Matlab Tutorial

Overview

- Help
- General
  - Scripts and Functions
- Vectors and Matrices
- Flow Control
- Plotting
- Loading and Saving
- Numerical Integration Example
- Symbolic Toolbox

Help

- demos
- help (list of help topics)
- help help (help on help)
- lookfor integration
- Documentation
  - helpdesk
  - ../matlab/help/pdf_doc/matlab/using_ml.pdf
  - ../ matlab/help/pdf_doc/matlab/getstart.pdf
  - ../matlab/help/pdf_doc/matlab/ref/Refbook.pdf
- http://www.mathworks.com

Basic math function in Matlab

abs(x)：absolute value of a scalar or the vector length

angle(z)：the phase angle of a complex number z

sqrt(x)：the square root of x

real(z)：the real part of a complex number z

imag(z)：imaginary part of a complex number z

conj(z)：complex conjugate of a complex number z

round(x)：round toward nearest integer, round (4.5)=5, round (4.4)=4

fix(x)：round toward zero, fix(4.5)=4, fix(4.4)=4, fix(-4.5)=04

sign(x)：Signum function, sign(x)=-1 when x<0,

sign(x)=1 when x>0

Variables in Matlab

Variables can be used to store scalar, vector or matrix (numbers). The matlab variable can also be used to do various math operation, such as the math of row vector:

```
x = [1 3 5 2];
y = 2*x+1
y =
3 7 11 5
```

Variables in Matlab

We can modify the elements of a vector at well:

```
y(3) = 2  % modify the third element to be 2
y =
3 7 2 5
y(6) = 10  % add the sixth element
y =
3 7 2 5 0 10
y(4) = []  % delete the fourth element
y =
3 7 2 0 10
```

Note: the text after percentage sign '%' in matlab is regarded as comments only. It will not be regarded as matlab commands.

```
x(2)*3+y(4 ) % summation of the 2nd element of x and the 4th element of y
ans =
9
y(2:4)-1 %  subtraction of the sub-vector from the 2nd to the 4th in y
ans =
6 1 -1
```

2:4 means a vector from the 2nd, 3rd, and 4th elements of y

Similarly, this method can be used to generate a geometric series

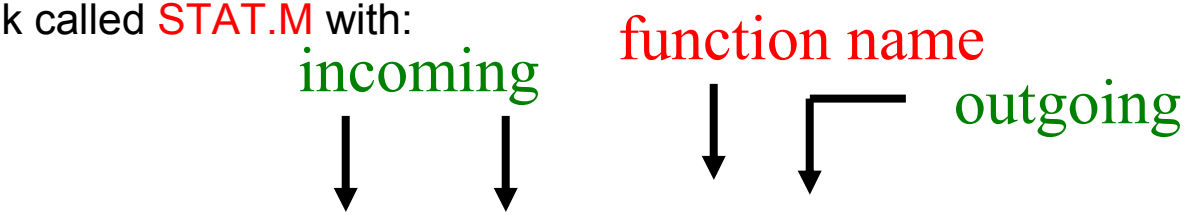
```
x = 7:16
 7 8 9 10 11 12 13 14 15 16
```

General

- Two ways to use Matlab
  - Command Window
    - acos(-1) or batch
    - ; up arrow … , = space
  - Scripts and Functions
    - Scripts and Functions called in command window
    - Developed as text file with .m extension
      - help function or help script
      - Editor/Debugger
      - variables “declared” in function not available in work space
      - variables used in script are “global” variables

» help function

FUNCTION Add new function.
New functions may be added to MATLAB's vocabulary if they are expressed in terms of other existing functions. The commands and functions that comprise the new function must be put in a file whose name defines the name of the new function, with a filename extension of '.m'. At the top of the file must be a line that contains the syntax definition for the new function. For example, the existence of a file on disk called STAT.M with:




```
function [mean,stdev] = stat(x)
%STAT Interesting statistics.
n = length(x);
mean = sum(x) / n;
stdev = sqrt(sum((x - mean).^2)/n);
```

defines a new function called STAT that calculates the mean and standard deviation of a vector. The variables within the body of the function are all local variables. See SCRIPT for procedures that work globally on the work-space.

» help script

SCRIPT About MATLAB scripts and M-files.
A SCRIPT file is an external file that contains a sequence of MATLAB statements. By typing the filename, subsequent MATLAB input is obtained from the file. SCRIPT files have a filename extension of ".m" and are often called "M-files".
To make a SCRIPT file into a function, see FUNCTION.

- Requires no input
- Provides no explicit output
- All variables used are available in the work space
- Like a main program
- .m file must be in your current path (help path)
    - path(path,a:\matlab)

Matrices

- help matfun
- help elmat
- useful commands:
  - eye(n)
  - ones(n)
  - zeros(n)
  - reshape(A,row,col)

- A=[2, 0; 0, 1; 3, 3]
- A =[A,[1;2;3]]
- A =(1:2,1:2)

```
» eye(2)
ans =
     1     0
     0     1

» ones(2)
ans =
     1     1
     1     1

» zeros(2,1)
ans =
     0
     0
```

```
» A = [2,0; 0,1; 3,3]
A =
     2     0
     0     1
     3     3

» A =[A,[1;2;3]]
A =
     2     0     1
     0     1     2
     3     3     3

» A(1:2,1:2)
ans =
     2     0
     0     1
```

Vectors

- Special case of matrices
  - a = [0 1 2 3 4] same as a = [0,1,2,3,4]
  - same as a = [0:4]
  - b = [0, .5, 1, 1.5, 2, 2.5] same as b = [0:.5:2.5]
  - c = [.1, .1, .1, .1, .1] same as c = ones(1,5) * 0.1
- R = R(:) ensures R is a column vector

- 0.1 * a = a.*c = [0, 0.1, 0.2, 0.3, 0.4] (.* array multiply)
- a * c`  (` is transpose) = 1.0

Flow Control

- if elseif else end
- switch case otherwise end
- for end
- while end
- break

```
if rem(n,2) ~= 0
      M = odd_magic(n)
elseif rem(n,4) ~= 0
      M = single_even_magic(n)
else
      M = double_even_magic(n)
end
```

```
switch (rem(n,4)==0) + (rem(n,2)==0)
      case 0
            M = odd_magic(n)
      case 1
            M = single_even_magic(n)
      case 2
            M = double_even_magic(n)
      otherwise
            error('This is impossible')
end
```

```
j = 1;
for i = 0 : 0.1 : 5
    t(j) = i;
    x(j) = exp(i);
    j = j+1;
end
```

```
t = [0: .1 : 5];
x = exp(t);
```

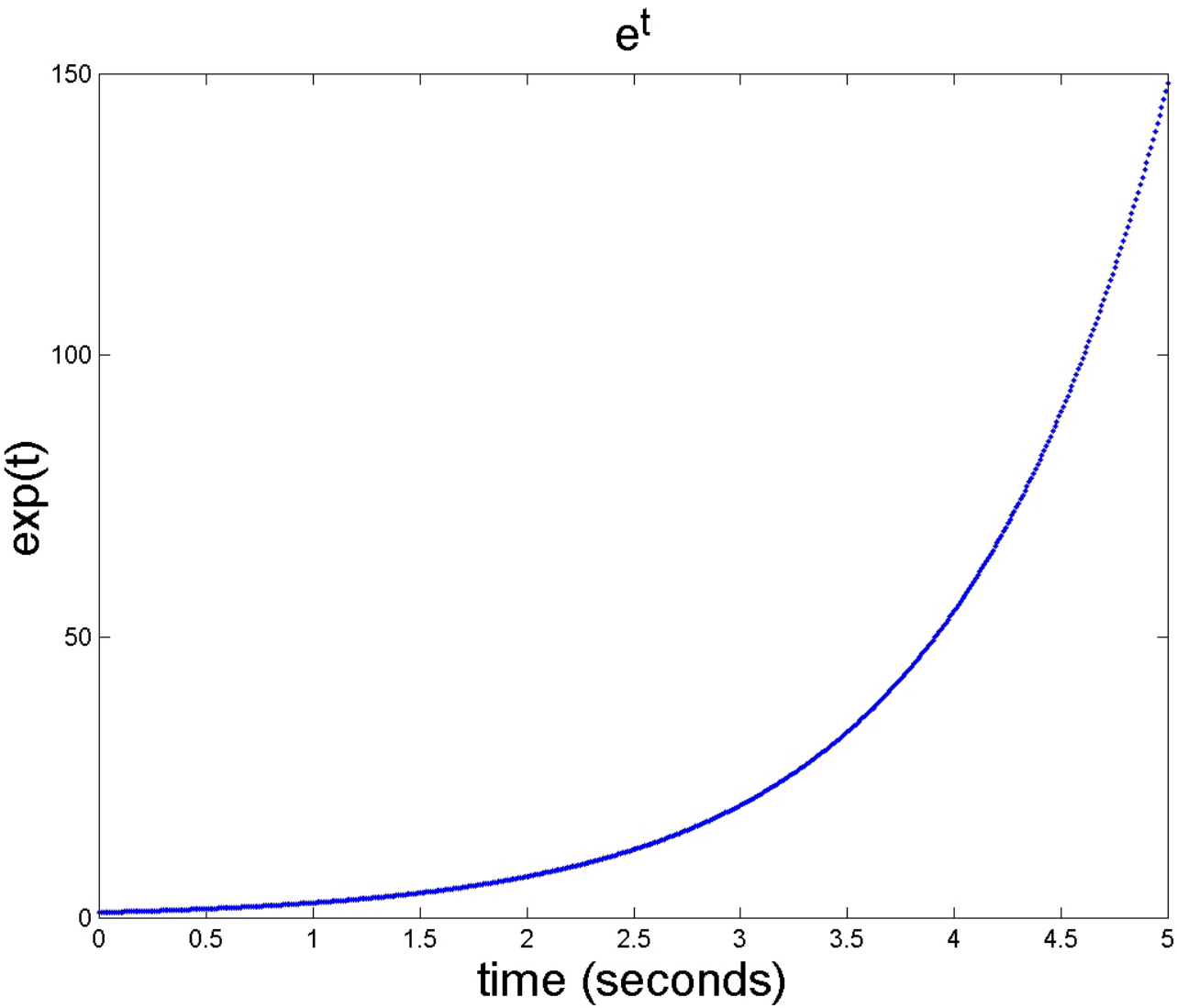
```
while b–a > eps*b
    x = (a+b)/2;
    fx = x^3–2*x–5;
    if sign(fx) == sign(fa)
        a = x; fa = fx;
    else
        b = x; fb = fx;
    end
end
```

Plotting

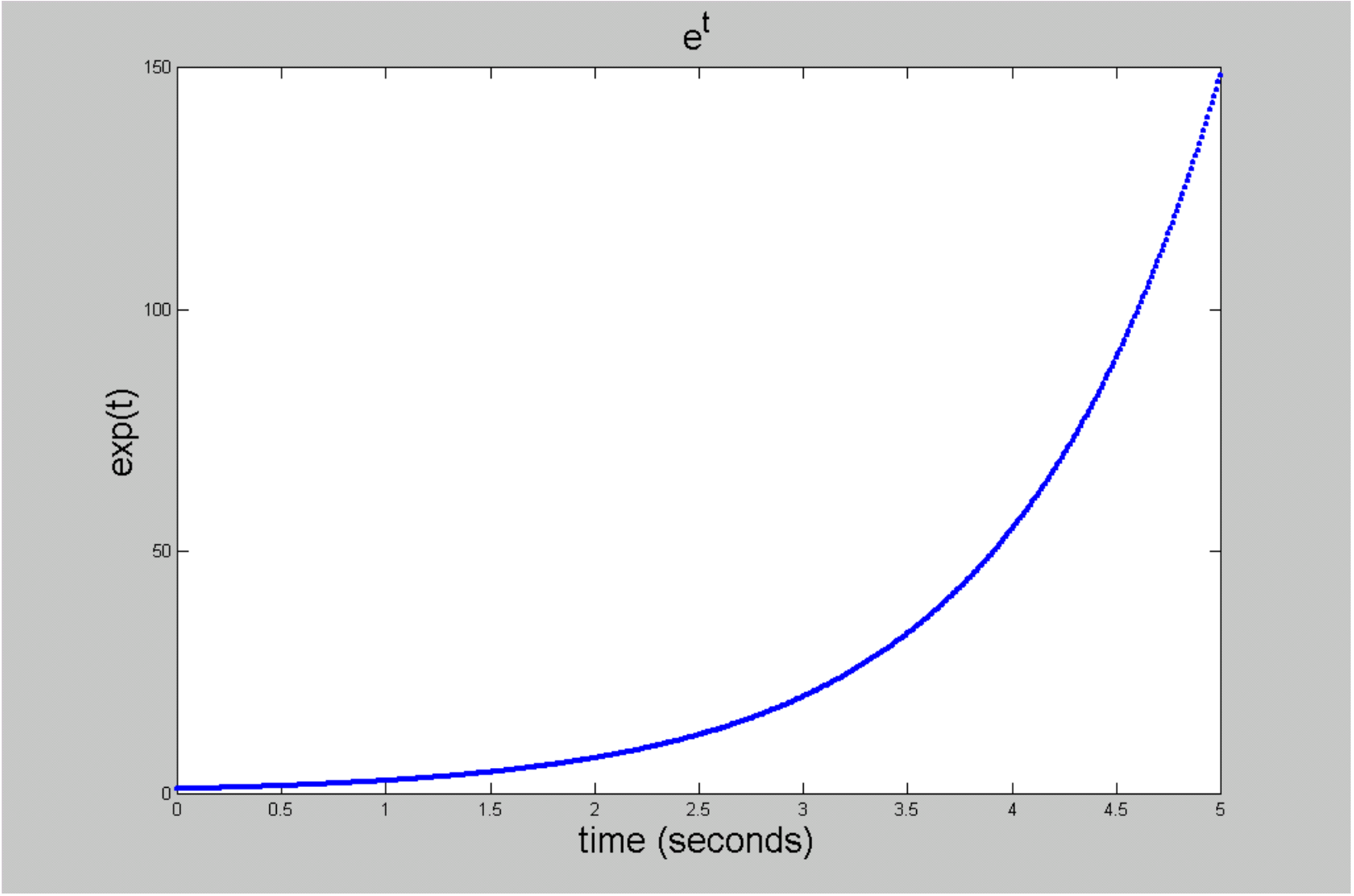
- help plot
- plot(x,y) plot(x,y,'r') plot3(x,y,z)
- axis([xmin xmax ymin ymax])
- title('This is the title')
- xlabel('x axis label')
- ylabel('y axis label')
- print -djpeg99 'd:\temp\plotexample.jpg'

```
» t = [0:.01:5];
» x = exp(t);
» plot(t,x)
» title(`e^t`)
» xlabel(`time (seconds)`)
» ylabel(`exp(t)`)
```

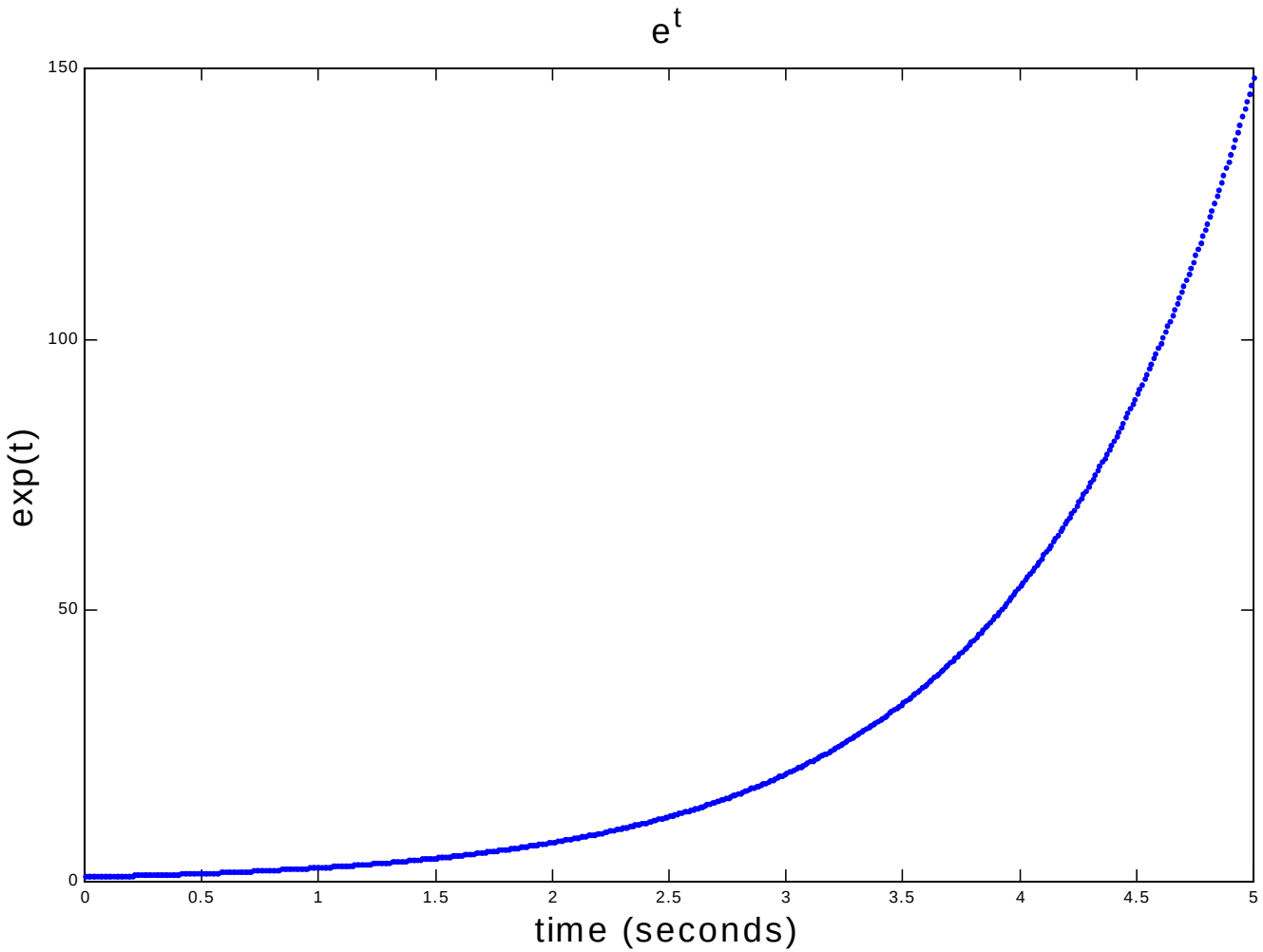
Saved as a .jpg



Copied as Bitmap



Copied as Windows WMF



Loading and Saving

- Load and Save in binary and ASCII
  - ASCII easier to share with other applications
- save ‘d:\temp\my_matrix.txt’ A -ascii -double
- load ‘d:\temp\my_matrix.txt’
  - becomes variable my_matrix
  - loads rows and columns of numbers only
- Be care in naming your variables
- save variable_name (saves all variables in the current workspace

```
» x = [0:10]`;
» y = [x  x.^2  x.^3  x.^4]

y =
         0         0         0         0
         1         1         1         1
         2         4         8        16
         3         9        27        81
         4        16        64       256
         5        25       125       625
         6        36       216      1296
         7        49       343      2401
         8        64       512      4096
         9        81       729      6561
        10       100      1000     10000
```



```
» save `d:\temp\y1.txt` y -ascii
```

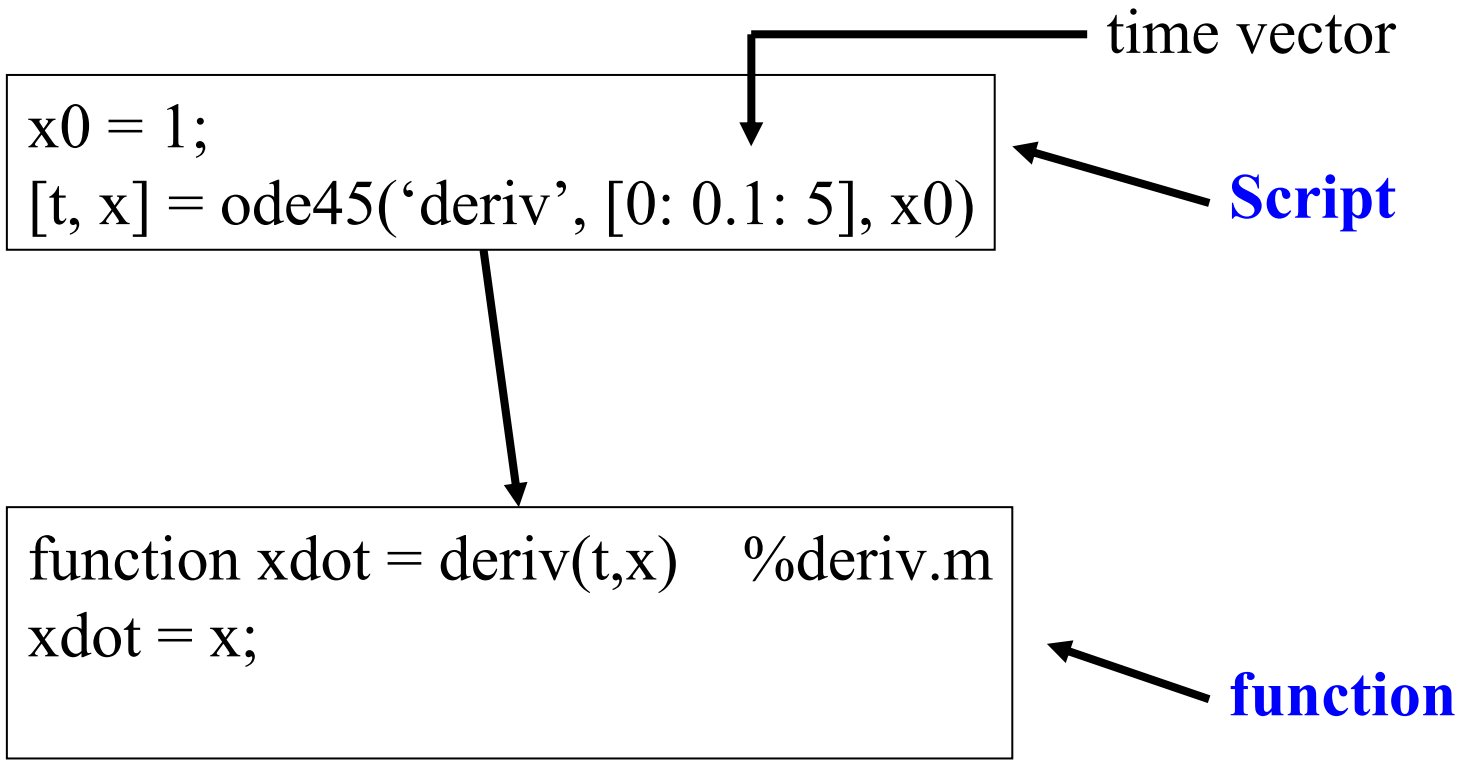
Contents of y1.txt

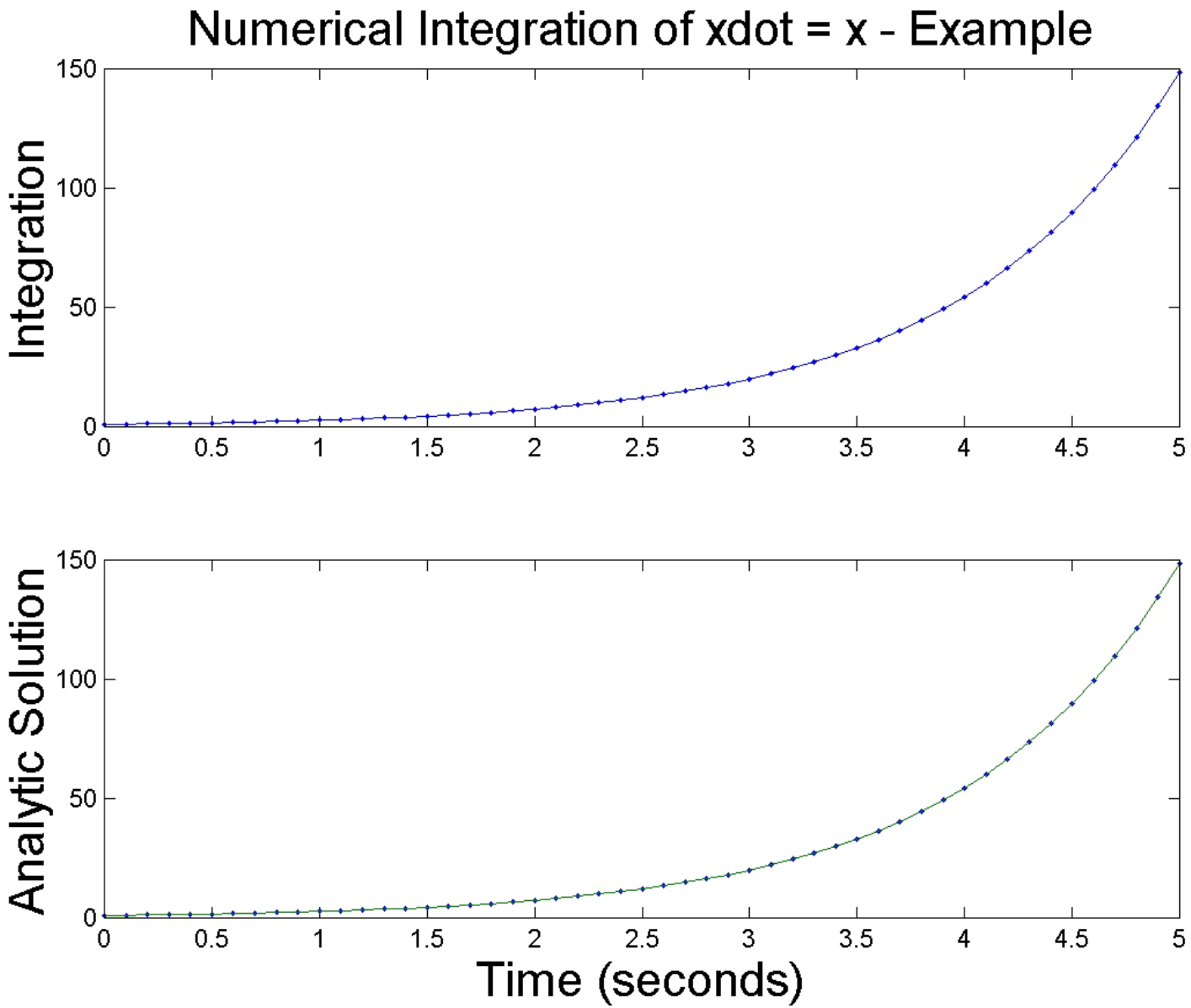
0.0000000e+000	0.0000000e+000	0.0000000e+000	0.0000000e+000
1.0000000e+000	1.0000000e+000	1.0000000e+000	1.0000000e+000
2.0000000e+000	4.0000000e+000	8.0000000e+000	1.6000000e+001
3.0000000e+000	9.0000000e+000	2.7000000e+001	8.1000000e+001
4.0000000e+000	1.6000000e+001	6.4000000e+001	2.5600000e+002
5.0000000e+000	2.5000000e+001	1.2500000e+002	6.2500000e+002
6.0000000e+000	3.6000000e+001	2.1600000e+002	1.2960000e+003
7.0000000e+000	4.9000000e+001	3.4300000e+002	2.4010000e+003
8.0000000e+000	6.4000000e+001	5.1200000e+002	4.0960000e+003
9.0000000e+000	8.1000000e+001	7.2900000e+002	6.5610000e+003
1.0000000e+001	1.0000000e+002	1.0000000e+003	1.0000000e+004

Numerical Integration

$$\dot{x} = x$$

$$x = e^t + c \qquad \text{Let } x(0) = 1 \text{ so, } c = 0$$




Numerical Integration of xdot = x - Example
Integration
Analytic Solution
Time (seconds)
0
50
100
150
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

Numerical Integration (cont.)

- Use Options to set the tolerance!!
- Two-body Example:

```
Clear   %Script to numerically integrate the 2-body equation
global MU
MU=3.986005e14;

time = [0:20:10000] %Integration time in seconds
R0 = [-2436450.0 -2436450.0 6891037.0];
V0 = [5088.611 -5088.611 0.0];
x0 = [R0' V0'];

tol = 1e-8;
options = odeset('RelTol',tol,'AbsTol',[tol tol tol tol tol tol]);
[t,x] = ode45('two_body', time, x0, options);

clf
subplot(2,1,1), plot(t,x(:,7),'r'), title('Radius vs Time'), ylabel('Rad Mag (m)');
subplot(2,1,2), plot(t,x(:,8),'b'), title('Velocity vs Time'), ylabel('Vel Mag (m/s)');
```

```
function xdot = two_body(t,x)  %derivative function for numerical integrator

%  x(1) = Ri (Radius Vector in the i direction)
%  x(2) = Rj
%  x(3) = Rk
%  x(4) = xdot(1) = Vi (Velocity vector in the i direction)
%  x(5) = xdot(2) = Vj
%  x(6) = xdot(3) = Vk
%         xdot(4) = Ai (Acceleration vector in the i direction)
%         xdot(5) = Aj
%         xdot(6) = Ak

MU = 3.98004e14;

r = norm(x(1:3));

xdot = [x(4);
        x(5);
        x(6);
        -MU * x(1) / r^3;
        -MU * x(2) / r^3;
        -MU * x(3) / r^3];
```

Numerical Simulation

- Solving differential equations by numerical integration
- Euler, Runge-Kutta, etc.
- Available in Matlab, Mathematica and Fortan (or in MS Visual C)
- Use these to examine nonlinear vibration problems that do not have analytical expressions for solutions

Euler or Tangent method (undo the derivative)

$$\frac{dx(t_i)}{dt} = \lim_{\Delta t \to 0} \frac{x(t_{i+1}) - x(t_i)}{\Delta t}$$

$$\Delta t = t_{i+1} - t_i$$

Undo the derivative

$$\text{solve } \dot{x}(t) = ax(t), x(0) = x_0$$

$$\frac{x_{i+1} - x_i}{\Delta t} = ax_i, x_0$$

$$x_i = x(t_i), \Delta t = t_{i+1} - t_i$$

$$x_{i+1} = x_i + (ax_i)\Delta t$$

Example solve

dx/dt=-3x, x(0)=1 numerically

$$a = -3, \text{ take } \Delta t = 0.5$$

$$x_0 = 1$$

$$x_1 = x_0 + (0.5)(-3)(x_0) = -0.5$$

$$x_2 = x_1 + (0.5)(-3)(x_1) = 0.25$$

$$\vdots \qquad \vdots \qquad = \quad \vdots$$

Analytical Solution

$$x(t) = Ae^{\lambda t}$$

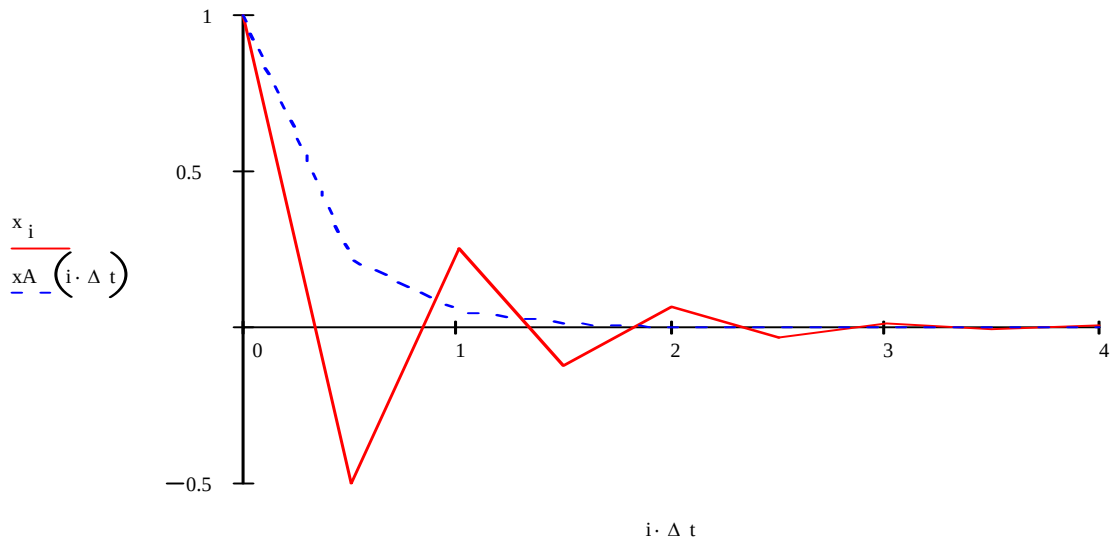
$$\dot{x}(t) = -3x(t) \Rightarrow -\lambda Ae^{\lambda t} = -3Ae^{\lambda t}$$

$$\Rightarrow \lambda = 3,$$

$$x(0) = x_0 = 1 = Ae^0 \Rightarrow A = 1 \text{ so that}$$

$$\underline{x(t) = e^{-3t}}$$

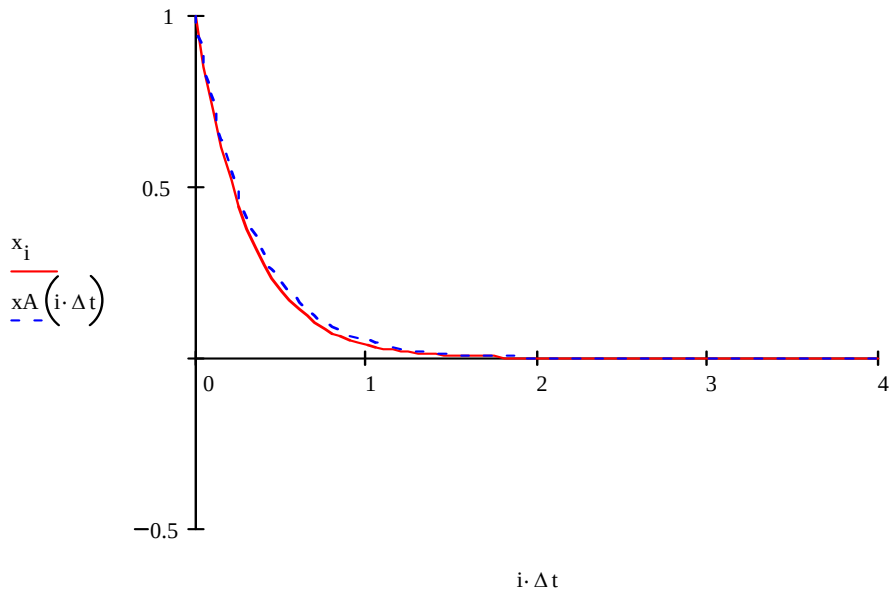
For $\Delta t = 0.5$



Time step is too large to capture the response

For $\Delta t = 0.05$

$$x_{i+1} := x_i - 3 \cdot x_i \cdot \Delta t$$



Numerical solution of the 2nd order equation of vibration:

$$m\ddot{x} + c\dot{x} + kx = 0 \Rightarrow$$

$$\text{Let } x_1 = x, \quad x_2 = \dot{x} \Rightarrow$$

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -\frac{c}{m} x_2 - \frac{k}{m} x_1$$

Write as a 1st order vector equation

$$\dot{\mathbf{x}} = A\mathbf{x}, \text{ where}$$

$$A = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix}, \quad \mathbf{x}(t) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{x}_0 = \begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix}$$

$$\mathbf{x}(t_{i+1}) = \mathbf{x}(t_i) + \Delta t A \mathbf{x}(t_i)$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta t A \mathbf{x}_i$$

Usually use Runge-Kutta

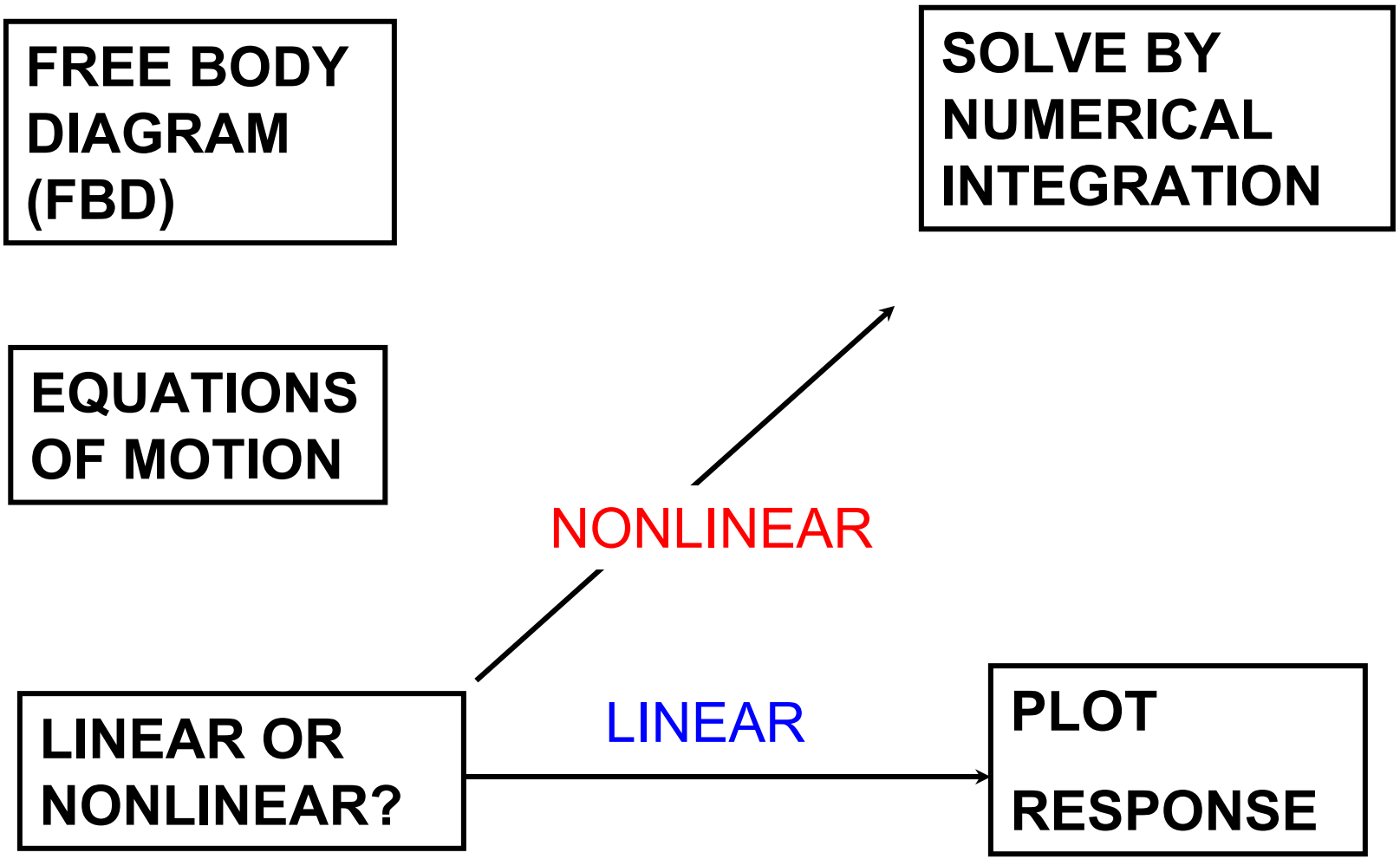
- Most use first order form
- Often picks Δt
- Works for nonlinear equations too

$$\ddot{x} + f(x, v) = 0$$

$$\begin{bmatrix} x_{i+1} \\ v_{i+1} \end{bmatrix} = \begin{bmatrix} x_i + v_i \Delta t \\ v_i - f(x_i, v_i)\Delta t \end{bmatrix}$$

Check Matlab [t,y]=ode23('fun', tspan, y0)

WHEN TO USE MATLAB



Solving the MDOF problems using computer codes

Alternately multiply by -1:

$$A = \begin{bmatrix} K & 0 \\ 0 & -M \end{bmatrix}, \quad B = \begin{bmatrix} 0 & K \\ K & C \end{bmatrix}$$

Symmetric, generalized eigenvalue problem

```
[X,D]=eig(A,B)%in matlab
```

Numerical integration for the time response

$$M\ddot{\mathbf{x}}(t)+C\dot{\mathbf{x}}(t)+K\mathbf{x}(t)=\mathbf{F}(t),\mathbf{x}(0)=\mathbf{x}_0,\dot{\mathbf{x}}(0)=\mathbf{v}_0 \Rightarrow$$

$$\begin{cases}\dot{\mathbf{y}}_1=\mathbf{y}_2\\ \dot{\mathbf{y}}_2=-M^{-1}K\mathbf{y}_1-M^{-1}C\mathbf{y}_2+M^{-1}\mathbf{F},\mathbf{y}_1(0)=\mathbf{x}_0,\mathbf{y}_2(0)=\mathbf{v}_0\end{cases}$$

$$\Rightarrow \dot{\mathbf{y}}(t)=A\mathbf{y}(t)+\mathbf{f}(t),\mathbf{y}(0)=\mathbf{y}_0$$

Where:

$$\mathbf{y}(t)=\begin{bmatrix}\mathbf{y}_1\\ \mathbf{y}_2\end{bmatrix},A=\begin{bmatrix}0 & I\\ -M^{-1}K & -M^{-1}C\end{bmatrix},\mathbf{f}(t)=\begin{bmatrix}0\\ M^{-1}\mathbf{F}(t)\end{bmatrix}$$

MATLAB Code

MAKE AN M-FILE called simulation.m

```
function ydot=simulation(t,y)
M=[9 0;0 1];C=[9 -1;-1 1];K=[27 -3;-3 3];
F=[0;0;0; sin(3*t)];
A=[zeros(2,2) eye(2,2);-M\K -M\C];
ydot=A*y+F;
% now type the following in the command window:
y0=[0.1;0;0;0];ti=0;tf=50;
[time,sol]=ode45('simulation',[ti tf],y0);
subplot(2,1,1),plot(time,sol(:,1))
xlabel('time s'),ylabel('y1 m'),title('dof 1')
subplot(2,1,2),plot(time,sol(:,2)),
xlabel('time s'),ylabel('y2 m'),title('dof 2')
```

Comments

- Can calculated the eigenvalue problem
- Can use it to write the free and forced response
- Can also simulate the free and forced response numerically (approximation)
- Can use numerical simulation to solve for nonlinear mdof systems (extension of sdof case)

Symbolic Toolbox

- help symbolic
- Example

```
syms x y z mu
r = sqrt(x^2 + y^2 + z^2)
U = mu/r

acell_x = diff(U,x)
acell_y = diff(U,y)
acell_z = diff(U,z)
```

```
r = (x^2+y^2+z^2)^(1/2)

U = mu/(x^2+y^2+z^2)^(1/2)

acell_x = -mu/(x^2+y^2+z^2)^(3/2)*x

acell_y = -mu/(x^2+y^2+z^2)^(3/2)*y

acell_z = -mu/(x^2+y^2+z^2)^(3/2)*z
```