EXPLORATIONS

Chapter 11

Features Used

3-D Graphing, **deSolve( ), expand( ), tExpand( ), solve( ), limit( ), colDim( ), rowDim( ), Func, For, EndFor, newMat( ), @n1, @1, ., +**

Setup

◆ 1, setFold(emag)

Electromagnetics

This chapter describes how to use the TI-89 to solve Laplace's equation for two-dimensional electrostatic problems. The separation of variables method provides exact solutions but is restricted to specific geometries. The relaxation method is applicable to any geometry, but it provides only an approximate solution. This chapter uses the function **lap( )** which is defined in *Chapter 10: Vector Calculus.*

Topic 52: Separation of Variables

Electrostatic potentials satisfy Laplace's equation, $\nabla^2 v=0$, in charge-free regions. A typical two-dimensional problem is shown in Figure 1.

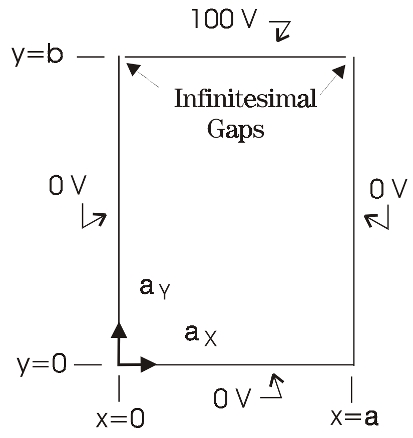


Figure 1. Geometry for 2-D electrostatic problem

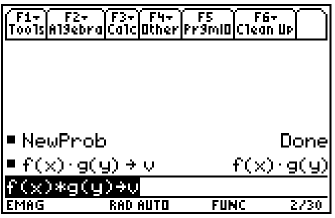
In two-dimensional, rectangular geometries, the potential varies with x and y, that is, v=v(x,y). The assumption that v is the product of two functions, each of which depending upon only a single variable, leads to v=f(x)g(y).

1. Clear the TI-89 by pressing 2nd F6 **2:NewProb** ENTER.

2. Enter the expression for **v** as shown in screen 1.

f (**x**) × **g** (**y**) STO▸ **v**

(1)



```
■ NewProb                         Done
■ f(x)·g(y) → v              f(x)·g(y)
f(x)*g(y)→v
EMAG        RAD AUTO     FUNC     2/30
```

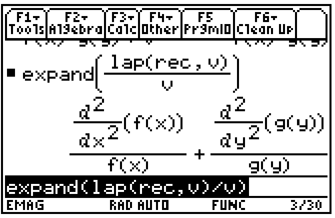
3. Applying the Laplacian operator to **v** and dividing the result by **v** leads to

$$\frac{\nabla^2 v}{v} = \frac{\frac{\partial^2 f(x)}{\partial x^2}}{f(x)} + \frac{\frac{\partial^2 g(y)}{\partial y^2}}{g(y)} = 0$$

Enter the expression using the function **lap** from Topic 49 (Chapter 10) as shown in screen 2.

CATALOG **expand(lap** (**rec** , **v**) ÷ **v**)

(2)



■ expand$\left(\frac{\text{lap(rec, v)}}{v}\right)$

$$\frac{\frac{d^2}{dx^2}(f(x))}{f(x)} + \frac{\frac{d^2}{dy^2}(g(y))}{g(y)}$$

```
expand(lap(rec,v)/v)
EMAG        RAD AUTO     FUNC     3/30
```

4. Since the first term depends only on **x** and the second only on **y** with their sum equal to zero, both terms must be constant. When the x-dependent term is set equal to the constant -k^2, the result is the differential equation

$$\frac{\partial^2 f(x)}{\partial x^2} + k^2 f(x) = 0$$

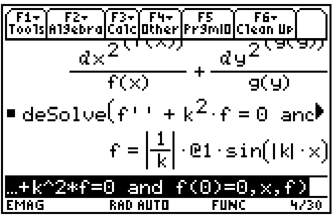
Use **deSolve()** to enter this equation with the boundary condition f(0)=0.

CATALOG **deSolve(f** 2nd [′] 2nd [′] + **k** ^ **2** × **f** = **0**
CATALOG **and f** (**0**) = **0** , **x** , **f**)

Screen 3 shows the output form is f(x)=$\left|\frac{1}{k}\right|$sin(|k|x)

times the constant $\left|\frac{1}{k}\right|$. (The constant does not affect the solution.)

(3)



$$\frac{dx^2}{f(x)} + \frac{dy^2}{g(y)}$$

■ deSolve(f'' + k^2·f = 0 and▸

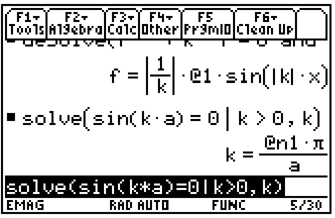
$$f = \left|\frac{1}{k}\right| \cdot @1 \cdot \sin(|k| \cdot x)$$

```
…+k^2*f=0 and f(0)=0,x,f)
EMAG        RAD AUTO     FUNC     4/30
```

5. This equation also equals zero at the other boundary, or f(a)=sin(ka)=0. Find the allowed values of **k** to be **k**=nπ/a as shown in screen 4.

CATALOG **solve(** 2nd [SIN] **k** × **a**) = **0** | **k** 2nd [>] **0** , **k**)

(4)



$$f = \left|\frac{1}{k}\right| \cdot @1 \cdot \sin(|k| \cdot x)$$

■ solve(sin(k·a) = 0 | k > 0, k)

$$k = \frac{@n1 \cdot \pi}{a}$$

```
solve(sin(k*a)=0|k>0,k)
EMAG        RAD AUTO     FUNC     5/30
```

***Note**: @1 represents an arbitrary integer. The screen may show @2, @3, and so forth, if **deSolve()** has been executed prior to this.*

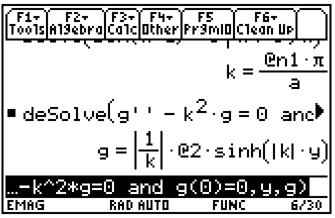
6. When the first term in the separated form of Laplace's equation equals -k^2, the second term in the equation must equal k^2. This gives a second differential equation

$$\frac{\partial^2 g(y)}{\partial y^2} - k^2 g(y) = \frac{\partial^2 g(y)}{\partial y^2} - \left(\frac{n\pi}{a}\right)^2 g(y) = 0$$

which is entered as shown in screen 5.

CATALOG **deSolve(g** 2nd [′] 2nd [′] − **k** ^ **2** × **g** = **0**
CATALOG **and g** (**0**) = **0** , **y** , **g**)

(5)



```
k = @n1·π / a
■ deSolve(g'' - k²·g = 0 and▸
g = |1/k|·@2·sinh(|k|·y)
…-k^2*g=0 and g(0)=0,y,g)
```

The solution is of the form g(y)= $\left|\frac{1}{k}\right|$sinh(ky) times the constant $\left|\frac{1}{k}\right|$.

7. Since **k**=nπ/a from the boundary conditions on f(x), g(y)=sinh(nπy/a). But it is not possible for this solution of g(y) to satisfy the upper boundary condition for arbitrary y=b (see top of Figure 1). Since the solutions for each **n** satisfy Laplace's equation, a linear combination of solutions must satisfy it also; such a solution is

$$v(x,y) = \sum_{n=1}^{\infty} c_n \sin\left(\frac{n\pi x}{a}\right)\sinh\left(\frac{n\pi y}{a}\right)$$

At the upper boundary where y=b, this takes the form of a Fourier series in x

$$v(x,b) = 100 = \sum_{n=1}^{\infty} c_n \sin\left(\frac{n\pi x}{a}\right)\sinh\left(\frac{n\pi b}{a}\right)$$

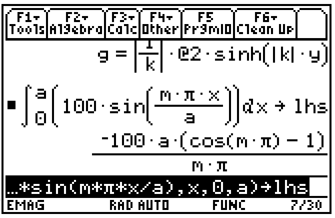
The constant c$_n$ is determined by usual Fourier techniques. Both sides of the equation are multiplied by sin(mπx/a) and integrated over the range 0≤x≤a. The left-hand side (LHS) integral becomes

$$\text{LHS} = \int_{x=0}^{a} 100\sin\left(\frac{m\pi x}{a}\right)dx = \frac{100a(1-\cos m\pi)}{m\pi}$$

as shown in screen 6.

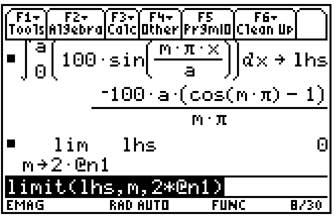
2nd [∫] **100** 2nd [SIN] **m** 2nd [π] **x** ÷ **a**) , **x** , **0** , **a**)
STO▸ **lhs**

(6)



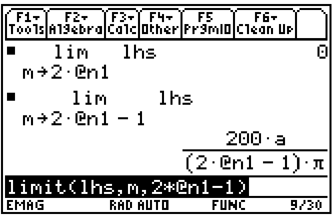
```
g = |1/k|·@2·sinh(|k|·y)
■ ∫₀ᵃ(100·sin(m·π·x / a))dx → lhs
-100·a·(cos(m·π) - 1) / m·π
…*sin(m*π*x/a),x,0,a)→lhs
```

8. When m=2@n2 is even, **lhs** = 0 (screen 7). When m=2@n2-1 is odd, **lhs** $= \dfrac{200a}{m\pi}$ (screen 8).



```
■ ∫[0,a](100·sin(m·π·x / a)) dx → lhs
                 -100·a·(cos(m·π) - 1)
                 ─────────────────────
                         m·π
■    lim     lhs                     0
  m→2·@n1
limit(lhs,m,2*@n1)
EMAG        RAD AUTO     FUNC     8/30
```

(7)



```
■    lim     lhs                     0
  m→2·@n1
■      lim      lhs
  m→2·@n1 - 1
                          200·a
                      ─────────────
                      (2·@n1 - 1)·π
limit(lhs,m,2*@n1-1)
EMAG        RAD AUTO     FUNC     9/30
```

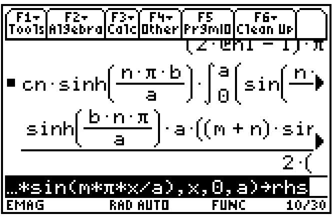
(8)

9. The right hand side integral, given as

$$\int_{x=0}^{a}\sum_{n=1}^{\infty} \mathrm{cn}\sinh\left(\frac{n\pi b}{a}\right)\sin\left(\frac{n\pi x}{a}\right)\sin\left(\frac{m\pi x}{a}\right)dx$$

requires more careful evaluation. An interchange of the summation and integration leads to a series of integrals, each with a different value of **n**

$$\sum_{n=1}^{\infty} \mathrm{cn}\sinh\left(\frac{n\pi b}{a}\right)\int_{x=0}^{a}\sin\left(\frac{n\pi x}{a}\right)\sin\left(\frac{m\pi x}{a}\right)dx$$



```
                             (2·@n1 - 1)·π
■ cn·sinh(n·π·b / a)·∫[0,a](sin(n·▸
   sinh(b·n·π / a)·a·((m + n)·sir▸
   ───────────────────────────────
                               2·(
…*sin(m*π*x/a),x,0,a)→rhs
EMAG        RAD AUTO     FUNC    10/30
```

(9)

The integral can be calculated directly by the TI-89 as shown in screen 9.

cn [×] [CATALOG] **sinh(n** [2nd] [π] **b** [÷] **a** [)] [2nd] [∫] [2nd] [SIN] **n** [2nd] [π] **x** [÷] **a** [)] [×] [2nd] [SIN] **m** [2nd] [π] **x** [÷] **a** [)] [,] **x** [,] **0** [,] **a** [)] [STO▸] **rhs**

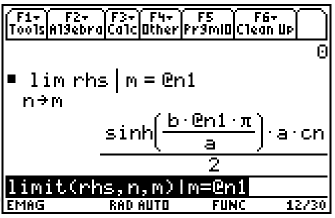
Evaluation of **rhs** for **n≠m** requires the **tExpand** command for the trigonometric functions only; in addition **n=@n1**=integer and **m=@n2**=integer. Enter the following key strokes for this evaluation.

[CATALOG] **sinh(n** [2nd] [π] **b** [÷] **a** [)] **a** [×] [(] [(] **m** [+] **n** [)] [×] [CATALOG] **tExpand(** [2nd] [SIN] **m** [2nd] [π] [−] **n** [2nd] [π] [)] [)] [−] [(] **m** [−] **n** [)] [×] [CATALOG] **tExpand(** [2nd] [SIN] **m** [2nd] [π] [+] **n** [2nd] [π] [)] [)] [)] [×] **cn** [÷] [(] **2** [×] [(] **m** [+] **n** [)] [×] [(] **m** [−] **n** [)] [2nd] [π] [)] [|] **n** [=] [◆] [STO▸] **n1** [CATALOG] **and m** [=] [◆] [STO▸] **n2.**

10. The result shows RHS=0 when **n≠m** (top of Screen 10).

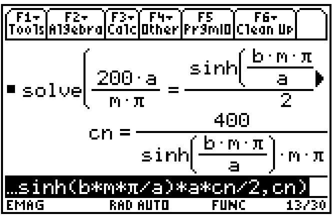
For the case **n=m** the evaluation is much simpler as shown in screen 10.

[CATALOG] **limit(rhs , n , m** [)] [|] **m** [=] [◆] [STO▸] **n1**



```
                                      0
■ lim rhs | m = @n1
  n→m
              sinh(b·@n1·π / a)·a·cn
              ──────────────────────
                        2
limit(rhs,n,m)|m=@n1
EMAG        RAD AUTO     FUNC    12/30
```

(10)



```
■ solve(200·a/(m·π) = sinh(b·m·π/a ▶ /2
                    cn = 400/(sinh(b·m·π/a)·m·π)
…sinh(b*m*π/a)*a*cn/2,cn)
EMAG          RAD AUTO        FUNC       13/30
```

(11)

11. The constant **cn** is determined by equating the evaluations of **lhs** and **rhs** of previous calculations (screen 11).

    CATALOG **solve(200a** ÷ (**m** 2nd [π]) = CATALOG **sinh(b** × **m** × 2nd [π] ÷ **a**) × **a** × **cn** ÷ **2** , **cn**)

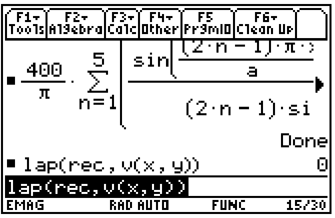
12. These results are combined to give the electrostatic potential within the region as

$$v(x,y) = \frac{400}{\pi}\sum_{n=odd}\frac{\sin\left(\frac{n\pi x}{a}\right)\sinh\left(\frac{n\pi y}{a}\right)}{n\sinh\left(\frac{n\pi b}{a}\right)}$$

    This is implemented with a finite number (5) of terms. Due to the TI-89's single step size of 1, the summation index **n** must be replaced by **2n-1** which takes on only odd integer values for consecutive integer values of **n**.

    Calculate the electrostatic potential as shown on the top of screen 12.

    400 ÷ 2nd [π] CATALOG **Σ(4** 2nd [SIN] (**2n** − **1**) 2nd [π] **x** ÷ **a**) CATALOG **sinh(** (**2n** − **1**) 2nd [π] **y** ÷ **a**) ÷ ((**2n** − **1**) CATALOG **sinh(** (**2n** − **1**) 2nd [π] **b** ÷ **a**)) , **n** , **1** , **5**) STO▸ **v** (**x** , **y**)



```
■ 400/π · Σ(n=1, 5) sin((2·n − 1)·π·▸/a) / ((2·n − 1)·si ▶
                                              Done
■ lap(rec, v(x, y))                              0
lap(rec,v(x,y))
EMAG          RAD AUTO        FUNC       15/30
```

(12)

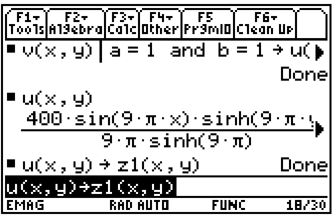
13. To check that this solution satisfies Laplace's equation, use the function **lap** from Topic 49 (screen 12).

    lap (**rec** , **v** (**x** , **y**))

    Observe (after awhile) that the result is zero, as it must be!

Topic 53: 3D Potential Graphs

How does **v(x,y)** from Topic 52 vary throughout the region? 3-D graphing provides such a view, but it requires specific values for **a** and **b**.



```
■ v(x, y) | a = 1 and b = 1 → u( ▶
                                  Done
■ u(x, y)
  400·sin(9·π·x)·sinh(9·π·y ▶
      9·π·sinh(9·π)
■ u(x, y) → z1(x, y)              Done
u(x,y)→z1(x,y)
EMAG          RAD AUTO        FUNC       18/30
```

(13)

1. For simplicity, let a=b=1 as shown in the top of screen 13.

   The result is stored as a new function **u(x,y)** by first defining **u(x,y)**.

   v (**x** , **y**) | **a** = **1** CATALOG **and b** = **1** STO▸ **u** (**x** , **y**)

   Then verify that the function is correct.

   u (**x** , **y**)

2. Define the graph function **z1(x,y)=u(x,y)** so that the plotting is faster than it would be by plotting **v(x,y)|a=1** and **b=1** (bottom of screen 13).

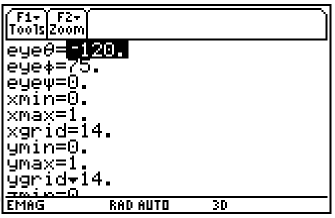
   u (**x** , **y**) STO▶ **z1** (**x** , **y**)

   The height **z** represents the magnitude of the function **v(x,y)**.

3. Press MODE and set **Graph** mode to**3D**. Use ◆ [WINDOW] to set the window variable values as shown in screens 14 and 15.

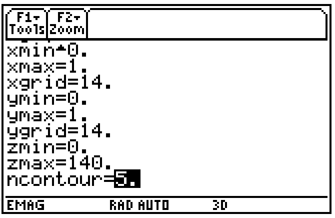
   eyeθ= ⁻120
   eyeϕ=75
   eyeψ=0
   xmin=0
   xmax=1
   xgrid=14

   ymin=0
   ymax=1
   ygrid=14
   zmin=0
   zmax=140
   ncontour=5



```
eyeθ=-120.
eyeϕ=75.
eyeψ=0.
xmin=0.
xmax=1.
xgrid=14.
ymin=0.
ymax=1.
ygrid↓14.
zmin=0
EMAG      RAD AUTO      3D
```

(14)

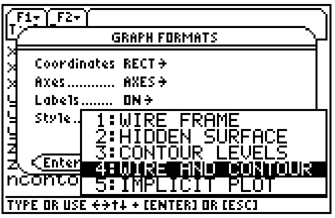


```
xmin↑0.
xmax=1.
xgrid=14.
ymin=0.
ymax=1.
ygrid=14.
zmin=0.
zmax=140.
ncontour=5.
EMAG      RAD AUTO      3D
```

(15)

4. From the Window Editor, use ◆ | to set the graph formats as shown in screen 16.

   Coordinates: RECT
   Axes: AXES
   Labels: ON
   Style: WIRE AND CONTOUR

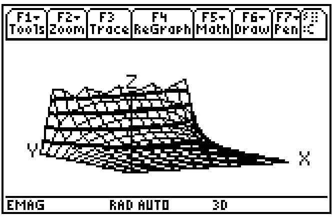


```
GRAPH FORMATS
Coordinates RECT→
Axes........ AXES→
Labels...... ON→
Style..  1:WIRE FRAME
         2:HIDDEN SURFACE
         3:CONTOUR LEVELS
         4:WIRE AND CONTOUR
         5:IMPLICIT PLOT
Enter
TYPE OR USE ←→↑↓ + [ENTER] OR [ESC]
```

(16)

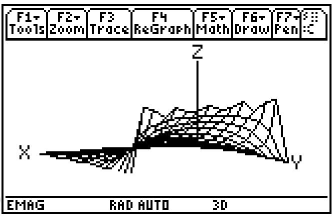
5. Press ◆ [GRAPH]. Wait a few minutes for the graph to be calculated. The results will look like screen 17.

   The voltage **v(x,y)** is graphed with equipotential contours every 20 volts. Once the calculations with the contours are made, the contours can be turned on and off.



(17)

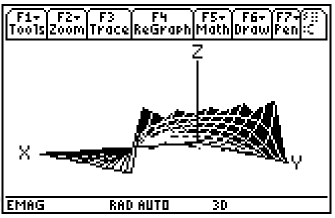
6. Press ◆ | and set **Style** to **WIRE FRAME** to turn off the contours. In the Window Editor, set **eyeθ**=60 and press ◆ [GRAPH] for a view from "behind" the graph, that is, from the region of **y**>1 (screen 18).



(18)

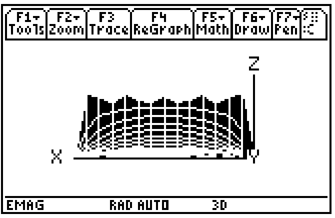
7. Press ◆ | and set **Style** to **HIDDEN SURFACE** to give a different nature to the graph (screen 19).

   The variation of the amplitude along the upper edge is due to the limited five-term Fourier Series approximation of the 100 volts. More terms would make this smoother.



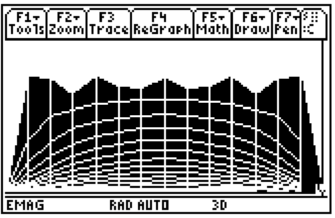
(19)

8. Press [Y] to look along the y-axis directly at these variations from the back of the graph (screen 20).



(20)

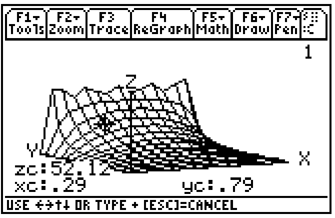
Use [×] to toggle between expanded and normal views (screen 21).



(21)

Use ◀ ▶ ▲ ▼ or change values in the Window Editor to change the viewing angle.

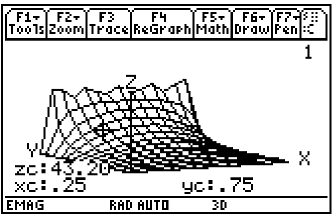
9. Return to the original viewing angle by pressing **0**. Then, in the Window Editor, set **eyeθ**=-120 and press [◆] [GRAPH]. To find the voltage at (x,y) coordinates, press [F3] **Trace** which places at cursor at the center of the x,y space and provides the corresponding z-value. Move the cursor ±y with ▲ ▼ and ±x with ◀ ▶. The voltage at x=.29 y=.79 is 52.12 V as shown in screen 22.



zc:52.12
xc:.29 yc:.79
USE ←→↑↓ OR TYPE + [ESC]=CANCEL

(22)

10. To see the voltage at a given position, just type the **x** and **y** coordinates. For example, type **0.25** [ENTER] **0.75** [ENTER] to see that the value at x=0.25 and y=0.75 is 43.20 V (screen 23).

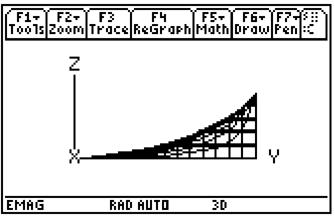


zc:43.20
xc:.25 yc:.75

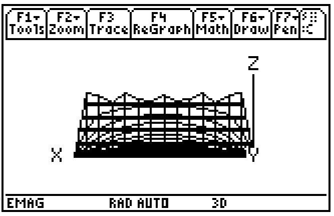
(23)

11. Press [◆] [I] and set **Style** to **WIRE AND CONTOUR** to display the equipotential contours on the wire frame.

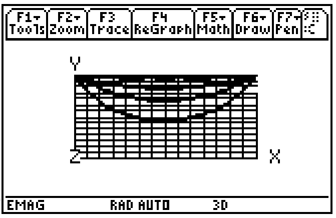
Press [X], [Y], and [Z] for views from those axes (screens 24, 25 and 26). The z-axis view (from above the graph) provides the standard view of the equipotentials on the x-y plane. However, [F2] **5:ZoomSqr** must be used to recompute the graph if the proper ratio of the **x** and **y** dimensions is required.



(24)



(25)



(26)

Topic 54: Relaxation Method

An alternate, but approximate, method which works for all geometries is known as the relaxation method. The procedure is simple: a solution for the voltage v(x,y) is guessed, an iterative set of approximate calculations operates upon this solution, and the “guessed” solution is modified by each successive set of calculations in a way which allows the guessed solution to “relax” to the correct solution.

Electrostatic voltages satisfy Laplace’s equation as $\nabla^2 v = 0$. Finite differences approximate the Laplacian by the central difference form as

$$\begin{aligned}\nabla^2 v &= \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \\ &\approx \frac{v_R + v_L - 2v_o}{\Delta x^2} + \frac{v_T + v_B - 2v_o}{\Delta y^2} \\ &= \frac{v_R + v_L + v_T + v_B - 4v_o}{\Delta x^2} = 0\end{aligned}$$

where Δx=Δy. For non-zero increments, Δx,Δy≠0, this equation is solved for the central node voltage as

$$v_o = \frac{v_R + v_L + v_T + v_B}{4}$$

This equation states that the voltage at the center node is equal to the average of the node voltages around it. This equation is valid at all nodes. The solution region is divided into rectangular grids, and the equation for v_o is applied successively to each node. This procedure is repeated until the node voltages approach the correct solution as a limiting value.

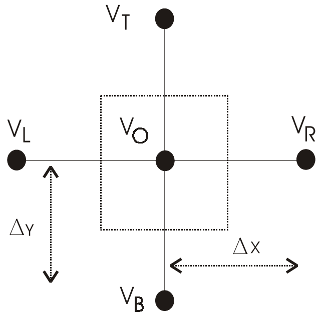


Figure 2. Finite difference cell

The solution region is divided into square grids. Each node is described by two integers, an x-coordinate and a y-coordinate. The x-coordinates are numbered from left to right; the y-coordinates are numbered from bottom to top. The nodes show a great similarity to the elements of a matrix in their geometric arrangement and in their numbering scheme. The storing of node voltages in a matrix provides a convenient and visual display of the voltages. The matrix row-column numbering scheme is used to simply and systematically apply the node voltage calculations to all of the cells. A second matrix is used to identify those boundary nodes at which the voltage is fixed and must not be changed by the calculations.

The TI-89 handles matrices and repetitive calculations very handily. The node voltages are stored in a matrix named **volt**; the geometry data is stored in a matrix named **geo**. Define a function called **relax** to perform the calculations.

```
:relax(v,geo,ntot)
:Func
:Local n,ntot,nrow,ncol
:Local nrowmax,ncolmax
:colDim(v)→ncolmax
:rowDim(v)→nrowmax
:For n,1,ntot
: For nrow,2,nrowmax-1
:   For ncol,2,ncolmax-1
:(v[nrow-1,ncol]+v[nrow+1,ncol]+v[nrow,ncol-1]+v[nrow,ncol+1])/4*geo[nrow,ncol]+v[nrow,ncol]*(1-
  geo[nrow,ncol])→v[nrow,ncol]
:   EndFor
: EndFor
:EndFor
:v
:EndFunc
```

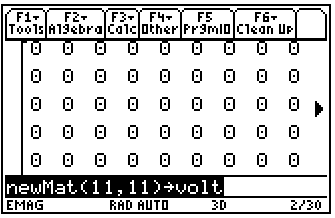
The function **relax** accepts the voltage matrix (**v**), the geometry matrix (**geo**), and the desired number of sets of repeated calculations (**ntot**) as arguments. For simplicity, all interior nodes of the voltage matrix are set to zero as the initial guess. Since the nodes on the edges of the matrix are fixed, the calculations are limited to interior points by the row and column sizes. The single instruction within the nested loops implements the discrete Laplacian at a node. The results of this calculation are multiplied by the corresponding element of the geometry matrix, 0 for nodes that are fixed or 1 for nodes that are modified by calculations. In addition, the nodal voltage is multiplied by 1−**geo** so that the original nodal voltage is stored in those nodes which are to remain fixed. This calculated node voltage is stored in **v**, and calculations move to the next node. With the completion of calculations, **v** is displayed on the Home screen.

1. Clear the TI-89 by pressing 2nd F6 **2:NewProb** ENTER.

2. For numeric results, the region of screen 27 is divided into 10x10 square grids resulting in 11x11 matrices for the voltage and geometry matrices. On the Home screen, create the voltage matrix filled with zeros (see screen 27).

   CATALOG **newMat(11** , **11**) STO▶ **volt**

(27)



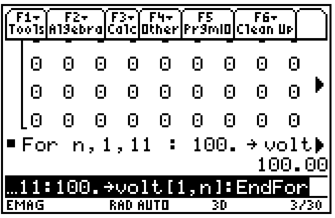
```
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 ▶
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
newMat(11,11)→volt
EMAG        RAD AUTO     3D       2/30
```

3. Set the 100 V boundary voltage.

   CATALOG **For n** , **1** , **11** 2nd [:] **100** . STO▶ **volt** 2nd [[] **1** , **n** 2nd []] 2nd [:] CATALOG **EndFor**

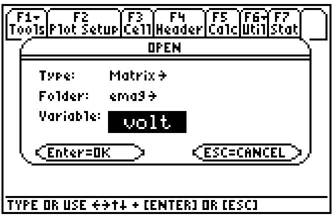
   Screen 28 shows that floating point results are returned because of the decimal point in 100.

(28)



```
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 ▶
0 0 0 0 0 0 0 0 0
■ For n,1,11 : 100. → volt▶
                      100.00
…11:100.→volt[1,n]:EndFor
EMAG        RAD AUTO     3D       3/30
```

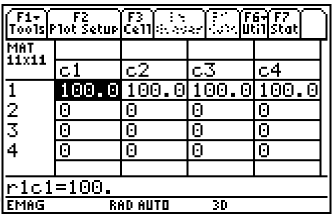
4. To verify that the matrix has been entered correctly, press [APPS] **6:Data/Matrix Editor 2:Open**. Specify **Type** as **Matrix**, **Folder** as **emag**, and **Variable** as **volt** (screens 29 and 30).

(29)


OPEN

Type: Matrix →
Folder: emag →
Variable: volt

TYPE OR USE ←→↑↓ + [ENTER] OR [ESC]

(30)


MAT 11x11	c1	c2	c3	c4
1	100.0	100.0	100.0	100.0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

r1c1=100.

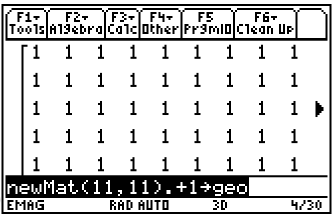
EMAG RAD AUTO 3D

***Note**: The 100's can be entered directly into this display instead of the entry line **For...EndFor** instruction.*

5. On the Home screen, create the geometry matrix filled with 1's (screen 31).

   [CATALOG] **newMat(11** [,] **11** [)] [.] [+] **1** [STO▸] **geo**

***Note**: The [.] [+] tells the TI-89 to do a dot addition. That is, add 1 to each of the elements in the new, all-zeroes matrix. If the [.] is omitted and just the [+] is used, the 1 will be treated as the identity matrix and then added to the new matrix, which is not wanted.*

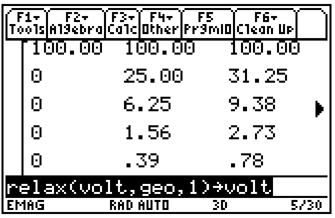
(31)


```
[1  1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1  1 ▸
 1  1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1  1
newMat(11,11).+1→geo
EMAG      RAD AUTO      3D      4/30
```

6. Since the function **relax** automatically excludes the first and last rows, zeros need not be inserted for a rectangular solution region. For non-rectangular regions, zeros must be inserted into **geo** at boundary locations. Do one iteration as shown in screen 32.

   relax [(] **volt** [,] **geo** [,] **1** [)] [STO▸] **volt**

   The results are stored in **volt** through the command line since **relax** is a function and cannot change any stored variables.

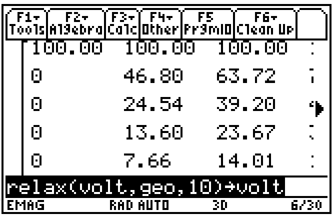
(32)


```
 100.00   100.00   100.00
 0        25.00    31.25
 0        6.25     9.38    ▸
 0        1.56     2.73
 0        .39      .78
relax(volt,geo,1)→volt
EMAG      RAD AUTO      3D      5/30
```

7. Repeat function **relax** 10 times (screen 33).

   relax [(] **volt** [,] **geo** [,] **10** [)] [STO▸] **volt**

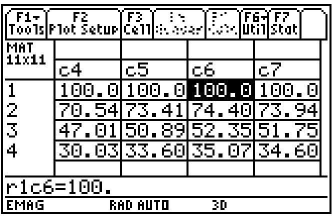
   It will take a while to get the results.

(33)


```
 100.00   100.00   100.00
 0        46.80    63.72
 0        24.54    39.20   ▸
 0        13.60    23.67
 0        7.66     14.01
relax(volt,geo,10)→volt
EMAG      RAD AUTO      3D      6/30
```

8. The matrix can be viewed by pressing [APPS] **6:Data/Matrix Editor 1:Current** as shown in screen 34.

   Since the structure is symmetric, the voltages should show symmetry about **c6** when sufficient iterations have been completed.



MAT 11x11	c4	c5	c6	c7
1	100.0	100.0	100.0	100.0
2	70.54	73.41	74.40	73.94
3	47.01	50.89	52.35	51.75
4	30.03	33.60	35.07	34.60

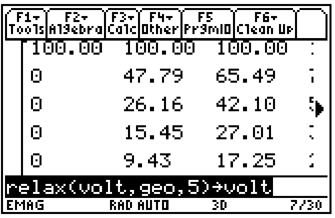
r1c6=100.

(34)

***Note**: Press [◆] [I] and select **5** to show four columns.*

9. Since it isn't exactly symmetric, return to the Home screen and repeat the function **relax** five more times as shown in screen 35.

   relax [(] **volt** [,] **geo** [,] **5** [)] [STO▸] **volt**

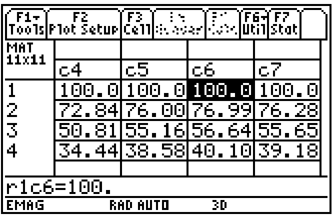


```
100.00   100.00   100.00   :
0        47.79    65.49    :
0        26.16    42.10    5▸
0        15.45    27.01    :
0        9.43     17.25    :
relax(volt,geo,5)→volt
```

(35)

10. Return to the Matrix Editor to check the symmetry (screen 36).

    It's not exactly there yet , but close enough.



MAT 11x11	c4	c5	c6	c7
1	100.0	100.0	100.0	100.0
2	72.84	76.00	76.99	76.28
3	50.81	55.16	56.64	55.65
4	34.44	38.58	40.10	39.18

r1c6=100.

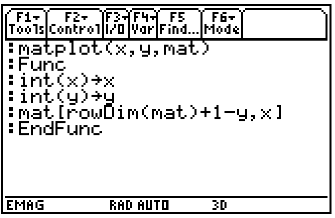
(36)

Topic 55: 3D Graphs of Tabular Data

As with the separation of variables method, a graph of the voltage from Topic 54 is helpful. However, matrix data cannot be graphed directly because it exists only for discrete values of row and column variables. But with the **int()** function, values of the graphing variable within a range can be converted to an integer for which row or column matrix data exists.

1. Create the function **matplot** as shown in screen 37 to convert the data.

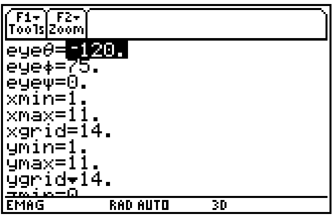
   The **x,y** coordinates of the data matrix **mat** are converted to integer values that provide the row and column indices. In order that the **y** index begins at the bottom of the matrix, it is reflected about the center row of the matrix.



```
:matplot(x,y,mat)
:Func
:int(x)→x
:int(y)→y
:mat[rowDim(mat)+1-y,x]
:EndFunc
```

(37)

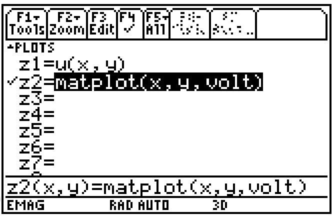
2. Press [◆] [WINDOW] and set the window variable values as shown in screen 38 as well as **zmin**=0, **zmax**=140, and **ncoutour**=0. Be sure the TI-89 is in the 3D graph mode.



```
eyeθ=-120.
eyeφ=75.
eyeψ=0.
xmin=1.
xmax=11.
xgrid=14.
ymin=1.
ymax=11.
ygrid▾14.
zmin=0
```

(38)

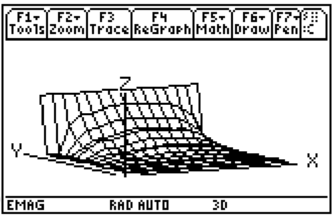
3. Press ◆ [Y=] to display the Y= Editor. Define **z2(x,y)=matplot(x,y,volt)** to graph the relaxation method solution (screen 39). Deselect **z1**.



```
▸PLOTS
 z1=u(x,y)
✓z2=matplot(x,y,volt)
 z3=
 z4=
 z5=
 z6=
 z7=
z2(x,y)=matplot(x,y,volt)
```

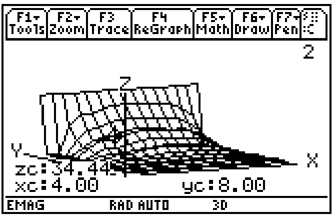
(39)

4. Press ◆ [GRAPH] to see the solution.



(40)

5. It looks nearly like the graph with the other method. A comparison with the exact solution of separation of variables using F3 **Trace** shows that the approximate voltage at x=1/4∗**xmax**=4 and y=3/4∗**ymax**=8 is v=34.44 V compared to the exact value of 43.20 V (screen 23). This inaccuracy is due to the adjustment of the x- and y-coordinates to take the integer plotting routine into account and the approximate nature of the relaxation method.



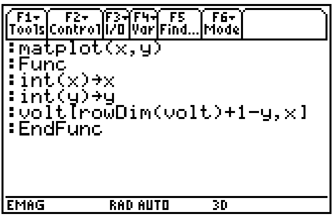
```
zc:34.44
xc:4.00          yc:8.00
```

(41)

Tips and Generalizations

This chapter showed how to solve and display problems using the TI-89 that are normally solved on larger computers. The relaxation method is general enough to solve other configurations by simply changing **volt** and **geo**.

The **matplot( )** function can be used to display a 3D graph of any matrix of data. It is faster, though less general, if the matrix to be graphed is explicitly stated in the function rather than passed as a parameter. Screen 42 shows how this is done for the relaxation problem.



```
:matplot(x,y)
:Func
:int(x)→x
:int(y)→y
:volt[rowDim(volt)+1-y,x]
:EndFunc
```

(42)

Although finding the potential over a 2D surface is useful, sometimes knowing what happens on a line is enough. Chapter 12 looks at some common ways of analyzing transmission lines.