The Inverse Conductivity Problem for a Hexagonal Network

Brett Sovereign
Rice University

October 25, 1990

1 Introduction

1.1 Overview

In this paper I explain the procedure by which the conductivities of a symmetrical hexagonal network can be determined from measurements of currents on the boundary due to imposed voltages. The existence of a unique solution to this inverse conductivity problem for any discrete network is shown in Curtis and Morrow (1990). This particular class of networks could be useful in more accurately approximating the continuous problem for a circular surface. In the end I show some of the results of the reconstruction for the simplest cases.

1.2 The Network

Let Ω be a network of resistors (lines) and conductors (intersections of lines), resembling a hexagonal tiling of the plane.

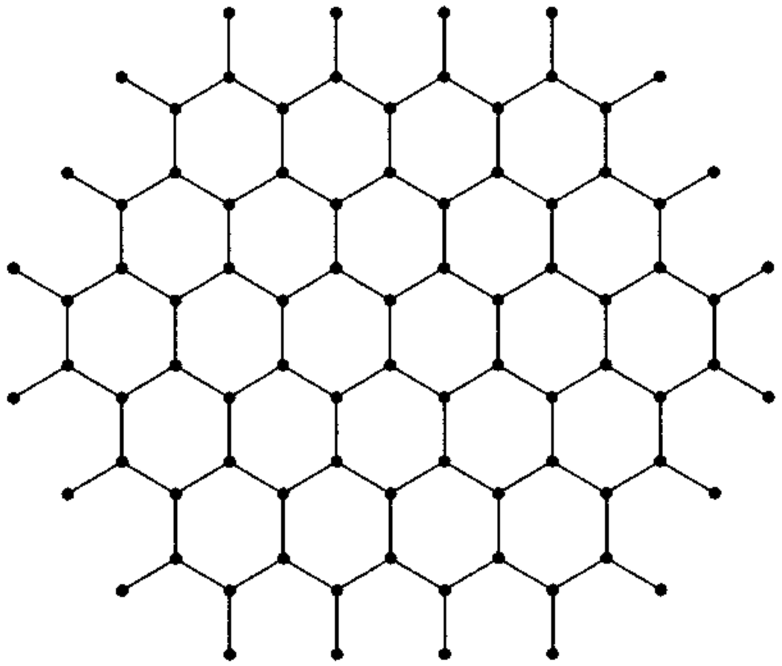


Figure 1

In this arrangement, each interior node is connected by three resistors to three nodes in the network, and the set of interior nodes is denoted by Ω_0. Two nodes are **adjacent** if they are connected by a single resistor. Each boundary node is connected by one resistor to one interior node, and the set is denoted by $\partial\Omega$. The set of resistors is denoted by Ω_1, and an individual resistor is often referred to by the nodes which it connects, such as PQ. A **corner** on the boundary of a hexagonal network occurs between two resistors on the boundary when they are connected to two interior nodes which are adjacent. A **side** of a boundary is the set of boundary nodes (and accompanying resistors) which lie between two corners, as in the below figure.

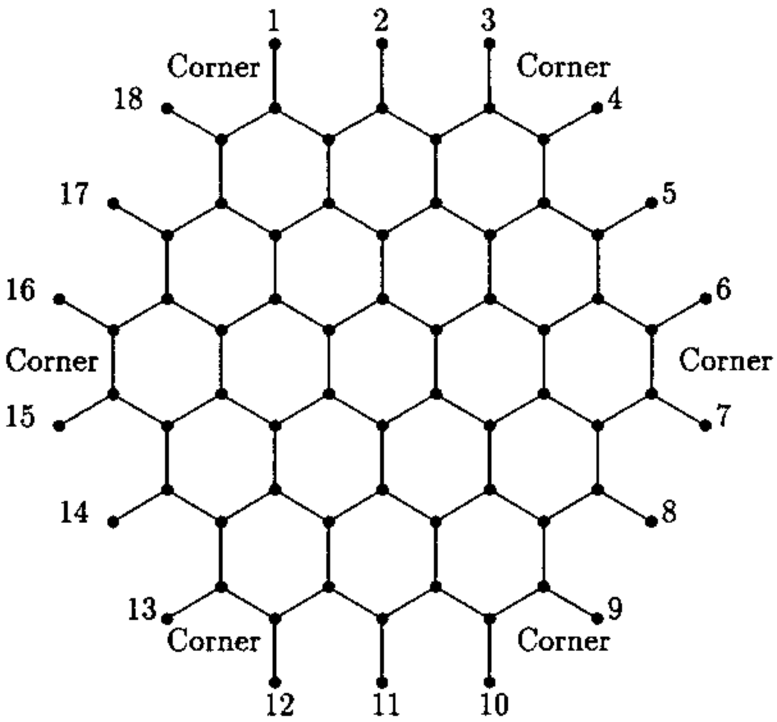
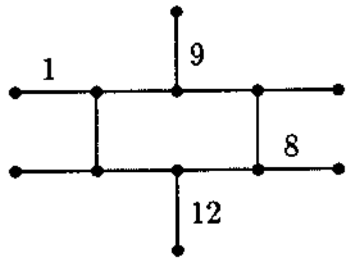


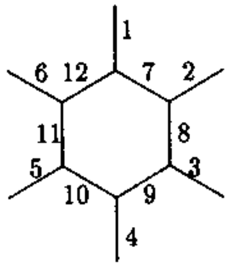
Figure 2

A **symmetrical hexagonal network** has exactly six sides, each with n boundary nodes. The size of these networks are completely determined by n, having $6 * n$ boundary nodes, $6 * n^2$ interior nodes, and $9 * n^2 + 3 * n$ total resistors. When discussing the network in general, there are several logical ways to number the nodes and resistors. For purposes of later discussion, the conductors can be broken into n

circular shells, with two distinct parts: the **spoke**, the circle of conductors radiating out from the center, and the **base**, the circle of conductors between the spokes. Shell numbering starts from the upper left spoke on the boundary and counts clockwise, and then moves to the base conductor to the immediate right and continues, moving into the center. Rectangular numbering considers the network as a modified rectangle, and numbers left to right, top to bottom, first the horizontal conductors, and then the vertical ones.



Rectangular Indexing



Shell Indexing

In general, the boundary nodes and resistors considered by themselves will be numbered clockwise from the upper left corner.

1.3 The Forward Problem

Each resistor PQ has a conductivity $\gamma(PQ) > 0$ associated with it, and each node has a real voltage $u(P)$ associated with it. Given a set of conductivities for our given network, we wish to find the linear mapping between voltages imposed on the boundary and the resulting currents, known as the Dirichlet to Neumann map. By

Ohm's law the current I through a given resistor PQ is given by

$$I(PQ) = \gamma(PQ)(u(p) - u(q))$$

Then for each interior node, Kirkhoff's Law applies

$$\sum_{Q \sim P} \gamma(PQ)(u(P) - u(Q)) = 0$$

Where $Q \sim P$ indicates that Q is adjacent to P.

This equation can be rewritten as

$$[\sum_{Q \sim P} \gamma(PQ)] * u(P) - \sum_{Q \sim P} (\gamma(PQ) * u(Q)) = 0$$

Thus the net current flow through any interior node is zero, and the $6 * n^2$ equations (one for each interior node of Ω_0) can be written as a matrix equation

$$Au = b$$

in which A is a $6 * n^2$ by $6 * n^2$ matrix with each diagonal entries being the sum of γ over the neighboring interior resistors, u the vector of voltages at each interior node, and b the vector of $\gamma(PQ) * u(Q)$ for boundary resistors PQ moved to the right hand side of the equation.

If $u(Q)$ for every boundary node Q then $u(P) = 0$ at all interior nodes P by the maximum principle (see Curtis and Morrow). This implies the non-singularity of A and so the above matrix equation has a unique solution. By setting A and b for the given conductivity and boundary potentials, we can solve for the interior potentials, and then calculate the currents at the boundary.

The linear mapping from voltages to currents is represented by the matrix Λ, which is dimension $6 * n$ by $6 * n$. $\Lambda(e_i)$ represents the currents across the boundary resistors due to a voltage of 1 at the ith boundary node and zero elsewhere. Thus $\Lambda_{i,j}$ is the current flow through boundary resistor i due to a voltage of 1 at boundary node j.

2 The Inverse Problem

2.1 The Algorithm

Now we work from the other direction: given the Dirichlet to Neumann map (in the form of a matrix Λ), we wish to calculate the conductivity throughout the network. By rewriting Kirkhoff's Law once again

$$(\sum_{Q\sim P} \gamma(PQ))u(P) = \sum_{Q\sim P} \gamma(PQ)u(Q)$$

we see that Kirkhoff's Law is a weighted average. Since $\gamma(PQ) > 0$, all four terms of this equation are non-zero. Given any three of the terms, the fourth follows automatically.

Given the following boundary information:

1. The values of $u(P)$ on sides 1–6:
2. The values of $\gamma\frac{\partial u}{\partial n}$ (the currents) on sides 5 and 6.

There is a unique solution u that satisfies Kirkhoff's Law with this information. Simply use the 4-point formula and work from sides 5 and 6.

In particular, setting u to zero on sides 1,4–6, except for $u(P_n) = 1$ on side 1 and $\gamma\frac{\partial u}{\partial n} = 0$ on sides 5 and 6 gives you a unique solution u that is zero up to the dotted line on the figure

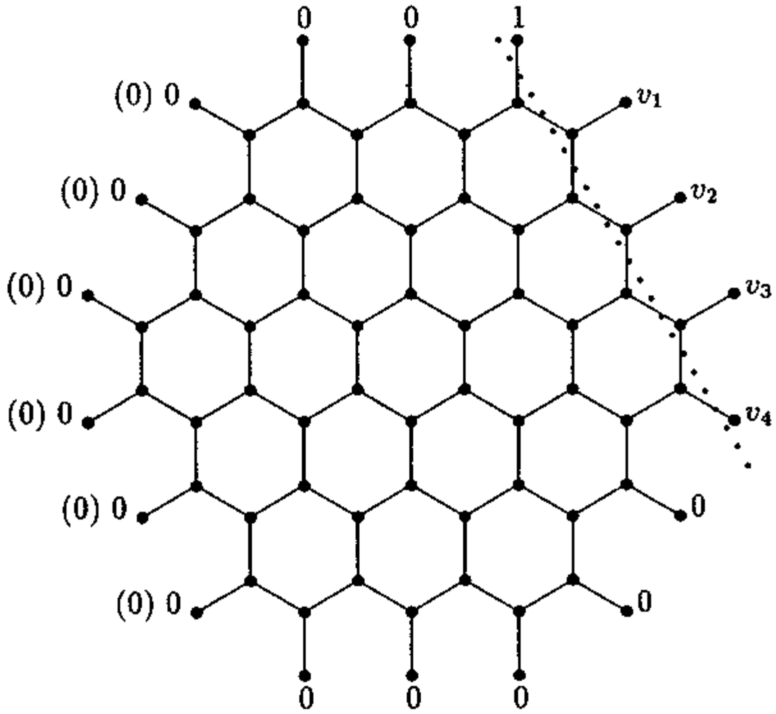


Figure 3

In general, there must be a series of voltages v_i on side 2 and part of side 3 such that

$$u(P_i) = 1$$
$$u(P_j) = v_i$$

For P_j located on sides 2 and 3 and above the dotted line The values v_i can be calculated from the overdetermined system:

$$\Lambda(e_3) + v_1 * \Lambda(e_4) + v_2 * \Lambda(e_5) + v_3 * \Lambda(e_6) + v_4 * \Lambda(e_7) = 0$$

for boundary nodes 13–18 (sides 5 and 6). A unique solution for the v_i can be obtained by selecting from the above system the equations for side 5, and the adjacent boundary node on side 6. The uniqueness is shown by a similar argument to the rectangular case (see [Curtis and Morrow])

Once the right-hand voltages are determined, the currents are calculated by

$$\Lambda(e_3) + v_1 * \Lambda(e_4) + v_2 * \Lambda(e_5) + v_3 * \Lambda(e_6) + v_4 * \Lambda(e_7) = C_i$$

this time at the boundary nodes with a non-zero voltage. Recalling the equation for current through a resistor, we can now determine the conductors γ_1 and γ_2, associated with P_3 and P_7 respectively, in the following way:

$$C_1 = (1 - 0) * \gamma_1$$
$$C_5 = (v_4 - 0) * \gamma_2$$

By rotating the figure counterclockwise so that side 2 becomes side 1, and repeating the above process, we calculate the first and third conductor on each side. The middle conductor on each side is found by changing the position of the 1 voltage to the middle of side 1, and solving for the (now five) voltages on the left side.

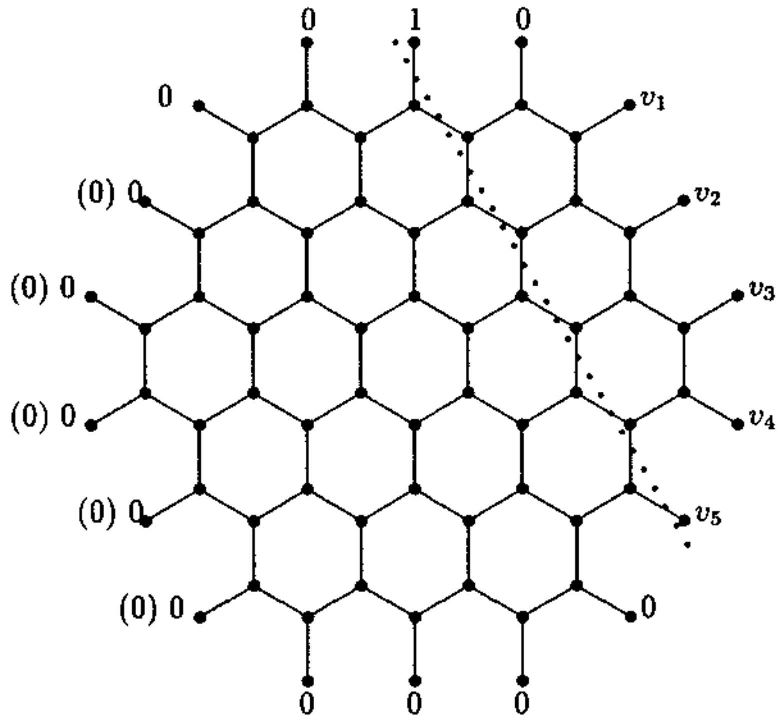


Figure 4

In this way, the boundary conductors are calculated.

We return to the initial conditions, and using our knowledge of the boundary conductors, calculate the base.

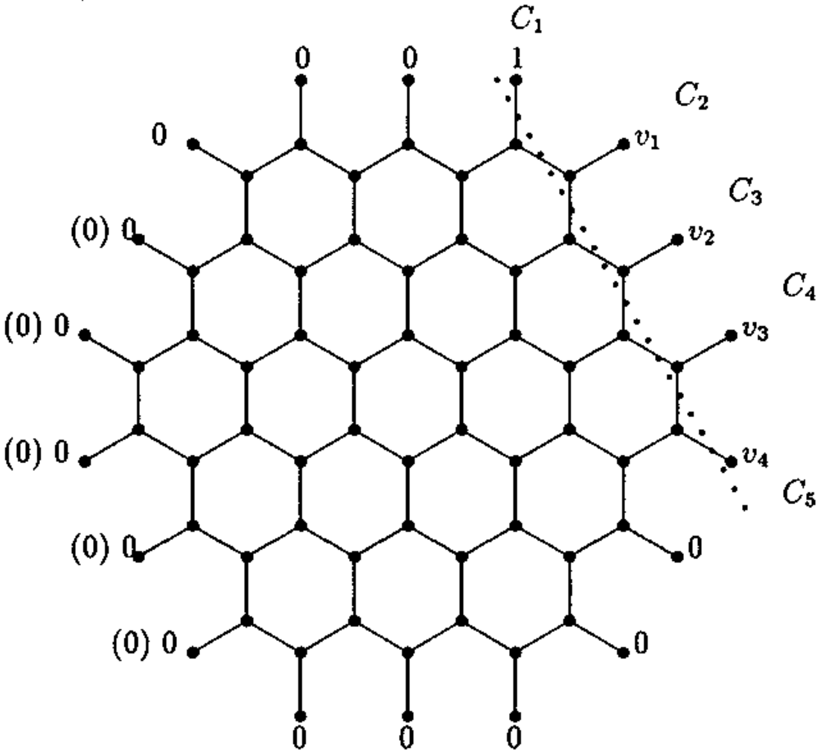


Figure 5

The interior shells of conductors are found in a similar fashion, using previously calculated information to determine the spokes, rotating until all spokes are found, and then determining the base. The procedure for determining the xth shell in an n-symmetrical hexagonal network is the same as calculating the boundary shell for an x-symmetrical hexagonal network. In order to put a non-zero voltage on the ith node of the xth shell, we place a 1 voltage on the (i)th boundary node of the entire network, and calculate inwards.

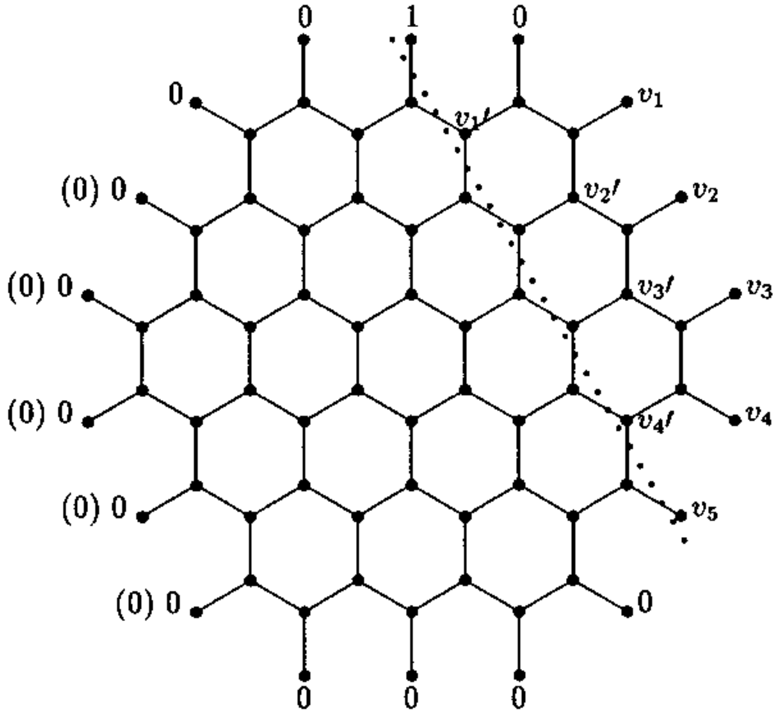


Figure 6

2.2 Some Results

The algorithm was programmed for the two simplest cases, the 1- and 2-hexagon networks. Given an initial set of conductances, a lambda matrix was generated in double precision (14 decimal places) which was then used to recalculate the γ's, and the deviation from the given conductance was found. For $\gamma = 1$ for all conductors, there was no measurale error for the 1-hex case, but for the 2-hex case there was an max error $\cong$ 1.1exp-13. For conductances which equalled the rectangular index number (γ(conductor No. 1)= 1, the error was $\cong 3.7\exp-13$ for 1-hex, and $5.4\exp-8$ for 2-hex. With given conductances which equalled the shell index, the error was $\cong 1.3\exp-12$ for 1-hex, and $4.4\exp-12$ for 2-hex.

References

[Curtis and Morrow] Edward Curtis and James Morrow, "The Dirichlet to Neumann Map for a Resistor Network"

A Programs

```
c  Forward solver for symmetric hexagonal network
c      File notation
c      side, spiral$ (=y shell index) and g array = user specified
c      Lambda matrix (for inspection)= user specified
c      Lambda matrix (for inverse program)= fort.total cond
c      Conductances (for comparison in inverse)= fort.total cond +1
       implicit double precision (a-h,o-y)
       parameter(mside=3,mlda=54,mbsize=18,mrsize=90)
       integer side,tnode,tcond,iband
       character*16, filename,xy,spiral
       double precision a(mlda,mlda),g(mrsize+1),abe(mlda,mlda)
       integer ipvt(24)
       double precision lambda(mbsize,mbsize),pot(mlda)
       integer bn(mbsize),bcond(mbsize),rsize,bounnode
c read conductivities into g
       print *,'keyboard or file?'
       read *, xy
       if (xy.ne.'f') then
          goto 10
       end if
       print *,'input file name?'
       read *,filename
       open(unit=15,file=filename,status='old')
       read (15,*) side,spiral
```

```
      if (side.gt.mside) then
         print *,'case too large'
         stop
      end if
      tcond=3*side*(3*side+1)
      do 5 i=1,tcond
         read (15,*) g(i)
5     continue
      goto 20
10    print *,'number of boundary nodes on a side?'
      read *,iside
      if (iside.gt.mside) then
         print *,'case too large'
         stop
      end if
      side=iside
      if =iside
      tcond=3*side*(3*side+1)
      print *,'spiral or rectangular?'
      read *,xy
      if (xy.eq.'s') then
         spiral='y'
      else
         spiral='n'
      end if
      print *,'constant, increasing,  or specified?'
      read *,xy
      if (xy.eq.'i') then
         do 12 i=1,tcond
            g(i)=i
12       continue
         goto 20
```

```
      else if (xy.eq.'c') then
         do 13 i=1,tcond
            g(i)=1
 13      continue
         goto 20
      end if
 15   do 17 i=1,tcond
         print *,'conductor #',i
         read *,g(i)
 17   continue
c convert to rectangular indexing if spiral
 20   if (spiral.eq.'y') then
         call convert(g,side,0)
      end if
      print *,'output filename?'
      read *,filename
      g(mrsize+1)=0
      call  setmat(side,g,a)
      rsize=2*side*(3*side+1)
      tnode=6*side**2
c put a matrix into band storage
      iband=4*side-1
      do 110 i=1,tnode
         j1=max0(1,i-iband)
         j2=min0(tnode,i+iband)
         do 100 j=j1,j2
            k=j-i+iband+1
            abe(i,k)=a(i,j)
 100     continue
 110  continue
c factor banded matrix
      lda=mlda
```

```
      call dnbfa(abe,lda,tnode,iband,iband,ipvt,info)

c set up boundary node-resistor/interior node link

      do 200 y=1,6
         do 190 x=1,side
            bounnode=(y-1)*side+x
            bcond(bounnode)=0
            bn(bounnode)=0
            if (y.eq.1) then
               bn(bounnode)=2*x
               bcond(bounnode)=rsize+x
            else if (y.eq.4) then
               bn(bounnode)=tnode-2*x+1
               bcond(bounnode)=tcond-x+1
            else if (y.eq.2) then
               do 120 i=1,x
                  bn(bounnode)=bn(bounnode)+(2*side+2*I-1)
                  bcond(bounnode)=bcond(bounnode)+(2*side+2*I-1)+1
 120           continue
            else if (y.eq.3) then
               bn(bounnode)=tnode/2
               bcond(bounnode)=rsize/2
               do 130 i=1,x
                  bn(bounnode)=bn(bounnode)+4*side+1-2*i
                  bcond(bounnode)=bcond(bounnode)+4*side+2-2*i
 130           continue
            else if (y.eq.5) then
               bn(bounnode)=tnode+1
               bcond(bounnode)=rsize+1
               do 140 i=1,x
                  bn(bounnode)=bn(bounnode)-(2*side+2*i-1)
```

```
               bcond(bounnode)=bcond(bounnode)-(2*side+2*i-1)-1
 140        continue
          else if (y.eq.6) then
            bn(bounnode)=tnode/2+1
            bcond(bounnode)=rsize/2+1
            do 150 i=1,x
               bn(bounnode)=bn(bounnode)-(4*side+1-2*i)
               bcond(bounnode)=bcond(bounnode)-(4*side+2-2*i)
 150        continue
          end if
 190    continue
 200  continue
c
      iband=4*side-1
      do 210 i=1,6*side
c     initialize right vectors
         do 205 j=1,tnode
            pot(j)=0
 205     continue
c     set right hand side for bounnode=i
         inode=bn(i)
         icond=bcond(i)
         pot(inode)=g(icond)
c     solve for interior potentials
         call dnbsl(abe,lda,tnode,iband,iband,ipvt,pot,0)
c     find lambda matrix
         do 207 k=1,6*side
            knode=bn(k)
            kcond=bcond(k)
            if (k.eq.i) then
               lambda(i,i)=g(kcond)*(1-pot(knode))
            else
```

```
              lambda(k,i)=-g(kcond)*pot(knode)
           end if
 207     continue
 210  continue
c     print conductances
      if (filename.eq.'skip') then
         goto 321
      end if
      open (unit=16,file=filename,status='new')
      write (16,*)
      write (16,410)'nodes on a side=',side
      write (16,410)'total conductors=',tcond,'total nodes=',tnode
      write (16,410)'assigned conductances (rectangular indexing)'
      write (16,*)
      write (16,400)(g(i),i=1,tcond)
      write (16,*)
c     print lambda matrix
      print *,'print lambda matrix?'
      read *,xy
      if (xy.eq.'n') then
         goto 321
      end if
 310  call prmatr(lambda,18,18,6*side,6*side)

c     print 'pure' lambda matrix for inverse problem
 321  if1=tcond
      do 230 i=1,6*side
         do 240 j=1,6*side
            write (if1,*)lambda(j,i)
 240     continue
 230  continue
 400  format(1p,5(d20.14,1x))
```

```
 410  format(4x,a,i3,2x)
      stop
      end



      SUBROUTINE SETMAT(side,g,a)

      integer side,node,nn,en,wn,sn,nc,sc,ec,wc
      integer rowsize,rsize,x,y
      double precision g(91),a(54,54)


      rsize=2*side*(3*side+1)

c top half of network
      do 70 y=1,side
c     number of columns=rowsize
         rowsize=2*side+2*y-1
c     column=x
         do 65 x=1,rowsize
c set boundary flags
            inw=0
            ine=0
            inf=0
c     locate node(x,y)
            node=0
            do 10 i=1,y-1
               node=node+(2*side+2*i-1)
 10         continue

            node=node+x
```

```
c             write (20,*)'node=',node
c check if x is odd or even
            if (mod(x,2).ne.0) then
               goto 20
            end if
c
c even x  north node/conductor
            sc=0
c     flag if row 1 since on boundary
            if (y.eq.1) then
               inf=-1
            end if
c     find north node
            nn=node-(rowsize)+1
c     find north conductor
            nc=rsize
            do 15 i=1,y-1
               nc=nc+(side-1+i)
 15         continue
            nc=nc+(x/2)
c     assign value to a matrix (except if flagged)
            if (inf.ne.-1) then
               if (nc.eq.0) then
                  a(node,nn)=0
               end if
               a(node,nn)=-g(nc)
            end if
            goto 30
c
c odd x  south node/conductor
 20         nc=0
c     determine if middle row
```

```
            if (y.eq.side) then
c     find node if middle
               sn=node+(rowsize)
c     find if is not middle
            else
               sn=node+2*side+2*y
            end if
c     find south conductor
            sc=rsize
            do 25 i=1,y
               sc=sc+side-1+i
 25         continue
            sc=sc+(x+1)/2
c     assign value to a matrix
            if (sc.eq.0) then
               a(node,sn)=0
            goto 30
            end if
            a(node,sn)=-g(sc)
c
c east conductor/node
 30         ec=0
c     flag if last node on row
            if (x.eq.(rowsize)) then
               ine=-1
            end if
c     find east node
            en=node+1
c     find east conductor
            do 35 i=1,(y-1)
               ec=ec+(2*side+2*i-1)+1
 35         continue
```

```
            ec=ec+x+1
c     assign value to a matrix (skip if flagged)
            if (ine.ne.-1) then
               if (ec.eq.0) then
                  a(node,en)=0
               end if
               a(node,en)=-g(ec)
            end if
c
c west conductor/node
 40         wc=0
c     flag if first node on row
            if (x.eq.1) then
               inw=-1
            end if
c     find west node
            wn=node-1
c     determine if last node on row
            if (x.eq.(rowsize)) then
               goto 45
            else
c     if F, then simple calculation of west conductor
               wc=ec-1
               goto 55
            end if
c     if T, then calculate west conductor
 45         do 50 i=1,(y-1)
c
               wc=wc+(2*side+2*i-1)+1
 50         continue
            wc=wc+x
c     assign value to a matrix (except if flagged)
```

```
 55         if (inw.ne.-1) then
               if (wc.eq.0) then
                  a(node,wn)=-g(wc)
               end if
               a(node,wn)=-g(wc)
            end if

c put diagonal entry into a
 60         if (ec.eq.0) then
               ec=91
            end if
            if (wc.eq.0) then
               wc=91
            end if
            if (nc.eq.0) then
               nc=91
            end if
            if (sc.eq.0) then
               sc=91
            end if
      a(node,node)=g(ec)+g(wc)+g(nc)+g(sc)
 65      continue
 70   continue

c bottom half of hexagonal network

      do 120 y=side+1,2*side
c     number of columns in row(y)=rowsize
         rowsize=6*side-2*y+1

         do 115 x=1,rowsize
c      write (20,*)'(x=',x,',y=',y,')'
```

```
c     set boundary flags to zero
            inf=0
            ine=0
            inw=0
            node=3*side**2
c     determine node(x,y)
            do 75 i=side+1,y-1
               node=node+(6*side-2*i+1)
 75         continue
            node=node+x
c      write (20,*)'node=',node
c     check to see if x is even or odd
            if ((mod(x,2)).ne.0) then
               goto 80
            end if
c even x south node/conductor
            nc=0
c     flag if last row since on boundary
            if (y.eq.(2*side)) then
               inf=-1
            end if
c     determine south node
            sn=node+(rowsize)-1
c     determine south conductor
            sc=rsize+side*(3*side-1)/2

            do 77 i=side+1,y
               sc=sc+3*side+1-i
 77         continue
            sc=sc+x/2
c     assign value to a matrix unless flagged
            if (inf.ne.-1) then
```

```
            a(node,sn)=-g(sc)
          end if
          goto 90

c odd x north node/conductor
 80       sc=0
c     determine if middle row or not
          if (y.gt.(side+1)) then
c     if F, determine north node
            nn=node-(rowsize)-1
c     if T, determine north node
          else
            nn=node-(rowsize)
          end if
c     determine north conductor
          nc=rsize+(3*side-1)*side/2
          do 85 i=side+1,y-1
            nc=nc+3*side+1-i
 85       continue
          nc=nc+(x+1)/2
c     assign value to a matrix
          a(node,nn)=-g(nc)
c east node/conductor
 90       ec=0
c     flag if on end of row
          if (x.eq.(rowsize)) then
            ine=-1
          end if
c     determine east node
          en=node+1
c     determine east conductor
          ec=side*(3*side+1)
```

```
          do 95 i=side+1,y-1
             ec=ec+(6*side-2*i+1)+1
 95       continue
          ec=ec+x+1
c     assign value to a matrix unless flagged
          if (ine.ne.-1) then
             a(node,en)=-g(ec)
          end if
c west node/conductor
          wc=0
c     flag if first node of row
          if (x.eq.1) then
             inw=-1
          end if
c     determine west node
          wn=node-1
c     determine if last node on row
          if (x.eq.(rowsize)) then
             goto 100
          else
c     if F, simple conductor calculation
             wc=ec-1
             goto 110
          end if
c     if T, calculate west conductor

 100      wc=side*(3*side+1)
          do 105 i=side+1,y-1
             wc=wc+(6*side-2*i+1)+1
 105         continue
          wc=wc+x
c     assign value to a matrix unless flagged
```

```
 110        if (inw.ne.-1) then
               a(node,wn)=-g(wc)
            end if
c put diagonal matrix entry
            if (ec.eq.0) then
               ec=91
            end if
            if (wc.eq.0) then
               wc=91
            end if
            if (nc.eq.0) then
               nc=91
            end if
            if (sc.eq.0) then
               sc=91
            end if
            a(node,node)=g(nc)+g(sc)+g(ec)+g(wc)
c move to next x
 115     continue
c move to next y
 120  continue
      return
      end


      SUBROUTINE prmatr(mat,maxrow,maxcol,row,col)

c     this subroutine prints out the elements of the matrix mat
c     with dimensions row by col

      implicit undefined(a-z)
      integer maxrow,maxcol,row,col
```

```
      double precision mat(maxrow,maxcol)
      integer i,j,k,i1,i2,block,l,space

c
      space=3
c
 20      format(a)
c
      block=int(col/5)+1
      if(mod(col,5).eq.0)block=block-1
      do 50 j=1,block
         i1=(j-1)*5+1
         if(j.eq.block.and.mod(col,5).ne.0)then
            i2=(j-1)*5+mod(col,5)
         else
            i2=j*5
         endif

            write(16,30)('column',i,i=i1,i2)
 30      format(6x,5(a,i11,4x))
         do 40 k=1,row
            if(i2.eq.j*5)then
               write(16,60)k,(mat(k,i),i=i1,i2),k
            else
               write(16,70)k,(mat(k,i),i=i1,i2)
            endif
 40      continue
         do 45 l=1,space
            write(16,*)
 45      continue
 50   continue
 60   format(i3,5(1x,d20.14),i3)
```

```fortran
 70   format(i3,5(1x,d20.14))
      stop
      end



      SUBROUTINE convert(g,side,dir)
c converts conductances from spiral indexing into rectangular form
c (dir=0) or vice-versa (dir=1)
      implicit double precision (a-h,o-z)
      integer scond2,rcond2,start,dir
      integer tcond, scond, rcond, thcond, i,j,x,y,sidemargin1
      integer wideside,side,shell,topmargin1,topmargin2,sidemargin2
      double precision g(3*side*(3*side+1)),h(200)
c transfer input into temporary array
      tcond=3*side*(3*side+1)
      do 10 i=1,tcond
         h(i)=g(i)
 10   continue
      thcond=2*tcond/3
c start from outside shell and work inwards
      do 20 shell=side,1,-1
         if (shell.eq.side) then
            start=0
            topmargin1=0
            topmargin2=0
            sidemargin1=0
            sidemargin2=0
         else
            start=start+3*wideside
            topmargin1=topmargin1+(side-shell)*2+2*side
            topmargin2=topmargin2+side+(side-shell-1)
```

```
          sidemargin1=sidemargin1+2
          sidemargin2=sidemargin2+1
        end if
        wideside=2*shell-1
c     work around shell first time (spokes first)
        do 30 y=1,6
          do 40 x=1,shell
            scond=start+(y-1)*shell+x
            rcond=0
            if (y.eq.1) then
              rcond=thcond+topmargin2+sidemargin2+x
            else if (y.eq.4) then
              rcond=tcond-topmargin2-sidemargin2-x+1
            else if (y.eq.2) then
              rcond=topmargin1
              do 50 j=1,x
                rcond=rcond+(2*shell+2*j-1)+2*sidemargin1+1
 50           continue
              rcond=rcond-sidemargin1
            else if (y.eq.6) then
              rcond=topmargin1
              tx=shell +1-x
              do 60 j=1,tx-1
                rcond=rcond+(2*shell+2*j-1)+2*sidemargin1+1
 60           continue
              rcond=rcond+sidemargin1+1
              else if (y.eq.3) then
                rcond=thcond/2
                do 70 j=1,x
                  rcond=rcond+(4*shell+1-2*j)+2*sidemargin1+1
 70             continue
                rcond=rcond-sidemargin1
```

```
                  else if (y.eq.5) then
                     rcond=thcond/2
                     tx=shell+1-x
                     do 80 j=1,tx-1
                        rcond=rcond+(4*shell+2-2*j)+2*sidemargin1
 80                  continue
                     rcond=rcond+sidemargin1+1
                  end if
c     do the conversion
                  if (dir.eq.0) then
                     g(rcond)=h(scond)
                  else
                     g(scond)=h(rcond)
                  end if
 30         continue
            start=start+6*shell
            do 90 x=1,wideside
               scond=start+x
               scond2=start+3*wideside+x
               rcond=topmargin1+sidemargin1+2+x
               rcond2=thcond-topmargin1-sidemargin1-x-1
               if (dir.eq.0) then
                  g(rcond)=h(scond)
                  g(rcond2)=h(scond2)
               else
                  g(scond)=h(rcond)
                  g(scond2)=h(rcond2)
               end if
 90         continue
            start=start+wideside
            do 100 x=1,wideside
               scond=start+x
```

```
      scond2=start+x+3*wideside
      if (mod(x,2).eq.1) then
         rcond=thcond+topmargin2
         do 110 j=1,(x-1)/2+2
            rcond=sidemargin2*2+shell +j-1+rcond
110      continue
         rcond=rcond-sidemargin2
         rcond2=tcond-rcond+thcond+1
      else
         rcond=topmargin1
         do 120 j=1,x/2+1
            rcond=2*sidemargin1+rcond+(2*shell+2*j)
120      continue
         rcond=rcond-sidemargin1-1
         rcond2=thcond-rcond+1
      end if
      if (dir.eq.0) then
         g(rcond)=h(scond)
         g(rcond2)=h(scond2)
      else
         g(scond)=h(rcond)
         g(scond2)=h(rcond2)
      end if
100 continue
    start=start+wideside
    do 130 x=1,wideside
      scond=start+x
      scond2=start+x+3*wideside
      if (mod(x,2).eq.1) then
         rcond=thcond/2
         do 140 j=1,(x-1)/2+1
            rcond=rcond+2*sidemargin1+(4*shell-2*j+2)
```

```
 140              continue
                  rcond=rcond-sidemargin1-1
                  rcond2=thcond-rcond+1
               else
                  rcond=thcond+side*(3*side+3)/2
                  do 150 j=1,x/2
                     rcond=rcond+2*sidemargin2+(2*shell-j)
 150              continue
                  rcond=rcond-sidemargin2
                  rcond2=tcond-rcond+thcond+1
               end if
               if (dir.eq.0) then
                  g(rcond)=h(scond)
                  g(rcond2)=h(scond2)
               else
                  g(scond)=h(rcond)
                  g(scond2)=h(rcond2)
               end if
 130        continue
            start=start+wideside
 20      continue
         return
         end



c Inverse solver for 1 hexagon case
      implicit double precision (a-h,o-y)
      double precision lambda(6,6),u(2),e(3),f(3,6),pot(6),
     $     v(2,6),g(12),h(12),diff(12)
      integer top,bnode(100),jnode(100,6),tnode
```

```
      character*16 filename,xy
c
c ask for filename
      tnode=6
      print *,'output filename?'
      read *,filename
c read lambda in from file
      do 10 i=1,6
         do 20 j=1,6
            read (12,*) lambda(j,i)
 20      continue
 10   continue
c take top=1 to 6 solve for exterior conductances
      do 30 top=1,6
         call getvolt(lambda,1,top,1,u,e,bnode)
         do 35 i=1,2
c             print *, u(i)
            v(i,top)=u(i)
 35      continue
         do 40 i=1,3
            f(i,top)=e(i)
            jnode(i,top)=bnode(i)
 40      continue
         g(jnode(1,top))=f(1,top)
 30   continue
c get interior conductances
      do 50 top=1,6
         pot(top)=v(1,top)-f(2,top)/g(jnode(2,top))
         g(top+6)=-f(1,top)/pot(top)
 50   continue
c convert into rectangular indexing
      call convert (g,1,0)
```

```
c compare with initial conductances
      ix=13
      do 300 i=1,12
         read (ix,*) h(i)
         diff (i)=dabs(h(i)-g(i))
 300  continue
c print conductances into file
      open (unit=15, file=filename,status='new')
      write (15,210) 'conductances (differences)'
      write (15,200) (g(i),diff(i),i=1,12)
      close (unit=15)
 200  format(1p,2(d20.14,1x,d20.14,3x))
 210  format(4x,a)

      stop
      end


c Inverse solver for 7 hexagon case
      implicit double precision (a-h,o-y)
      double precision g(42),lambda(12,12),u(4),v(4,6),e(6),f(6,6)
      double precision pot(8,6),h(42),diff(42)
      integer top,inode(100),jnode(100,6),knode(100,6)
      character*16 filename
c read from file
      do 10 i=1,12
         do 20 j=1,12
            read (42,*) lambda(j,i)
 20      continue
 10   continue
      print *, 'output file name?'
      read *,filename
```

```
c
      do 30 top=1,6
         call getvolt (lambda,2,top,2,u,e,inode)
         do 40 i=1,3
            v(i,top)=u(i)
 40      continue
         do 50 i=1,4
            f(i,top)=e(i)
 50      continue
         do 60 i=1,4
            jnode(i,top)=inode(i)
 60      continue
c     get exterior conductances
         g(inode(1))=f(1,top)
         g(inode(4))=f(4,top)/v(3,top)
 30   continue
c
      do 70 top=1,6
         pot(1,top)=v(1,top)-f(2,top)/g(jnode(2,top))
         pot(2,top)=v(2,top)-f(3,top)/g(jnode(3,top))
         do 80 i=1,3
            inode(i)=mod(3*top+i-2,18)+13
            knode(i,top)=inode(i)

 80      continue
         g(inode(1))=-f(1,top)/pot(1,top)
         g(inode(2))=-g(inode(1))+f(2,top)/pot(1,top)
         g(inode(3))=-pot(1,top)/pot(2,top)*g(inode(2))
 70   continue
      do 90 top=1,6
         call getvolt(lambda,2,top,1,u,e,inode)
         do 100 i=1,6
```

```
           f(i,top)=e(i)
 100     continue
         do 110 i=1,5
           v(i,top)=u(i)
 110     continue
         ihg=3*(top-1)+13
         pot(1,top)=-f(1,top)/g(ihg)
         pot(2,top)=-f(2,top)/g(2*top)
         ihh=ihg+1
         g(30+top)=-g(ihg)+(pot(2,top)-pot(1,top))*g(ihh)
     $          /pot(1,top)
         pot(3,top)=v(1,top)-f(3,top)/g(jnode(2,top))
         pot(5,top)=pot(3,top)+(-f(3,top)+(pot(3,top)-pot(2,top))*
     $          g(12+3*top))/g(knode(2,top))
         pot(6,top)=v(2,top)-f(4,top)/g(jnode(3,top))
 90    continue
c get last conductances
       do 120 top=1,6
         igh=mod(top,6)+31
         pot(4,top)=pot(5,top)+((pot(5,top)-pot(3,top))*g(knode(2,top))+
     $          (pot(5,top)-pot(6,top))*g(knode(3,top)))/g(igh)
         g(36+top)=-pot(1,top)/pot(4,top)*g(30+top)
 120   continue
c  convert into rectangular coordinates
       call convert(g,2,0)
c  compare to original
       if1=43
       do 193 i=1,42
         read (if1,*) h(i)
         diff(i)=dabs(h(i)-g(i))
 193   continue
       open (unit=15,file=filename,status='new')
```

```
      write (15,210)'conductances (differences)'
      write (15,200)(g(i),diff(i),i=1,42)
      close(unit=15)
 200  format(1p,2(d20.14,3x,d20.14,1x))
 210  format(4x,a)
      stop
      end



      SUBROUTINE getvolt(lambda,side,top,loc,u,e,bnode)
c input lambda matrix, nodes on a side, which side one voltage is imposed
c which node on that side is one (loc=1-side)
c output voltage (u) and currents
      implicit undefined (a-z)
      integer i,top,loc, band,j,side,wideband,topband
      integer gnode(100),bnode(100),ipvt(100),info
      double precision lambda(6*side,6*side),e(3*side-2*loc+2)
      double precision a(100,100),u(2*side-loc+1)
      character*16 xy
c
      topband=side-loc+1
      band=2*side-loc+1
      wideband=topband+band
c determine boundary nodes involved
      do 5 i=1,topband
         bnode(i)=(top-1)*side+loc+i-1
c         print *,bnode(i)
 5    continue
      do 10 i=1,band
         bnode(i+topband)=mod(top*side+i-1,6*side)+1
         gnode(i)=mod(bnode(i+topband)+3*side-1,6*side)+1
```

```
c         print *,bnode(i+topband),gnode(i)
 10   continue
c      read *,xy
      do 20 i=1,band
         do 30 j=1,band
            a(i,j)=lambda(gnode(i),bnode(j+topband))
 30      continue
         u(i)=-lambda(gnode(i),bnode(1))
 20   continue
c factor a matrix
      call dgefa(a,100,band,ipvt,info)
c solve for voltages
      call dgesl(a,100,band,ipvt,u,0)
c get exterior currents
      do 40 i=1,wideband
         e(i)=lambda(bnode(i),bnode(1))
         do 50 j=1,band
            e(i)=e(i)+u(j)*lambda(bnode(i),bnode(j+topband))
 50      continue
 40   continue
      return
      end
```