LEBANESE AMERICAN UNIVERSITY
School of Engineering
Department of Electrical and Computer Engineering

ELE443 Control System LAB
Fall 2013

Lecture 2: Array Mathematical Operations, Random Numbers & Symbolic Math

Joe Khalifeh

Addition & Subtraction

- Addition and subtraction are performed between matrices having the same dimensions.
  - A=[7 5 9;2 3 6]
    A =
    7 5 9
    2 3 6
  - B=[4 8 1;3 5 4]
    B =
    4 8 1
    3 5 4
  - C=A+B
    C =
    11 13 10
    5 8 10

    D=A-B
    D =
    3 -3 8
    -1 -2 2

Must have the same dimensions

Array Multiplication

- Matrix multiplication is defined for matrices A and B such that the number of columns of A is equal to the number of rows of B.
- Let A be m-by-n matrix and B is a p-by-q matrix. The matrix multiplication A*B is defined iff n=p.
- In this case, the resulting **matrix C=A*B is an m-by-q matrix.**
- **Note that matrix multiplication** is not commutative.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \qquad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}$$

$$A * B = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} \end{bmatrix}$$

Array Multiplication

- Consider the following Example:

  - A=[7 8 9;3 2 8;5 4 -2];
    B=[1;-9;3];

    C=A*B

    C = -38
            9
          -37

A(3x3)*B(3x1)
=C(3x1)

Inverse of a matrix

- The inverse of matrix M is denoted by M^{-1} and is defined such that $M*M^{-1}= M^{-1}*M=I$ where I is the identity matrix.
- M is invertible (i.e. M^{-1} exists) iff M is nonsingular (i.e. it has a nonzero determinant)
- The inverse of M is calculated using the command inv(M).
- Consider the following examples

A=[1 0 0;0 2 1;2 -2 1]

$$A = \begin{matrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 2 & -2 & 1 \end{matrix}$$

det(A)

ans =4

inv(A)

$$Ans = \begin{matrix} 1 & 0 & 0 \\ 0.5 & 0.25 & -0.25 \\ -1 & 0.5 & 0.5 \end{matrix}$$

A*inv(A)

$$ans = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

E-B-E Operations

- Element by element E-B-E operations are performed by typing a dot (.) before the operator.
- Addition and subtraction are E-B-E operations.
- E-B-E multiplication, division and power are done between matrices having the same dimensions.

$$A = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \quad B = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix}$$

$$A.*B = \begin{bmatrix} a_1 b_1 & a_2 b_2 & a_3 b_3 \end{bmatrix}$$

$$A./B = \begin{bmatrix} a_1 / b_1 & a_2 / b_2 & a_3 / b_3 \end{bmatrix}$$

$$A.\wedge B = \begin{bmatrix} a_1^{\,b_1} & a_2^{\,b_2} & a_3^{\,b_3} \end{bmatrix}$$

Array Analysis

Function	Description	Example
mean(V)	For vectors, MEAN(V) is the mean value of the elements in V. if V is a matrix then MEAN(V) will return a vector containing the mean of each row)	V=[5 8 9 10]; mean(V)=8 A=[1 3;2 1] mean(V)=[2 1.5]
mx=max(M) mn=min(M)	For vectors, mx (mn) is the largest (smallest) element in X. For matrices, mx (mn) is a row vector containing the maximum (minimum) element from each column.	M = 2 3 4 5 8 9 -1 0 7 max(M) ans =5 8 9
rank(M)	provides an estimate of the number of linearly independent rows or columns of a matrix A.	M=[1 7 8;2 6 4]; rank(M) ans =2

Array Analysis

Function	Description	Example
rref(M)	produces the reduced row echelon form of A.	M=[1 7 8;2 6 4]; rref(M) ans = 1 0 -2.5 0 1 1.5
[V,D] = eig(X)	produces a diagonal matrix V of eigenvalues and a full matrix D whose columns are the corresponding eigenvectors.	A=[1 3;2 1] [D,V]=eig(A) D = 0.7746 -0.7746 0.63246 0.63246 V = 3.4495 0 0 -1.4495
sum(X)	is the sum of the elements of vector X.	X=[1 2 3]; S=sum(X) S =6

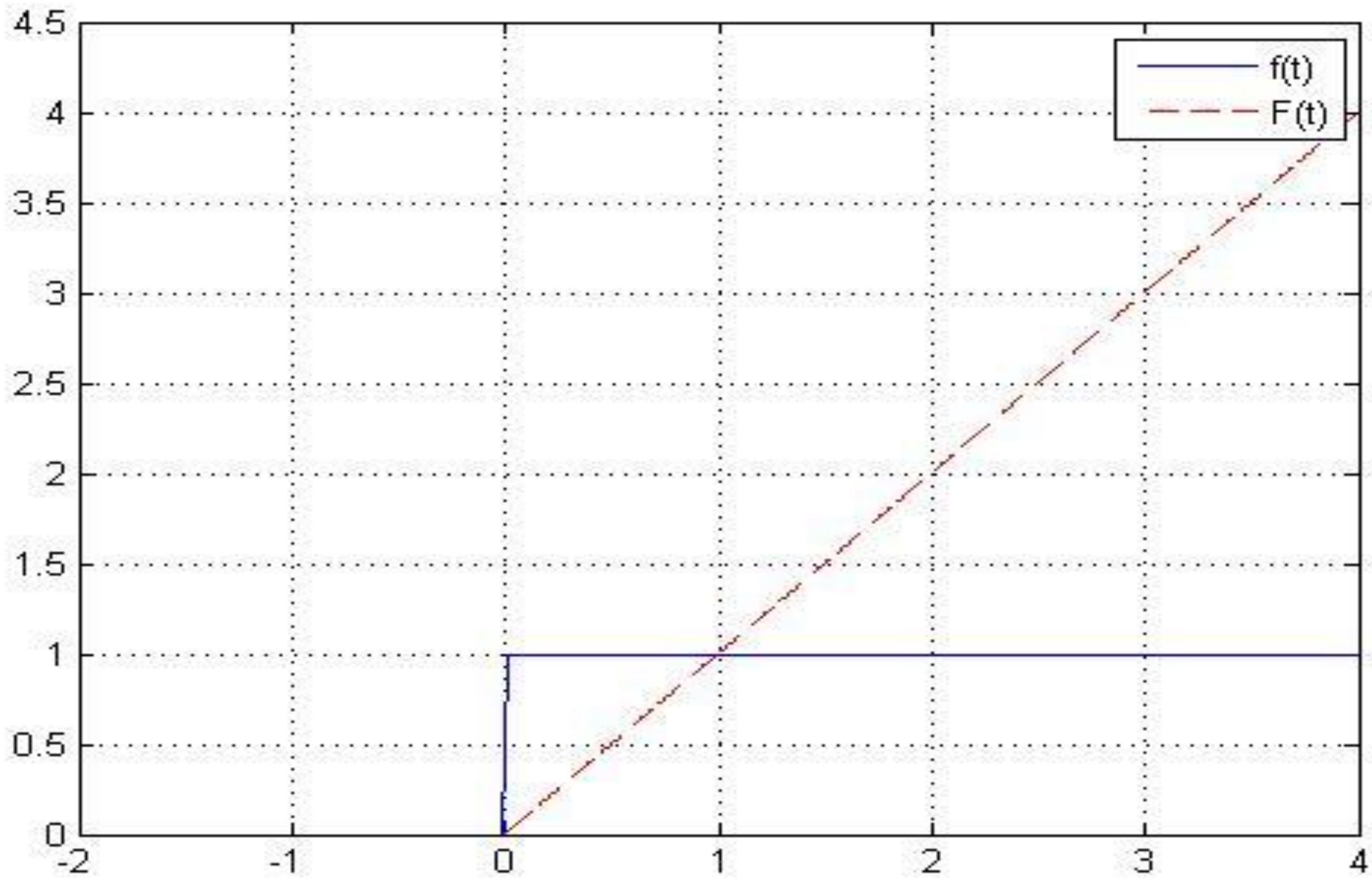
Array Analysis (Application)

Function	Description	Example

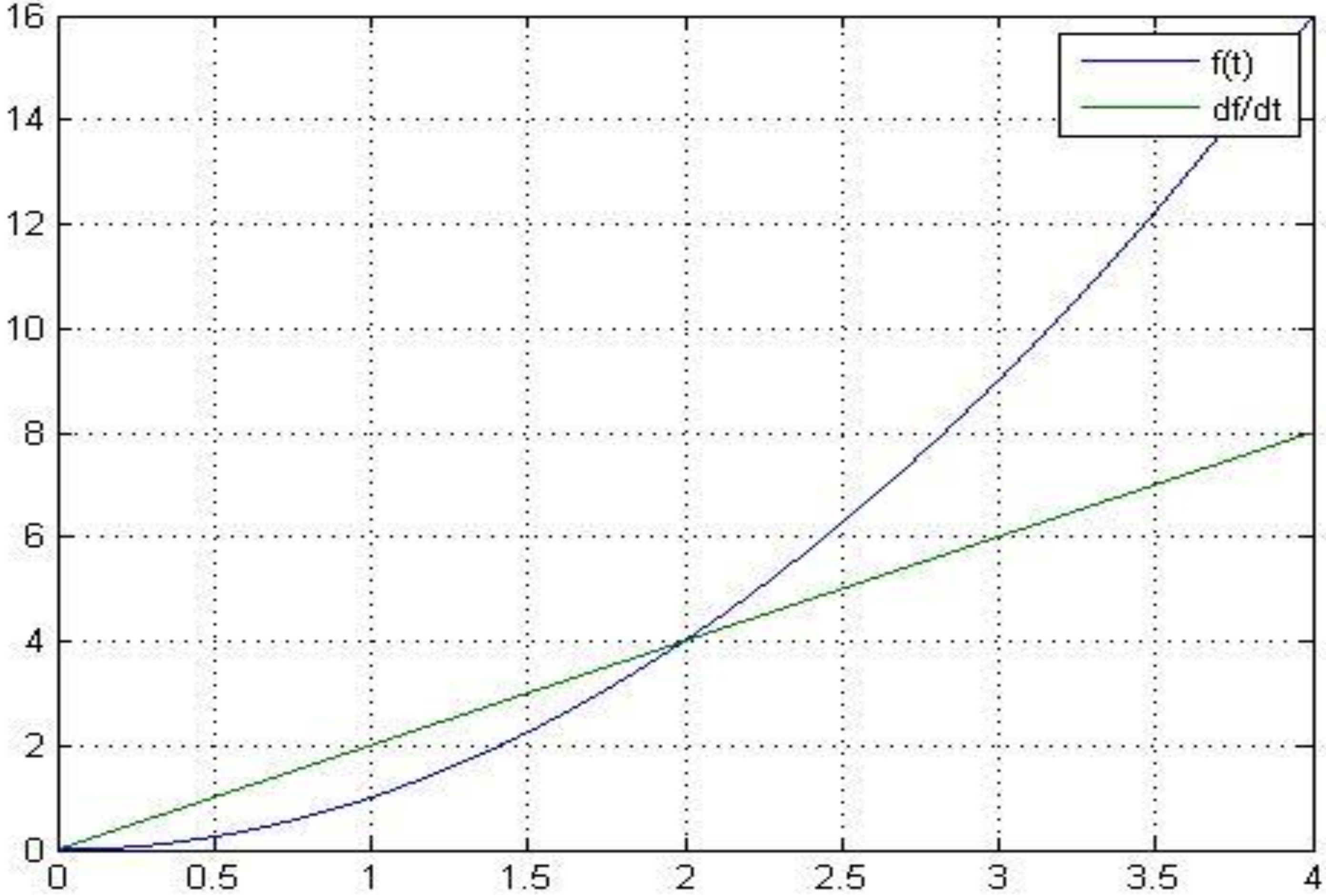
cumsum(x)	returns the cumulative sum along different dimensions of an array.
	It can be used for numerical integration.
		dt=0.01;
		t=-2:dt:2;
		f=0.5*(sign(t)+1);
		F=cumsum(f)*dt;

diff(x)	calculates differences between adjacent elements of x.
	It can be used for numerical differentiation.
	<u>Note:</u> if size(x,2)=n, then size(diff(x),2)=n-1.
		dt=1e-3;
		t=0:dt:4;
		f=t.^2;
		Df=diff(f)/dt;
		t_d=t(1:size(t,2)-1);
		plot(t,f,t_d,Df),grid

Numerical Integration of a function



Numerical differentiation of a function



Array Analysis

Function	Description	Example
expm(M)	is the matrix exponential of the square matrix X.	M=[2 1;5 -4]; expm(M) ans =13.976 2.0718 10.359 1.5453
exp(M)	computes the exponential of X element-by-element	M=[2 1;5 -4]; exp(M) ans = 7.3891 2.7183 148.41 0.018316

Array Analysis

- Note that matrix exponential (expm in MATLAB) is computed according to Taylor series expansion of exponential function, where the operand is a matrix and not a scalar.

$$\exp(M) = I + M + \frac{M^2}{2!} + \frac{M^3}{3!} + \ldots = \sum_{i=0}^{\infty} \frac{M^i}{i!}$$

- That's why expm is only applied to square matrices because it's a linear combination of power of matrices.

Array Analysis

- **std(V)**: returns the standard deviation of the elements of vector V.
- **median(V)**: For vectors, median(V) is the median value of the elements in V.
- **sort(X)**: For vectors, it sorts the elements of X in ascending order. For matrices, it sorts each column of X in ascending order.
- **det(M)**: returns the determinant of the matrix M.
- **dot(V1,V2)**: returns the scalar product of the vectors V1 and V2
- **cross(V1,V2)**: returns the cross product of the vectors V1 and V2.

Random numbers

- **"rand"** function generates uniformly distributed pseudo-random numbers.
  - random numbers are between 0 and 1.
- **rand(N)**
  - returns an N-by-N matrix containing pseudo-random values drawn from a uniform distribution on the unit interval i.e. [0;1].
- **rand(M,N)**
  - returns a random M-by-N matrix in the range [0 1]

- Generate random numbers in the interval [a b], let:
  a=2; b=6;
  (b-a)*rand(1,5)+a `%generates 5 numbers`
  ans = [3.6746 5.3849 4.1006 2.8106 4.6885]
  - 1-by-5 matrix.

Random numbers

- **randint:** Generate matrix of uniformly distributed random integers.
- **randint(M,N,[a,b])** generates an M-by-N in a given range (between a and b, a and b included).
  - randint(2,3,[0,5])

    ans =

0	4	0
9	7	8

- **randn**: Generates **normally distributed** random numbers.

Random numbers

- **randsrc**: Used for generating non-uniform distributed random numbers.
- randsrc(M,N,[A,B,C;p_0,p_1,p_2])
  - Returns a M-by-N matrix having elements 'A', 'B', and 'C' with a probabilities p_0, p_1 and p_2 respectively.
- As an Example:
  - data=randsrc(2,3,[0,1;0.2,0.8])

    data =

    1 0 1

    1 1 1

- **randperm(n)**: returns a random permutation of the integers 1:n.
  - randperm(5)
    ans = 5 2 3 4 1

Symbolic Math

- It defines variables that don't have necessarily a defined scalar or numerical value.
- Symbolic objects can be used as independent variables.
- For Example we can use the command syms such that:
  - syms x y z
  - Therefore x, y and z are three Symbolic Objects.

- A symbolic object can be:
  - A variable with no pre-assigned numerical value.
    - Ex: syms x y t
  - A number.
    - Ex: a=sym(2)
  - An expression made of symbolic variable/numbers.
    - Ex: syms x y; z=sqrt(x+y)

- A symbolic expression is a mathematical expression made of one or several symbolic objects.

Creating Symbolic Objects

- We use the commands **sym** and **syms** to create symbolic objects.
- To create one Symbolic Object:
  - a=sym(2)
    - a has a symbolic-numerical value
  - b=sym('gamma')
    - b has a symbolic-string value.

- To create multiple symbolic objects:
  - syms x y z

Symbolic v.s Numerical

- Symbolic objects: a and b

  a=sym(1);

  b=sym(2);

  f=a/b

  f = ½

- Numerical objects: A and B:

  A=1;

  B=2;

  F=A/B

  F =0.500

Symbolic to Numeric

- Some symbolic expressions can have numerical values resulting from numerical operations.
- To convert from symbolic to numerical objects, we use the command **`double(S)`**.
- Example:

```
a=sym(3);
b=1/a
b  = 1/3
B=double(b)
B = 0.33333
```

findsym command

- Used to find and enumerate symbolic variables present in an expression.

- Command Syntax:

  - findsym(S)
    - Displays all symbolic variables found in S, in alphabetical order.

  - findsym(S,n)
    - Displays the first *n* symbolic variables found in S, in **default order**
      - **Default order** for one letter variables: Start from *x* and list the others in the order of their closeness to *x*.

subs command

- To substitute a variable in a symbolic expression we use the command **subs**. Consider the following example:
  - syms x y;
    f=2*x+log(y);
    subs(f,x,2);
    ans = 4+log(y)
    subs(f,[x y],[2 1])
    ans = 4
- The Factor command:
  - sym x
    G=-1/2*exp(-x)*cos(x)-1/2*exp(-x)*sin(x)
    factor(G)
    ans = -1/2*exp(-x)*(cos(x)+sin(x))
- See also: expand, simplify, pretty.

Solving equations (Symbolic)

- Solving algebraic equations.
- Example (2 equations, 2 unknowns)
  - [x y]=solve('2*x+a*y-1=0','b*x+2*y=0')

    x =-2/(b*a-4)

    y =1/(b*a-4)*b

- Solving Differential equations:
- 'Dn' represents the n^{th} order derivative operator
- Consider the Example: $x'' + a^2 x = 0$, IC: $x(0)=1$, $x'(0)=1$
  - x = dsolve('D2x+a^2*x=0','x(0)=1, Dx(0)=1')

    x = 1/a*sin(a*t)+cos(a*t)

Calculating Derivatives (Symbolic)

- Partial Derivatives. Consider the function given by:
  - f = sin(x^2+y)
- Partial derivative w.r.t. x
  - f_x=diff(f,x)
    f_x =2*cos(x^2+y)*x
- 2nd Partial derivative w.r.t. x
  - f2_x=diff(f,x,2)
    f2_x = -4*sin(x^2+y)*x^2+2*cos(x^2+y)

Calculating Integrals (Symbolic)

- Integration. Consider the function given by:
  - f=sin(x)*exp(-x)

- Definite integral (i.e. Integrate f(x) over the range [0,2])
  - F=int(f,x,0,2)
    F =-1/2*exp(-2)*cos(2)-1/2*sin(2)*exp(-2)+1/2

- Indefinite Integral
  - G=int(f,x)
    G =-1/2*exp(-x)*cos(x)-1/2*sin(x)*exp(-x)

Inline function

- Used for multivariable expressions.
- Returns a function that takes several arguments. As an example:

  - D=inline('2*x*y+sin(x)','x','y')
    D = Inline function:
    D(x,y) = 2*x*y+sin(x)

  - D(1,2)
    ans = 4.8415