Solution of Laplace's Equation Using Matrix Inversion

Solution of Laplace's Equation Using Matrix Inversion

For $\rho = 0$ and $\nabla \varepsilon = 0$,

The direct solution to Laplace's Equation $\nabla^2 V = 0$

leads to the following matrix equation

$$[C][V] = [F]$$

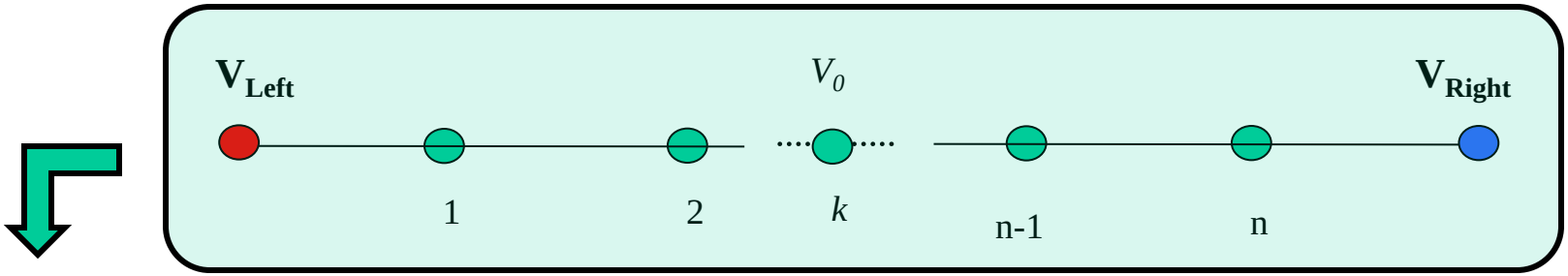
where for n unknown potential points in the solution domain

$[C]$ is an n x n matrix of coefficients for the potenials.

$[V]$ is an n x 1 vector of unknown potentials.

$[F]$ is an n x 1 vector of known potentials or zeroes.

Matrix Solution for a 1D Problem Uniform Discretization



for the point on the left boundary

$0 + V_C + 0 = V_{Left}$

for the point on the right boundary

$0 + V_C + 0 = V_{Right}$

$$V_C = \frac{1}{2}[V_L + V_R] \quad \Rightarrow \quad V_L - 2V_C + V_R = 0 \,,$$

or for a fixed potenial point $0 + V_C + 0 = V_{\text{Fixed potential}}$

for the point next to the left boundary, point (1)

$$0 - 2V_{C(1)} + V_{R(2)} = -V_{Left}$$

for the point next to the right boundary, point (n)

$$V_{L(n-1)} - 2V_{C(n)} + 0 = -V_{Right}$$

for any intermediate point

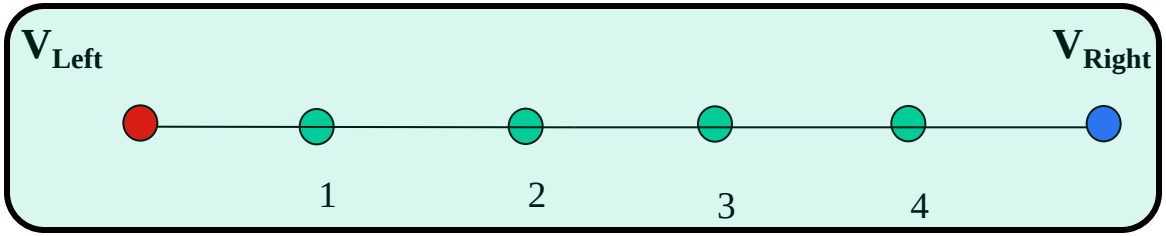
$$V_L - 2V_C + V_R = 0$$

for a point k with fixed potential V_0

$$0 + V_{C(k)} + 0 = V_0$$

$$\begin{bmatrix} -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} V_1 \\ \cdots \\ V_{i-1} \\ V_i \\ V_{i+1} \\ V_k \\ \cdots \\ V_n \end{bmatrix} = \begin{bmatrix} -Vleft \\ 0 \\ 0 \\ 0 \\ 0 \\ V_0 \\ 0 \\ -Vright \end{bmatrix}$$

Illustrative Example for the Matrix Solution of a 1D Problem



for point # 1: $-2V_1 + V_2 = -V_{Left}$

for point # 2: $V_1 - 2V_2 + V_3 = 0$

for point # 3: $V_2 - 2V_3 + V_4 = 0$

for point # 4: $V_3 - 2V_4 = -V_{Right}$

$$\begin{bmatrix} -2 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} = \begin{bmatrix} -V_{left} \\ 0 \\ 0 \\ -V_{right} \end{bmatrix}$$

For any point, a general expression can be written as

$a_k V_{k-1} + b_k V_k + c_k V_{k+1} = d_k$

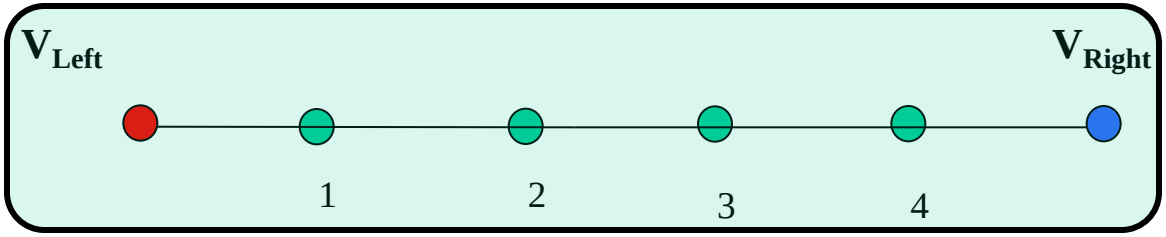
where the coefficients $a_k, b_k, c_k,$ and d_k can be stored in a one dimensional vectors.

Thus the allocation of the 2D matrix is not necessary.

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 \\ 0 & a_3 & b_3 & c_3 \\ 0 & 0 & a_4 & b_4 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix}$$

Tri-Diagonal Matrix

Solution of the Tri-Diagonal Matrix



From equation # 1: $b_1V_1 + c_1V_2 = d_1 \rightarrow V_1 = (d_1 - c_1V_2)/b_1$

In equation # 2: $a_2V_1 + b_2V_2 + c_2V_3 = d_2$

$$(b_2 - \frac{a_2c_1}{b_1})V_2 + c_2V_3 = (d_2 - \frac{a_2d_1}{b_1})$$

$$\rightarrow b_2'V_2 + c_2V_3 = d_2' \rightarrow V_2 = (d_2' - c_2V_3)/b_2'$$

In equation # 3: $a_3V_2 + b_3V_3 + c_3V_4 = d_3$

$$(b_3 - \frac{a_3c_2}{b_2'})V_3 + c_3V_4 = (d_3 - \frac{a_3d_2'}{b_2'})$$

$$\rightarrow b_3'V_3 + c_3V_4 = d_3' \rightarrow V_3 = (d_3' - c_3V_4)/b_3'$$

In equation # 4: $a_4V_3 + b_4V_4 = d_4$

$$(b_4 - \frac{a_4c_3}{b_3'})V_4 = (d_4 - \frac{a_4d_3'}{b_3'}) \rightarrow b_4'V_4 = d_4' \rightarrow V_4 = d_4'/b_4'$$

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 \\ 0 & a_3 & b_3 & c_3 \\ 0 & 0 & a_4 & b_4 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix}$$

Tri-Diagonal Matrix Coefficients

$$b_2' = (b_2 - \frac{a_2 c_1}{b_1}) \,,\; b_3' = (b_3 - \frac{a_3 c_2}{b_2'}) \,,\; b_4' = (b_4 - \frac{a_4 c_3}{b_3'})$$

$$b_i' = (b_i - \frac{a_i c_{i-1}}{b_{i-1}'})$$

$$d_2' = (d_2 - \frac{a_2 d_1}{b_1}) \,,\; d_3' = (d_3 - \frac{a_3 d_2'}{b_2'}) \,,\; d_4' = (d_4 - \frac{a_4 d_3'}{b_3'})$$

$$d_i' = (d_i - \frac{a_i d_{i-1}'}{b_{i-1}'})$$

$$V_1 = (d_1 - c_1 V_2)/b_1 \quad , \quad V_2 = (d_2' - c_2 V_3)/b_2'$$

$$V_3 = (d_3' - c_3 V_4)/b_3' \quad , \quad V_4 = (d_4' - 0)/b_4'$$

$$V_i = (\frac{d_i' - c_i V_{i+1}}{b_i'})$$

Tri-Diagonal Matrix – MatLab Function Routine

$$b_i' = (b_i - \frac{a_i c_{i-1}}{b_{i-1}'}) \quad , \quad d_i' = (d_i - \frac{a_i d_{i-1}'}{b_{i-1}'}) \quad , \quad V_i = (\frac{d_i' - c_i V_{i+1}}{b_i'})$$

```
Function V=tri_diag(a,b,c,d,n)
for i=2:n
   b(i)=b(i)-(a(i)*c(i-1))/b(i-1);
   d(i)=d(i)-(a(i)*d(i-1))/b(i-1);
end
V(n)=d(n)/b(n);
for i=n-1:-1:1
   V(i)=(d(i)-c(i)*V(i+1))/b(i);
end
```

$$\begin{bmatrix} b_1 & c_1 & & & & & \\ a_2 & b_2 & c_2 & & & & \\ & a_3 & b_3 & c_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & a_i & b_i & c_i & \\ & & & & \ddots & \ddots & \\ & & & & & a_n & b_n \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ \vdots \\ V_i \\ \vdots \\ V_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_i \\ \vdots \\ d_n \end{bmatrix}$$

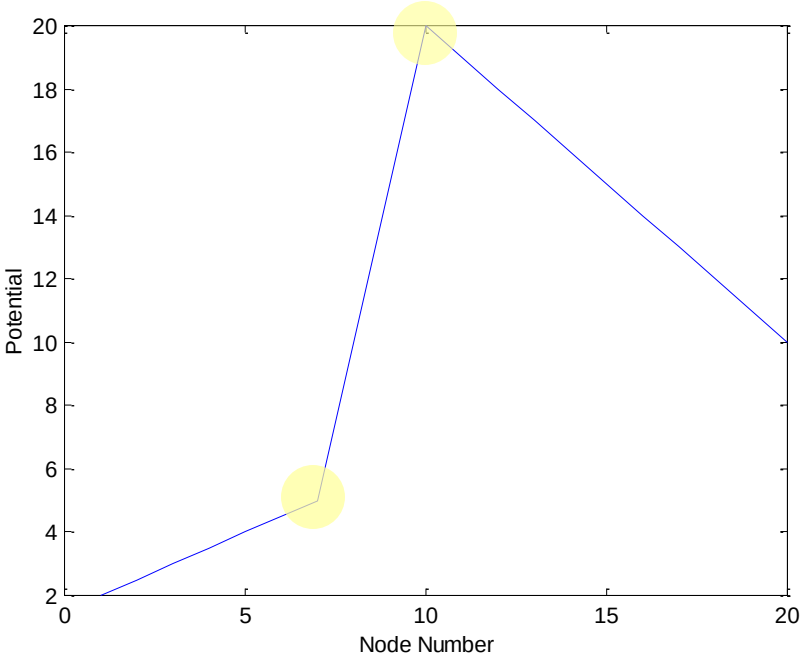
Vectorized form

```
Function V=tri_diag(a,b,c,d,n)
b(2:n)=b(2:n)-(a(2:n)*c(1:n-1))/b(1:n-1);
d(2:n)=d(2:n)-(a(2:n)*d(1:n-1))/b(1:n-1);
V(n)=d(n)/b(n);
for i=n-1:-1:1
   V(i)=(d(i)-c(i)*V(i+1))/b(i);
end
```

Driver for The Tri_Diag.m Function

```
% Tri-Diag_test.m  By: Dr. Atef Elsherbeni
% n number of total points in the domain
% If the potential at the boundaries is
% not defined or defined in a different
% manner, then the matrix set up for the
% boundary points will be different
n = 20; vleft = 2; vright = 10;
b(1) = 1 ; c(1)= 0; d(1) = vleft;
for i = 2:n-1
   a(i)=1; b(i)=-2 ; c(i)=1; d(i)=0;
end
a(n)= 0; b(n)= 1; d(n) = vright;

% nodes with fixed potential
no2 = n/2
a(no2)=0;b(no2)=1;c(no2)=0;d(no2)=20;
no3 = n/3
a(no3)=0;b(no3)=1;c(no3)=0;d(no3)=5;

v = tri_diag(a,b,c,d,n);

plot([1:n],[v(1:n)]);
xlabel (' Node Number');
ylabel('Potential');
```

$$\begin{bmatrix} b_1 & c_1 & & & & & \\ a_2 & b_2 & c_2 & & & & \\ & a_3 & b_3 & c_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & a_i & b_i & c_i & \\ & & & & \ddots & \ddots & \\ & & & & & a_n & b_n \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ \vdots \\ V_i \\ \vdots \\ V_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_i \\ \vdots \\ d_n \end{bmatrix}$$



Generating Matrix Equation for a 2D Problem

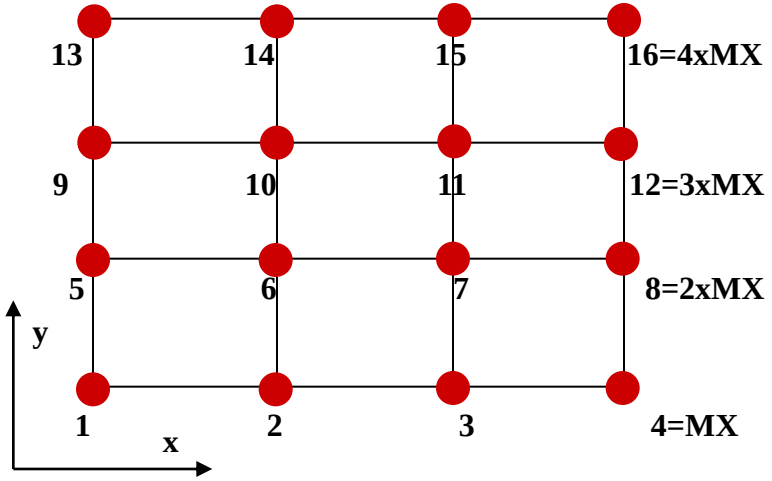
For uniform discretizations and a homogeneous media

$$V_T + V_B - 4V_C + V_L + V_R = 0$$

Or in general

$$c_t V_T + c_b V_B + c_c V_C + c_l V_L + c_r V_R = c$$

Where *C's* are coefficients.

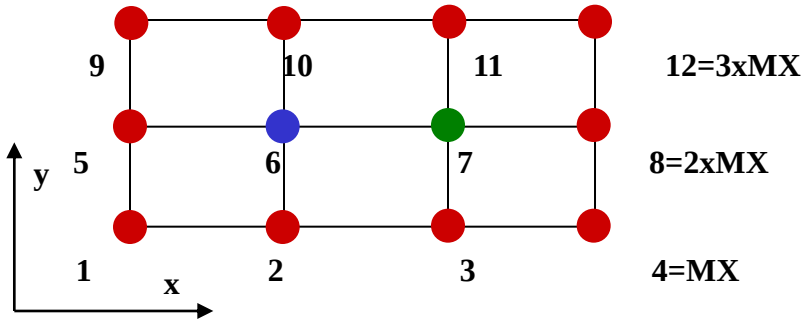


The equations from all nodes can be written in the following matrix form

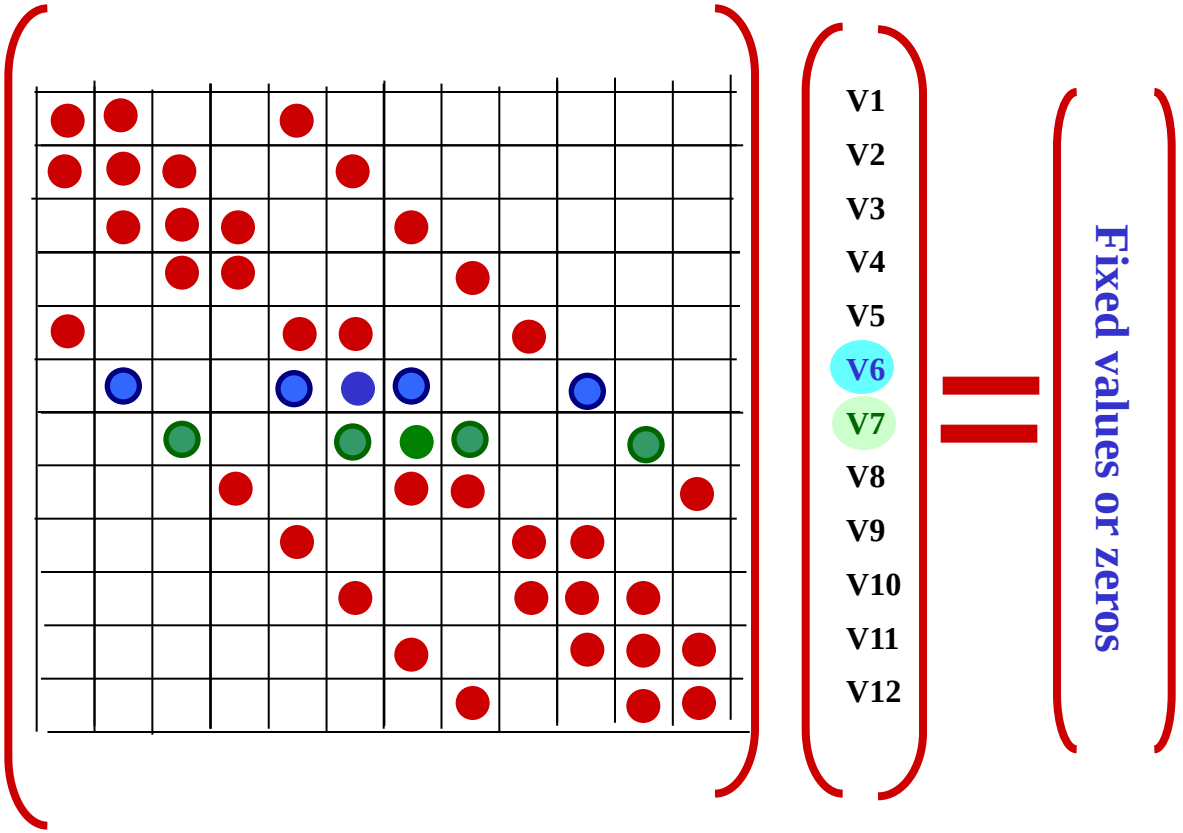
Coefficients	Unknown Potentials	=	Zeroes or Known Potential at fixed points

Generating the Coefficient Matrix

For points on this sample mesh the following matrix elements are filled with circles. Other elements are zeroes.



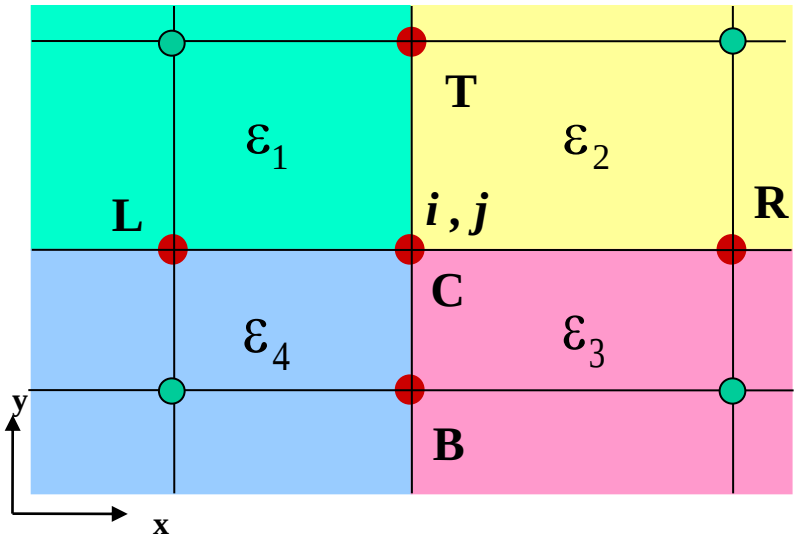
The result is a sparse matrix for the coefficients.



Potentials and the Media Properties in Terms of Node Coordinates

$$V(i,j) = C_T(i,j)V(i,j+1) + C_L(i,j)V(i-1,j) + C_B(i,j)V(i,j-1) + C_R(i,j)V(i+1,j)$$

$$V(i,j) - C_T(i,j)V(i,j+1) - C_L(i,j)V(i-1,j) - C_B(i,j)V(i,j-1) - C_R(i,j)V(i+1,j) = 0$$



$$C_T(i,j) = \frac{[\varepsilon_{(i-1,j)} + \varepsilon_{(i,j)}]/2}{[\varepsilon_{(i-1,j)} + \varepsilon_{(i,j)} + \varepsilon_{(i,j-1)} + \varepsilon_{(i-1,j-1)}]}$$

$$C_L(i,j) = \frac{[\varepsilon_{(i-1,j)} + \varepsilon_{(i-1,j-1)}]/2}{[\varepsilon_{(i-1,j)} + \varepsilon_{(i,j)} + \varepsilon_{(i,j-1)} + \varepsilon_{(i-1,j-1)}]}$$

$$C_B(i,j) = \frac{[\varepsilon_{(i-1,j-1)} + \varepsilon_{(i,j-1)}]/2}{[\varepsilon_{(i-1,j)} + \varepsilon_{(i,j)} + \varepsilon_{(i,j-1)} + \varepsilon_{(i-1,j-1)}]}$$

$$C_R(i,j) = \frac{[\varepsilon_{(i,j-1)} + \varepsilon_{(i,j)}]/2}{[\varepsilon_{(i-1,j)} + \varepsilon_{(i,j)} + \varepsilon_{(i,j-1)} + \varepsilon_{(i-1,j-1)}]}$$

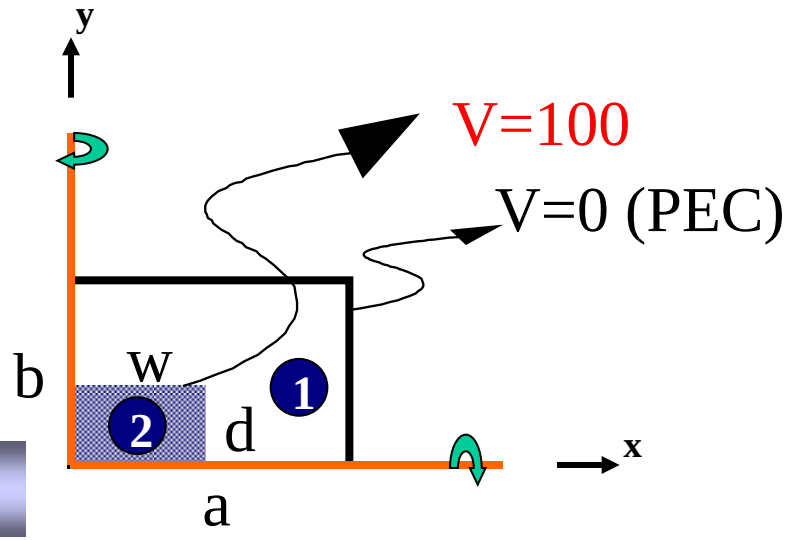
Matrix Solution for a Rectangular Coaxial Cable

1 **For general nodes in uniform media**

$$c_C V_C - c_B V_B - c_L V_L - c_R V_R - c_T V_T = 0$$

$$c_B = c_L = c_R = c_T = \frac{1}{4} \quad , \quad c_C = 1$$

Computational domain



2 **For constant potential PEC object**

$$c_C V_C - c_B V_B - c_L V_L - c_R V_R - c_T V_T = V_{\text{constant}} = 100$$

$$c_B = c_L = c_R = c_T = 0 \quad , \quad c_C = 1$$

Matrix Solution for a Rectangular Coaxial Cable

3 **PEC boundaries**

$$C_C V - C_B V_B - C_L V_{LC} - C_R V_R - C_T V_T = 0$$

$$C_B = C_L = C_R = C_T = 0 \quad , \quad C_C = 1$$

Symmetry along the y-axis

$$C_C V_C - C_B V_B - 2C_R V_R - C_T V_T = 100 \quad (2)$$

$$= 0 \quad (1)$$

$$C_B = C_R = C_T = \frac{1}{4} \quad , \quad C_C = 1$$

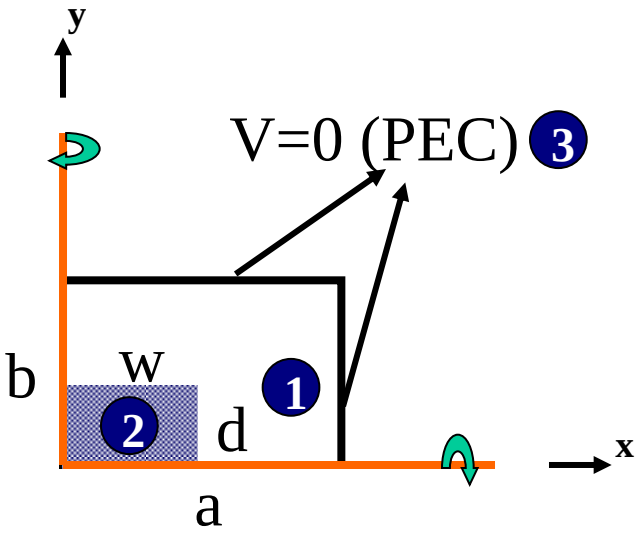
Symmetry along the x-axis

$$C_C V_C - C_L V_L - C_R V_R - 2C_T V_T = 100 \quad (2)$$

$$= 0 \quad (1)$$

$$C_L = C_R = C_T = \frac{1}{4} \quad , \quad C_C = 1$$

Computational domain



Matrix Shape and Size

- **The matrix form**

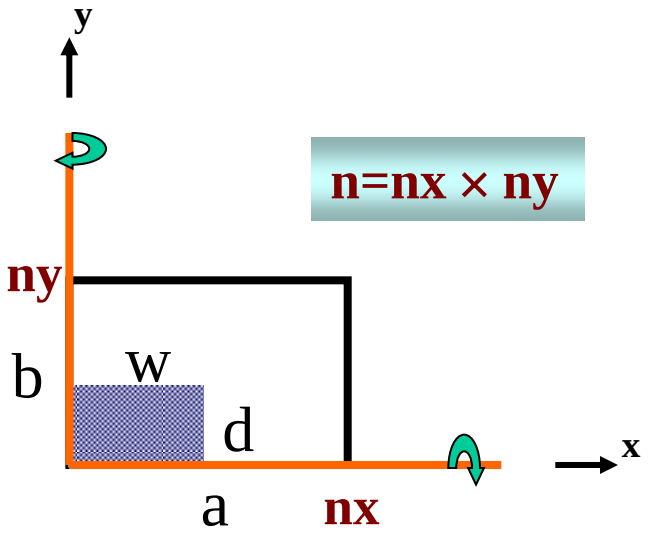
$$[C][V]=[F]$$

C **: Coefficients Matrix (n×n)**

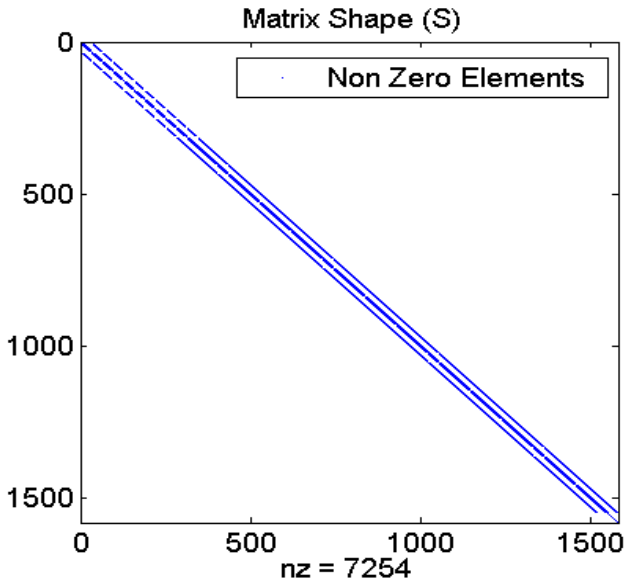
V **: Unknown Potential**

F **: Right hand side of equation**

Computational domain



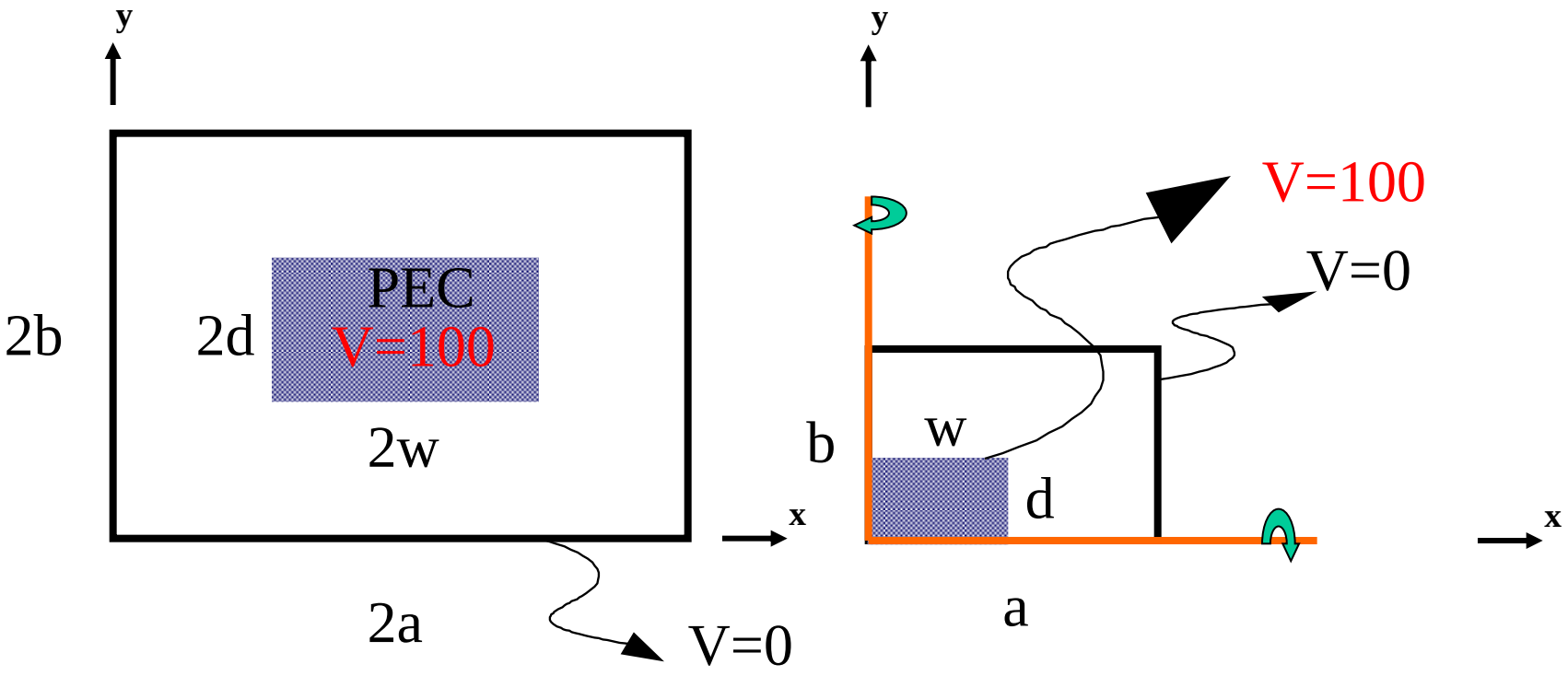
- **"C" is a sparse matrix of size (n×n)**
- **For nx = 51 and ny = 31, n = 1581**
- **Total number of elements n^2 = 2,499,561**
- **Non-zero elements = 7254= 0.29%**



Example - Rectangular Coaxial Cable

Original problem

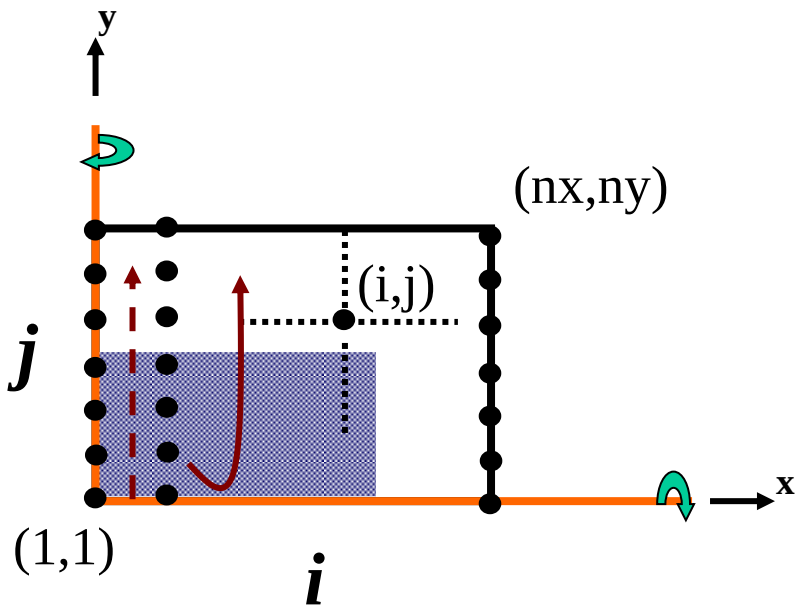
Computational domain



$a = 2.5, b = 1.5, w = .5, d = .25$, h=.025 All dimensions are in mm.

Node Assignment and Coefficient Matrix Filling

```
C = sparse(n,n);    % Matrix initialization
jj = 0;
for i = 1:nx
   for j = 1:ny
      jj = jj + 1;
      if i == 1 & j > nd & j < ny     % Symmetry - y axis
         C(jj,jj) = 1;
         C(jj,jj+ny) = -0.5;  % Right node
         C(jj,jj-1) = -0.25;  % Bottom node
         C(jj,jj+1) = -0.25;  % Top node
         F(jj) = 0;
         .
         .
     elseif  i == nx     % Right boundary (now PEC)
         C(jj,jj) = 1;
         F(jj) = 0;
         .
         .
    elseif  i >= 1 & i <= nw & j >= 1 & j <= nd  % Constant potential
         C(jj,jj) = 1;
         F(jj) = vcond;
        .
        .
   end
end

V = C\F.';          % Matrix solution (.') for matrix transpose
% V = inv(C)*F.';
```

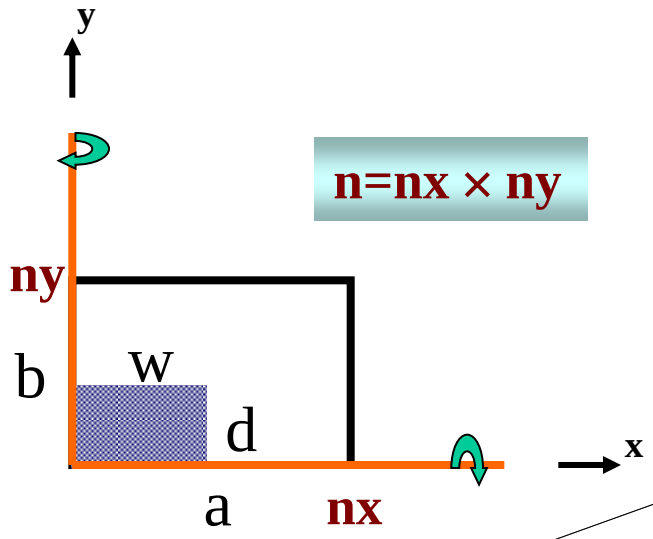


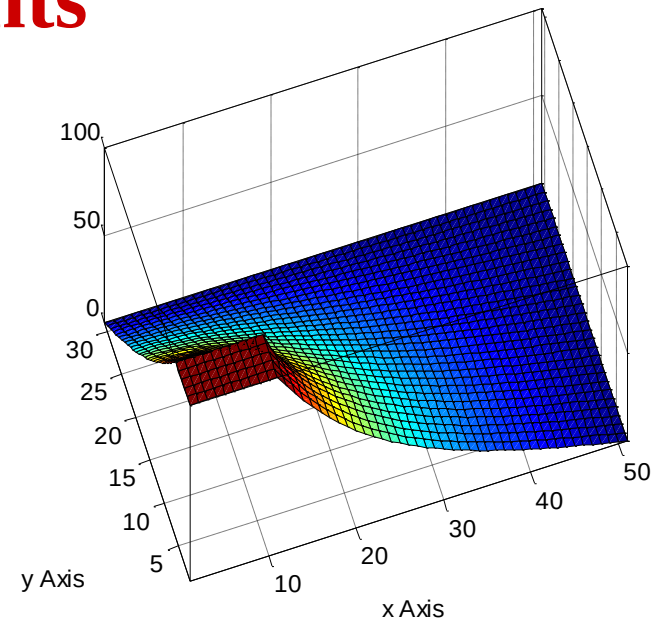
jj for node *(i , j) = (i-1)×ny+j*

```
% Putting the solution in the vector form into a physical grid
layout
jj = 0;
for i = 1:nx
   for j = 1:ny
      jj = jj + 1;
      Voltage(i,j) = V(jj);
   end
end
```

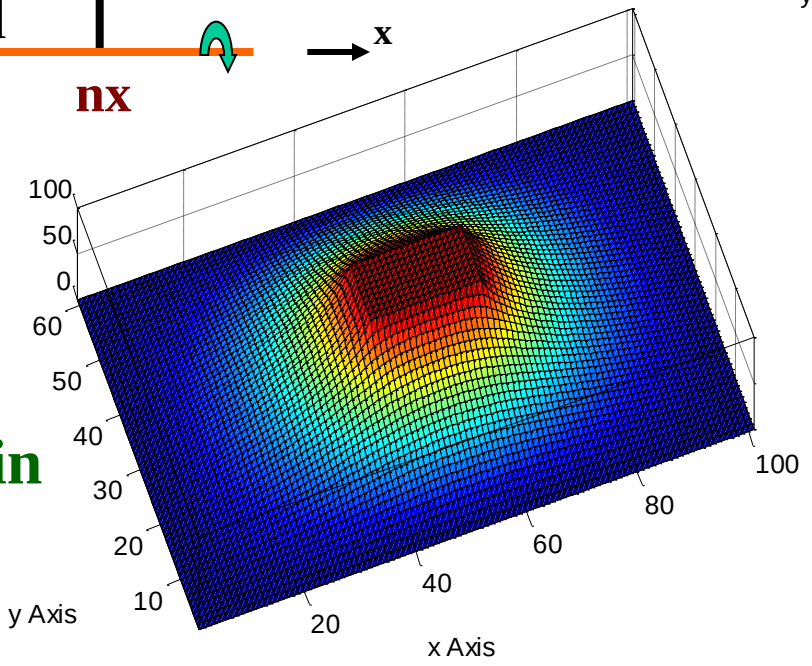
Expected Results

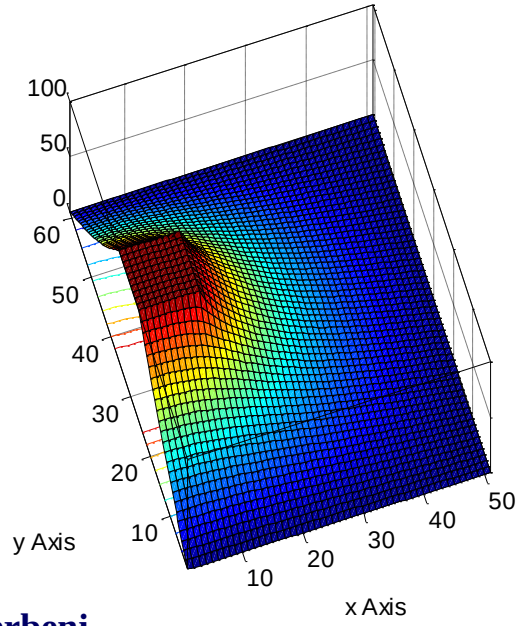
Computational domain





Full domain





CPU Time Comparison

nx = 102, ny = 62, delta = .025
Matrix size (n) = 6324*6324

Matrix Solution

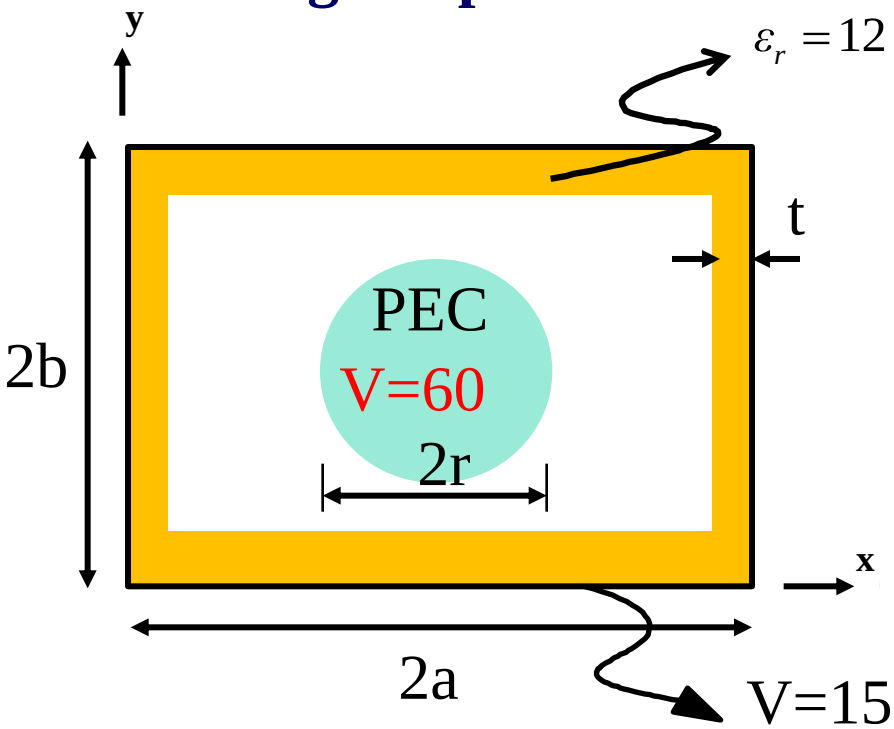
Matrix Initialization	Inverse Technique	CPU Time [Sec.]
Sparse	C\F.`	0.343
Zeros	C\F.`	0.499
Sparse	Inv(C)*F.`	10.73
Zeros	Inv(C)*F.`	10.82

Iterative Solution

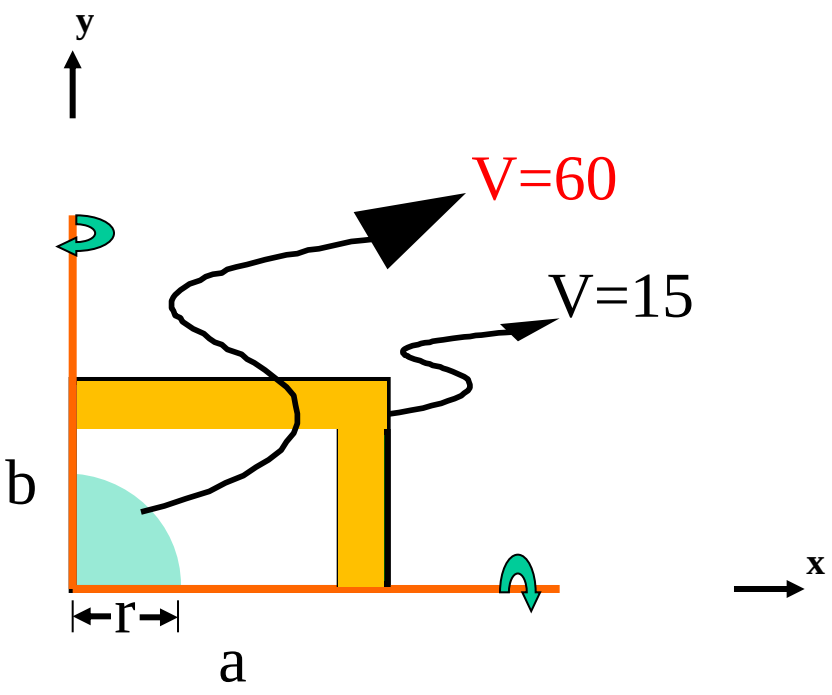
Number of Iterations	CPU Time [Sec.] Vectorized Code
1000	0.128
3000	0.343

Assignment - Coaxial Cable

Original problem



Computational domain



$a = 8, b = 6, r = 1.8,\ t = 0.4$ All dimensions are in cm.

For the above coaxial cable geometry (left figure), use FD technique to develop Matlab codes to solve for the potential distribution inside the outer conducting boundary of the cable. Use only one quarter of the problem as shown above (right figure) for the potential computations. Two codes to be developed based on:

a) Iterative procedure

b) matrix inversion procedure

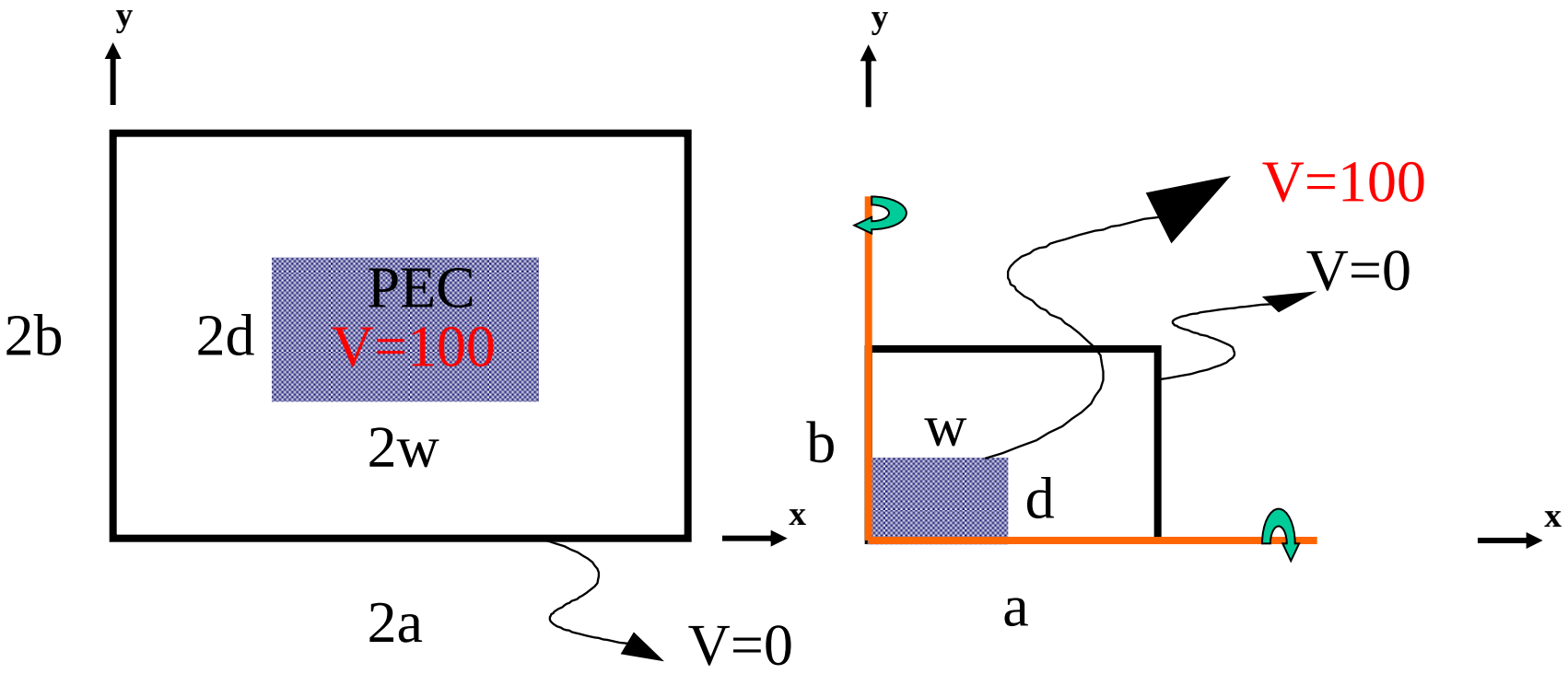
Compare the CPU times required as illustrated in the previous slide.

End of Lecture

Example - Rectangular Coaxial Cable

Original problem

Computational domain



$a = 2.5, b = 1.5, w = .5, d = .25,$ All dimensions are in mm.

Matlab Code for the Rectangular Coaxial Problem 1/3

```
%Rect_Coax_Matrix.m
% Solution of a shielded rectangular coaxial
% one quarter of the domain is used for field computation
% Different forms for the coefficient matrix (sparse, full)
% two procedures for the matrix solution procedure
% Started on September 4, 2004
% Last modified Sept. 3, 2007
% by Dr. Atef Elsherbeni -- atef@olemiss.edu
%
clear all; ; close all; setfigures;
DateTime_Start = datestr(now)
tstart = cputime;
%% Input Data:
%% ------------
a = 2.5;            % outer length along x
b = 1.5;            % outer length along b
w = 0.5;            % center conductor length along x
d = 0.25;           % center conductor length along y
h = 0.05;           % incerement of the segmentation
vcond = 100;        % potential on the center conductor
time = cputime;
nx = (a/h)+1;  ny = (b/h)+1;
nw = (w/h)+1;    nd = (d/h)+1;
n = nx*ny;
%% Initializing the Matrix
F(1:n)= 0;
%F= zeros(n,1);
%fsize = size(F)

 S = sparse(n,n);
 %S = zeros(n,n);
```

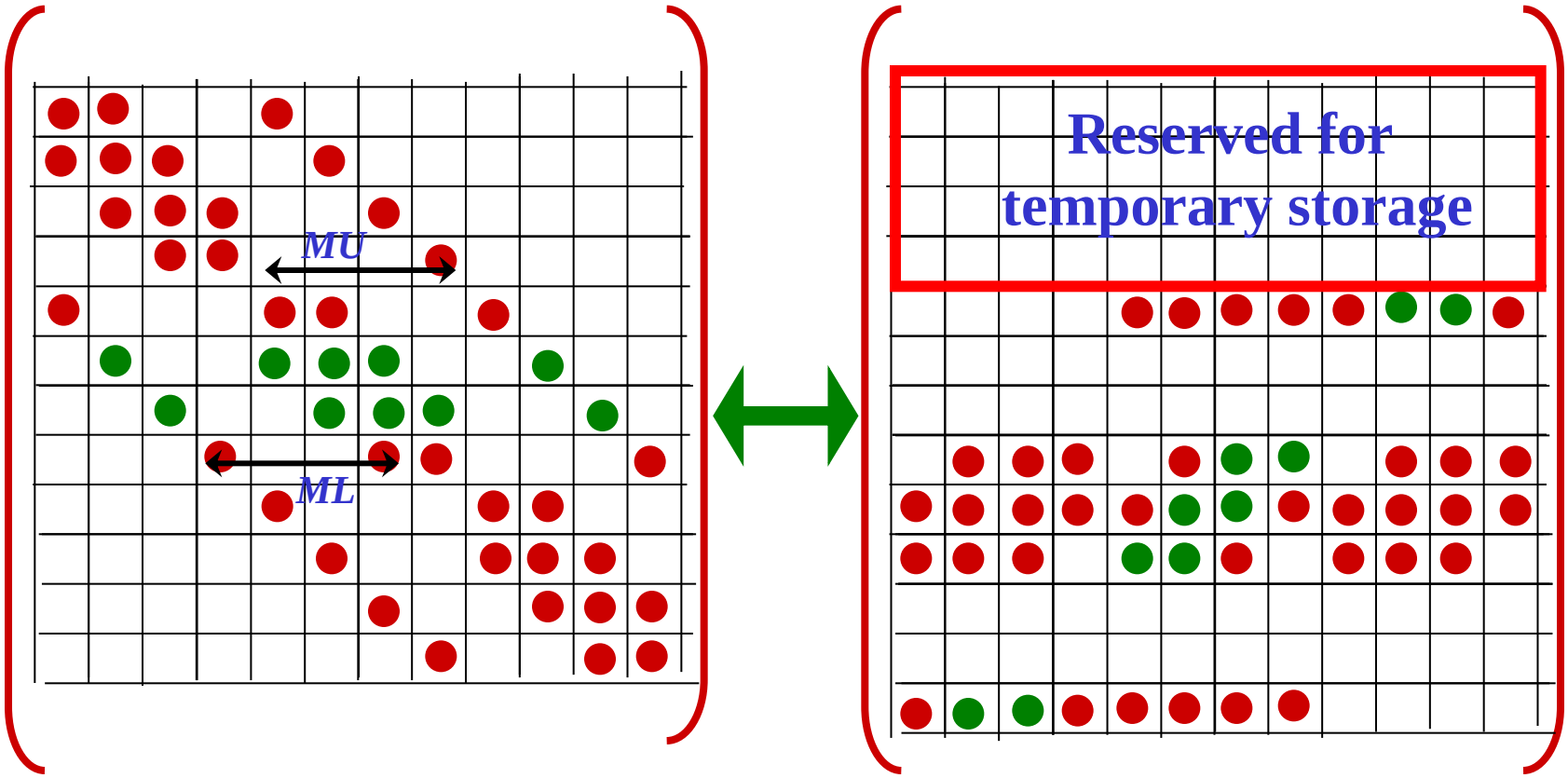
Matlab Code for the Rectangular Coaxial Problem 2/3

```
%% Matrix filling
jj = 0;
for i = 1:nx
    for j = 1:ny
        jj = jj + 1;
        if i == 1 & j > nd & j < ny     % Symmetry around y axis
            S(jj,jj) = 1;
            S(jj,jj+ny) = -0.5;  % Right node
            S(jj,jj-1) = -0.25;  % Bottom node
            S(jj,jj+1) = -0.25;  % Top node
            F(jj) = 0;
        elseif j == 1  & i > nw & i < nx     % Symmetry around x axis
            S(jj,jj) = 1;  F(jj) = 0;
            S(jj,jj-ny) = -0.25;  % left node
            S(jj,jj+ny) = -0.25;  % Right node
            S(jj,jj+1) = -0.5;    % Top node
        elseif i == nx     % Right boundary (now PEC)
            S(jj,jj) = 1;  F(jj) = 0;
        elseif j == ny       % top boundary (now PEC)
            S(jj,jj) = 1;  F(jj) = 0;
        elseif i >= 1 & i <= nw & j >= 1 & j <= nd  % Constat potential region
            S(jj,jj) = 1;  F(jj) = vcond;
        else             % General point
            S(jj,jj) = 1; F(jj) = 0;
            S(jj,jj-ny) = -0.25;     % left node
            S(jj,jj+ny) = -0.25;     % Right node
            S(jj,jj-1) = -0.25;     % Bottom node
            S(jj,jj+1) = -0.25;    % Top node
        end
    end
end
```

Matlab Code for the Rectangular Coaxial Problem 3/3

```
%V = S\F.';  % Matrix solution (.') for the transpoze

V = inv(S)*F.';

% Putting the solution in the vector form into a physical grid layout
jj = 0;
for i = 1:nx
    for j = 1:ny
        jj = jj + 1;
        Voltage(i,j) = V(jj,1);
    end
end
vt = Voltage.';  % Note the transpose operation
figure(1);
surf(vt);  View(-15,35);     % plot the potential distribution
xlabel('x Axis'); ylabel('y Axis'); title('Potential Distribution');
box on; axis equal, colorbar;

vup=flipud(vt); vtt=vt(2:ny,:); vft=cat(1,vup,vtt);
figure(2);
surfc(vft); View(-15,35);box on; axis equal, colorbar;
xlabel('x Axis'); ylabel('y Axis'); title('Potential Distribution');

vlr=fliplr(vft); vlt=vft(:,2:nx); vlf=cat(2,vlr,vlt);
figure(3);
surfc(vlf); View(-15,35);box on; axis equal, colorbar;
xlabel('x Axis'); ylabel('y Axis'); title('Potential Distribution');

tend = cputime;          % end time for the matrix solution
disp('Total Conputational Time in Seconds')
(tend - tstart)
```

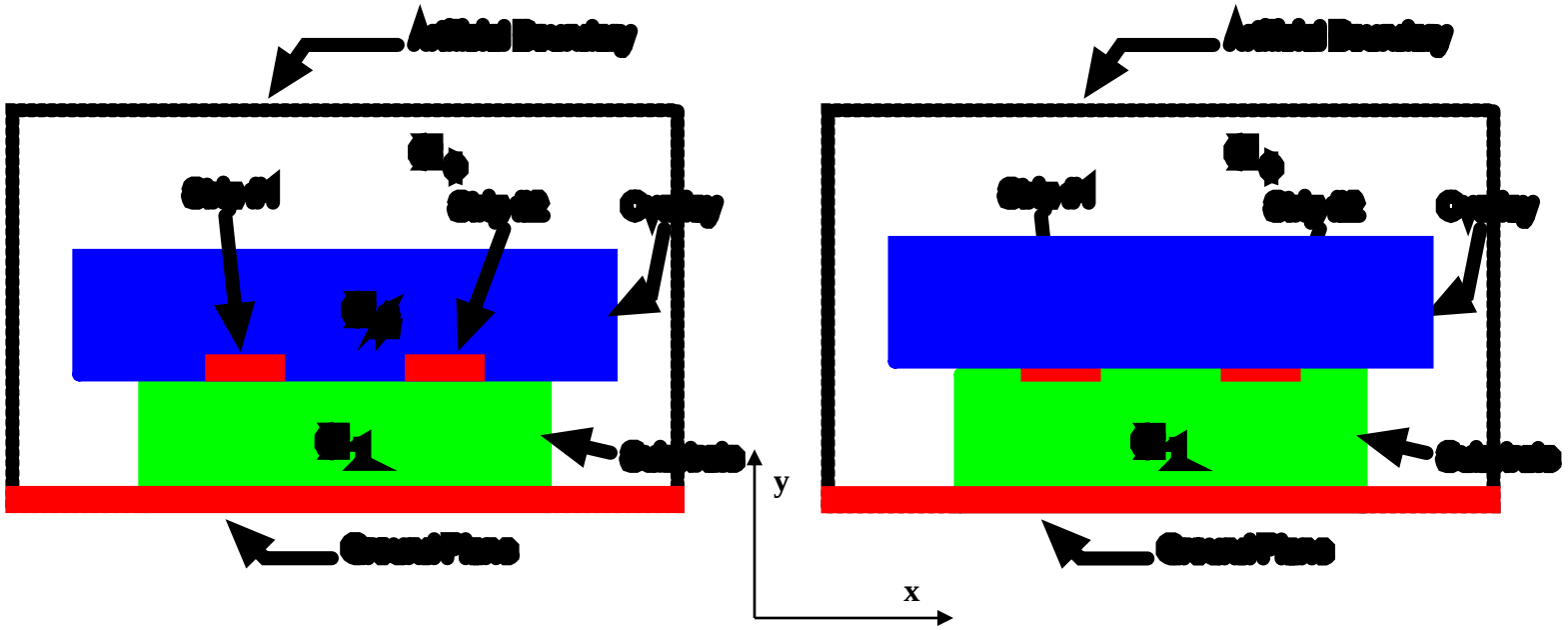
Converting the Coefficient Matrix to a Banded Matrix



$$A(i,j) \Leftrightarrow A_{banded}(ML + MU + 1 + i - j, j)$$

Two Geometries of Coupled Lines with Overlay

Numerical results



Coupled Microstrip lines with dielectric overlays with and without air gap due to the thickness of the conductors.

Results for the Potential Coupled Lines with Overlay