributed polygonal network (Ω) which has been triangulated where the conductivity (γ_T) of each triangle is constant. Also, let there be a function U on Ω. Let $\partial\Omega$ be the boundary of Ω and ∂T_i be the boundary of each triangle. Define a function $L(U)$ such that

$$L(U(x,y)) = 0 = \gamma_1 \frac{\partial U_1}{\partial n_1} + \gamma_2 \frac{\partial U_2}{\partial n_2} \qquad (x,y) \in (\partial T_1 \cap \partial T_2) \tag{1.0.1}$$

$$L(U(x,y)) = 0 = \nabla\left(\gamma_{T_i} \nabla U\right) \qquad (x,y) \notin (\partial T_1 \cap \partial T_2) \tag{1.0.2}$$

where T_1 and T_2 are two triangles of Ω, and n_1 and n_2 are the outward normals for T_1 and T_2, respectively.

The Dirichlet inner product for two functions, U and V, is defined as

$$D_\gamma (U,V) = \int_\Omega \gamma \nabla U \cdot \nabla V \tag{1.0.3}$$

$D_\gamma (U,U)$ is called the Dirichlet norm of U. $D_\gamma (U,U)$ is "potentially definite" if $D_\gamma (U,U) \geq 0$ and $D_\gamma (U,U) = 0$ iff U is constant [2] .

If Ω is a lumped network then the Dirichlet inner product for two functions U and V defined on each node of Ω will be given by the quadratic form

$$Q(U,V) = \sum_{\forall i,j} g_{ij} (v_i - u_j) \tag{1.0.4}$$

where i and j are nodes of the lumped network and v_i and u_j are the values of U and V at those nodes.

1.1. **An upper network for piecewise constant conductivities.** Assume U is the solution of the Dirichlet problem, U is piecewise linear on $\partial\Omega$ and V is a linear function which is equal to U on $\partial\Omega$. (Where Ω is a triangulated distributed network) Let $W = U - V$.

$$D_\gamma (V,V) = D_\gamma (U-W, U-W) = \int_\Omega \gamma |U-W|^2 \tag{1.1.1}$$

$$= \int_\Omega \gamma |\nabla U|^2 - 2\int_\Omega \gamma \nabla U \cdot \nabla W + \int_\Omega \gamma |\nabla W|^2 \tag{1.1.2}$$

It will be shown (Lemma 1) that

$$\int_\Omega \gamma \nabla U \cdot \nabla W = \int_{\partial\Omega} \gamma W \frac{\partial U}{\partial n} \tag{1.1.3}$$

which implies

$$D_\gamma (U,W) = \int_\Omega \gamma \nabla U \cdot \nabla W = \int_{\partial\Omega} \gamma W \frac{\partial U}{\partial n} = 0 \tag{1.1.4}$$

since W = 0 on $\partial\Omega$. This reduces 1.1.2 to

$$D_\gamma(V,V) = D_\gamma(U,U) + D_\gamma(W,W) \tag{1.1.5}$$

This implies

$$D_\gamma(V,V) \geq D_\gamma(U,U) \tag{1.1.6}$$

since the Dirichlet functional is positive definite.

Lemma 1. *Let U be a γ-harmonic function and W be a continuos function on a triangulated distributed network Ω with constant conductivities on each triangle.*

$$\int_\Omega \gamma \nabla U \cdot \nabla W = \int_{\partial\Omega} \gamma W \frac{\partial U}{\partial n} \tag{1.1.7}$$

Proof. Consider the dirichlet inner product for a triangle (T_1) which has constant conductivity γ_1 and two γ-harmonic functions, U_1 and W_1. The Dirichlet inner product for U_1 and W_1 on this triangle is

$$D_\gamma(U_1,W_1) = \int_{T_1} \gamma_1 \nabla U_1 \cdot \nabla W_1 \tag{1.1.8}$$

By Green's theorem

$$= \int_{\partial T_1} \gamma_1 W_1 \frac{\partial U_1}{\partial n_1} \tag{1.1.9}$$

Assume another triangle has constant conductivity (γ_2) and shares an edge (C) with the first triangle. Also, assume that this triangle has two γ-harmonic functions U_2 and W_2 such that boundary condition 1.0.1 is satisfied on the shared edge, C.

Then the Dirichlet inner product for the combined set of the two triangles ($\Omega_{1,2}$) is

$$D_\gamma(U,W) = \int_{\Omega_{1,2}} \gamma \nabla U \cdot \nabla W = \int_{T_1} \gamma_1 \nabla U_1 \cdot \nabla W_1 + \int_{T_2} \gamma_2 \nabla U_2 \cdot \nabla W_2 \tag{1.1.10}$$

By 1.1.9

$$= \int_{\partial T_1} \gamma_1 W_1 \frac{\partial U_1}{\partial n_1} + \int_{\partial T_2} \gamma_2 W_2 \frac{\partial U_2}{\partial n_2} \tag{1.1.11}$$

By 1.0.1 this integral along C is 0. This leaves only the integral for the boundary of the combined region $(\partial\Omega_{1,2})$. Or

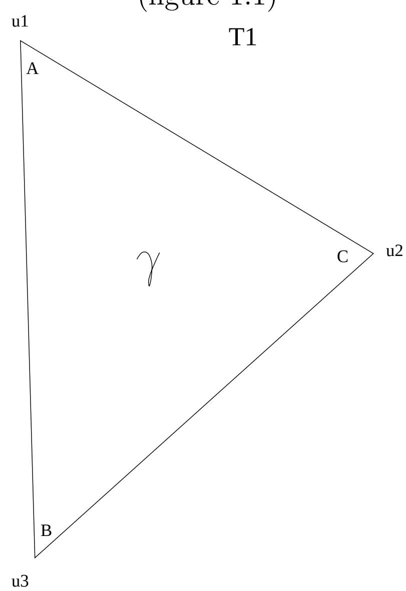
$$D_\gamma(U,W) = \int_{\Omega_{1,2}} \gamma \nabla U \cdot \nabla W = \int_{\partial\Omega_{1,2}} \gamma W \frac{\partial U}{\partial n} \tag{1.1.12}$$

This process follows for any other triangles that are added onto the region.

□

Let T_1 be a triangle with constant conductivity γ, angles A, B, and C, and voltages u_1, u_2, and u_3 at their respective points. (See figure 1.1.) It has been shown that if $V = U$ on the boundary of T_1, then $D(V,V) \geq D(U,U)$. Let U be piecewise linear on ∂T_1. The Dirichlet norm for a linear function V on T_1 is

(figure 1.1)



$$D_{\gamma T}(V,V) = \int_T \gamma \left|\nabla V\right|^2 = \gamma \left|\nabla V\right|^2 A_T \tag{1.1.13}$$

where A_T is the area of the triangle.

To find the weights (g_A, g_B, and g_C) for an upper network for T_1 we will employ a method similar to Duffin's method for finding weights for an upper network with uniform conductivity [2] . Since D_T is a potentially definite quadratic form we can write

$$D_{\gamma T}(V,V) = g_C(u_1-u_2)^2 + g_B(u_1-u_3)^2 + g_A(u_2-u_3)^2 \tag{1.1.14}$$

To determine g_A, g_B, and g_C we set $u_2 = u_3 = 0$ and 1.1.13 equal to 1.1.14 leaving

(1.1.15)
$$g_C u_1^2 + g_B u_1^2 = \gamma \left|\nabla V\right|^2 A_T = \gamma \left|\nabla U\right|^2 \left(\frac{h}{2} h \cot B + \frac{h}{2} h \cot C\right) = \gamma \left(\frac{u_1}{h}\right)^2 \left(\frac{h^2}{2} \cot B + \frac{h^2}{2} \cot C\right)$$

where h is the height of the triangle using the side opposite A as its base.

This is equivalent to

(1.1.16)
$$\frac{\gamma}{2}\left(\cot B + \cot C\right) = g_C + g_B$$

By symmetry, we obtain

(1.1.17)
$$\frac{\gamma}{2}\left(\cot A + \cot C\right) = g_A + g_C$$

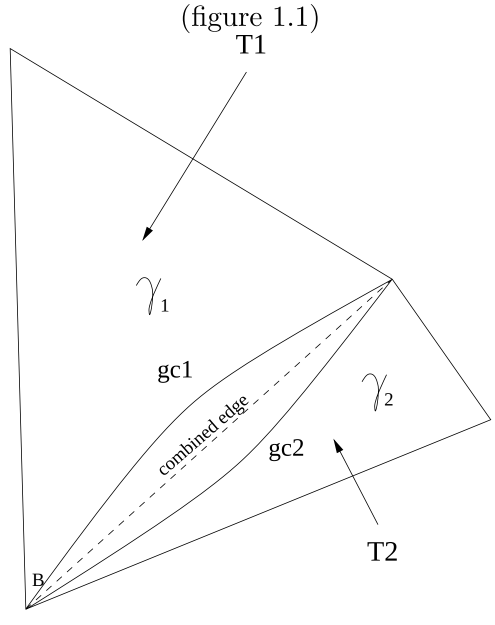
(1.1.18)
$$\frac{\gamma}{2}\left(\cot A + \cot B\right) = g_A + g_B$$

The solution for this set of equations is

(1.1.19)
$$g_i = \frac{\gamma}{2} \cot i$$

where i is one of the edges of T_1.

(figure 1.1)

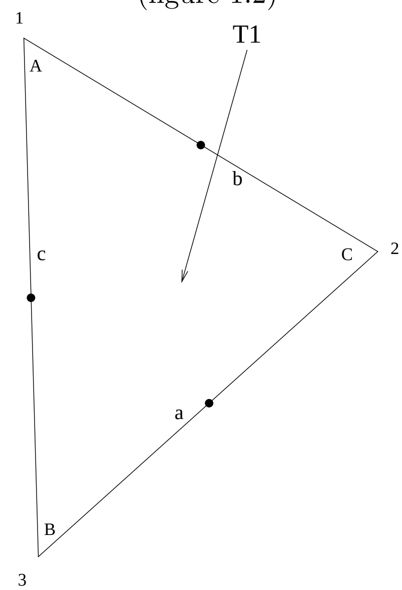


Suppose two triangles in Ω share an edge C. (See figure 1.1) Then, g along this edge in the lumped network is the sum of g_c of each of the two triangles.

From 1.1.6 the Dirichlet norm of V on this network is greater than or equal to the Dirichlet norm of U on the lumped network Ω. This provides an *upper network* for Ω.

1.2. **A lower network for piecewise constant conductivities.** Now we will find a network and function on this network whose Dirichlet norm is a lower bound for that of U on Ω. Let a, b, and c be the edges of a triangle T_1, and let v_a, v_b, and v_c be the potential at the midpoint of a, b, and c, respectively. (See figure 1.2) Let V_1 be a linear function on T_1. Let w_a be the current which flows into edge a, and likewise for w_b and w_c. Then w_a, w_b, and w_c can be expressed as linear functions of v_a, v_b, and v_c. That is

(figure 1.2)



$$w_a = g_{ab}\left(v_a - v_b\right) + g_{ac}\left(v_a - v_c\right) \tag{1.2.1}$$

The Dirichlet norm of V_1 on T_1 is

$$D_{\gamma T_1}\left(V_1, V_1\right) = g_{ab}\left(v_a - v_b\right)^2 + g_{ac}\left(v_a - v_c\right)^2 + g_{bc}\left(v_b - v_c\right)^2 \tag{1.2.2}$$

To determine g_{ab}, g_{ac}, and g_{bc}, let v_1, v_2, and v_3 be the values of the function V_1 on the vertices of T_1. Substituting 1.1.19 into 1.1.14 we obtain

$$D_{\gamma T_1}\left(V_1, V_1\right) = \gamma_{T_1}\left(\left(v_1 - v_2\right)^2 \cot C + \left(v_1 - v_3\right)^2 \cot B + \left(v_2 - v_3\right)^2 \cot A\right) \tag{1.2.3}$$

Side a joins vertices 2 and 3 so v_a is the average of v_2 and v_3. That is, $v_a = \frac{1}{2}\left(v_2 + v_3\right)$, and similar for v_b and v_c. Or $2\left(v_a - v_b\right) = \left(v_2 - v_1\right)$, and so on. This implies

$$D_{\gamma T_1}(V_1, V_1) = 2\gamma_{T_1}\cot C\,(v_a - v_b)^2 + 2\gamma_{T_1}\cot B\,(v_a - v_c)^2 + 2\gamma_{T_1}\cot A\,(v_b - v_c)^2 \tag{1.2.4}$$

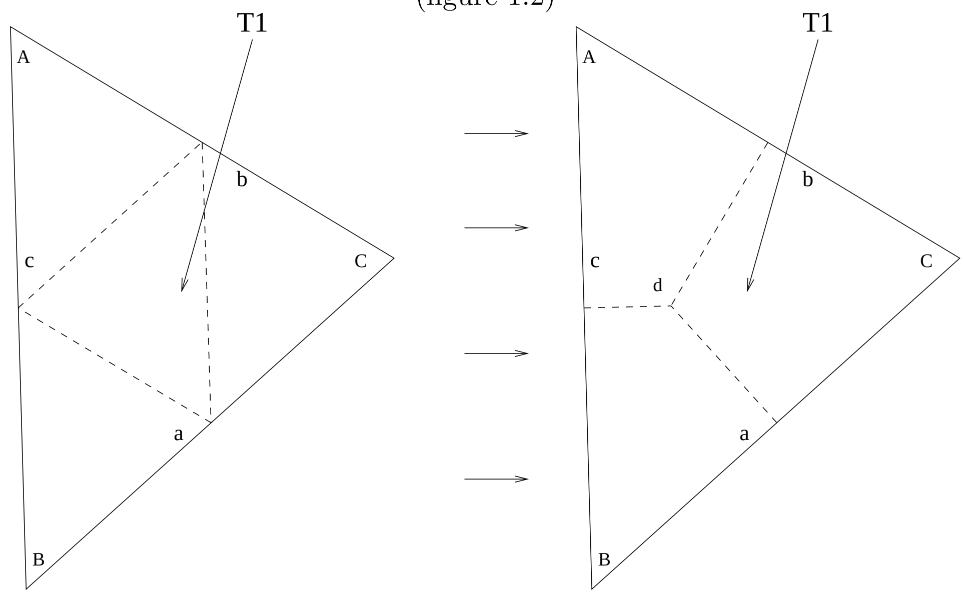
Setting 1.2.2 equal to 1.2.4 we obtain

$$g_{ab} = 2\gamma_{T_1}\cot C \tag{1.2.5}$$

$$g_{ac} = 2\gamma_{T_1}\cot B \tag{1.2.6}$$

$$g_{bc} = 2\gamma_{T_1}\cot A \tag{1.2.7}$$

(figure 1.2)



This network is equivalent to a network with a 4th node d inserted in the middle of the triangle such that each of the midpoints of a, b, and c are connected to d. (See figure 1.2) The transformation from the original 3 node network to the 4 node network is called a Δ-Y transformation. This new network has conductivities γ_{ad}, γ_{bd}, and γ_{cd} where

$$\gamma_{ad} = g_{ac}g_{ab}\left(\frac{1}{g_{ab}} + \frac{1}{g_{ac}} + \frac{1}{g_{bc}}\right) = 2\gamma_{T_1}\frac{\tan A + \tan B + \tan C}{\tan B \tan C} = 2\gamma_{T_1}\tan A \tag{1.2.8}$$

This comes from the trigonometric identity (Lemma 2)

$$\frac{\tan A + \tan B + \tan C}{\tan A \tan B \tan C} = 1 \qquad \forall(A, B, C)|A + B + C = \pi \tag{1.2.9}$$

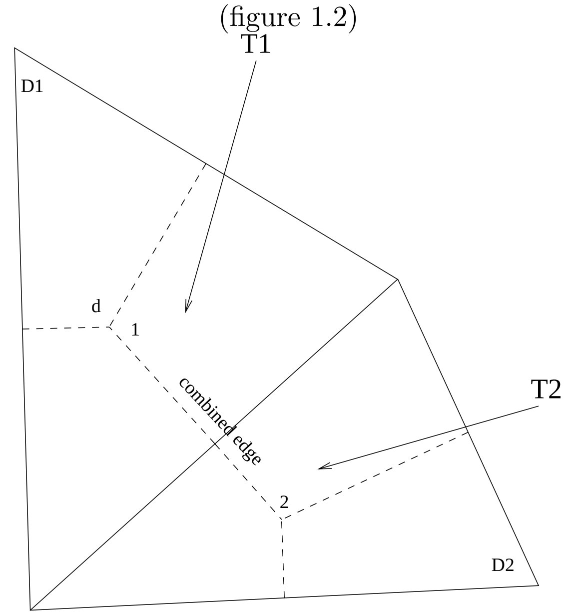
Likewise we obtain

$$\gamma_{bd} = 2\gamma_{T_1}\tan B \tag{1.2.10}$$

$$\gamma_{cd} = 2\gamma_{T_1}\tan C \tag{1.2.11}$$

Consider another triangle T_2 with conductivity γ_{T_2} which has an adjacent edge with T_1. (See figure 1.2) Call the angles opposite the combined edge D_1 and D_2. The conductivity for this combined edge $(1,2)$ is

(figure 1.2)



$$\gamma_{1,2} = \frac{1}{\frac{1}{\gamma_{1d}} + \frac{1}{\gamma_{2d}}} = \frac{1}{\frac{1}{\frac{2}{\gamma_{T_1}}\tan D_1} + \frac{1}{\frac{2}{\gamma_{T_2}}\tan D_2}} = \frac{1}{\frac{\gamma_{T_1}}{2}\cot D_1 + \frac{\gamma_{T_2}}{2}\cot D_2} \tag{1.2.12}$$

A network with these conductivities provides a *lower network* for Ω. Lemma 3 and arguments similar to those on Duffin's page 806 [2] may be used to show that the Dirichlet norm of V on this network is less than or equal to that of U on Ω.

Lemma 2.

$$\frac{\tan A + \tan B + \tan C}{\tan A \tan B \tan C} = 1 \qquad \forall (A,B,C) | A+B+C = \pi \tag{1.2.13}$$

Proof. (By J. Morrow) Let $A+B+C=\pi$ then

$$0 = \tan\pi = \tan(A+B+C) = \frac{\tan A + \tan(B+C)}{1 - \tan A \tan(A+B)} \tag{1.2.14}$$

which implies

$$\tan A + \tan(B+C) = 0 = \tan A + \frac{\tan B + \tan C}{1 - \tan B \tan C} \tag{1.2.15}$$

which implies

$$\tan A + \tan B + \tan C - \tan A \tan B \tan C = 0 \tag{1.2.16}$$

The result follows.

□

Lemma 3. *Let W be a piecewise constant vector field with a continuous normal component across edges such that div $W = 0$. Let U be the solution to the Dirichlet problem on a triangulated region. Then*

$$D_\gamma(U) \geq \frac{\left(\int_{\partial\Omega} UW_n\right)^2}{\int_\Omega \gamma |W|^2} \tag{1.2.17}$$

Proof. Using Schwarz' inequality we get

$$\left(\int_\Omega \gamma^{\frac{1}{2}} W \cdot \gamma^{\frac{1}{2}} \nabla U\right)^2 \leq \left(\int_\Omega \gamma |W|^2\right)\left(\int_\Omega \gamma |\nabla U|^2\right) \tag{1.2.18}$$

Which implies

$$\frac{\left(\int_\Omega \gamma W \cdot \nabla U\right)^2}{\int_\Omega \gamma |W|^2} \leq \int_\Omega \gamma |\nabla U|^2 \tag{1.2.19}$$

From lemma 4, we have

$$\int_\Omega \gamma W \cdot \nabla U = \int_{\partial\Omega} \gamma U W_n \tag{1.2.20}$$

Combining the above two equations we get the result. Since

$$D_\gamma(U, U) = \int_\Omega \gamma |\nabla U|^2 \tag{1.2.21}$$

□

Lemma 4. *Let W be a piecewise constant vector field with a continuous normal component across edges such that div $W = 0$. Let U be the solution to the Dirichlet problem. Then*

$$\int_{\Omega} \gamma W \cdot \nabla U = \int_{\partial\Omega} \gamma U W_n \tag{1.2.22}$$

Proof. Let T_1 be a triangle of Ω with conductivity γ_1. Then by Green's theorem

$$\int_{T_1} \gamma_1 W \cdot \nabla U = \int_{\partial T_1} \gamma_1 U W_{n_1} \tag{1.2.23}$$

Consider another Triangle T_2 which has conductivity γ_2 and shares an edge C with T_1. Let Ω_{12} be the region of T_1 and T_2. Then

$$\int_{\Omega_{12}} \gamma W \cdot \nabla U = \int_{T_1} \gamma_1 W \cdot \nabla U + \int_{T_2} \gamma_2 W \cdot \nabla U = \int_{\partial T_1} \gamma_1 U W_{n_1} + \int_{\partial T_2} \gamma_2 U W_{n_2} \tag{1.2.24}$$

Along C, this integral is

$$\int_{C} \gamma_1 U W_{n_1} + \gamma_2 U W_{n_2} = 0 \tag{1.2.25}$$

This comes from the the hypothesis on W.

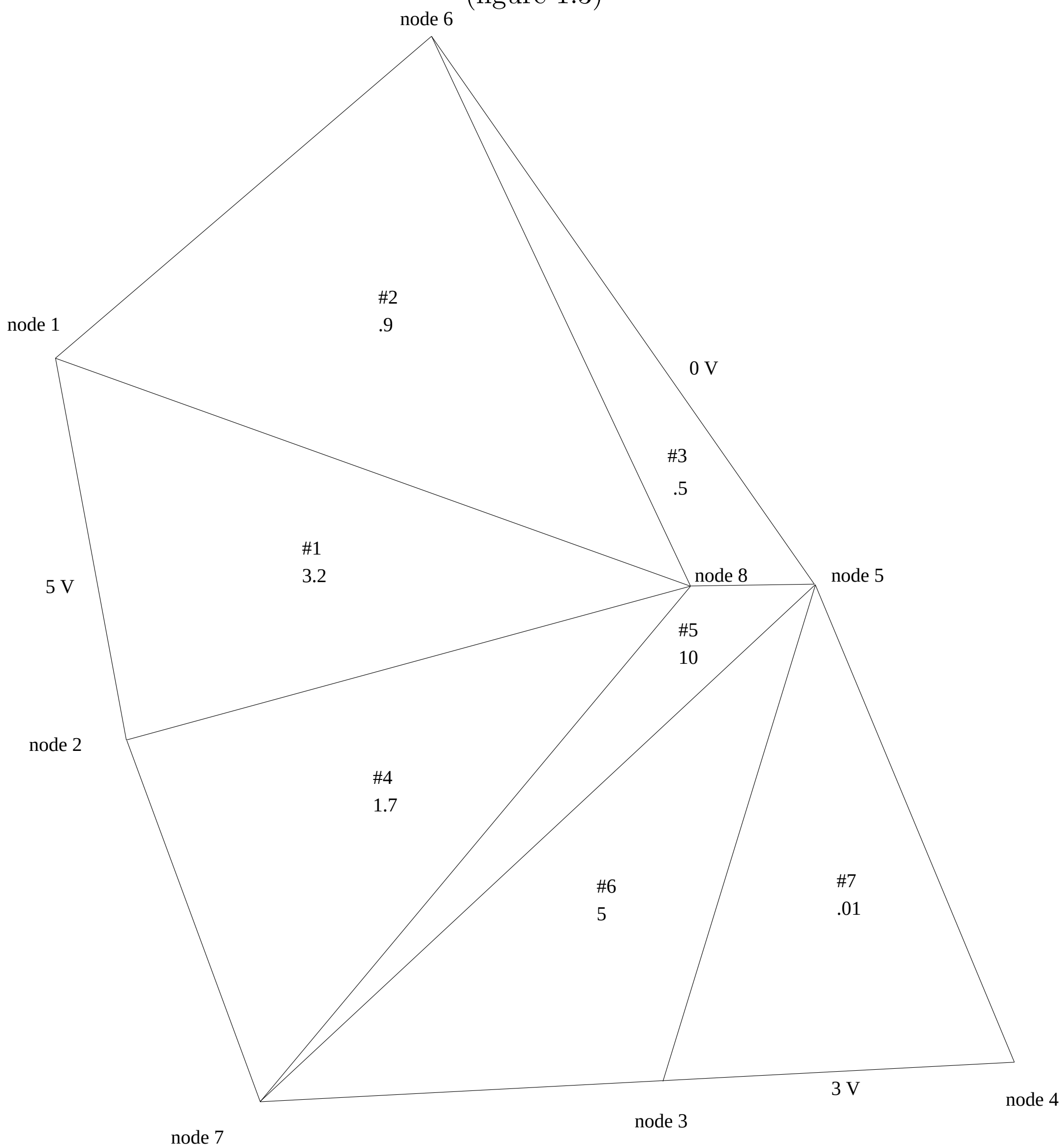
$$\int_{\Omega_{12}} \gamma W \cdot \nabla U = \int_{\partial\Omega_{12}} \gamma U W_n \tag{1.2.26}$$

This process follows for adding on other triangles. $\square$

1.3. **Calculations of piecewise constant conductivity.** Using figure 1.3 as an example, we get

	1	2	3	4	5	6	7	8
1	2.9956	-2.28504	0	0	0	-0.450001	0	-0.260562
2	-2.28504	4.91348	0	0	0	0	-1.21393	-1.41451
3	0	0	7.3173	-0.00595878	-2.98121	0	-4.33013	0
4	0	0	-0.00595878	0.00777863	-0.00181986	0	0	0
5	0	0	-2.98121	-0.00181986	12.9163	0.116576	5.1054	-15.1552
6	-0.450001	0	0	0	0.116576	0.768285	0	-0.43486
7	0	-1.21393	-4.33013	0	5.1054	0	9.17329	-8.73463
8	-0.260562	-1.41451	0	0	-15.1552	-0.43486	-8.73463	25.9998

(figure 1.3)



for the upper Kirchoff matrix. (The lower Kirchoff matrix is not displayed here since it is 14 by 14.) Using methods described in [1], we get 69.9055 for the upper bound of the Dirichlet norm, and 33.3273 for the lower bound.

References

[1] E. Curtis, D. Ingerman, J. Morrow, *Circular planar graphs and resistor networks*, submitted.
[2] R. J. Duffin, *Distributed and Lumped Networks*, Journal of Mathematics, Vol. 8, No. 5 (1959), 793-826.

Appendix A. Files used

A.1. **triangle.cxx.**

```
// FILE: triangle.cxx  (by Marc Pickett I  pickett@refuge)
// this program calculates the upper and lower Kirchoff matrices
// for triangulated distributed networks with piecewise constant
// conductivities.
// The usage of this command is:
// cat <file.triangle> | triangle
// file.triangle is the input file a sample of which is
// sample.triangle  or  sample2.triangle
// Note:  sample triangle is unusable because of its comments.

# include <iostream.h>
# include <stdlib.h>
# include <math.h>
#include <iomanip.h>   // Provides setw and setf

// the triangle struct
struct triangle
{
    double conduct;
    double angle[3];
    int points[3][2];
};

void initialize(int numma_of_triangles, triangle* &distrib);
void upper(
              int numma_of_triangles,
      triangle* &distrib,
           double** &upper_kirch,
      int nodes
          );
void lower(
              int numma_of_triangles,
      triangle* &distrib,
           double** &upper_kirch,
      int &nodes
          );
void init_matrix(double** &matrix, int nodes);
void display_mat(double** &mat, int rows, int columns);
int find_match(
                  int i,
                  int ii,
                  triangle* &distrib,
                  int numma_of_triangles,
  int &nodes
              );
void switcher(double** &mat, int a, int b, int nodes);
void kirch_diag ( double** kirch, int nodes );


main()
```

```
{
    int numma_of_triangles, nodes, low_nodes;
// nodes is the number of nodes in the triangle (and the upper network)
// low_nodes is the number of nodes in the lower network

    triangle* distrib;

    double** upper_kirch;
    double** lower_kirch;
// these are where the upper and lower kirchoff matrices are stored

    cin >> numma_of_triangles;
    cin >> nodes;

    distrib = new triangle[numma_of_triangles];

    initialize(numma_of_triangles, distrib);
// input the network

    low_nodes = numma_of_triangles;

    upper(numma_of_triangles, distrib, upper_kirch, nodes);
    lower(numma_of_triangles, distrib, lower_kirch, low_nodes);
// compute the upper and lower matrices

    cout << "The upper Kirchoff matrix is:" << endl;
    display_mat(upper_kirch, nodes, nodes);
    cout << "\n \n";
    cout << "The lower Kirchoff matrix is:" << endl;
    display_mat(lower_kirch, low_nodes, low_nodes);
    cout << "\n \n";
}


void initialize(int numma_of_triangles, triangle* &distrib)
// read in the triangles
{
    int i, ii;

    for ( i = 0; i < numma_of_triangles; i++)
    {
        cin >> distrib[i].conduct;
for ( ii = 0; ii <= 2; ii++)
{

    cin >> distrib[i].angle[ii];
    cin >> distrib[i].points[ii][0];
    cin >> distrib[i].points[ii][1];
    distrib[i].points[ii][0] -= 1;
    distrib[i].points[ii][1] -= 1;
```

```
    distrib[i].angle[ii] = distrib[i].angle[ii] * 3.14159/180;
}
    }

    return;
}


void upper(
              int numma_of_triangles,
      triangle* &distrib,
      double** &upper_kirch,
      int nodes
          )
// compute the upper matrix
{
    int i,ii;

    init_matrix(upper_kirch, nodes);

// the following for loop is calculating the new conductivities as
// given by my paper
    for ( i = 0; i < numma_of_triangles; i++)
    {
        for ( ii = 0; ii <= 2; ii++)
{
    upper_kirch[distrib[i].points[ii][0]]
               [distrib[i].points[ii][1]]
        -=  .5 * distrib[i].conduct * cot(distrib[i].angle[ii]);
    upper_kirch[distrib[i].points[ii][1]]
               [distrib[i].points[ii][0]]
        -=  .5 * distrib[i].conduct * cot(distrib[i].angle[ii]);
}
    }

    // put in the diagnal entries
    kirch_diag ( upper_kirch, nodes );


    return;
}

void lower(
              int numma_of_triangles,
      triangle* &distrib,
      double** &lower_kirch,
      int &low_nodes
          )
// compute the lower matrix
{
```

```
    int i, ii, going_to[numma_of_triangles][3];

// here I need to find which conductivities are shared between triangles
// and which ones goto the border.  For the ones that goto the border
// I need to create a new node (a border node.)A
// low_nodes is the total number of needed nodes (border nodes + boundary nodes)
    for ( i = 0; i < numma_of_triangles; i++ )
        for ( ii = 0; ii <= 2; ii++)
    going_to[i][ii] = find_match( i, ii, distrib,
                           numma_of_triangles, low_nodes );

    init_matrix(lower_kirch, low_nodes);

// the following for loop is calculating the new conductivities as
// given by my paper
    for ( i = 0; i < numma_of_triangles; i++)
    {
        for ( ii = 0; ii <= 2; ii++)
{
    if ( lower_kirch[i][going_to[i][ii]] == 0 )
        lower_kirch[i][going_to[i][ii]] = -2 * tan(distrib[i].angle[ii])
                                    * distrib[i].conduct;
    else
    {
        lower_kirch[i][going_to[i][ii]] =
    -(1/(1/-lower_kirch[i][going_to[i][ii]] +
    1/(2 * tan(distrib[i].angle[ii]) * distrib[i].conduct)));
            }
    lower_kirch[going_to[i][ii]][i] =
    lower_kirch[i][going_to[i][ii]];
}

    }
    // put in the diagnal entries
    kirch_diag ( lower_kirch, low_nodes );
}


void init_matrix(double** &matrix, int nodes)
// initializes a matrix.  Need more be said?
{
    int i,ii;

    matrix = new double*[nodes];

    for ( i = 0; i < nodes; i++)
    {
       matrix[i] = new double[nodes];
       for ( ii = 0; ii < nodes; ii++)
          matrix[i][ii] = 0;
```

```
    }

    return;
}

void display_mat(double** &mat, int rows, int columns)
// displays a matrix.
{
    int i,ii;

    cout << "\n";

    for ( i = 0; i < rows; i++)
    {
        for ( ii = 0; ii < columns; ii++)
    cout << setw(10) << mat[i][ii] << " ";

        cout << "\n";
    }

    return;
}

int find_match(
                  int i,
  int ii,
  triangle* &distrib,
  int numma_of_triangles,
  int &low_nodes
      )
// this function looks for a triangle which shares the edge ii of triangle i
// it calls this triangle iii and it's shared edge ia.
{
    int iii = 0, ia = 0;

//  a well conditioned while loop:
// basically this just looks that iii is a triangle not the same as
// i which shares edge iii of i.
    while ( ( iii < numma_of_triangles )
          &&
          ( ( iii == i )
  ||
    ( ( ( distrib[i].points[ii][0] != distrib[iii].points[ia][0] ) ||
        ( distrib[i].points[ii][1] != distrib[iii].points[ia][1] ) )
            &&
            ( ( distrib[i].points[ii][1] != distrib[iii].points[ia][0] ) ||
              ( distrib[i].points[ii][0] != distrib[iii].points[ia][1] ) )
      ) ) )
    {
       ia ++;
```

```
        if ( ia >= 3 )
        {
            ia = 0;
    iii++;
        }
     }

    if ( iii >= numma_of_triangles )
     {
         iii = low_nodes;
         low_nodes++;
     }

     return iii;
}



void switcher(double** &mat, int a, int b, int nodes)
// This switches 2 nodes in matrix mat.
// this fuction is unused in this program, but it would
// be useful if one wanted to place the boundary nodes at the
// top of the Kirchoff matrix.
{
    int i;
    double  temp2;
    double* temp = mat[a - 1];
    mat[a - 1] = mat[b - 1];
    mat[b - 1] = temp;

    for (i = 0; i < nodes; i++)
    {
        temp2 = mat[i][a - 1];
        mat[i][a - 1] = mat[i][b - 1];
mat[i][b - 1] = temp2;
    }

    return;
}


void kirch_diag ( double** kirch, int nodes )
// put the diagnal entries in a kirchoff matrix
{
    int i, ii;
    double row_sum = 0;

    for ( i = 0; i < nodes; i++ )
    {
        for ( ii = 0; ii < nodes; ii++ )
```

```
            row_sum -= kirch[i][ii];
        kirch[i][i] = row_sum;
row_sum = 0;
    }
}
```

A.2. **sample.triangle.** This is the sample input file used to make calculations with triangle.cxx for the example distributed network in this paper. To use this, the comments must be taken out.

```
//  Sample triangulated network input file for triangle.cxx
//  ALL the comments must be removed in order for this to work
//  sample2.triangle is a usable sample file (with out comments)

7  // number of triangles
8  // number of nodes

//  The order of input:
//  angle of edges (in degrees),   connects point,      to point


//   1st triangle
3.2 // conductivity of triangle
60 2 8 // angle A
85 1 8 // angle B
35 1 2  // angle C

//   2nd triangle
.9
60 8 6
75 1 8
45 1 6

//   3rd triangle
.5
115 6 5
55 6 8
10 5 8

//   4th triangle
1.7
60 2 8
85 7 8
35 7 2

//   5th triangle
10
20 5 8
```

```
130 7 5
30 7 8

//   6th triangle
5
40 5 3
110 5 7
30 7 3

//   7th triangle
.01
40 3 4
70 5 4
70 5 3
```

E-mail address: pickett@refuge.colorado.edu