Rutgers University
School of Engineering

Fall 2011

14:440:127 - Introduction to Computers for Engineers

Sophocles J. Orfanidis
ECE Department
orfanidi@ece.rutgers.edu

week 8

Weekly Topics

Week 1 - Basics – variables, arrays, matrices, plotting (ch. 2 & 3)
Week 2 - Basics – operators, functions, program flow (ch. 2 & 3)
Week 3 - Matrices (ch. 4)
Week 4 - Plotting – 2D and 3D plots (ch. 5)
Week 5 - User-defined functions (ch. 6)
Week 6 - Input-output processing (ch. 7)
Week 7 - Program flow control & relational operators (ch. 8)
Week 8 - Matrix algebra – solving linear equations (ch. 9)
Week 9 - Strings, structures, cell arrays (ch. 10)
Week 10 - Symbolic math (ch. 11)
Week 11 - Numerical methods – data fitting (ch. 12)
Week 12 – Selected topics

Textbook: H. Moore, *MATLAB for Engineers*, 2nd ed., Prentice Hall, 2009

Matrix Algebra

- dot product
- matrix-vector multiplication
- matrix-matrix multiplication
- matrix inverse
- solving linear systems
- least-squares solutions
- determinant, rank, condition number
- vector & matrix norms
- examples
- electric circuits
- temperature distributions

The dot product is the basic operation in matrix-vector and matrix-matrix multiplications

Operators and Expressions

operation	element-wise	matrix-wise
addition	`+`	`+`
subtraction	`-`	`-`
multiplication	`.*`	`*`
division	`./`	`/`
left division	`.\`	`\`
exponentiation	`.^`	`^`
transpose w/o complex conjugation		`.'`
transpose with complex conjugation		`'`

```
>> help /
>> help precedence
```

used in matrix algebra operations

```
>> A = [1 2; 3 4]
A =
     1     2
     3     4
```

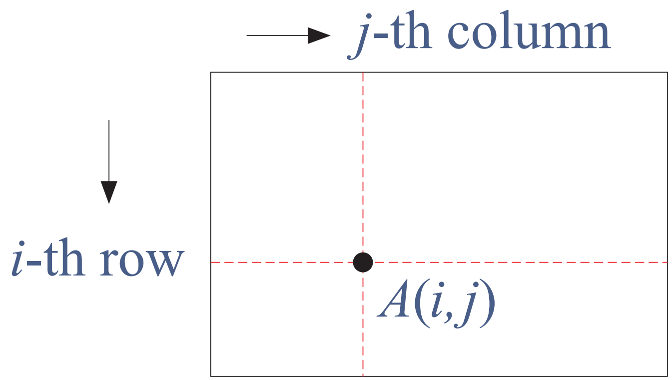
$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$$

```
>> [A, A.^2; A^2, A*A]          % form sub-blocks
ans =
     1     2  |   1     4
     3     4  |   9    16
  ------------+------------
     7    10  |   7    10       % note A^2 = A*A
    15    22  |  15    22
```

```
>> B = 10.^A;
>> [B, log10(B)]
ans =
          10         100           1           2
        1000       10000           3           4
```

$$B = \begin{bmatrix} 10^1 & 10^2 \\ 10^3 & 10^4 \end{bmatrix}$$

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$



matrix indexing convention

```
>> A = [1 2 3; 2 0 4; 0 8 5]
A =
     1     2     3
     2     0     4
     0     8     5

>> size(A)          % [N,M] = size(A), NxM matrix
ans =
     3     3
```

dot product

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

a, **b** must have the same dimension

$$\mathbf{a}^T\mathbf{b} = [a_1, a_2, a_3] \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = a_1b_1 + a_2b_2 + a_3b_3$$

$$\mathbf{a}^T\mathbf{b} = \mathbf{a}'\,\mathbf{b} = \mathbf{a} \cdot \mathbf{b} = \mathbf{a}\,.'*\,\mathbf{b}$$

math notations

MATLAB notation

dot product for complex-valued vectors

complex-conjugate transpose, or, hermitian conjugate of **a**

$$\mathbf{a}^\dagger \mathbf{b} = [a_1^*, a_2^*, a_3^*] \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = a_1^* b_1 + a_2^* b_2 + a_3^* b_3$$

$$\mathbf{a}^\dagger \mathbf{b} = \mathbf{a}^H \mathbf{b} = \mathbf{a}' * \mathbf{b}$$

math notations

MATLAB notation

for real-valued vectors, the operations `'` and `.'` are equivalent

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \\ -3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 4 \\ -5 \\ 2 \end{bmatrix}$$

$$[1,\ 2,\ -3]\begin{bmatrix} 4 \\ -5 \\ 2 \end{bmatrix} = 1 \times 4 + 2 \times (-5) + (-3) \times 2 = -12$$

```
>> a = [1; 2; -3];  b = [4; -5; 2];
>> a'*b
ans =
   -12
>> dot(a,b)              % built-in function
ans =                    % same as sum(a.*b)
   -12
```

matrix-vector multiplication

combine three dot product operations into a single matrix-vector multiplication

$$[4,\ 1,\ 2]\begin{bmatrix} 5 \\ -4 \\ -7 \end{bmatrix} = 2$$

$$[1,\ -1,\ 1]\begin{bmatrix} 5 \\ -4 \\ -7 \end{bmatrix} = 2 \quad \Rightarrow \quad \begin{bmatrix} 4 & 1 & 2 \\ 1 & -1 & 1 \\ 2 & 1 & 1 \end{bmatrix}\begin{bmatrix} 5 \\ -4 \\ -7 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ -1 \end{bmatrix}$$

$$[2,\ 1,\ 1]\begin{bmatrix} 5 \\ -4 \\ -7 \end{bmatrix} = -1$$

matrix-vector multiplication

combine three dot product operations into a single matrix-vector multiplication

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}$$

matrix-matrix multiplication

$$\begin{bmatrix} 4 & 1 & 2 \\ 1 & -1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ -4 \\ -7 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ -1 \end{bmatrix}$$

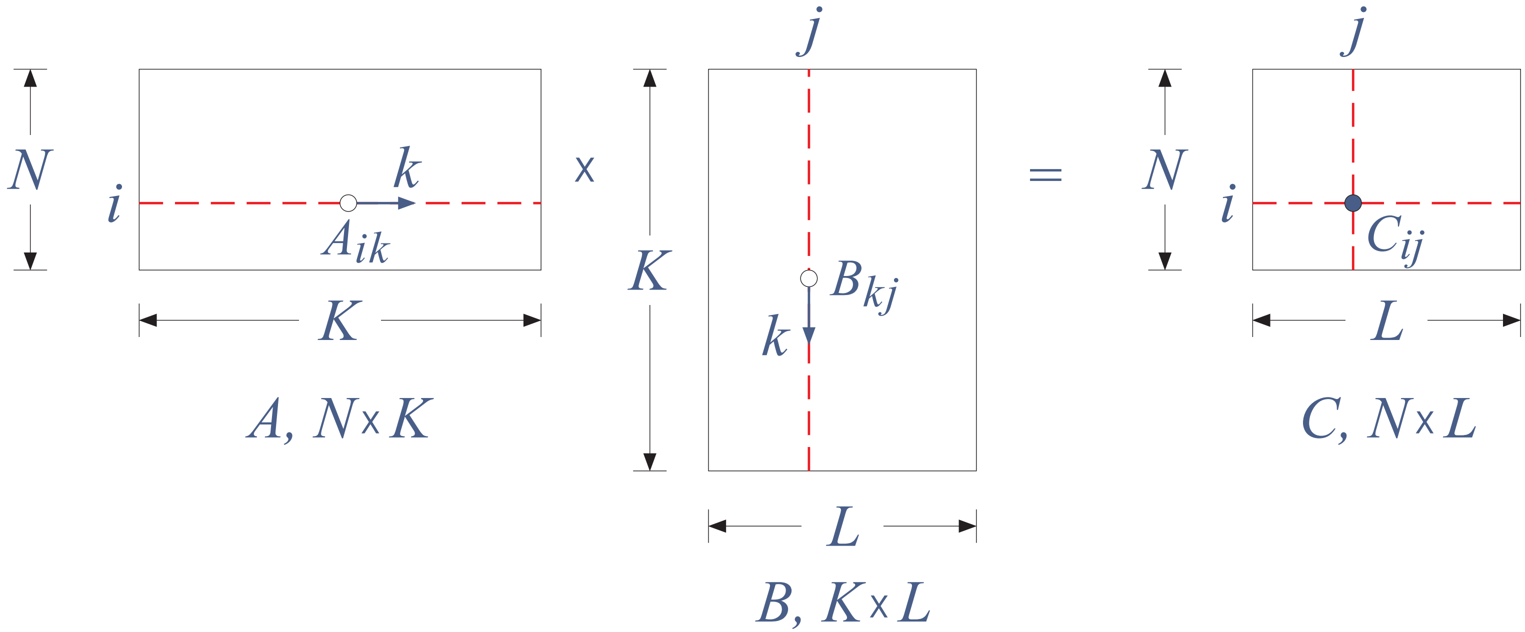
$$\begin{bmatrix} 4 & 1 & 2 \\ 1 & -1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \\ 3 \end{bmatrix}$$

$$\begin{bmatrix} 4 & 1 & 2 \\ 1 & -1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} -3 \\ 1 \\ 6 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

combine three matrix-vector multiplications into a single matrix-matrix multiplication

$$\begin{bmatrix} 4 & 1 & 2 \\ 1 & -1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 5 & -1 & -3 \\ -4 & 3 & 1 \\ -7 & 2 & 6 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 \\ 2 & -2 & 2 \\ -1 & 3 & 1 \end{bmatrix}$$

```
>> A = [4 1 2; 1 -1 1; 2 1 1]
A =
     4     1     2
     1    -1     1
     2     1     1

>> B = [5 -1 -3; -4 3 1; -7 2 6]
B =
     5    -1    -3
    -4     3     1
    -7     2     6

>> C = A*B
C =
     2     3     1
     2    -2     2
    -1     3     1
```



$$C_{ij} = \sum_{k=1}^{K} A_{ik} B_{kj}\,, \qquad 1 \le i \le N\,, \quad 1 \le j \le L$$

C(*i*,*j*) is the dot product of *i*-th row of *A* with *j*-th column of *B*

$$\begin{bmatrix} 4 & 1 & 2 \\ 1 & -1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 5 & -1 & -3 \\ -4 & 3 & 1 \\ -7 & 2 & 6 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 \\ 2 & -2 & 2 \\ -1 & 3 & 1 \end{bmatrix}$$

$$2 \times (-1) + 1 \times 3 + 1 \times 2 = 3$$

note:
A*B ≠ B*A

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

$$= \left[\begin{array}{c|c} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ \hline a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{array}\right]$$

Rule of thumb:

(NxK)x(KxM) --> NxM

A is NxK
B is KxM
then, A*B is NxM

vector-vector multiplication

$$[a_1, a_2, a_3]\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = a_1b_1 + a_2b_2 + a_3b_3$$

(1x3)x(3x1) --> 1x1 = scalar

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}[b_1, b_2, b_3] = \begin{bmatrix} a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_1 & a_2b_2 & a_2b_3 \\ a_3b_1 & a_3b_2 & a_3b_3 \end{bmatrix}$$

(3x1)x(1x3) --> 3x3

vector-vector multiplication

```
>> [1, 2, 3] * [2 -3 -1]'

ans =

    -7

>> [1, 2, 3]' * [2 -3 -1]

ans =

     2    -3    -1
     4    -6    -2
     6    -9    -3
```

row x column

column x row

solving linear systems

$$A x = b$$

Linear equations have a very large number of applications in engineering, science, social sciences and economics

Linear Programming – Management Science

Computer Aided Design – aerodynamics of cars, planes

Signal Processing in Communications and Control, Radar, Sonar, Electromagnetics, Oil Exploration, Computer Vision, Pattern & Face Recognition

Chip Design – millions of transistors

Economic Models, Finance, Statistical Models, Data Mining, Social Models

Markov Models – speech, biology, Google pagerank

Scientific Computing – solving very large problems

solving linear systems

$$\begin{aligned} 4x_1 + x_2 + 2x_3 &= 10 \\ x_1 - x_2 + x_3 &= 20 \\ 2x_1 + x_2 + x_3 &= 10 \end{aligned} \quad \Rightarrow \quad \begin{bmatrix} 4 & 1 & 2 \\ 1 & -1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 10 \\ 20 \\ 10 \end{bmatrix}$$

$$A\mathbf{x} = \mathbf{b}$$

matrix inverse

$$A\mathbf{x} = \mathbf{b} \quad \Rightarrow \quad \mathbf{x} = A^{-1}\mathbf{b} = A\backslash\mathbf{b}$$

always use the **`backslash`** operator to solve a linear system, instead of **`inv(A)`**

solving linear systems (using backslash)

$$\begin{aligned} 4x_1 + x_2 + 2x_3 &= 10 \\ x_1 - x_2 + x_3 &= 20 \\ 2x_1 + x_2 + x_3 &= 10 \end{aligned} \quad \Rightarrow \quad \begin{bmatrix} 4 & 1 & 2 \\ 1 & -1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 10 \\ 20 \\ 10 \end{bmatrix}$$

```
>> A = [4 1 2; 1 -1 1; 2 1 1];
>> b = [10 20 10]';
>> x = A\b
x =
   -30
    10
    60

>> norm(A*x-b)       % test - should be zero
ans =                % of the order of eps
     0
```

solving linear systems (using inv)

$$
\begin{aligned}
4x_1 + x_2 + 2x_3 &= 10 \\
x_1 - x_2 + x_3 &= 20 \\
2x_1 + x_2 + x_3 &= 10
\end{aligned}
\quad \Rightarrow \quad
\begin{bmatrix} 4 & 1 & 2 \\ 1 & -1 & 1 \\ 2 & 1 & 1 \end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}
=
\begin{bmatrix} 10 \\ 20 \\ 10 \end{bmatrix}
$$

```
>> A = [4 1 2; 1 -1 1; 2 1 1];
>> b = [10 20 10]';
>> inv(A)                        % same as A^(-1)
ans =
     2     -1     -3
    -1      0      2
    -3      2      5

>> x = inv(A) * b                % but prefer backslash
x =
   -30
    10
    60
```

solving linear systems – backslash and forwardslash

A of size **NxN** and invertible

X of size **NxK**

B of size **NxK**

equivalent

```
AX = B  -->  X = A\B = inv(A)*B
```

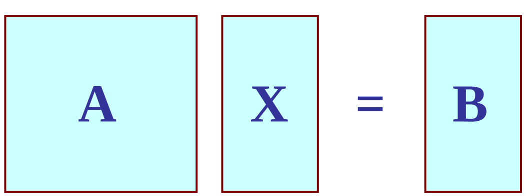
A of size **NxN** and invertible

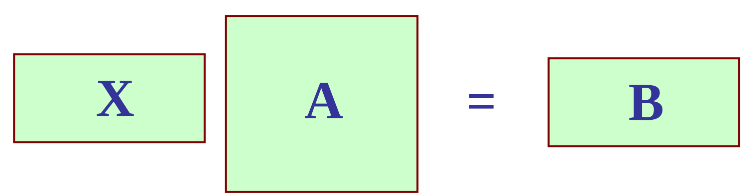
X of size **KxN**

B of size **KxN**

equivalent

```
XA = B  -->  X = B/A = B*inv(A)
```





solving linear systems – least-squares solutions

A of size **NxM**
x of size **Mx1** column
b of size **Nx1** column

```
x = A\b
```

will be discussed further in week-11

is a solution of **Ax=b** in a least-squares sense, i.e., **x** minimizes the norm squared:

(Ax-b)'*(Ax-b) = min

x may or may not be unique depending on whether the linear system **Ax=b** is over-determined, under-determined, or whether **A** has full rank or not

```
x = pinv(A)*b;
```

```
>> help \
>> help pinv
```

Invertibility, rank, determinants, condition number

The inverse **inv(A)** of an **NxN** square matrix **A** exists if its **determinant** is non-zero, or, equivalently if it has **full rank**, i.e., its **rank** is equal to the row or column dimension **N**

```
>> doc inv
>> doc det
>> doc rank
>> doc cond
```

```
a = [1 2 3]'; b = [4 5 6]';
A = [a, a+b, b]

A =
     1     5     4
     2     7     5
     3     9     6
```

det(A) = 0

```
>> det(A)

ans =
     0


>> rank(A)

ans =
     2
```

Invertibility, rank, determinants, condition number

The larger the **cond(A)** the more ill-conditioned the linear system, and the less reliable the solution.

```
A = [1,  5,            4
     2,  7 + 1e-8,     5
     3,  9,            6];
```

```
>> cond(A)
ans =
  3.3227e+009
```

```
A\[1; 2; 3]

ans =
     1
     0
     0
```

```
A\[1.001; 2.0002; 3.000003]

ans =
      30150.999185
     -30150.000183
      30150.000683
```

```
det(A) = -6.0000e-008
```

Determinant and inverse of a 2x2 matrix

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

$$\det(A) = ad - bc$$

Matrix Exponential

Used widely in solving linear dynamic systems

$$\exp(A) = \sum_{n=0}^{\infty} \frac{A^n}{n!} = 1 + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots$$

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
>> A = [1 2;3 4];

>> expm(A)      % matrix exponential
ans =
   51.9690   74.7366
  112.1048  164.0738

>> exp(A)       % element-wise exponential
ans =
    2.7183    7.3891
   20.0855   54.5982
```

```
>> doc expm
>> doc exp
```

Vector & Matrix Norms

L_1, L_2, and L_∞ norms of a vector

>> doc norm

can also be defined for matrices

$$\mathbf{x} = [x_1, x_2, \ldots, x_N]$$

$$\|\mathbf{x}\|_1 = \sum_{n=1}^{N} |x_n|$$

L_1 norm

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{n=1}^{N} |x_n|^2}$$

Euclidean, L_2 norm

$$\|\mathbf{x}\|_1 = \max\left(|x_1|, |x_2|, \ldots, |x_N|\right)$$

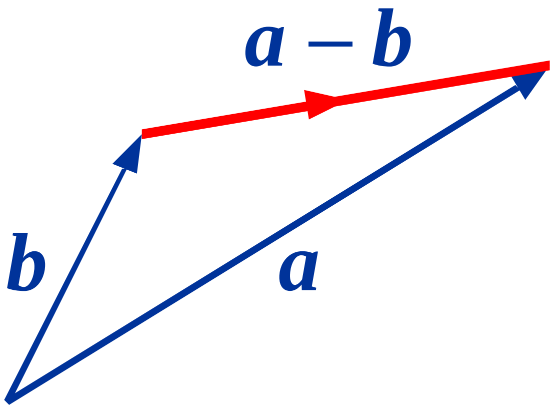
```
x = [1, -4, 5, 3]; p = inf;

switch p
   case 1
      N = sum(abs(x));                  % N = norm(x,1);
   case 2
      N = sqrt(sum(abs(x).^2));         % N = norm(x,2);
   case inf
      N = max(abs(x));                  % N = norm(x,inf);
   otherwise
      N = sqrt(sum(abs(x).^2));         % N = norm(x,2);
end
```

equivalent calculation using the built-in function **norm** :

useful for comparing two vectors or matrices

```
>> norm(a-b)     % a,b vectors of same size
>> norm(A-B)     % A,B matrices of same size
```

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$



$$\|\mathbf{a} - \mathbf{b}\|_2 = \mathrm{norm}(\mathbf{a} - \mathbf{b})$$

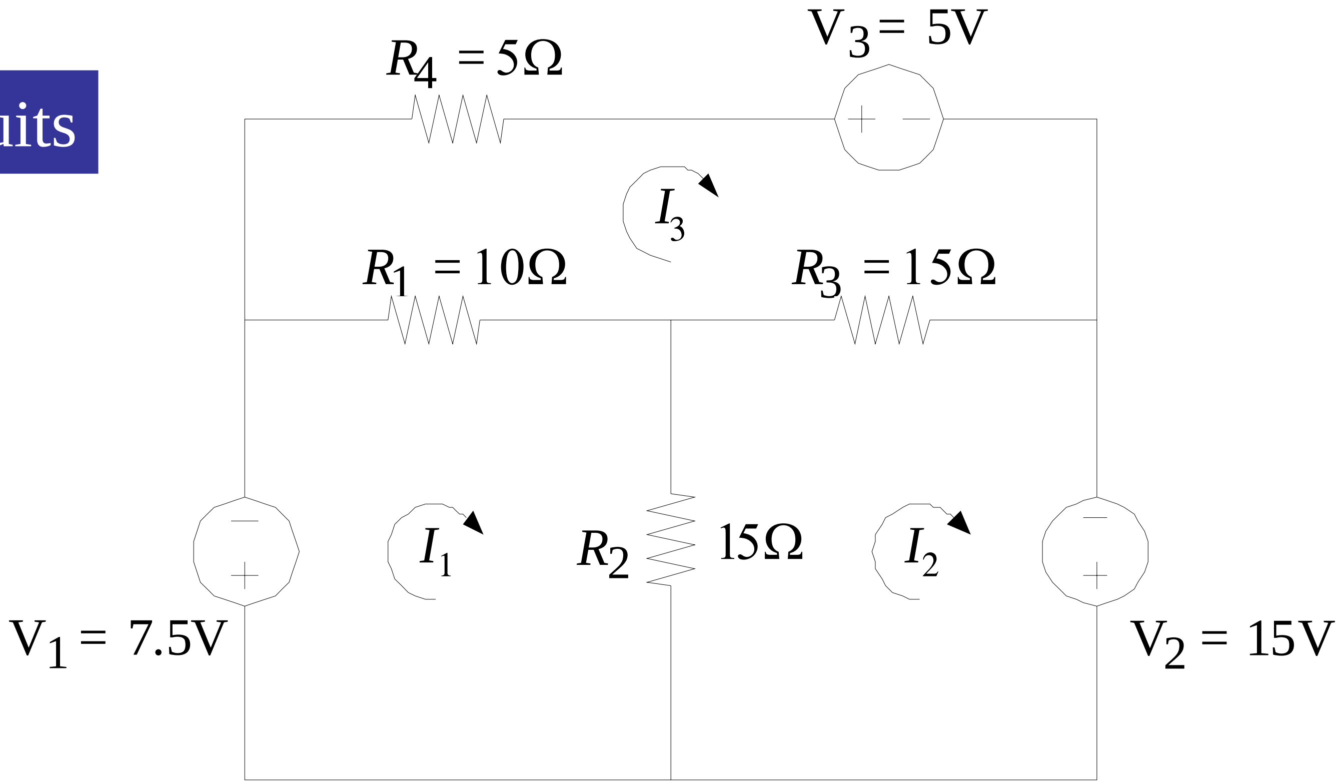
$$= \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + (a_3 - b_3)^2}$$

$$= \sqrt{(\mathbf{a} - \mathbf{b})'(\mathbf{a} - \mathbf{b})}$$

Euclidean distance

dot product

Electric Circuits



Kirchhoff's Voltage Law

$$R_1(I_1 - I_3) + R_2(I_1 - I_2) + V_1 = 0$$

$$R_2(I_2 - I_1) + R_3(I_2 - I_3) - V_2 = 0$$

$$R_4 I_3 + R_3(I_3 - I_2) + R_1(I_3 - I_1) + V_3 = 0$$

Electric Circuits

$$(R_1 + R_2)I_1 - R_2 I_2 - R_1 I_3 = -V_1$$

$$-R_2 I_1 + (R_2 + R_3)I_2 - R_3 I_3 = V_2$$

$$-R_1 I_1 - R_3 I_2 + (R_1 + R_3 + R_4)I_3 = -V_3$$

$$\begin{bmatrix} R_1 + R_2 & -R_2 & -R_1 \\ -R_2 & R_2 + R_3 & -R_3 \\ -R_1 & -R_3 & R_1 + R_3 + R_4 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} -V_1 \\ V_2 \\ -V_3 \end{bmatrix}$$

$$R_1 = 10, \quad R_2 = 15, \quad R_3 = 15, \quad R_4 = 5$$

$$V_1 = 7.5, \quad V_2 = 15, \quad V_3 = 10$$

$$\begin{bmatrix} R_1 + R_2 & -R_2 & -R_1 \\ -R_2 & R_2 + R_3 & -R_3 \\ -R_1 & -R_3 & R_1 + R_3 + R_4 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} -V_1 \\ V_2 \\ -V_3 \end{bmatrix}$$

$$\begin{bmatrix} 25 & -15 & -10 \\ -15 & 30 & -15 \\ -10 & -15 & 30 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} -7.5 \\ 15 \\ -5 \end{bmatrix}$$

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}$$

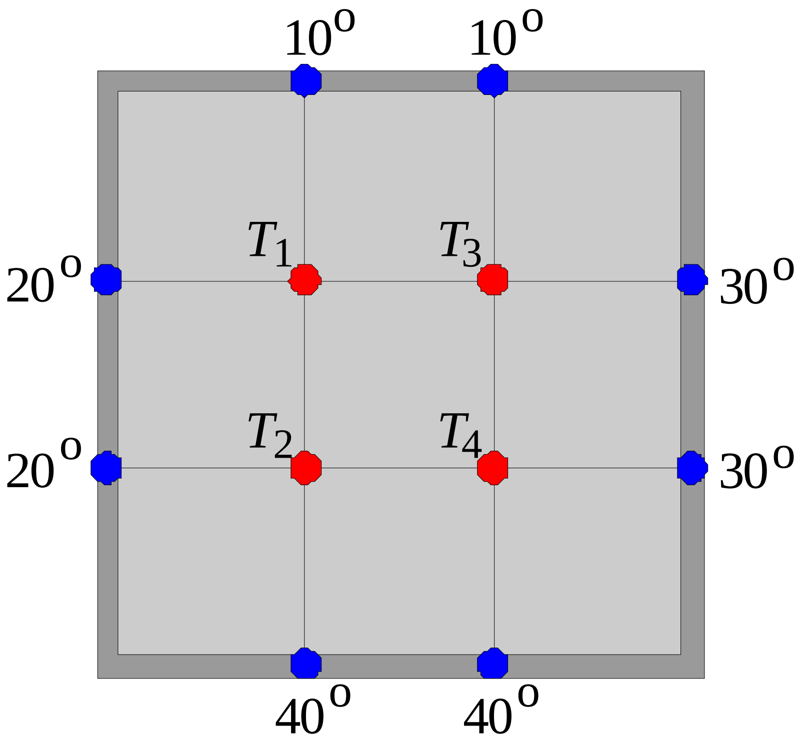
```
A = [25, -15, -10; -15, 30, -15; -10, -15, 30]
b = [-7.5; 15; -5]

A =
    25   -15   -10
   -15    30   -15
   -10   -15    30

b =
   -7.5000
   15.0000
   -5.0000

x = A\b

x =
    0.5000
    1.0000
    0.5000
```

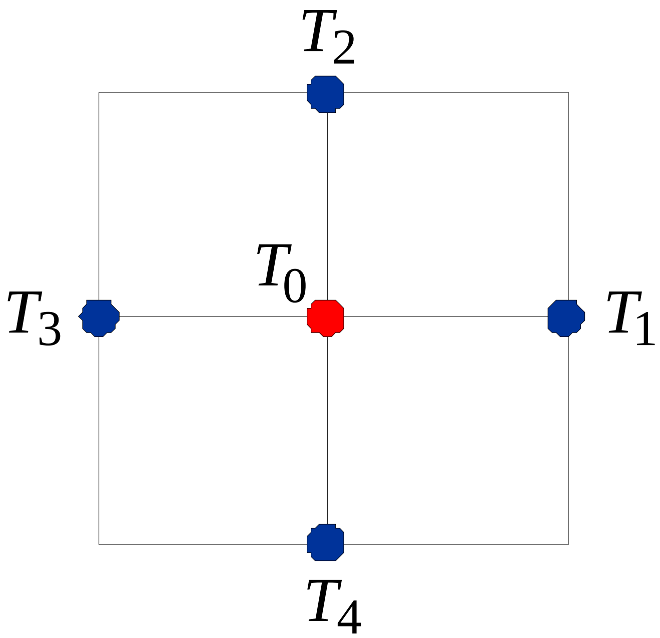
$$\mathbf{x} = \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 1.0 \\ 0.5 \end{bmatrix}$$

```
inv(A)

ans =
    0.2571    0.2286    0.2000
    0.2286    0.2476    0.2000
    0.2000    0.2000    0.2000

inv(sym(A)) --> (1/105) * [27   24   21
                           24   26   21
                           21   21   21]
```

$$\mathbf{x} = A^{-1}\mathbf{b} = \frac{1}{105}\begin{bmatrix} 27 & 24 & 21 \\ 24 & 26 & 21 \\ 21 & 21 & 21 \end{bmatrix}\begin{bmatrix} -7.5 \\ 15 \\ -5 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 1.0 \\ 0.5 \end{bmatrix}$$

Temperature Distribution

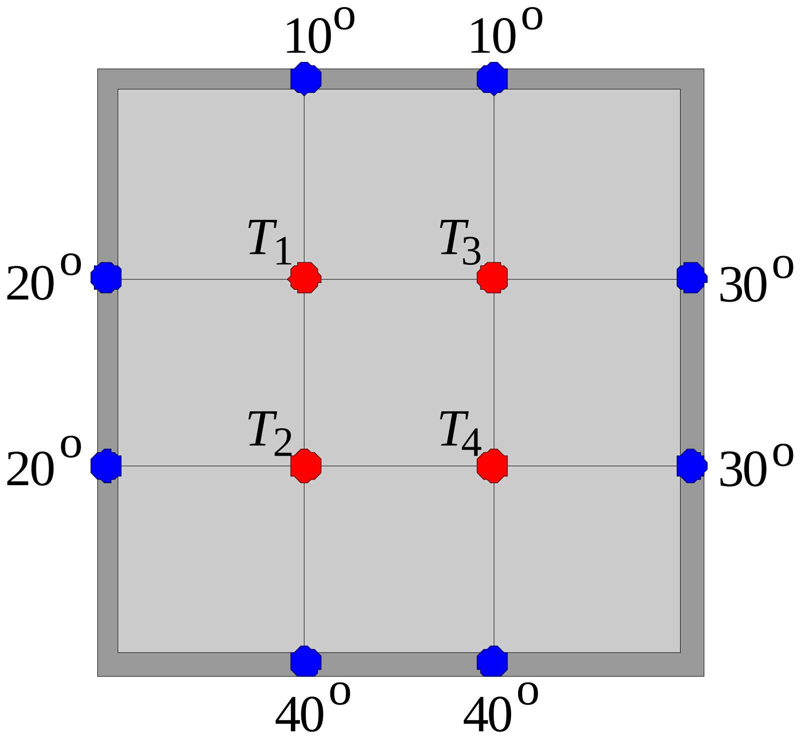




$$T_0 = \frac{1}{4}(T_1 + T_2 + T_3 + T_4)$$

follows from discretizing the Laplace equation

$$\nabla^2 T = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$



$$T_1 = \frac{1}{4}(10 + 20 + T_2 + T_3)$$

$$T_2 = \frac{1}{4}(20 + 40 + T_1 + T_4)$$

$$T_3 = \frac{1}{4}(10 + 30 + T_1 + T_4)$$

$$T_4 = \frac{1}{4}(30 + 40 + T_2 + T_3)$$

$$4T_1 - T_2 - T_3 = 30$$

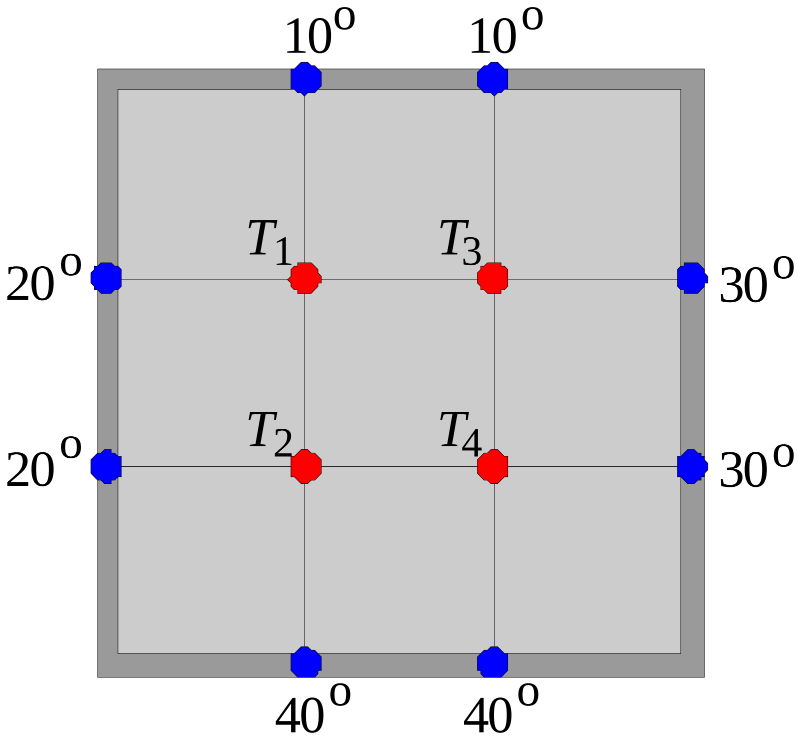
$$4T_2 - T_1 - T_4 = 60$$

$$4T_3 - T_1 - T_4 = 40$$

$$4T_4 - T_2 - T_3 = 70$$

$$\begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} 30 \\ 60 \\ 40 \\ 70 \end{bmatrix}$$

$$\mathbf{A}\ \mathbf{x} = \mathbf{b}$$

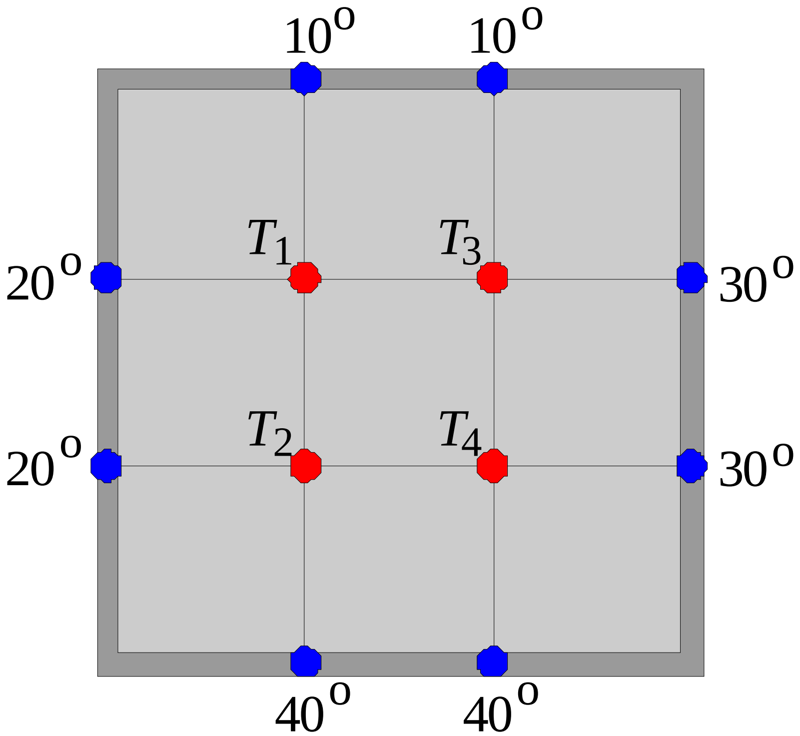


$$\begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} 30 \\ 60 \\ 40 \\ 70 \end{bmatrix}$$

A x = b

```
>> A = [ 4 -1 -1  0
        -1  4  0 -1
        -1  0  4 -1
         0 -1 -1  4];
>> b = [30; 60; 40; 70];
```

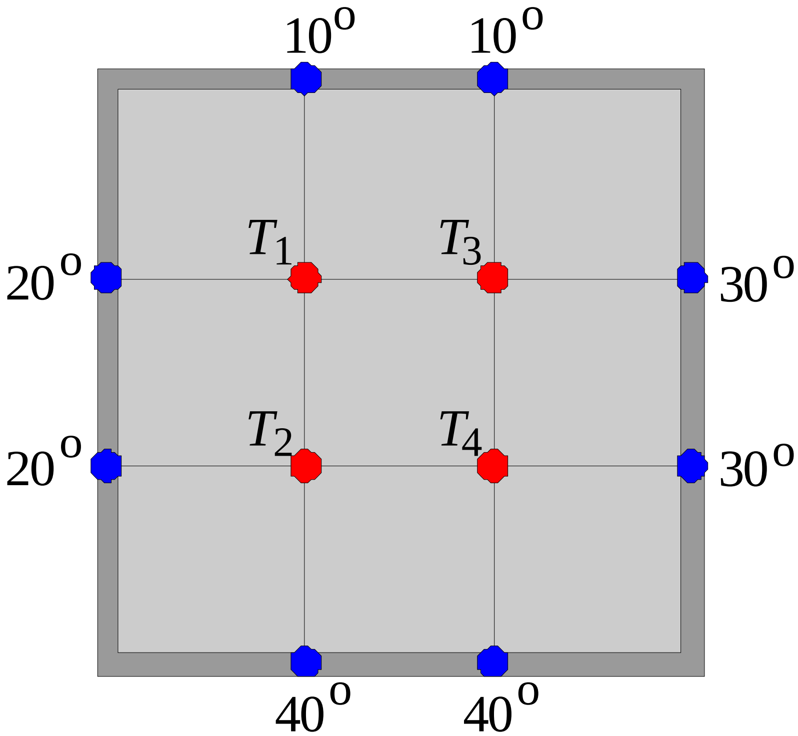
```
>> x = A\b

x =

   20.0000
   27.5000
   22.5000
   30.0000
```



display solution **T** in a rectangular pattern

nodes were numbered in column order

```
T = zeros(2,2);  % shape of T
T(:) = x

T =
   20.0000   22.5000
   27.5000   30.0000
```



$$\begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} 30 \\ 60 \\ 40 \\ 70 \end{bmatrix}$$

A x = b

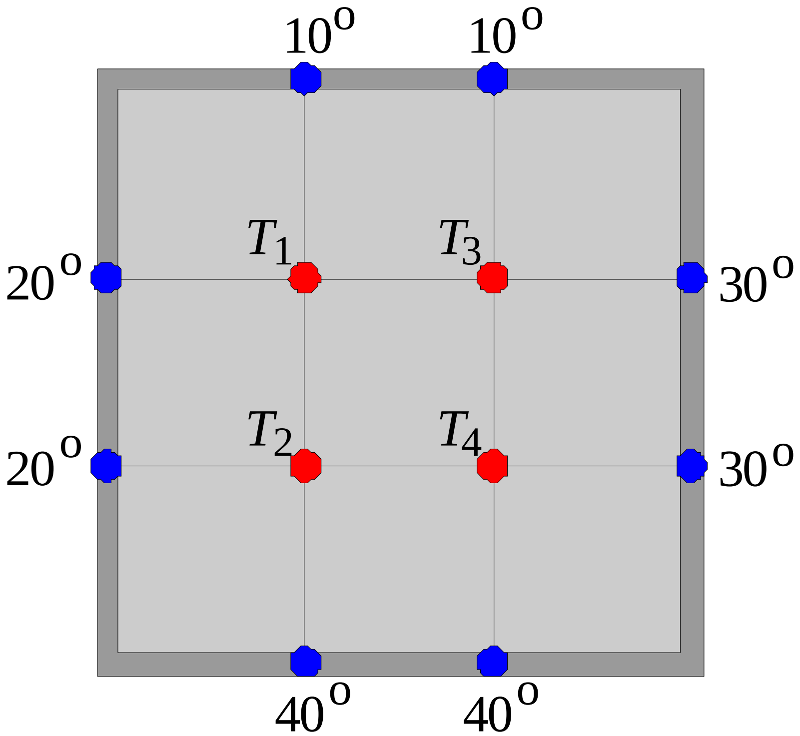
Rules for constructing **A** and **b**:

1) main diagonal is 4

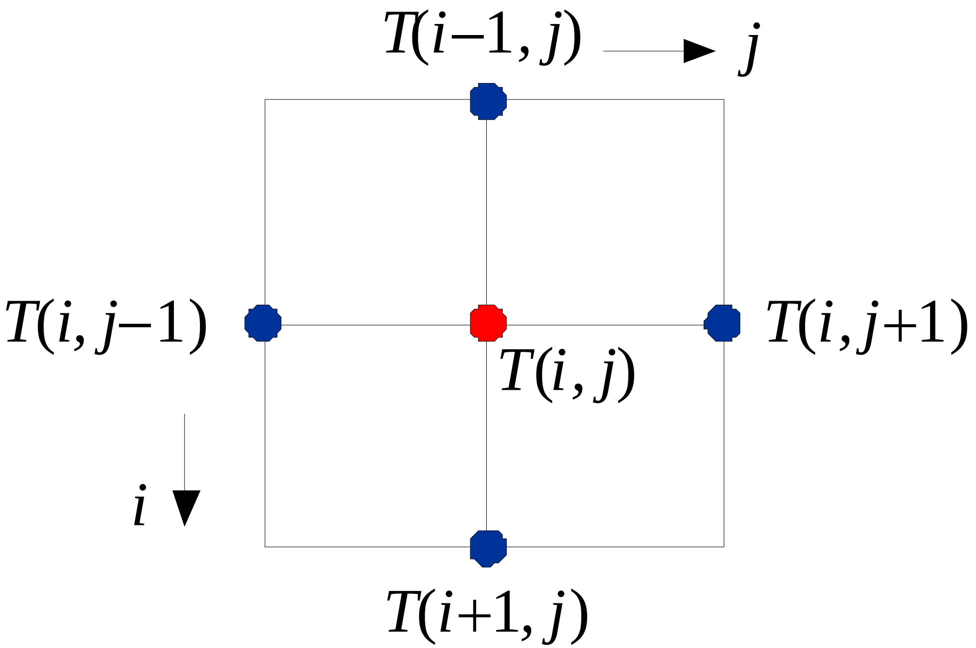
2) if nodes **i,j** are connected, then, -1, otherwise, 0

3) **b(i)** is sum of boundary values connected to node **i**

Iterative Solution



convenient for large number of subdivisions



used also to solve 2D electrostatics problems

$$T(i,j) = \frac{1}{4}\left[T(i+1,j) + T(i-1,j) + T(i,j+1) + T(i,j-1)\right]$$

```
N=4; M=4;
left=20; right=30; up=10; dn=40;
T(1,:) = repmat(up,1,M);   T(N,:) = repmat(dn,1,M);
T(:,1) = repmat(left,N,1); T(:,M) = repmat(right,N,1);

Tnew = T;

tol = 1e-4; K = 100;
for k=1:K,
  for i=2:N-1,
    for j=2:M-1,
      Tnew(i,j) = (T(i-1,j) + T(i+1,j) +...
                   T(i,j-1) + T(i,j+1))/4;
    end
  end
  if norm(Tnew-T) < tol, break; end
  T = Tnew;
end

T(1,[1,end]) = nan; T(end,[1,end]) = nan;
```

boundary values

iterate over internal nodes only

```
T =                                  % start-up
   NaN     10     10    NaN
    20      0      0     30
    20      0      0     30
   NaN     40     40    NaN

% converged after k = 19 iterations
% to within the specified tol = 1e-4

T =
       NaN   10.0000   10.0000       NaN
   20.0000   19.9999   22.4999   30.0000
   20.0000   27.4999   29.9999   30.0000
       NaN   40.0000   40.0000       NaN
```

```
T =              % after k=1 iteration
       NaN   10.0000   10.0000       NaN
   20.0000    7.5000   10.0000   30.0000
   20.0000   15.0000   17.5000   30.0000
       NaN   40.0000   40.0000       NaN

T =              % after k=2 iterations
       NaN   10.0000   10.0000       NaN
   20.0000   13.7500   16.2500   30.0000
   20.0000   21.2500   23.7500   30.0000
       NaN   40.0000   40.0000       NaN

T =              % after k=3 iterations
       NaN   10.0000   10.0000       NaN
   20.0000   16.8750   19.3750   30.0000
   20.0000   24.3750   26.8750   30.0000
       NaN   40.0000   40.0000       NaN
```

```
N=30; M=30;
left=0; right=0; up=0; dn=60;
tol = 1e-6; K = 5000;

% breaks out at k = 2475

[X,Y] = meshgrid(2:M-1, 2:N-1);

Z = T(2:M-1, 2:N-1);
surf(X,Y,Z);
```