MATLAB: *Numerical Analysis functions*

LU decomposition: use ***lu()*** function to solve system of linear equations. For example, solve the following system:

$$\begin{bmatrix} 3 & -0.1 & -0.2 \\ 0.1 & 7 & -0.3 \\ 0.3 & -0.2 & 10 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 7.85 \\ -19.3 \\ 71.4 \end{Bmatrix}$$

```
>> A=[3 -.1 -.2;.1 7 -.3;.3 -.2 10];
>> b=[7.85; -19.3; 71.4];
>> [L,U]=lu(A)

L =

    1.0000         0         0
    0.0333    1.0000         0
    0.1000   -0.0271    1.0000


U =

    3.0000   -0.1000   -0.2000
         0    7.0033   -0.2933
         0         0   10.0120

>> d=L\b

d =

    7.8500
  -19.5617
   70.0843

>> x=U\d

x =

    3.0000
   -2.5000
    7.0000
```

Syntax	Description
eig(A)	Finds Eigenvalues and eigenvectors of A.
eye(N)	Generates the N-by-N identity matrix
zeros(N)	Generates the N-by-N matrix of zeros
factorial(x)	Calculate the factorial of x-scalar.
inv(function)	Finds the inverse of non-singular matrix
polyfit(X,Y,N)	Least-squares polynomial regression of Nth order that fits Y.
poly(A)	Convert roots to polynomial
roots(p)	Find polynomial roots
polyval(p,x)	returns the value of a polynomial p evaluated at x
regression(x,y)	calculate the linear regression between input and output (1^{st} order)

<u>Example 1</u>: find first-order polynomial fit of the following data.

X	10	20	30	40	50	60	70	80
Y	25	70	380	550	610	1220	830	1450

1- Using ***polyfit()*** function:

```
>> x=[10 20 30 40 50 60 70 80];
>> y=[25 70 380 550 610 1220 830 1450];
>> a=polyfit(x,y,1)

a =

   19.4702 -234.2857
```

$$Y=19.4702X - 234.2857$$

2- Using ***regression()*** function

```
>> x=[10 20 30 40 50 60 70 80];
>> y=[25 70 380 550 610 1220 830 1450];
>> [R,A,B]=regression(x,y)

R =

    0.9383

A =

   19.4702

B =

 -234.2857
```

Correlation Coefficient (R)

slope (A)

y-intercept (B)

3- Evaluate the resulted polynomial at x =45.

```
>> x=[10 20 30 40 50 60 70 80];
>> y=[25 70 380 550 610 1220 830 1450];
>> a=polyfit(x,y,1)

a =

   19.4702 -234.2857

>> y=polyval(a,45)

y =

  641.8750
```

Example 2: find the roots of the following polynomial

$$f(x) = x^5 - 3.5x^4 + 2.75x^3 + 2.125x^2 - 3.875x + 1.25$$

Use ***roots()*** function to find the five roots of this 5th order polynomial

```
>> a=[ 1 -3.5 2.75 2.125 -3.875 1.25];
>> x=roots(a)

x =

   2.0000
  -1.0000
   1.0000 + 0.5000i
   1.0000 - 0.5000i
   0.5000
```

Now, use the found roots to reconstruct the original polynomial using ***poly(x)*** function

```
>> a=poly(x)

a =

    1.0000   -3.5000    2.7500    2.1250   -3.8750    1.2500
```

Numerical Integration:

1- Using ***Trapezoidal Rule***, use ***trapz()*** function to evaluate the integral from data. Consider the following example:

$$f(x) = 400x^5 - 900x^4 + 675x^3 - 200x^2 + 25x + 0.2$$

X	0.00	0.12	0.22	0.32	0.36	0.40

```
>> x=[0 .12 .22 .32 .36 .4];
>> y=0.2+25*x-200*x.^2+675*x.^3-900*x.^4+400*x.^5;
>> z=trapz(x,y)

z =

    0.5407
```

2- Using ***Adaptive Simpson quadrature*** (non-smooth functions), use ***quad()*** function to integrate functions using this algorithm. The ***quad()*** function syntax is:

q=quad(fun,a,b,tol,trace,p1,p2,…)

where, ***fun*** refers to function to be integrated, ***a*** and ***b*** are the lower and upper limits, respectively, ***tol*** is desired absolute error tolerance, ***trace*** $\neq 0$ means display additional information.

A- Use ***quad()*** function to integrate the following:

$$\int_0^2 \frac{1}{x^3 - 2x - 5} dx$$

```
>> F = @(x)1./(x.^3-2*x-5);
Q = quad(F,0,2)

Q =

   -0.4605
```

Note that using ***<u>quad()</u>*** function requires two steps: (1) enter the function [**@**(variable_of_integration] and (2) executing the ***quad()*** function.

B- Use ***quad()*** function to integrate:

$$\int_0^8 -0.0547x^4 + 0.864x^3 - 4.1562x^2 + 6.2917x + 2dx$$

```
>> F=@(x)-0.0547*x.^4+0.864*x.^3-4.1562*x.^2+6.2917*x+2;
>> z=quad(F,0,8)

z =

   34.2637
```

3- Using Lobatto Quadrature algorithm (smooth functions), use ***quadl()*** function. The ***quadl()*** function syntax is:

q=quadl(fun,a,b)

where, ***fun*** refers to function to be integrated, ***a*** and ***b*** are the lower and upper limits, respectively. Consider the following example:

Use quadl() function to integrate:

$$\int_2^8 (9 + 4cos^2(0.4t))(5e^{-0.5t} + 2e^{0.15t})dx$$

```
>> F=@(t)(9+cos(0.4*t).^2).*(5*exp(-0.5*t)+2*exp(0.15*t));
>> z=quadl(F,2,8)

z =

  281.5059
```

Ordinary Differential Equations (ODE)-Initial Value Problem (IVP):

command	Description	Syntax
ode15s	Stiff D.E.	[tout,yout]=ODE15S(odefun,tspan,y0)
ode23	Non-stiff D.E. (low order)	[tout,yout] = ODE23(odefun,tspan,y0)
ode23s	Stiff D.E. (low order)	[tout,yout]= ODE23S(odefun,tspan,y0)
ode23t	Moderately stiff D.E.(trapezoidal rule)	[tout,yout]= ODE23t(odefun,tspan,y0)
ode23tb	Stiff D.E. (low order) (implicit R.K.)	[tout,yout]= ODE23tb(odefun,tspan,y0)
ode45	Non-stiff D.E. (medium order)	[tout,yout]= ODE45(odefun,tspan,y0)
ode113	Non-stiff D.E. (variable order)	[tout,yout]= ODE113(odefun,tspan,y0)

Example:

Use **ode23s()** and ***ode45()*** to solve the following D.E.

$$\frac{d^2y}{dt} - \mu(1-y^2)\frac{dy}{dt} + y = 0, \quad \text{I.C. } y(0) = \frac{dy}{dt} = 1$$

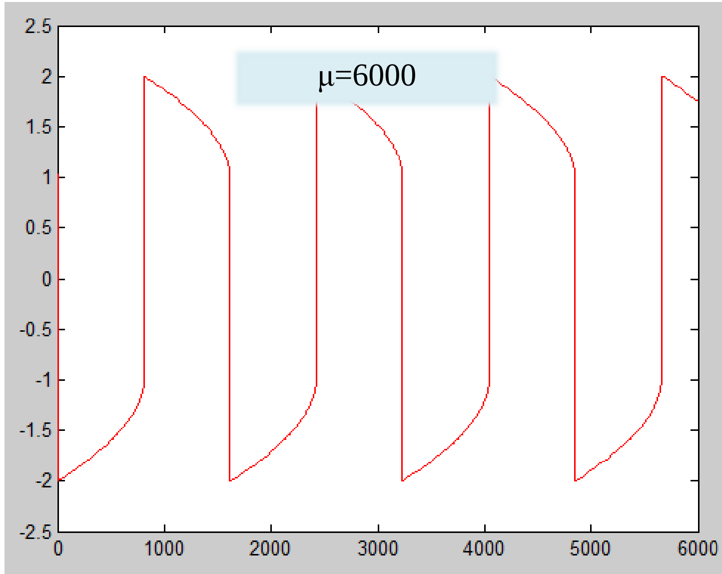
Convert given 2nd order D.E. into two 1st order D.E., such as:

$$\frac{dy}{dt} = x$$

$$\frac{dx}{dt} = \mu(1-y^2)x + y = 0$$

```
>> F=@(t,y,mu)[y(2);mu*(1-y(1)^2)*y(2)-y(1)];
>> [t,y]=ode23s(F,[0 6000],[1 1],[],1000);
>> plot(t,y(:,1),'-r')
```

The above code uses ***ode23s()*** because the given D.E. is very stiff when μ=1000. (observe the response below). In this example, μ=1000, time span (tspan) = [0,6000] and initial conditions was set to 1.




```
>> F=@(t,y,mu)[y(2);mu*(1-y(1)^2)*y(2)-y(1)];
>> [t,y]=ode45(F,[0 20],[1 1],[],1);
>> plot(t,y(:,1),'-r',t,y(:,2),'--b')
```

The above code uses ***ode45()*** because the given D.E. is smooth when μ=10 as suggested by the e response below).