Introduction to Applied Scientific Computing using MATLAB

Mohsen Jenadeleh

In this lecture, slides from MIT, Rutgers and Waterloo University are used to form the lecture slides

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
- iterative solutions of linear systems
- examples
- electric circuits
- temperature distributions

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
```

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

dot product

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

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

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

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

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

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

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

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

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

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

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

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

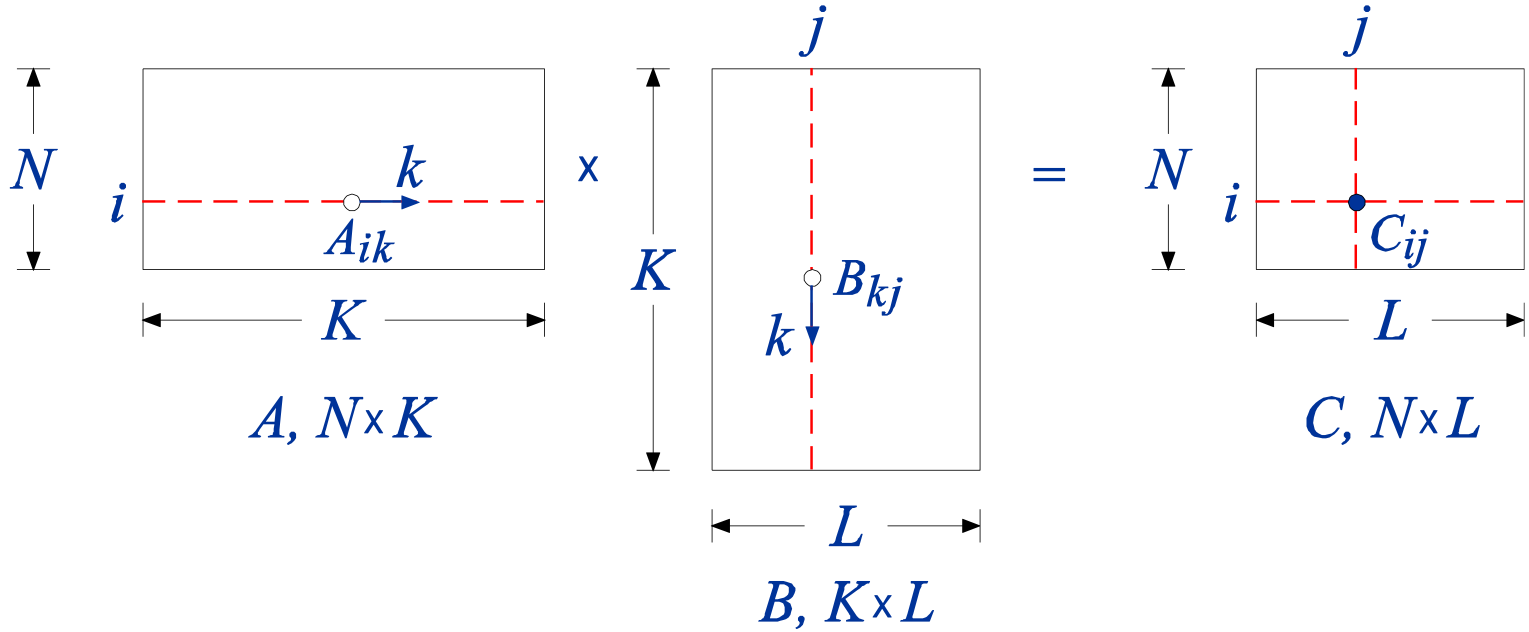
$$\begin{bmatrix} 4 & 1 & 2 \\ 1 & -1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \\ 3 \end{bmatrix}$$

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

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

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

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

row * column = scalar

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} [b_1, b_2, b_3] = \begin{bmatrix} a_1 b_1 & a_1 b_2 & a_1 b_3 \\ a_2 b_1 & a_2 b_2 & a_2 b_3 \\ a_3 b_1 & a_3 b_2 & a_3 b_3 \end{bmatrix}$$

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

column * row = matrix

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 = scalar

column x row = matrix

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, Communications, Control, Radar, Sonar, Electromagnetics, Oil Exploration, Computer Vision, Pattern & Face Recognition

Chip Design – millions of transistors on a chip

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

Markov Models – Speech, Biology, Google Pagerank

Scientific Computing – solving very large problems

the only practical way to solve very large systems is iteratively

solving linear systems

$$\begin{aligned} 2x_1 + x_2 &= 4 \\ x_1 + 5x_2 - x_3 &= 8 \\ x_1 - 2x_2 + 4x_3 &= 9 \end{aligned} \quad \Rightarrow \quad \begin{bmatrix} 2 & 1 & 0 \\ 1 & 5 & -1 \\ 1 & -2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 8 \\ 9 \end{bmatrix}$$

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

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)

$$\begin{aligned} 2x_1 + x_2 &= 4 \\ x_1 + 5x_2 - x_3 &= 8 \\ x_1 - 2x_2 + 4x_3 &= 9 \end{aligned} \quad \Rightarrow \quad \begin{bmatrix} 2 & 1 & 0 \\ 1 & 5 & -1 \\ 1 & -2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 8 \\ 9 \end{bmatrix}$$

```
>> A = [2 1 0; 1 5 -1; 1 -2 4];
>> b = [4 8 9]';
>> x = A\b            % solution of A*x = b
x =                   % x = A^-1 * b
    1                 % x = inv(A) * b
    2
    3

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

solving linear systems (using inv)

```
>> A = [2 1 0; 1 5 -1; 1 -2 4];
>> b = [4 8 9]';
>> inv(A)                        % same as A^(-1)
ans =
    0.5806   -0.1290   -0.0323
   -0.1613    0.2581    0.0645
   -0.2258    0.1613    0.2903

>> x = inv(A) * b                % but prefer backslash
x =                              % same as x = A^-1 * b
    1.0000
    2.0000
    3.0000

>> norm(A*x-b)
ans =
  1.8310e-015
```

```
>> inv(sym(A))
ans =
[ 18/31, -4/31, -1/31]
[ -5/31,  8/31,  2/31]
[ -7/31,  5/31,  9/31]
```

solving linear systems – back-slash and forward-slash

A of size **NxN** and invertible

X of size **NxK**

B of size **NxK**

equivalent

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

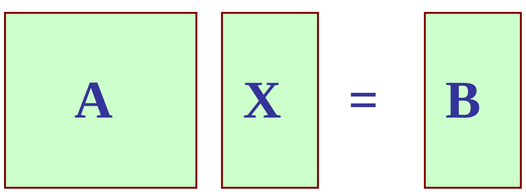
A of size **NxN** and invertible

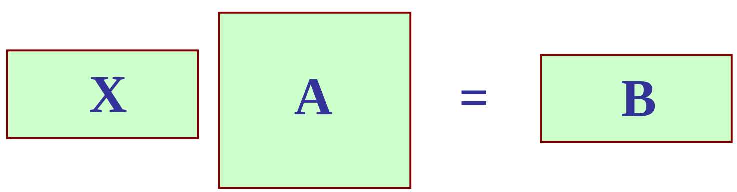
X of size **KxN**

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
```

pseudo-inverse

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

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

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

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

least-squares solutions - summary

Fundamental Theorem of Linear Algebra – what is it?

`A = NxM` matrix — `A' = MxN` matrix

`x = Mx1` column — `A'*A = MxM` matrix

`b = Nx1` column — `A'*b = Mx1` column

Assuming full rank for **A**, we have the following cases:

1. `N>M`, overdetermined case, (most common in practice)
   `x = A\b =` unique least-squares solution, same as
   `x = pinv(A)*b`, and
   `x = (A'*A)^(-1) * (A'*b)`

`x = A\b` is numerically the most accurate method

2. `N<M`, underdetermined case, (there are many solutions)
   `x=A\b`, `x=pinv(A)*b`, are two possible solutions

3. `N=M`, square invertible case, **x** is unique
   `x = A\b` is equivalent to `x = A^(-1)*b`

least-squares solutions - example

```
% overdetermined
% full-rank example

A = [1 2; 3 4; 5 6]
b = [4, 3, 8]';

x = A\b
% x = pinv(A)*b
% x = (A'*A)\(A'*b)

x =
    -1
     2
```

overdetermined system of

3 equations in 2 unknowns

$$x_1 + 2x_2 = 4$$
$$3x_1 + 4x_2 = 3$$
$$5x_1 + 6x_2 = 8$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 8 \end{bmatrix}$$

$$e = b - Ax = \begin{bmatrix} 4 \\ 3 \\ 8 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4 - x_1 - 2x_2 \\ 3 - 3x_1 - 4x_2 \\ 8 - 5x_1 - 6x_2 \end{bmatrix} = \text{error}$$

least-squares solutions - example

$$J = e^T e = (b - Ax)^T (b - Ax) = x^T (A^T A) x - 2x^T (A^T b) + b^T b = \min$$

$$\frac{\partial J}{\partial x} = 2A^T (Ax - b) = 0 \quad \Rightarrow \quad x_{\text{opt}} = (A^T A)^{-1} A^T b$$

inverse exists because ***A*** was assumed to have full rank

$$J_{\min} = J|_{x = x_{\text{opt}}} = b^T b - b^T A (A^T A)^{-1} A^T b$$

minimized value of ***J*** achieved at ***x* = *x*_opt**

least-squares solutions - example

```
A = [1 2; 3 4; 5 6]
b = [4, 3, 8]';

x_opt = (A'*A)\(A'*b)

J_min = b'*b - ...
  b'*A*inv(A'*A)*A'*b

x_opt =
      -1
       2
J_min =
       6
```

$$A^TA = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} 35 & 44 \\ 44 & 56 \end{bmatrix}$$

$$A^Tb = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 8 \end{bmatrix} = \begin{bmatrix} 53 \\ 68 \end{bmatrix}$$

$$b^Tb = [4,\ 3,\ 8] \begin{bmatrix} 4 \\ 3 \\ 8 \end{bmatrix} = 89$$

$$x_{\mathrm{opt}} = (A^TA)^{-1}A^Tb = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, \quad J_{\min} = b^Tb - b^TA^T(A^TA)^{-1}Ab = 6$$

least-squares solutions - example

$$e = b - Ax = \begin{bmatrix} 4 \\ 3 \\ 8 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4 - x_1 - 2x_2 \\ 3 - 3x_1 - 4x_2 \\ 8 - 5x_1 - 6x_2 \end{bmatrix} = \text{error}$$

$$J = (4 - x_1 - 2x_2)^2 + (3 - 3x_1 - 4x_2)^2 + (8 - 5x_1 - 6x_2)^2$$

$$= [x_1, x_2] \begin{bmatrix} 35 & 44 \\ 44 & 56 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - 2 \cdot [53,\ 68] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 89$$

$$= 35x_1^2 + 88x_1x_2 + 56x_2^2 - 106x_1 - 136x_2 + 89$$

$$= 35(x_1 + 1)^2 + 88(x_1 + 1)(x_2 - 2) + 56(x_2 - 2)^2 + 6$$

$$= [x_1 + 1,\ x_2 - 2] \begin{bmatrix} 35 & 44 \\ 44 & 56 \end{bmatrix} \begin{bmatrix} x_1 + 1 \\ x_2 - 2 \end{bmatrix} + 6\,, \quad x - x_{\text{opt}} = \begin{bmatrix} x_1 + 1 \\ x_2 - 2 \end{bmatrix}$$

least-squares solutions - example

$$J = 35(x_1+1)^2 + 88(x_1+1)(x_2-2) + 56(x_2-2)^2 + 6 \geq 6$$

J is minimized at $x_1 = -1$, $x_2 = 2$, with minimum value, $J = 6$

```
% we can also minimize J with fminsearch,
% i.e., the multivariable version of fminbnd

J = @(x) 35*(x(1)+1).^2 +...
         88*(x(1)+1).*(x(2)-2)+...
         56*(x(2)-2).^2 + 6;

x0 = [0,0]';    % arbitrary initial search point

[xmin,Jmin] = fminsearch(J,x0)

% xmin =              % Jmin = 6
%      -1.0000
%       2.0000
```

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., when 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.

The *condition number* of a matrix measures the sensitivity of the solution of a system of linear equations to errors in the data

```
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

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

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

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

Example: $$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-1} = \frac{1}{4-6} \begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ 1.5 & -0.5 \end{bmatrix}$$

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

>> doc norm

L_1, L_2, and L_∞ norms of a vector

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

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

L_1 norm

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

Euclidean, L_2 norm

$$\|\mathbf{x}\|_\infty = \max(|x_1|, |x_2|, \ldots, |x_N|)$$

L_∞ norm

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$$

used as distance measure between two vectors or matrices

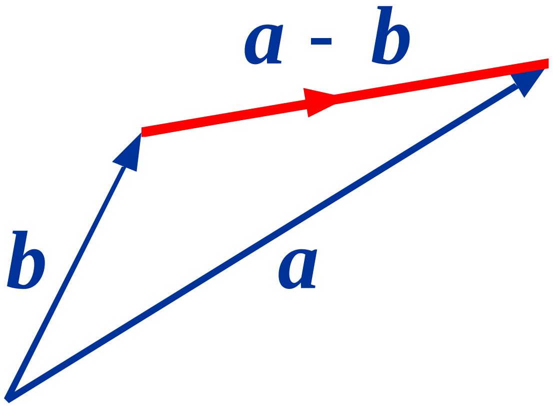
```
x = [1, -4, 5, 3]; p = inf;

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
```

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



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

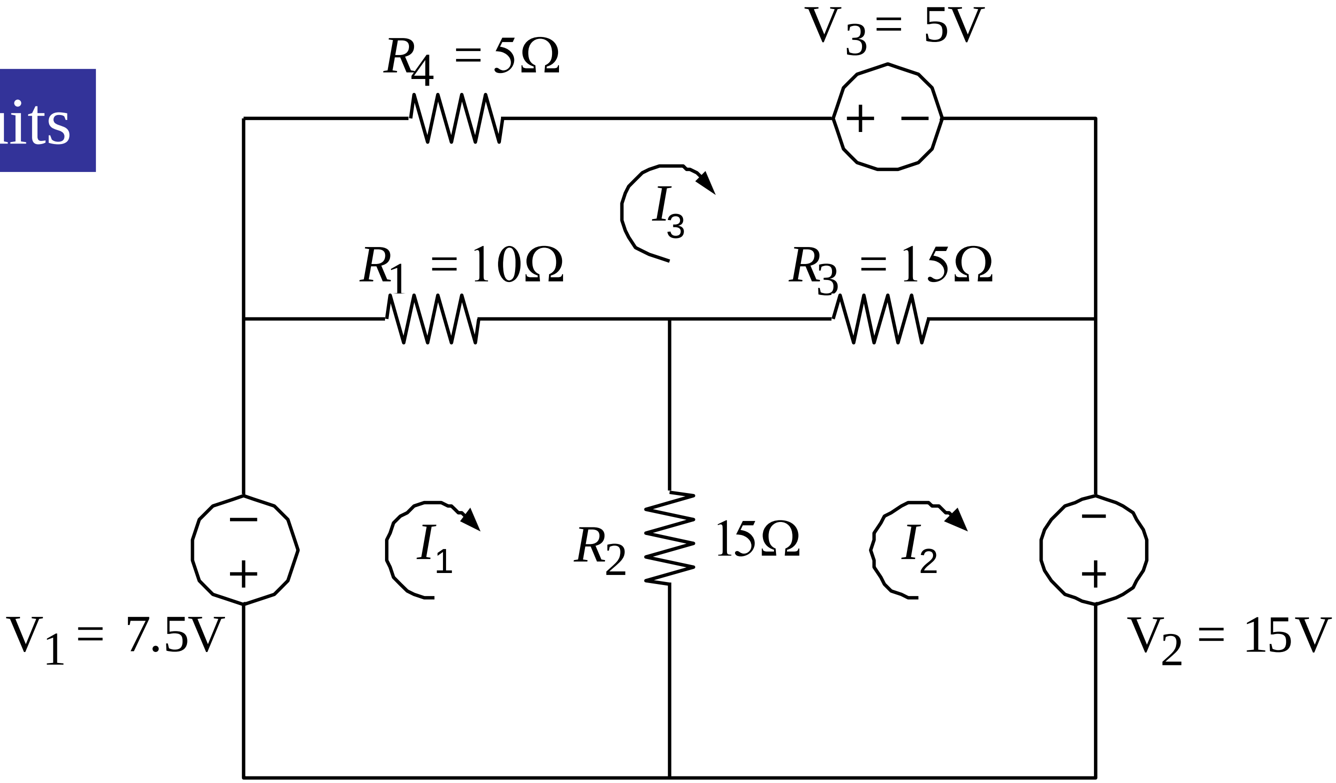
$$= \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + (a_3 - b_3)^2}$$

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

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]
```

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

Iterative solutions of linear systems **Ax=b**

the only practical way to solve very large linear systems is iteratively

Methods:

1. Jacobi method
2. Gauss-Seidel method
3. Relaxation methods
4. Conjugate Gradient method
5. Others

G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3/e, JHU Press, 1996.
D. S. Watkins, *Fundamentals of Matrix Computations*, 2/e, Wiley, 2002.
L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM, 1997.
A. Bjork, *Numerical Methods for Least Squares Problems*, SIAM, 1996.

scalar example illustrating the Jacobi method

rearrange

$$3x = 12$$

$$2x + x = 12$$

rearrange

$$2x = -x + 12$$

$$x = -0.5\,x + 6$$

turn it into a recursion

for $k = 1, 2, 3, \ldots$

$$x(k+1) = -0.5\,x(k) + 6$$

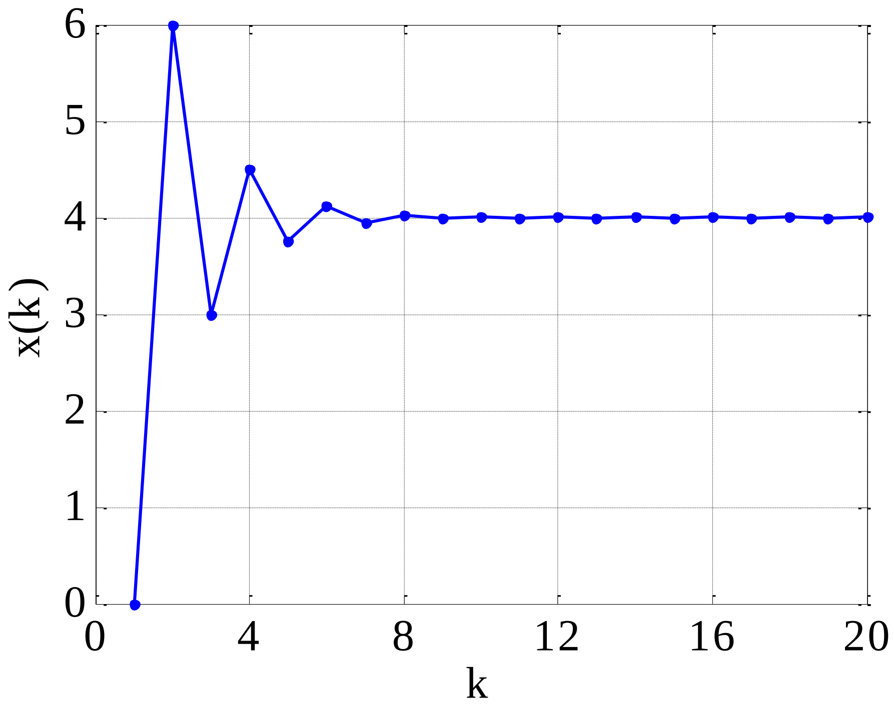
start with any $x(1)$,

$$x(2) = -0.5x(1) + 6$$

$$x(3) = -0.5x(2) + 6$$

$$x(4) = -0.5x(3) + 6, \quad \text{etc.}$$

```
x(1)=0;        % arbitrary
for k=1:19
  x(k+1) = -0.5*x(k) + 6;
end

k=1:20; plot(k,x,'b.-');
```




```
>> [k; x]'
    1   0.0000
    2   6.0000
    3   3.0000
    4   4.5000
    5   3.7500
    6   4.1250
    7   3.9375
    8   4.0313
    9   3.9844
   10   4.0078
   11   3.9961
   12   4.0020
   13   3.9990
   14   4.0005
   15   3.9998
   16   4.0001
   17   3.9999
   18   4.0000
   19   4.0000
   20   4.0000
```

```
tol=1e-10; x0=0;

x=x0; k=1;

while 1
    xnew = -0.5*x + 6;
    if abs(xnew-x)<=tol
        break;
    end
    x = xnew;
    k = k+1;
end

k, abs(x-4)

k =
    37
ans =
  5.8208e-011
```

forever while loop

```
tol=1e-10; x0=0;

x=x0; k=1;
xnew = -0.5*x+6;

while abs(xnew-x)>tol
    x = xnew;
    k = k+1;
    xnew = -0.5*x + 6;
end

k, abs(x-4)

k =
    37
ans =
  5.8208e-011
```

conventional while loop