RMIT University

OENG-1116: MODELLING & SIMUALTION OF ENGINEERING SYSTEMS

Week 2: Part-1

Examples of Modelling & Simulations: Falling Masses, Projectiles and Multi-DOF Mass-Spring-Damper Systems

Prof Pavel M. Trivailo

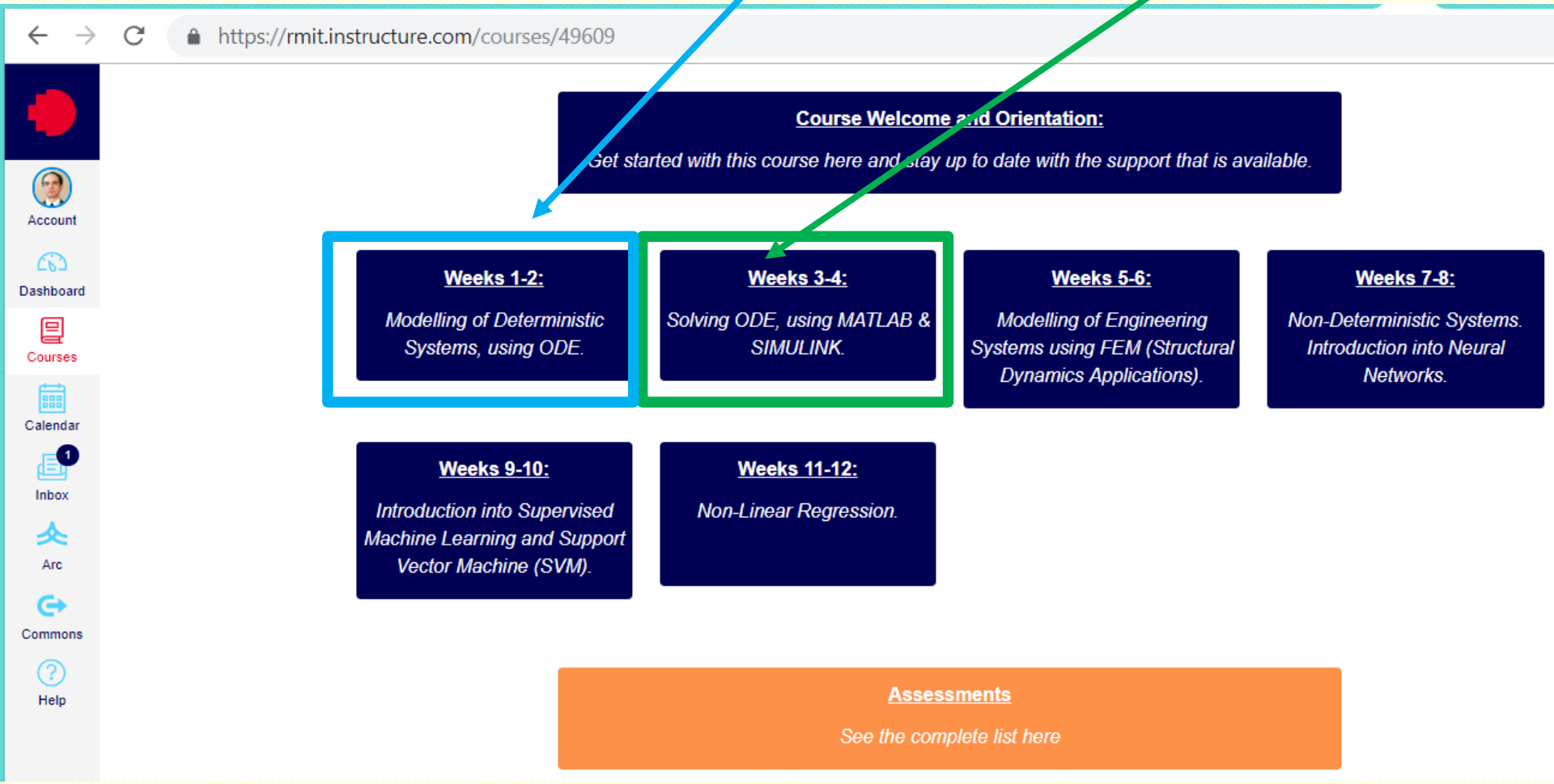
pavel.trivailo@rmit.edu.au

FEEDBACK:

QUESTIONS
HOME WORK
OBSERVATIONS
GENERAL DISCUSSIONS

CANVAS MATERIALS:

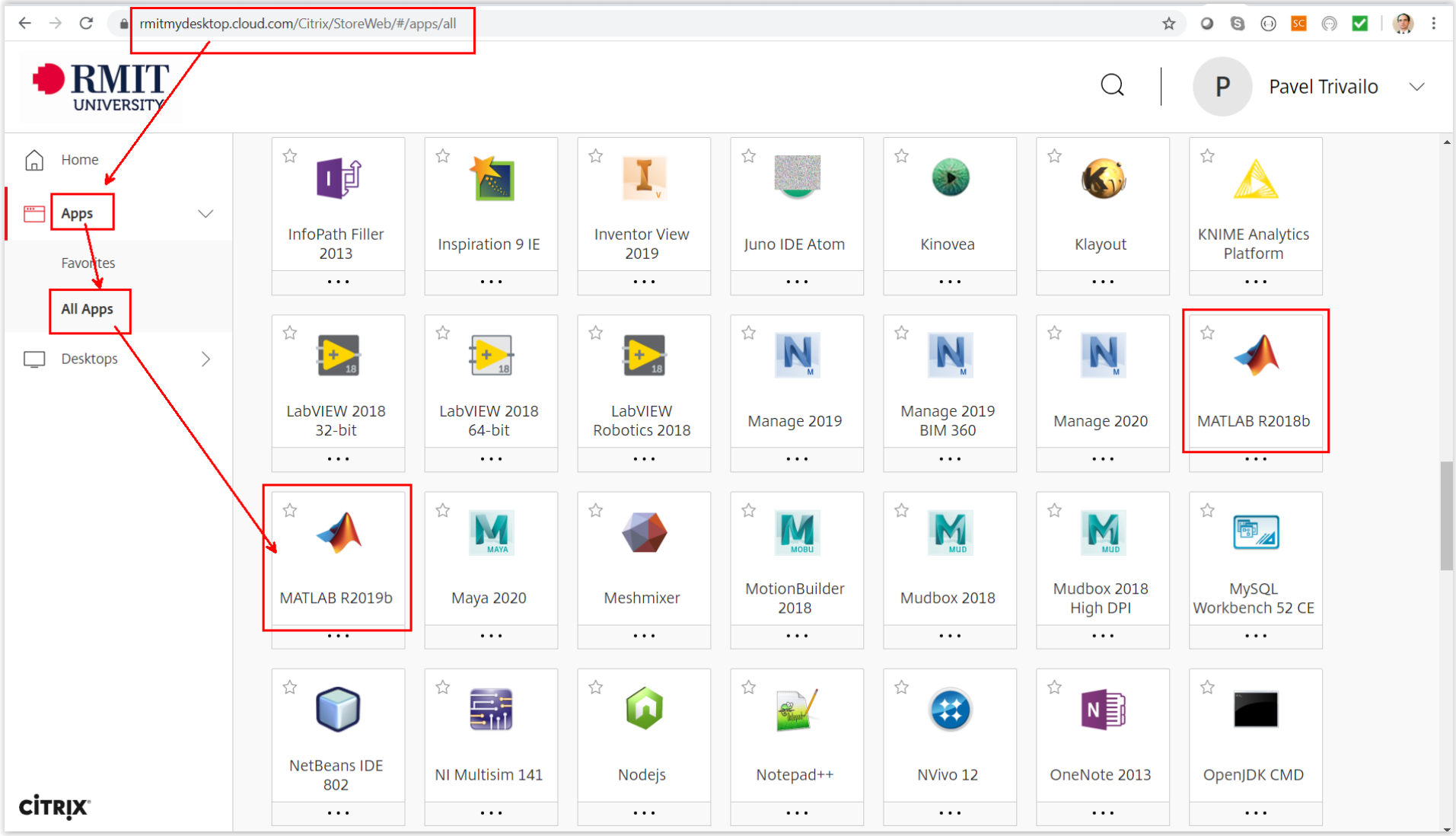
On the **Canvas**, under "**Week 1a, 1b**", "**Week 2a, 2b**":



MATLAB:

LICENCE ON YOUR COMPUTER and ACCESS vis MyDESKTOP

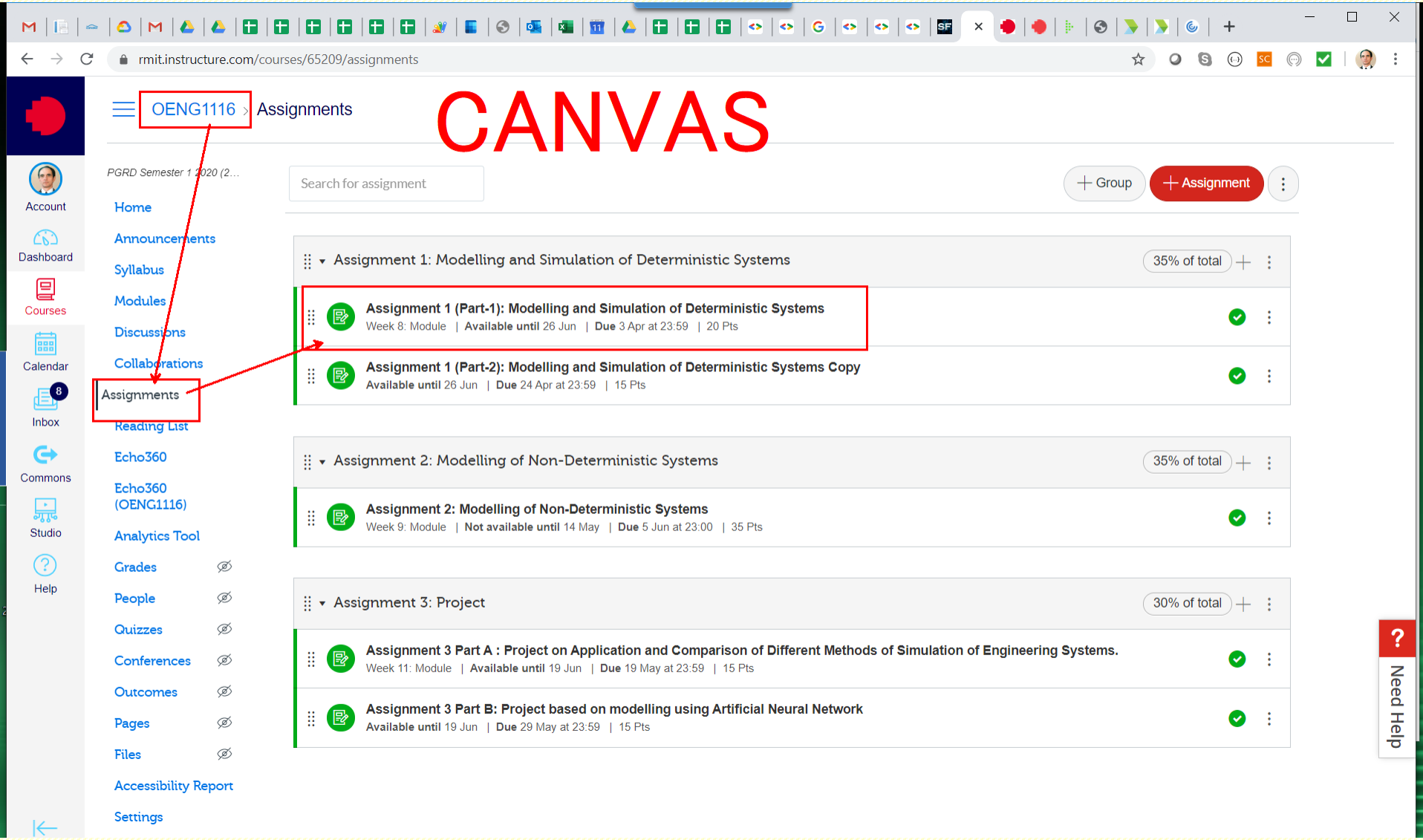
MyDesktop:


rmitmydesktop.cloud.com/Citrix/StoreWeb/#/apps/all
RMIT UNIVERSITY
Pavel Trivailo
Home
Apps
Favorites
All Apps
Desktops
InfoPath Filler 2013
Inspiration 9 IE
Inventor View 2019
Juno IDE Atom
Kinovea
Klayout
KNIME Analytics Platform
LabVIEW 2018 32-bit
LabVIEW 2018 64-bit
LabVIEW Robotics 2018
Manage 2019
Manage 2019 BIM 360
Manage 2020
MATLAB R2018b
MATLAB R2019b
Maya 2020
Meshmixer
MotionBuilder 2018
Mudbox 2018
Mudbox 2018 High DPI
MySQL Workbench 52 CE
NetBeans IDE 802
NI Multisim 141
Nodejs
Notepad++
NVivo 12
OneNote 2013
OpenJDK CMD
CITRIX

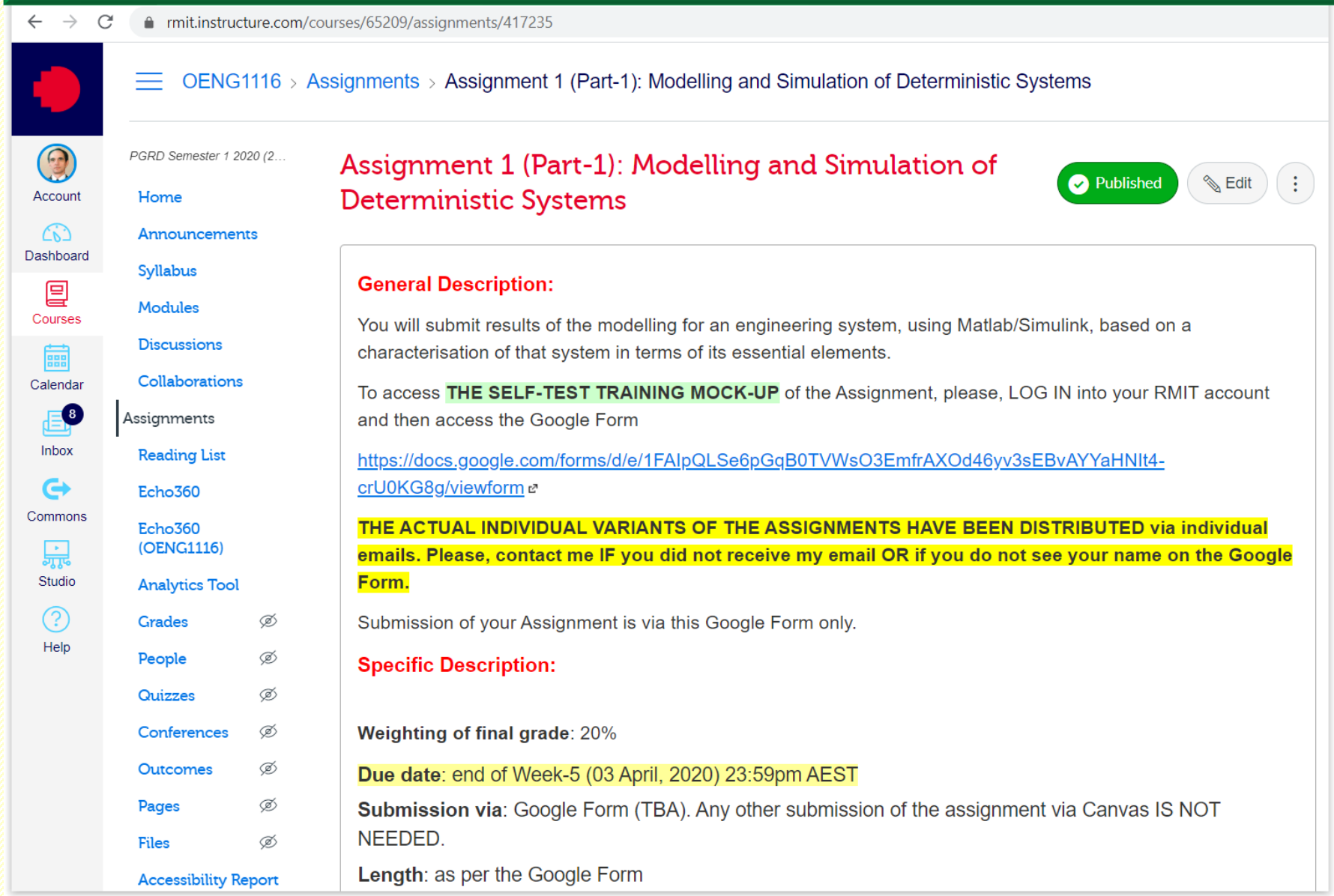
ASSIGNMENT-1

Released: individual links sent vial emails (should reach every student!)

CANVAS: “Assignments”:



CANVAS: “Assignment 1 (Part-1)”:



CANVAS: LINK TO THE MOCK UP of the Assignment-1, Part-1:

https://docs.google.com/forms/d/e/1FAIpQLSe6pGqB0TVWsO3EmfrAXOd46yv3sEBvAYYaHNIt4-crU0KG8g/viewform

Please, **DO NOT** re-type the link address, just click on the link.
Important: you must be currently LOGGED IN into RMIT ACCOUNT, and not private account, like iAM12345@gmail.com

CANVAS: “Assignment 1 (Part-1) Mock up”:

docs.google.com/forms/d/e/1FAIpQLSe6pGqB0TVWsO3EmfrAXOd46yv3sEBvAYYaHNIt4-crU0KG8g/viewform

OENG1116 Assignment-1-2020: Pt-1 of 2

(1) This is a Part-1 of the Individual Assignment-1 and is due on Friday, end of Week-5 (03 April, 2020) 23:59pm Melbourne time. Submit electronically. There is NO NEED to submit or re-submit your assignment via Canvas.

(2) Total weighting of the Assignment-1 is 35% (the max TOTAL score in the Course), which includes 20% for this Part-1 of the Assignment. Attempt ALL tasks. Each Question shows its associated UNSCALED points. Please, be aware that the TOTAL max UNSCALED value for the Part-1 will be SCALED to 20% of the Course, so your shown mark for this Assignment is an UNSCALED mark!!!

(3) You can only assume successful submission, if you (immediately after submission) see a confirmation message from Google. Save and keep it as a proof.

(4) Message from Google must contain "edit your response" linked line. Keep this message, as the "edit" line enables you to edit your initial submission up to the due date.

(5) This Assignment is employing the most modern Google technology, enabling your mark to be "delivered" to your "doorstep", via your individual emails, which is different from traditional Canvas method, requiring you to LOG IN into your account and click many buttons to see your result. So, Canvas is not used as the main "marks delivery vehicle". Saying this, I will still upload your marks to Canvas, but towards the end of the Course.

(6) Important: the multiple-choice tasks may have SEVERAL CORRECT ANSWERS and, if this is the case, in order to earn question points, you need to tick ALL correct answers and, at the same time, do not tick ANY incorrect answer. Partially selected correct answers will result in zero mark for the question

Your email address (**pavel.trivailo@rmit.edu.au**) will be recorded when you submit this form. Not you? Switch account

*Required

Deadline for A-1, Part-1:

The deadline for the submission of the ***Individual Variants*** **Assignment-1, Pt-1** (OENG1116) is end of **Week-5, Friday, 03 Apr 2020.**

The aim of this submission IS:

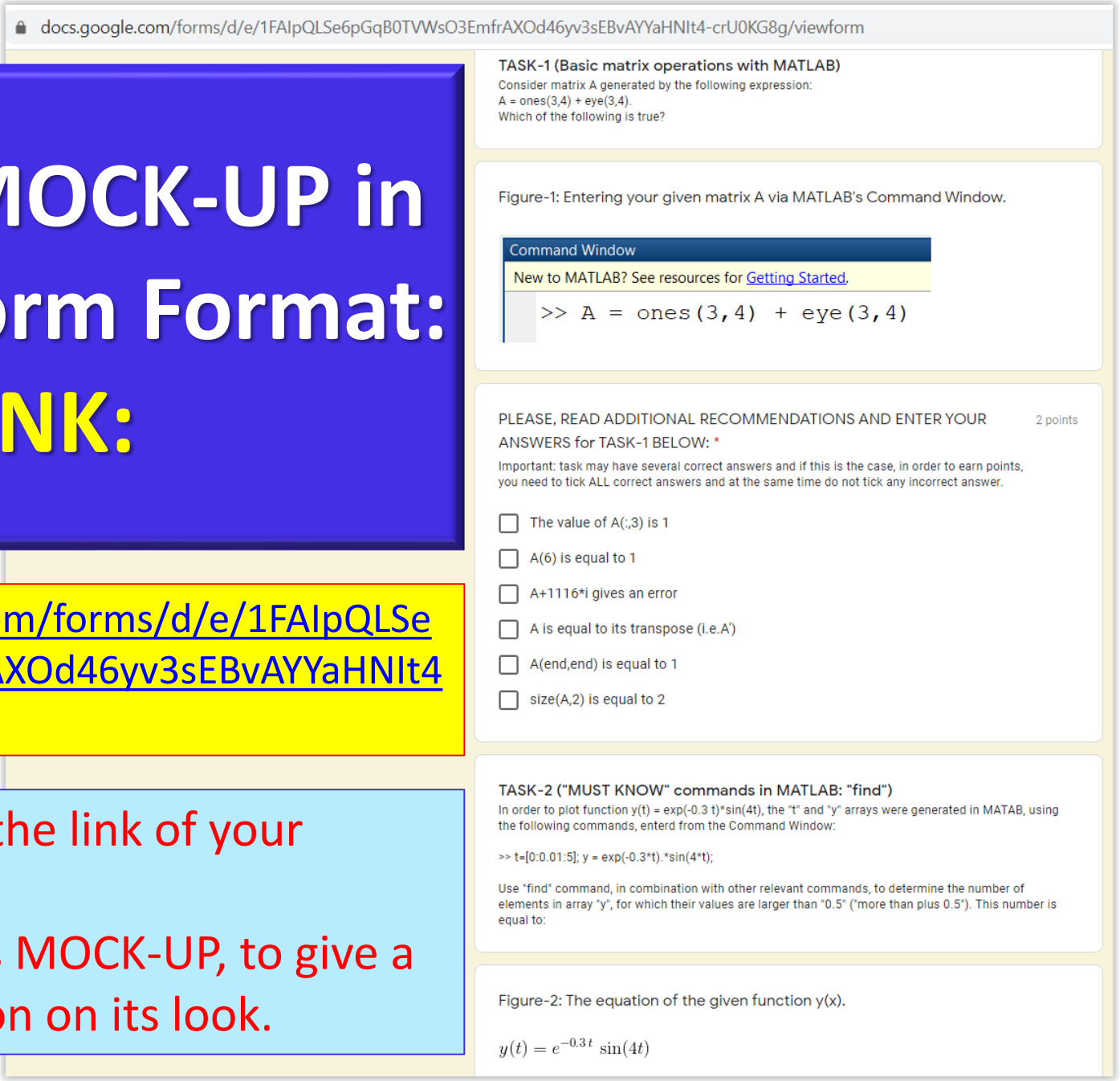
for each Master to receive a feedback from the Research Supervisor by the end of Week-6 to bring you certainty, peace of mind and to multiply your confidence!

Web Links to the Google Forms with your ***Individual Variants*** **Assignment-1, Pt-1** (OENG1116) will be sent via your Student Email shortly (before Sunday).
PLEASE, IMMEDIATELY REPORT TO ME (via email) IF YOU DID NOT RECEIVE THIS EMAIL FROM ME.

A1, Pt1 MOCK-UP in Google Form Format: LINK:

https://docs.google.com/forms/d/e/1FAIpQLSe6pGqB0TVWsO3EmfrAXOd46yv3sEBvAYYaHNIt4-crU0KG8g/viewform

Note: This is not the link of your assignment!
This is a link to its MOCK-UP, to give a general impression on its look.



A1, Pt1 : STRUCTURE, INSTRUCTIONS:

This is an individual assignment and its submission is via Google Form only.

In order to access the Assignment, you must be logged in into RMIT account.

If you attempt to do this from your private Gmail account, system would not be able to recognise your association with RMIT and access would not be granted.

Please, carefully read instructions on Assignment, which are also reproduced below for your convenience:

(1) This Individual Assignment is due on Friday, 03-April-2020, 23:59. Submit electronically.

(2) Total weighting of this A-1, Pt-1: 20%. Attempt ALL tasks. Questions have shown values.

(3) **You can only assume successful submission, if you (immediately after submission) receive confirmation message from Google. Keep it as a proof.**

(4) Email from Google must contain **EDIT** button. Keep this email, as it enables you to edit your initial submission up to the due date.

A1:

MATLAB MINI-TUTORIAL:

BASIC and NAVIGATION COMMANDS

ESSENTIAL COMMANDS:

TRY THESE EXAMPLES:

>> help pwd

>> doc pwd

>> help who

>> doc who

```
>> pwd
>> cd C:\OENG1116_W2a_MATLAB
>> a=[1:10]; b=-10:2:8;
>> who
>> size(a)
>> length(b)
>> find(b<0)
>> sprintf('PI = %6.2f',pi)
PI =   3.14
>> sprintf('PI = %20.18f',pi)
PI = 3.14159265358793100
```

ESSENTIAL COMMANDS:

TRY THESE EXAMPLES:

>> doc eye

>> doc zeros

>> doc ones

```
>> eye(3)
>> eye(3,3)
>> eye(2,7)
>> zeros(3,5)
>> ones(3,5)
>> a=[1:10]; b=-10:2:8;
>> mxA=[1:5; 10:10:50; 100:100:500];
>> mxA(2)
>> mxA(3)
>> mxA(4)
>> mxA(2,:)=[];
>> mxA
```

A1:

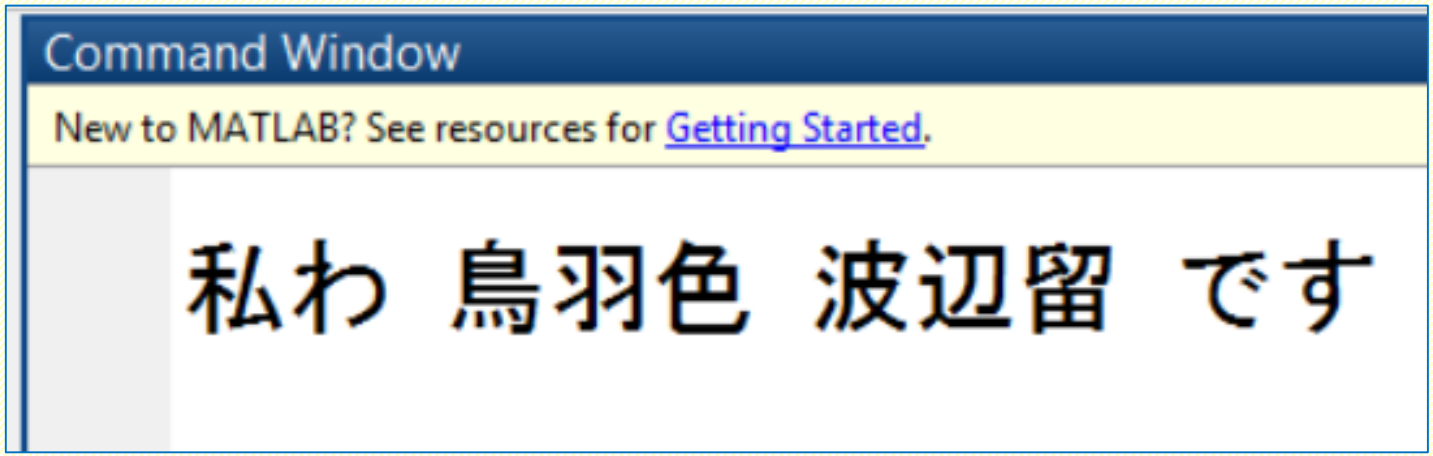
MATLAB MINI-TUTORIAL:

FORMATTED PRINT + LOOPS

MATLAB TRICK-1: Printing in Japanese!

```
% FORMATTED PRINT

clc; close('all');
pav0=[char(31169) char(12431)]; pav3=[char(12391) char(12377)];
pav12=[char(40165) char(32701) char(33394) char(32) ...
       char(27874) char(36794) char(30041)];
disp(sprintf('%s %s %s',pav0,pav12,pav3))
commandwindow
```

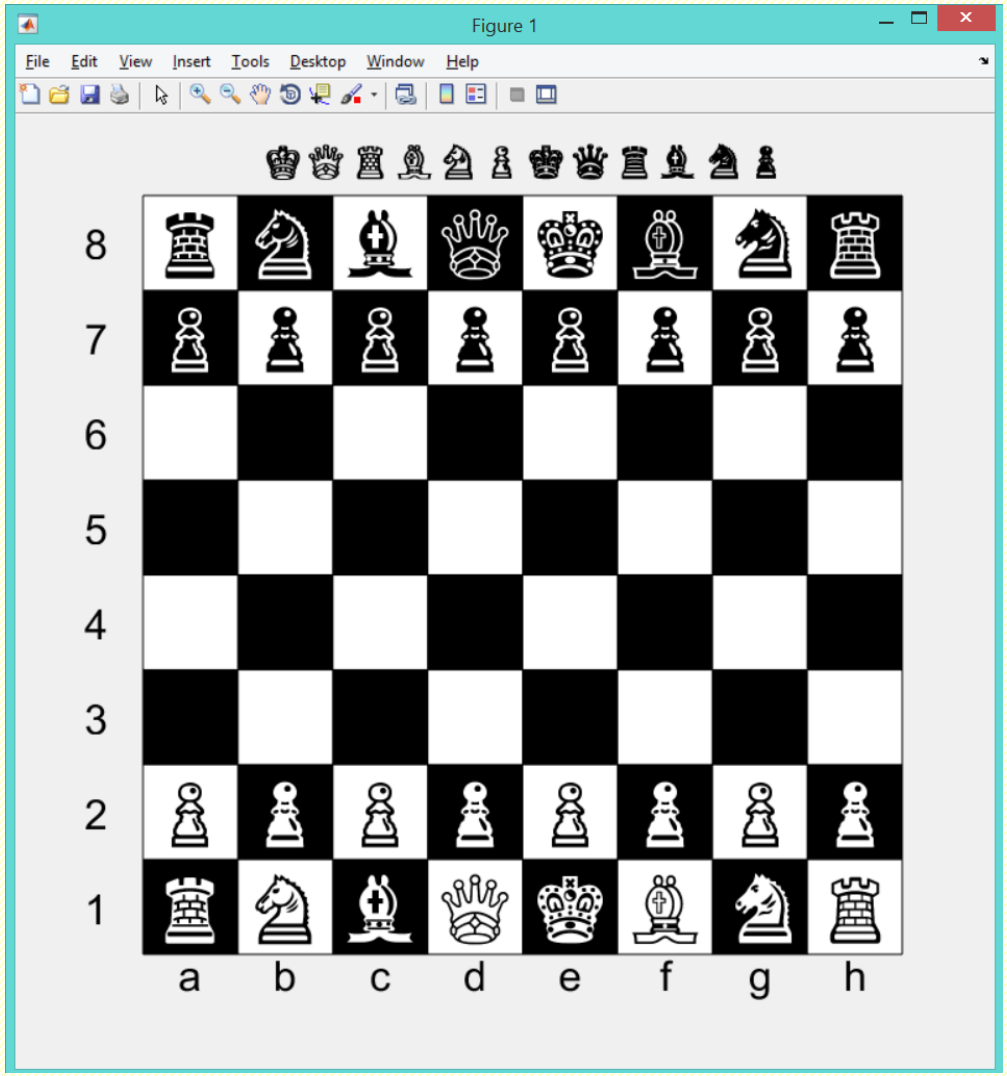
"**THREE DOTS**" Method to break long line without the error message



MATLAB TRICK:
Printing Chess Board!

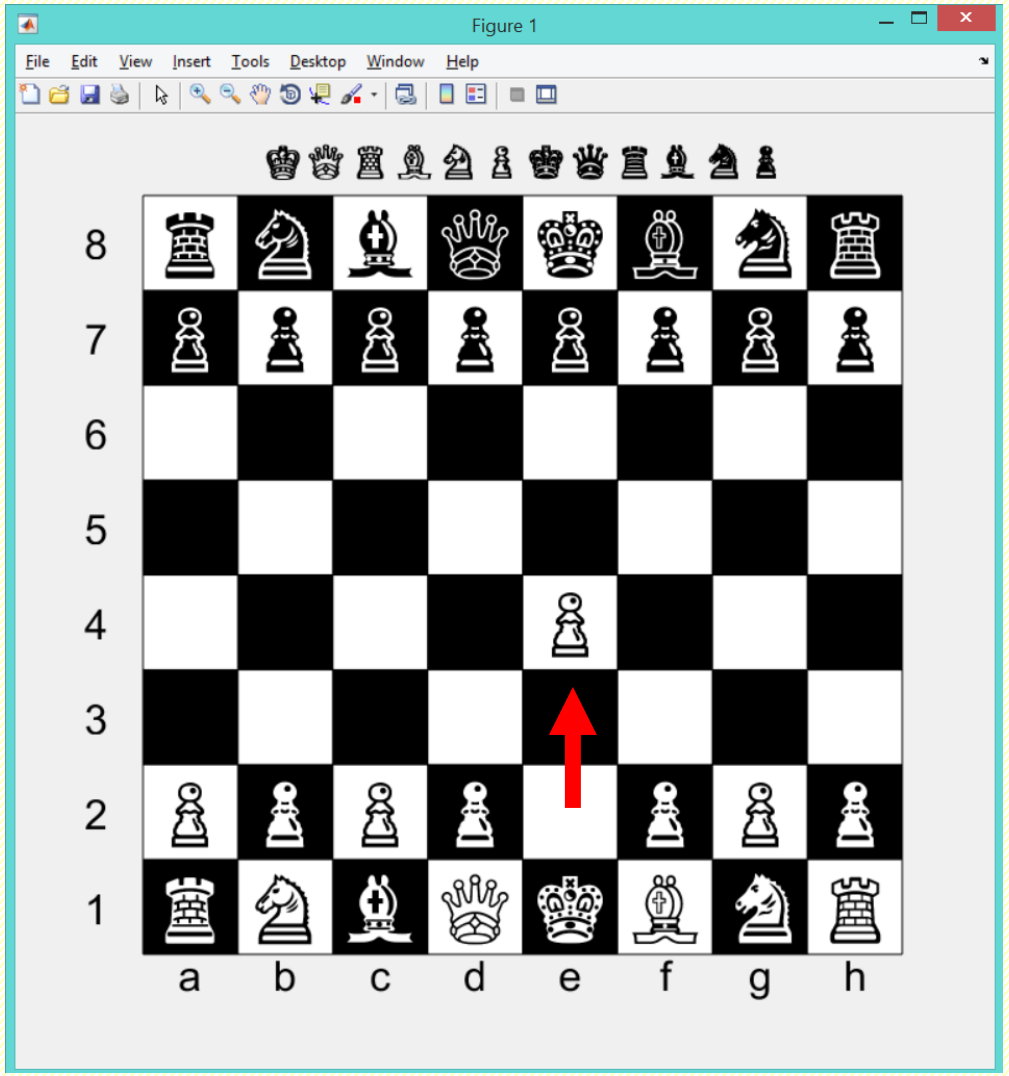
Programmed by:
Prof P.M.Trivailo

```
%%
% --- Prof P.M.Trivailo (C) 2019
t=1:1:9; x=sign(sin(pi*t));
pcolor(x'*x);
colormap(gray(2));
view([0 -90]);
axis ij;
axis square;
title(char(9812:9823),'FontSize',24);
```



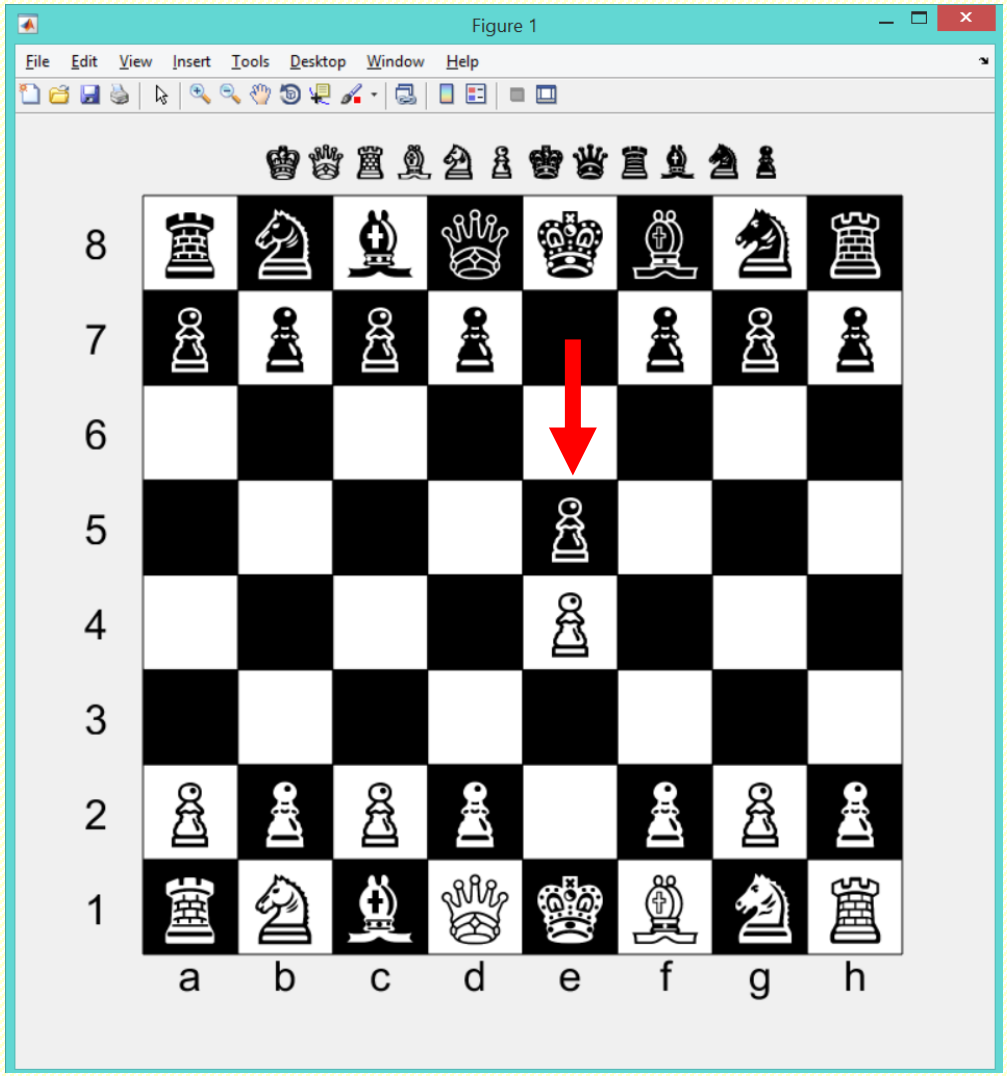
MATLAB TRICK:
Playing Chess in MATLAB!

1. e2-e4



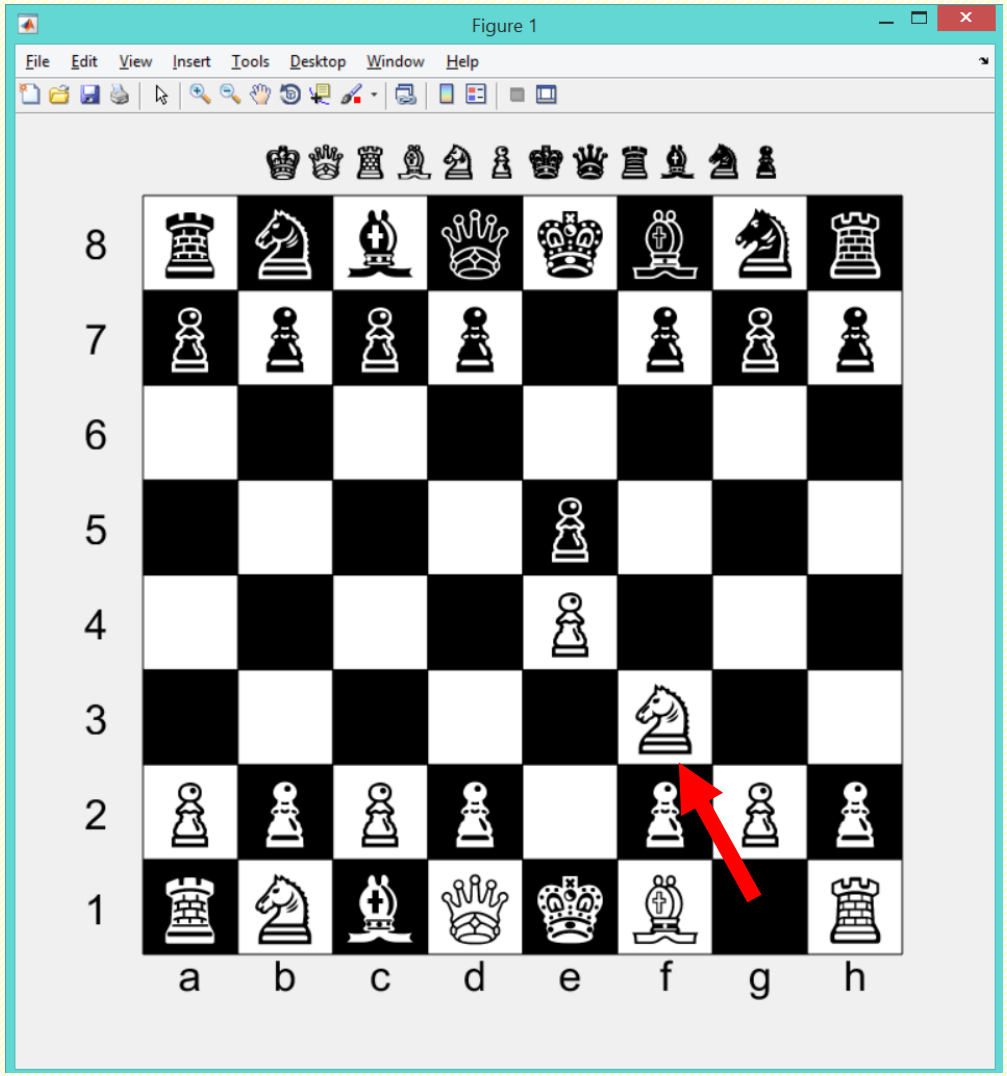
MATLAB TRICK:
Playing Chess in MATLAB!

1. e2-e4 e7-e5



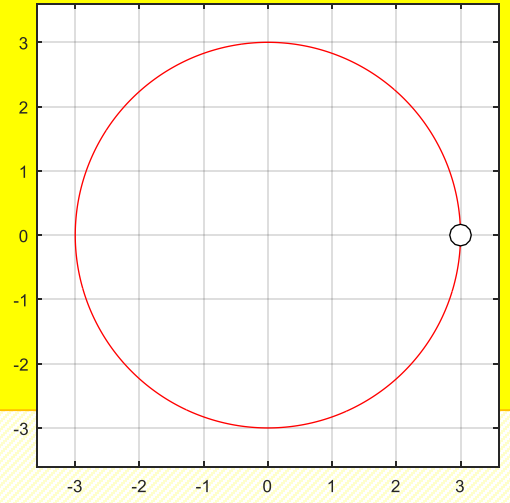
MATLAB TRICK:
Playing Chess in MATLAB!

1. e2-e4 e7-e5
2. Ng1-f3



MATLAB TRICK-2: Animation! ("for" loop and "set" commands)

```
% Plot basic circle
th=0:1:360; th=th*pi/180; R=3;
X=R*cos(th); Y=R*sin(th);

figure; plot(X,Y,'r');
axis([-1 1 -1 1]*R*1.2); axis square; grid on; hold on;

dot=line('XData',X(1),'YData',Y(1),...
    'Marker','o','MarkerSize',12,'MarkerFaceColor','w');

% Animate 'dot' marker
for i=1:length(th)
    set(dot,'XData',X(i),'YData',Y(i));
    drawnow; pause(0.01)
end
```



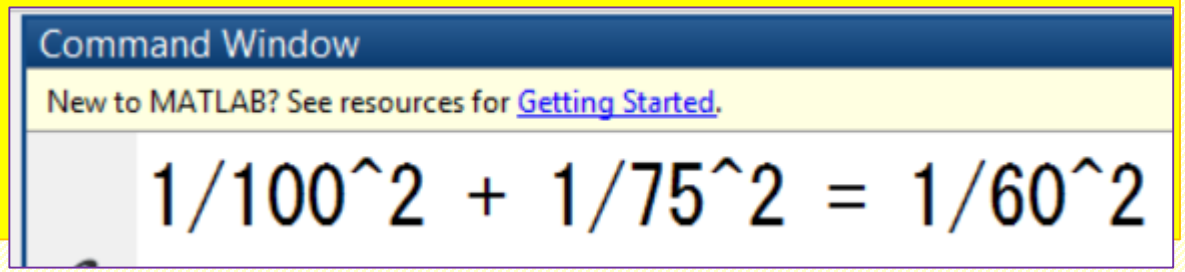
MATLAB TRICK-3: “for” Loops + “if”

Solve for integers a, b and c
for the range [0 120]:

$$\frac{1}{a^2} + \frac{1}{b^2} = \frac{1}{c^2}$$

MATLAB TRICK-3: “for” Loops + “if”

```
clear; close all; clc
C=120;
for c=1:C
    for a=1:C
        for b=1:C
            if 1/c^2-1/a^2-1/b^2 == 0,
              disp(sprintf('1/%3i^2 + 1/%i^2 = 1/%i^2 ',a,b,c))
            end
        end
    end
end
commandwindow
```



Solution:

$$\frac{1}{75^2}+\frac{1}{100^2}=\frac{1}{60^2}$$

PROOF

```
Command Window
New to MATLAB? See resources for Getting Started.

>> factor(100^2+75^2)

ans =

     5     5     5     5     5     5
```

$$LHS = \frac{1}{75^2} + \frac{1}{100^2} = \frac{100^2 + 75^2}{75^2 \times 100^2}$$

```
Command Window
New to MATLAB? See resources for Getting Started.

>> factor(75^2*100^2)

ans =

  Columns 1 thr

     2     2     2     2     3     3     5     5     5     5     5

  Columns 12 through 14

     5     5     5
```

$$LHS = \frac{5^6}{2^4 \times 3^2 \times 5^8} = \frac{1}{2^4 \times 3^2 \times 5^2} = \frac{1}{60^2}$$

$$\frac{1}{75^2} + \frac{1}{100^2} = \frac{1}{60^2} \quad \Leftarrow \text{verified !!!}$$

OVERVIEW:

WHAT YOU MUST KNOW BY NOW?

MATLAB Basics: “help”

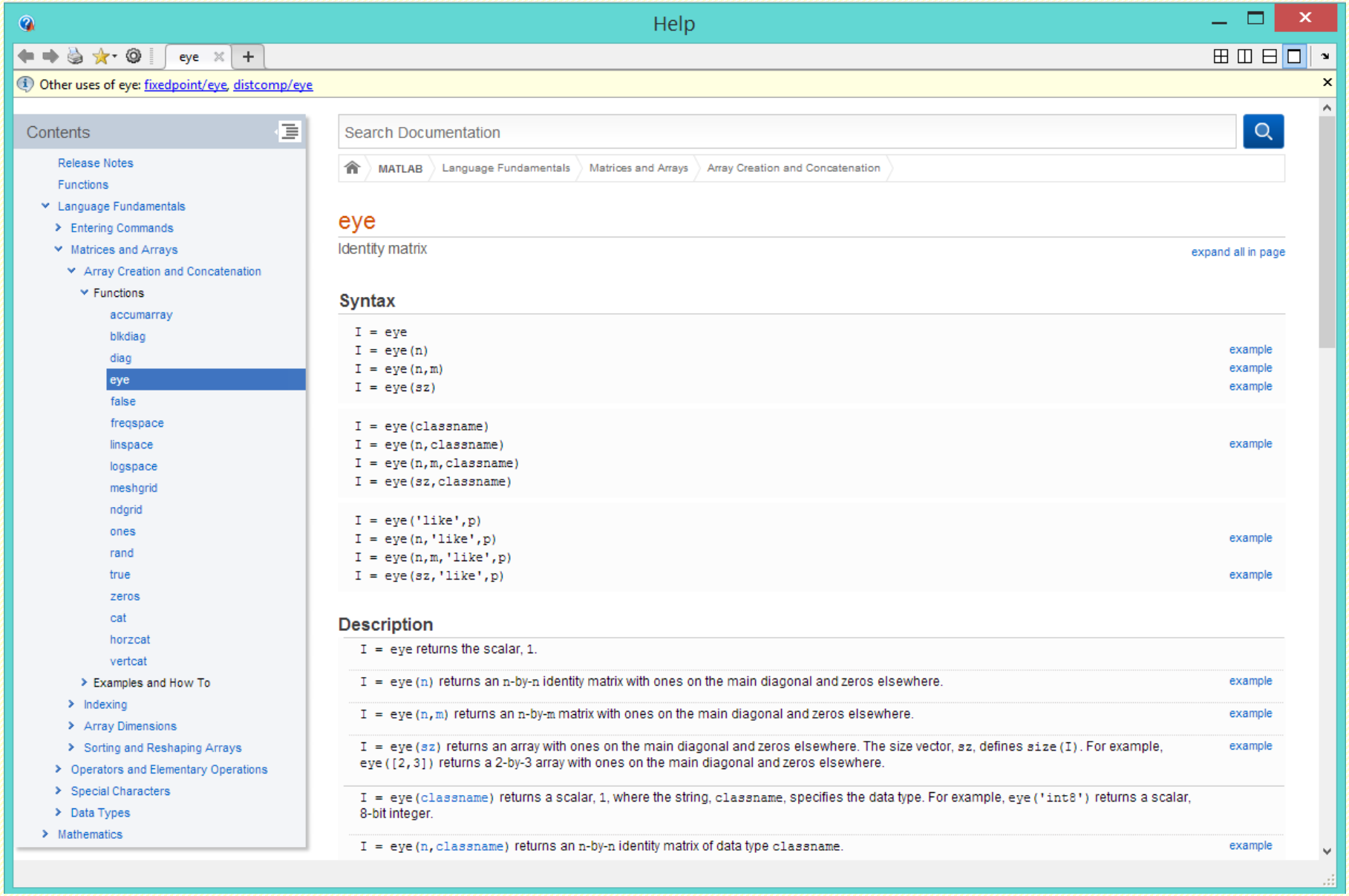
Command Window

New to MATLAB? See resources for Getting Started.

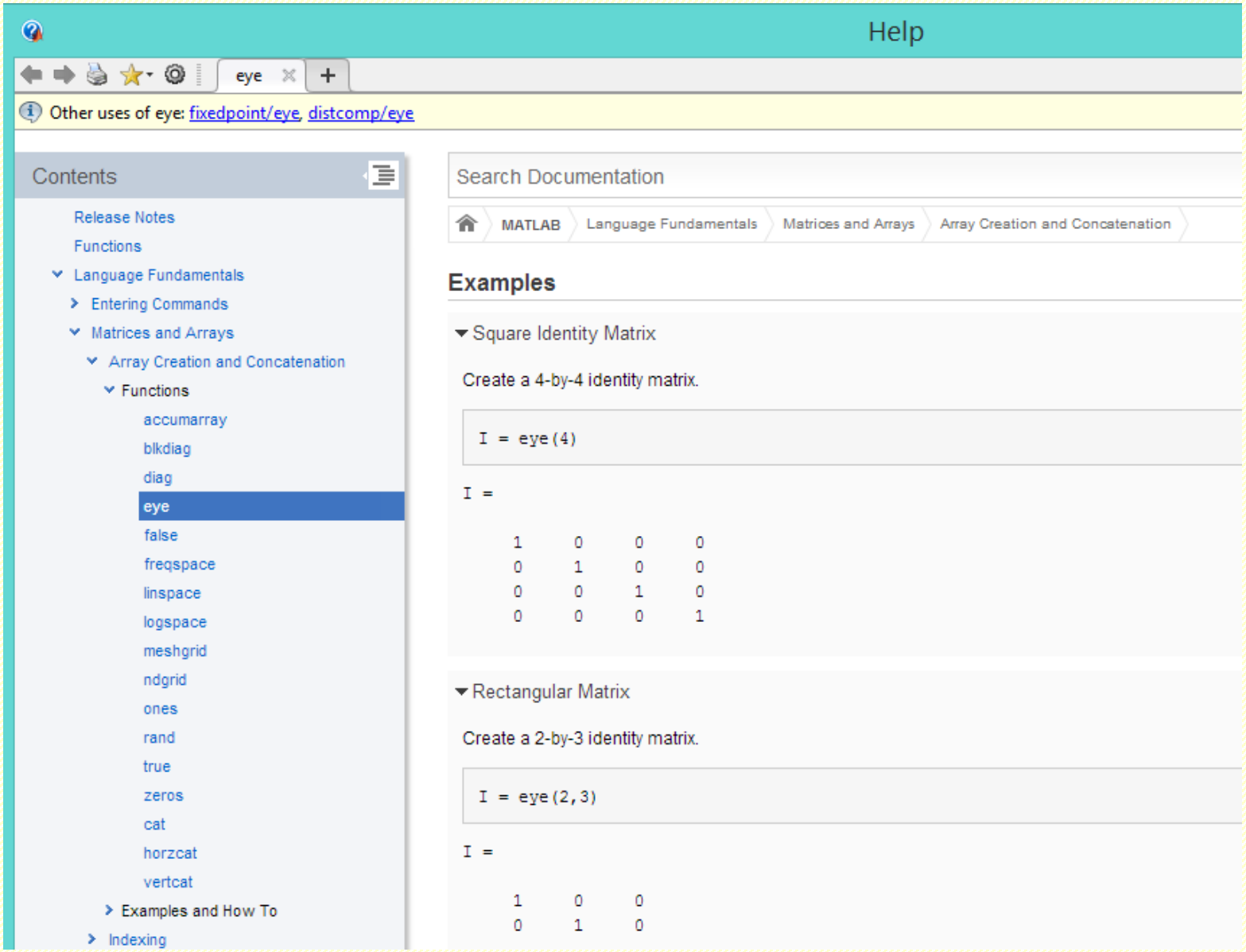
```
>> help grid
 grid   Grid lines.
    grid ON adds major grid lines to the current axes.
    grid OFF removes major and minor grid lines from the current axes.
    grid MINOR toggles the minor grid lines of the current axes.
    grid, by itself, toggles the major grid lines of the current axes.
    grid(AX,...) uses axes AX instead of the current axes.

    grid sets the XGrid, YGrid, and ZGrid properties of
    the current axes.

    AX.XMinorGrid = 'on' turns on the minor grid.

    See also title, xlabel, ylabel, zlabel, axes, plot, box.

    Reference page in Help browser
       doc grid
```

%% TRY also:
>> help clc
>> help close
>> help clear
>> help plot
>> help zeros
>> help eye
>> help ones
>> help grid
>> help xlabel
>> help title
>> help sprintf

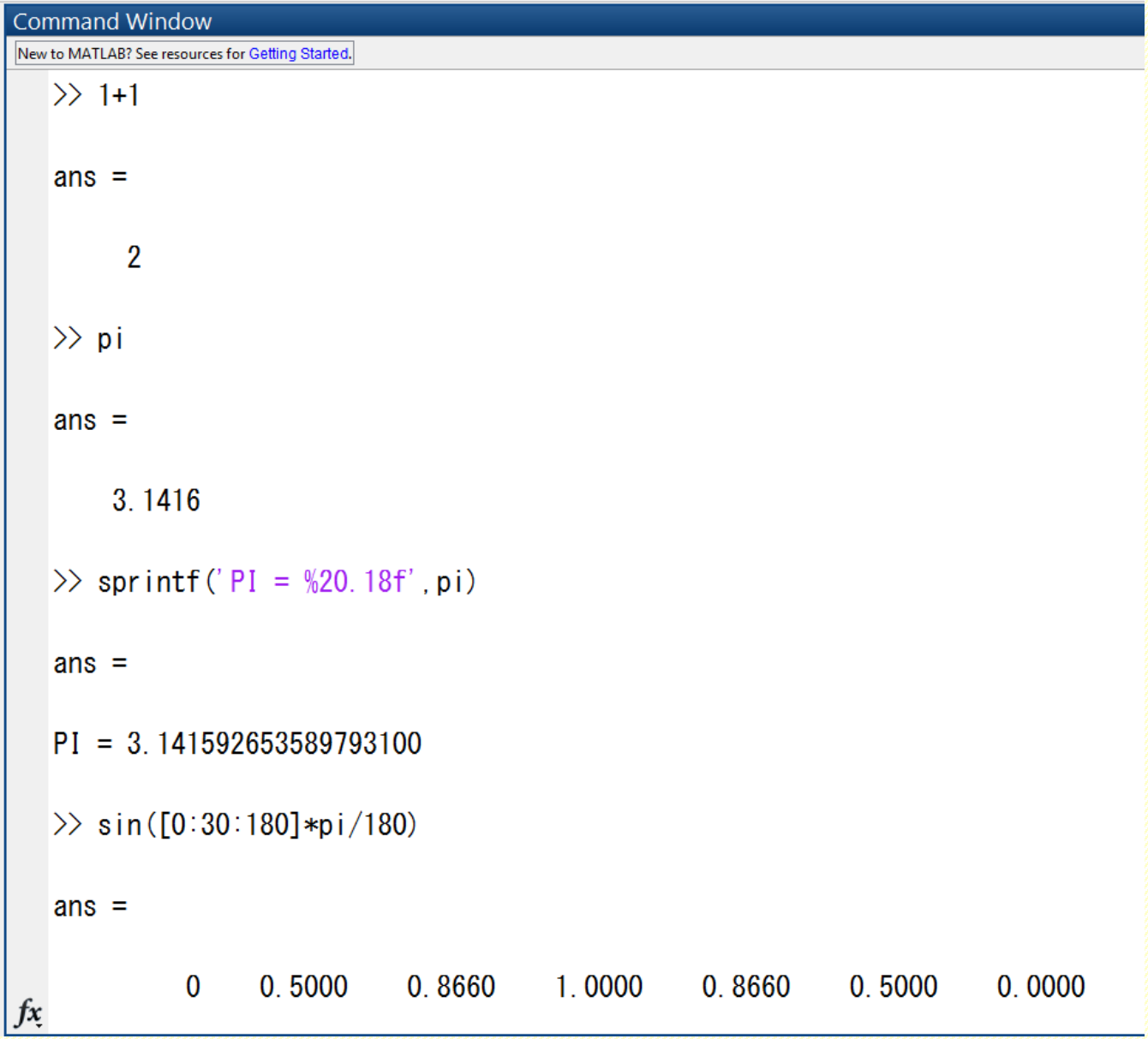
MATLAB Basics: "doc" ["doc eye" example]



MATLAB Basics: "doc" (cont'd)



MATLAB Basics: Command Window



MATLAB Basics: ARRAYS

Command Window

New to MATLAB? See resources for Getting Started.

```
>> a=[1:8; 10:10:80; 100:100:800; 1000:1000:8000]

a =

     1     2     3     4     5     6     7     8
    10    20    30    40    50    60    70    80
   100   200   300   400   500   600   700   800
  1000  2000  3000  4000  5000  6000  7000  8000

>> a(3,:)

ans =

   100   200   300   400   500   600   700   800

>> a(3:4,2:7)

ans =

   200   300   400   500   600   700
  2000  3000  4000  5000  6000  7000
```


>> a=[1:8; 10:10:80; 100:100:800; 1000:1000:8000]
>> a(3,:)
>> a(3:4,2:7)

MATLAB Basics: LOOPS

```
>> for i=1:10, disp(sprintf('i = %2i  i x PI = %g',i,i*pi)); end
```

Command Window

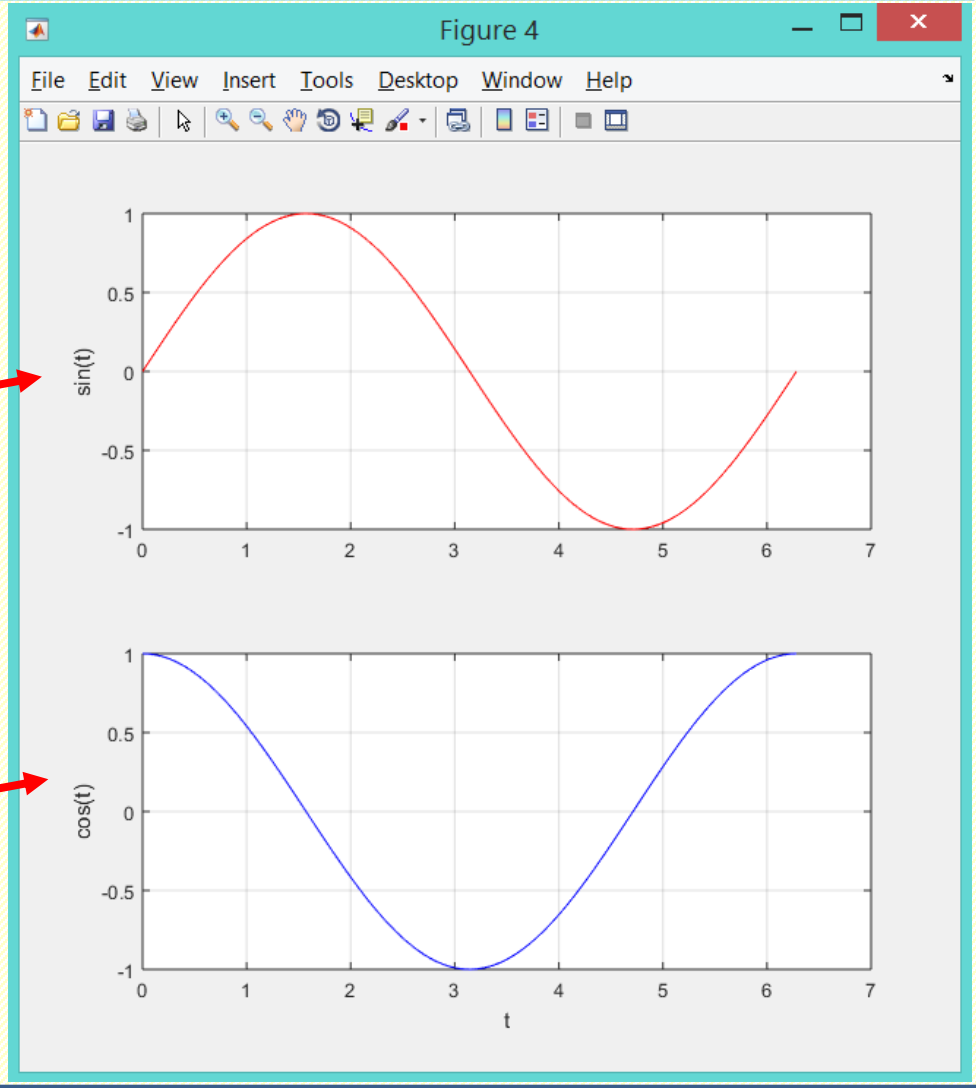
New to MATLAB? See resources for Getting Started.

```
>> for i=1:10, disp(sprintf('i = %2i  i x PI = %g', i, i*pi)); end
i =  1  i x PI = 3.14159
i =  2  i x PI = 6.28319
i =  3  i x PI = 9.42478
i =  4  i x PI = 12.5664
i =  5  i x PI = 15.708
i =  6  i x PI = 18.8496
i =  7  i x PI = 21.9911
i =  8  i x PI = 25.1327
i =  9  i x PI = 28.2743
i = 10  i x PI = 31.4159
fx >>
```

MATLAB Basics: *.m files

"MUST KNOW" Plotting Example

```
%%
t=[0:0.01:1]*2*pi;

figure

subplot(2,1,1);
plot(t,sin(t),'r');
ylabel('sin(t)')
grid on


subplot(2,1,2);
plot(t,cos(t),'b');
ylabel('cos(t)')
xlabel('t')
grid on
```



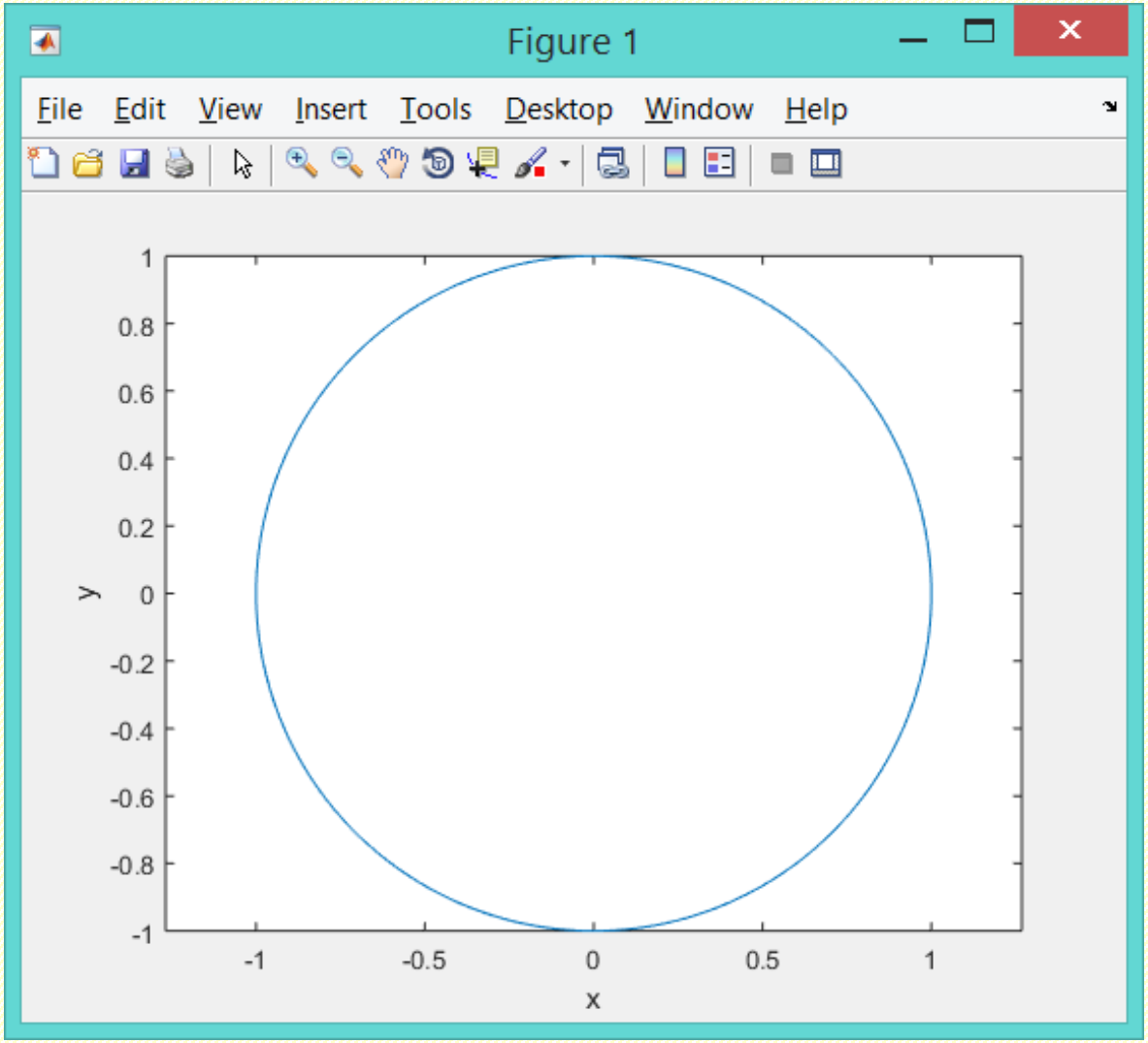
MATLAB Basics: *.m files

"MUST KNOW" Plotting Example

```
%%
th=[0:1:360]*pi/180;
x=1*cos(th);
y=1*sin(th);

figure;
h=plot(x,y);
xlabel('x');
ylabel('y');
axis equal

get(h)
```

Press **Ctrl+ENTER** to execute the cell

Press **Ctrl+Shift+ENTER** to execute the cell & advance to the next cell



MATLAB Basics: *.m files

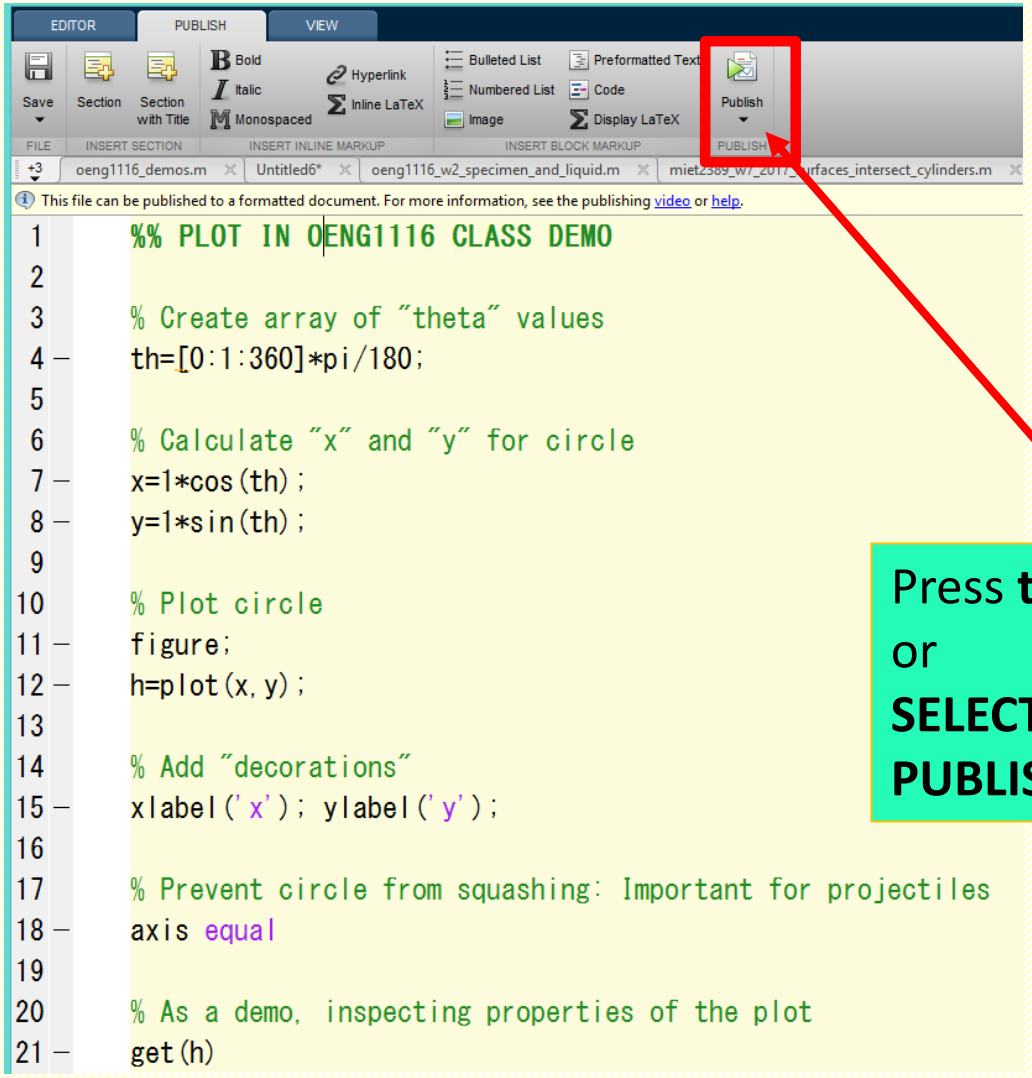
"MUST KNOW" Commands

NOTE:
This is my (PMT) demo with specific directories/files names.
Your example may involve different names.

```
>> pwd

>> cd C:\Users\owner\Documents\RMIT\__2018_1116\DEMOSinCLASS

>> ls

>> ls *.m

>> edit plot_circle

>> plot_circle
```

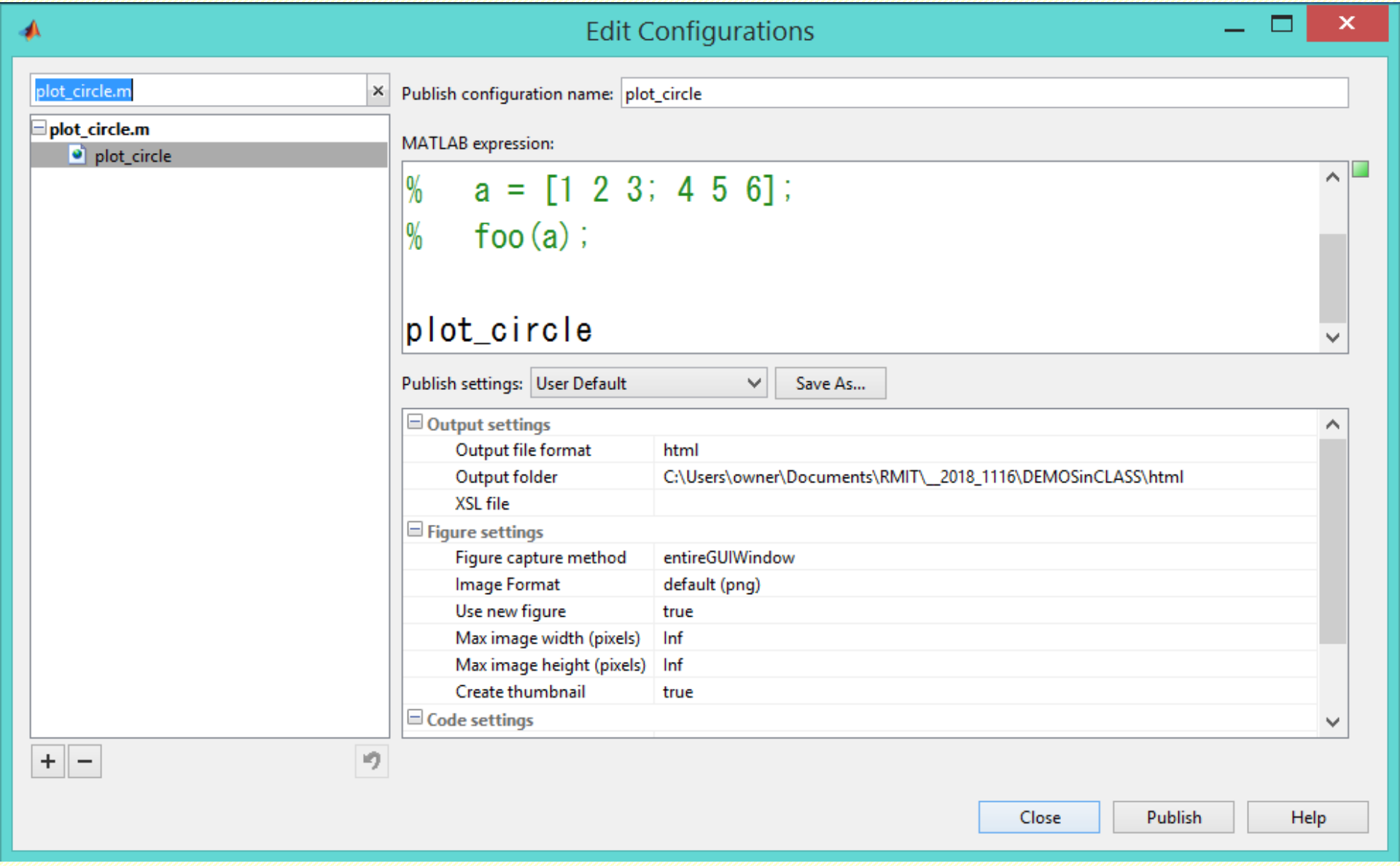
MATLAB Basics: PUBLISHING

Basic Publishing Example



Press **to PUBLISH**
or
SELECT
PUBLISHING OPTIONS

MATLAB Basics: PUBLISHING



Web Browser - PLOT IN OENG1116 CLASS DEMO

PLOT IN OENG1116 CLASS DEMO

Location: file:///C:/Users/owner/Documents/RMIT/__2018_1116/DEMOSinCLASS/html/plot_circle.html

```
% Create array of "theta" values
th=[0:1:360]*pi/180;

% Calculate "x" and "y" for circle
x=1*cos(th);
y=1*sin(th);

% Plot circle
figure;
h=plot(x,y);

% Add "decorations"
xlabel('x');
ylabel('y');

% Prevent circle from squashing: Important for projectiles

axis equal

% As a demo, inspecting properties of the plot
get(h)
```

BEGINNING OF THE PUBLISHED DOCUMENT:

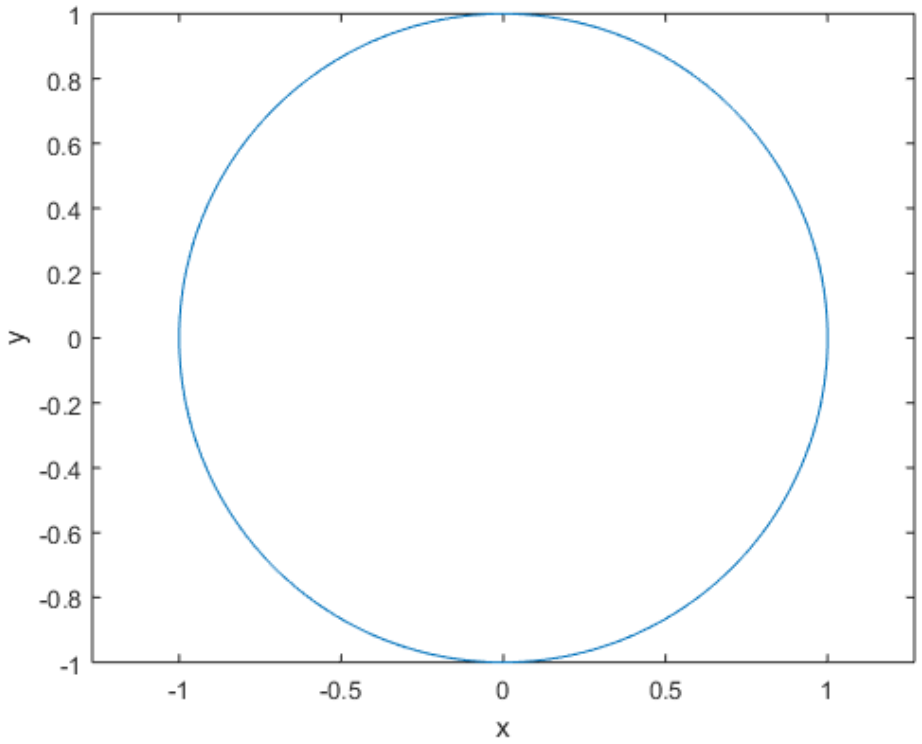
```
    AlignVertexCenters: 'off'
            Annotation: [1x1 matlab.graphics.eventdata.Annotation]
          BeingDeleted: 'off'
            BusyAction: 'queue'
         ButtonDownFcn: ''
              Children: []
              Clipping: 'on'
                 Color: [0 0.4470 0.7410]
             CreateFcn: ''
             DeleteFcn: ''
           DisplayName: ''
```

Web Browser - PLOT IN OENG1116 CLASS DEMO

PLOT IN OENG1116 CLASS DEMO

Location: file:///C:/Users/owner/Documents/RMIT/__2018_1116/DEMOSinCLASS/html/plot_circle.html

```
YDataSource: ''
      ZData: [1x0 double]
ZDataSource: ''
```

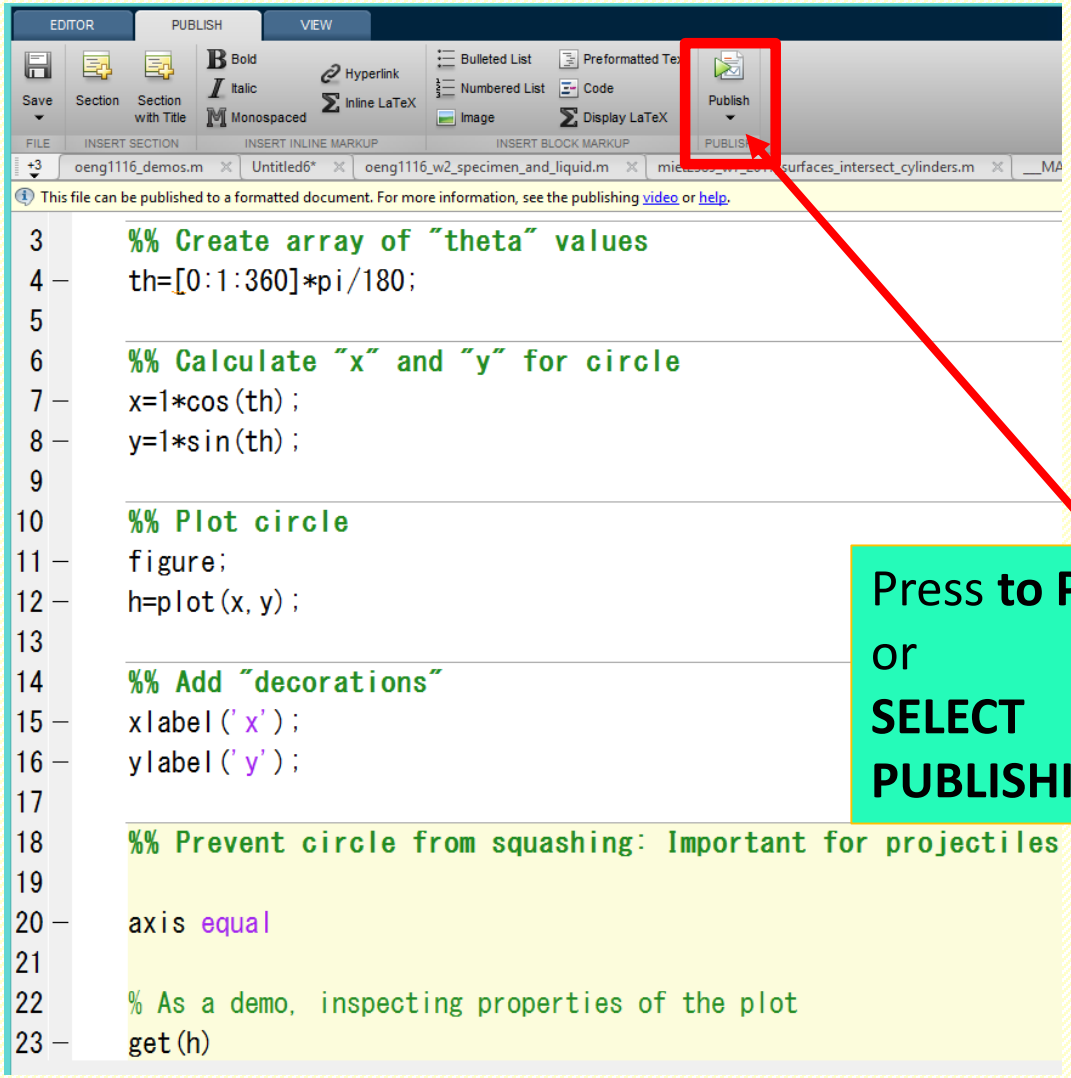


END OF THE PUBLISHED DOCUMENT:

Published with MATLAB® R2015a

MATLAB Basics: PUBLISHING

Slightly Modified Publishing Example



Press **to PUBLISH**
or
SELECT
PUBLISHING OPTIONS

Web Browser - PLOT IN OENG1116 CLASS DEMO

PLOT IN OENG1116 CLASS DEMO

Location: file:///C:/Users/owner/Documents/RMIT/__2018_1116/DEMOSinCLASS/html/plot_circle2.html

PLOT IN OENG1116 CLASS DEMO

Contents

BEGINNING OF THE PUBLISHED DOCUMENT:

Create array of "theta" values

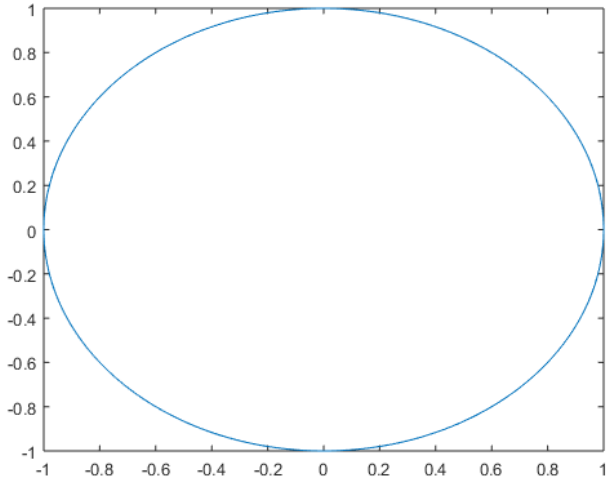
```
th=[0:1:360]*pi/180;
```

Calculate "x" and "y" for circle

```
x=1*cos(th);
y=1*sin(th);
```

Plot circle

```
figure;
h=plot(x,y);
```



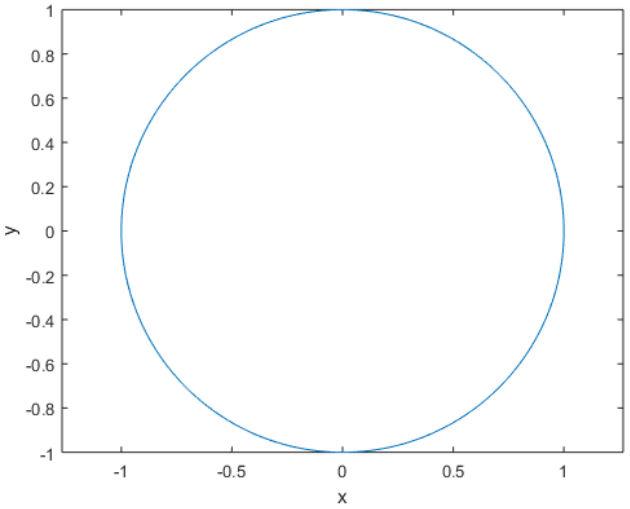
Web Browser - PLOT IN OENG1116 CLASS DEMO

PLOT IN OENG1116 CLASS DEMO

Location: file:///C:/Users/owner/Documents/RMIT/__2018_1116/DEMOSinCLASS/html/plot_circle2.html

```
          LineStyle: '-'
          LineWidth: 0.5000
             Marker: 'none'
     MarkerEdgeColor: 'auto'
     MarkerFaceColor: 'none'
         MarkerSize: 6
             Parent: [1x1 Axes]
      PickableParts: 'visible'
           Selected: 'off'
  SelectionHighlight: 'on'
                Tag: ''
               Type: 'line'
    UIContextMenu: []
           UserData: []
            Visible: 'on'
              XData: [1x361 double]
         XDataMode: 'manual'
       XDataSource: ''
              YData: [1x361 double]
       YDataSource: ''
              ZData: [1x0 double]
       ZDataSource: ''
```

END OF THE PUBLISHED DOCUMENT:



Published with MATLAB® R2015a

MATLAB Basics: Useful Commands

```
pwd % !!!!!!!!!!!!
path % !!!!!!!!!!!!

help
doc
diary

ls
who
save

plot([1 2],[10 15])
subplot(2,1,1)
grid on
xlabel ('x')
close; clear; clc
title('OENG1116')
legend('a','b')
```

```
line
patch
axis
axis square
axis equal

zeros(2,2)
eye(3)
ones(2,4)

size      % !!!!
length    % !!!!

interp1   % !!!!

eig       % !!!!

class     % !!!!
```

```
% comments and %% cells

format short
format long
sprintf     % !!!!!!!!!!!!!!!!!!!!!!
find        % !!!!!!!!!!!!!!!!!!!!!!

% Splitting long lines using "..."

ode45       % !!!!!!!!!!!!!!!!!!!!!!
% see examples in Lecture-2

% anonymous functions
zzz = @(a) a.^2          % !!!!!!!
zzz(1)
zzz(2)
zzz([1 2 3])
zzz([1 2 3; 4 5 6])
```

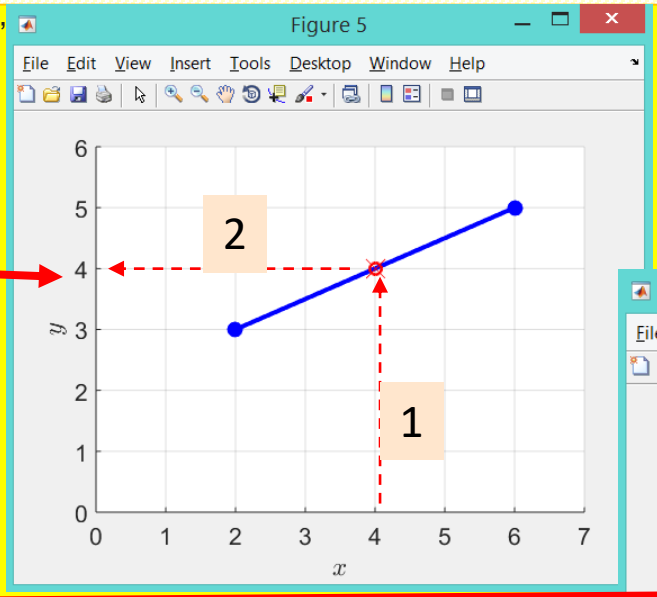
```
% !!!!!!!!!!
for
end


% !!!!!!!!!!
function


% !!!!!!!!!!
if
end


&    % !!! AND
|    % !!! OR
==   % !!!
```

MATLAB Basics: Useful Examples

%% Useful examples of using "**interp1**"

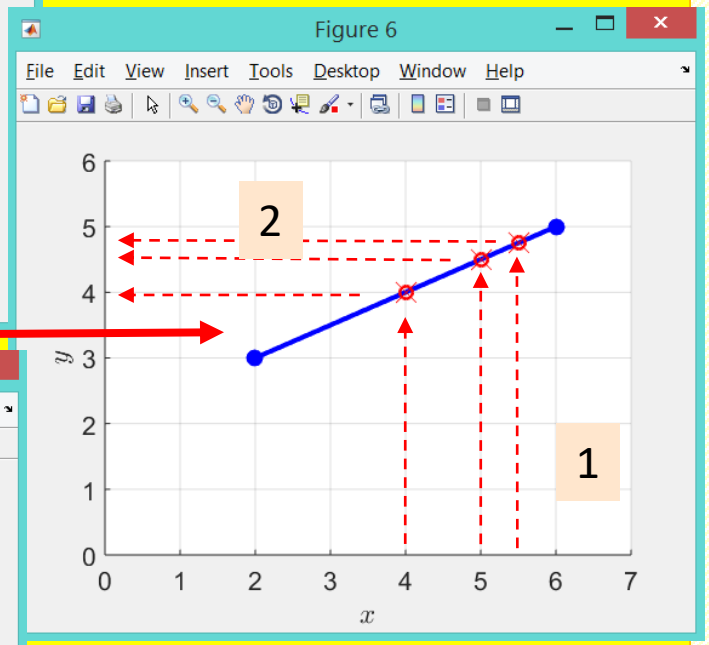
% Example-1

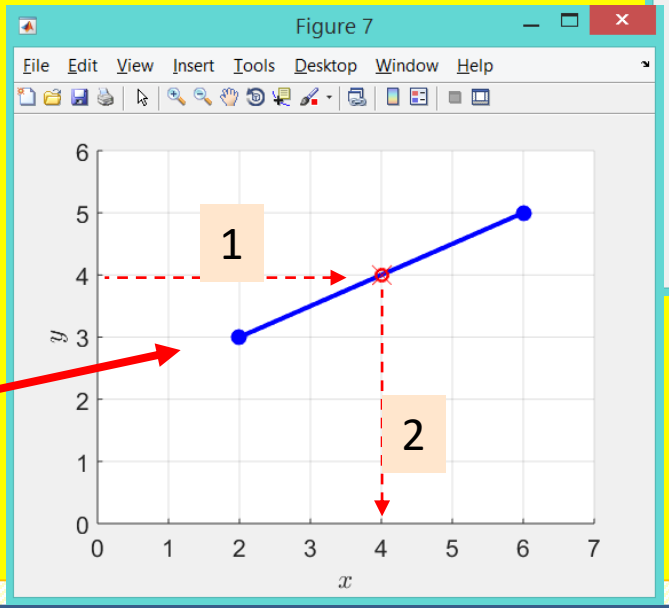
```
figure; hold on;
X=[2 6]; Y=[3 5]; plot(X,Y,'b-*');
Xi=4;
Yi=interp1(X,Y,Xi);
plot(Xi,Yi,'ro','LineWidth',3);
```



% Example-2

```
figure; hold on;
X=[2 6]; Y=[3 5]; plot(X,Y,'b-*');
Xi=[4 5 5.5];
Yi=interp1(X,Y,Xi);
plot(Xi,Yi,'ro','LineWidth',3);
```





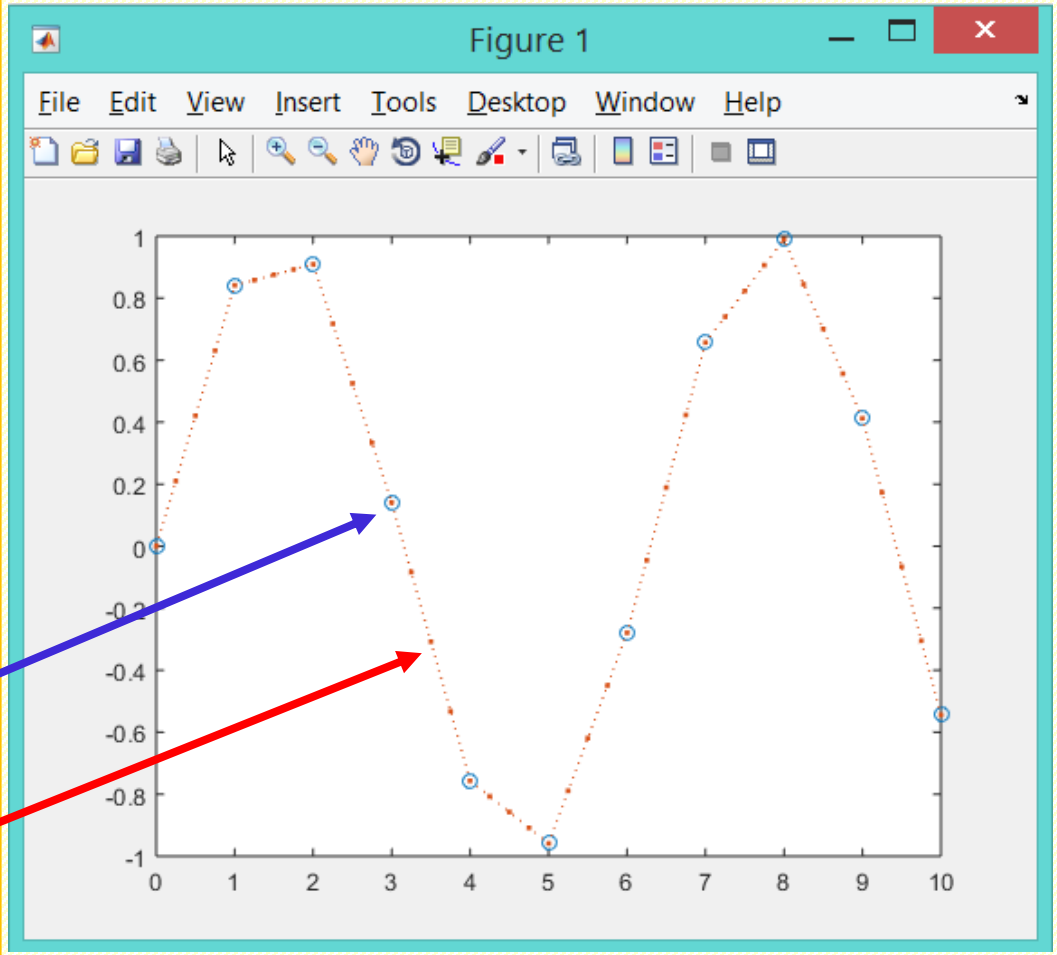
% Example-3

```
figure; hold on;
X=[2 6]; Y=[3 5]; plot(X,Y,'b-*');
Yi=[4];
Xi=interp1(Y,X,Yi);
plot(Xi,Yi,'rx','LineWidth',3);
```

MATLAB Basics: Useful Examples

%% Useful examples from MATLAB interp1 help

```
% Example-4
X = 0:10;
V = sin(X);

Xq = 0:.25:10;
Vq = interp1(X,V,Xq);

plot(X,V,'o',Xq,Vq,':.')
```

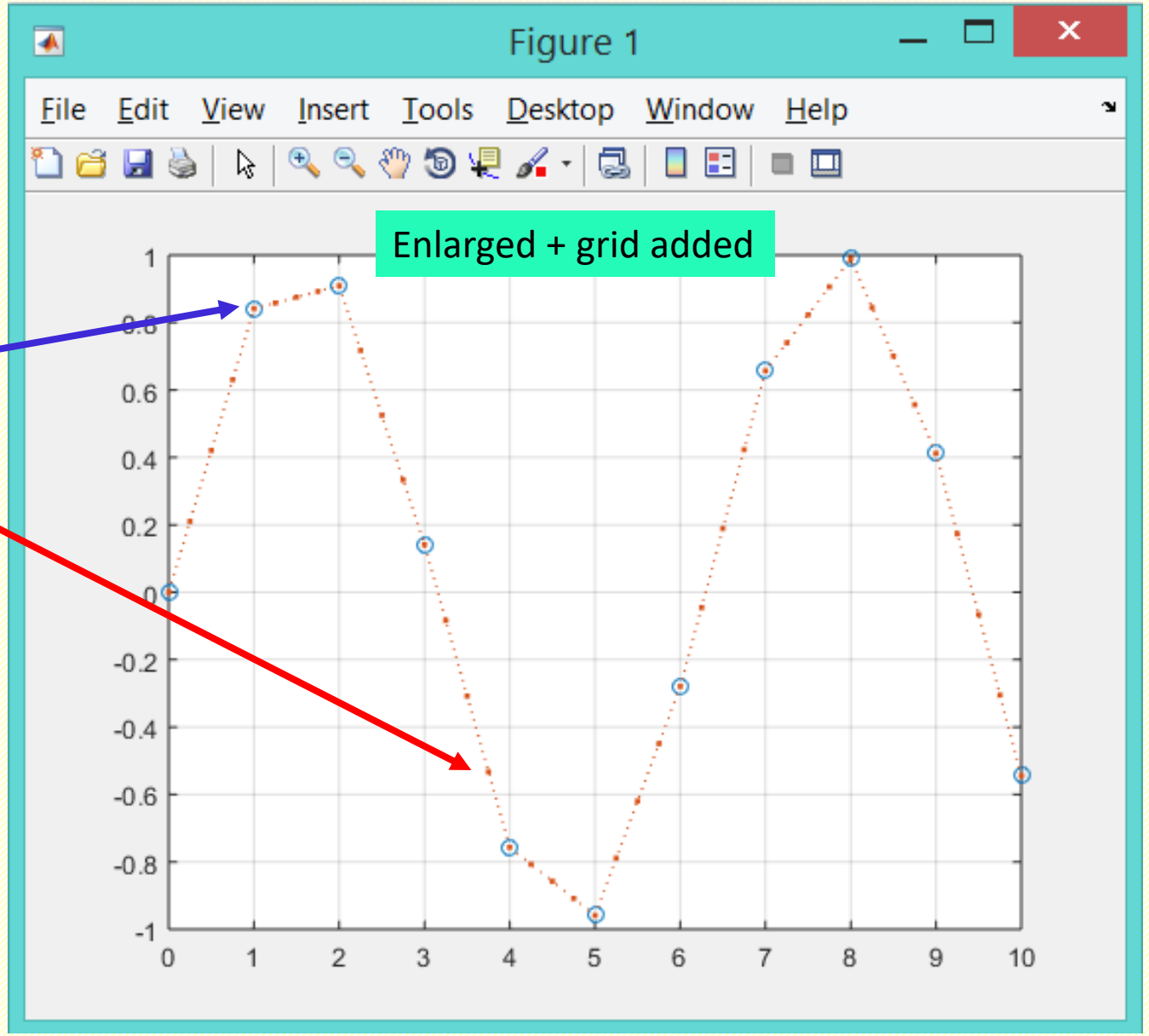


MATLAB Basics: Useful Examples

%% Useful examples from MATLAB **interp1** help

% Example-4

```
X = 0:10;
V = sin(X);

Xq = 0:.25:10;
Vq = interp1(X,V,Xq);

plot(X,V,'o',Xq,Vq,':.')

grid on
```



MATLAB Basics: Useful Examples

Command Window

New to MATLAB? See resources for Getting Started.

```
>> a=[1 2 3 4 5 6 -7 -8 9 10]

a =

     1     2     3     4     5     6    -7    -8     9    10

>> idx=find(a<0)

idx =

     7     8
```

LECTURE MATERIAL:

FIRST ORDER SYSTEMS

NEWTON's LAW OF COOLING (and HEATING):

MODELLING

Newton's Law of Cooling (and Heating):

The rate at which the temperature of an object changes is proportional to the difference between the temperature of the object and the temperature of the surroundings:

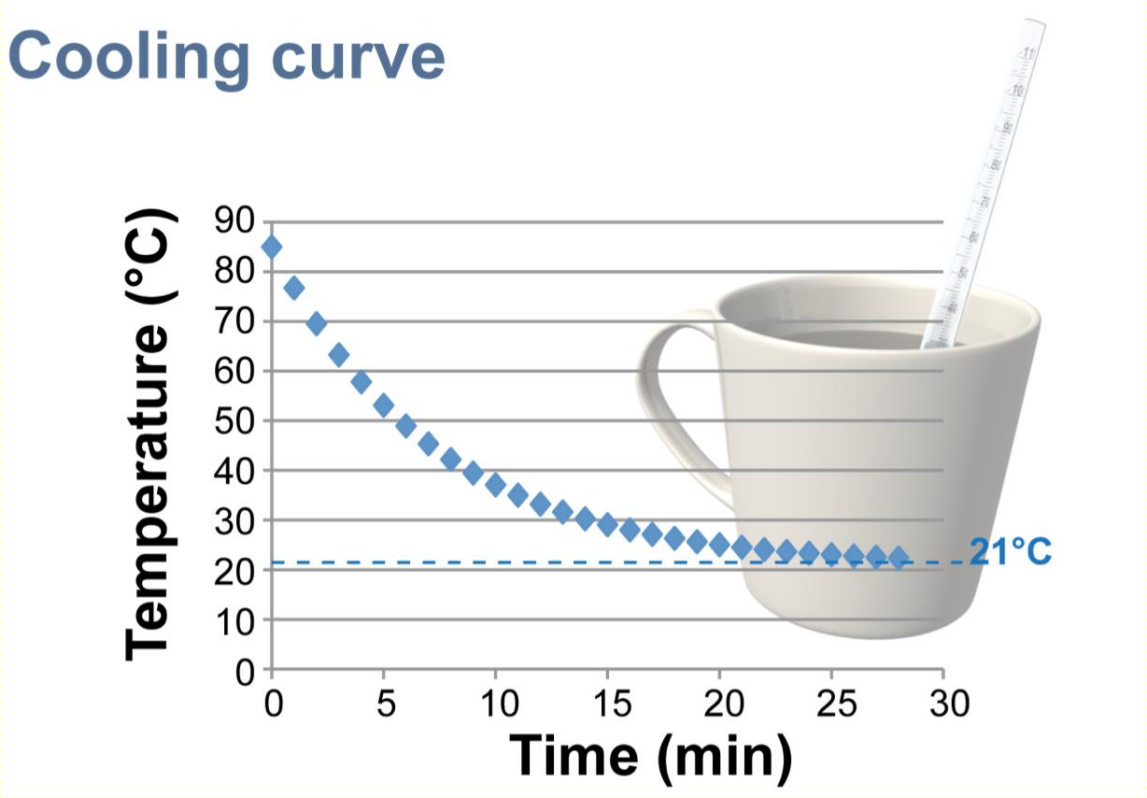
$$\frac{dT}{dt} = k(T - T_s)$$

where: t time; k - the constant of proportionality;
$T(t)$ - temperature of the object at time t;
$T_s(t)$ - temperature of the surrounding area at time t.

Courtesy: Texas Instruments, https://education.ti.com/-/media/97CF63CF732A4BB79FB2632514AD5159

Newton's Law of Cooling (and Heating): Example for Everyone to Understand

The rate of cooling depends on the temperature difference between the two objects.



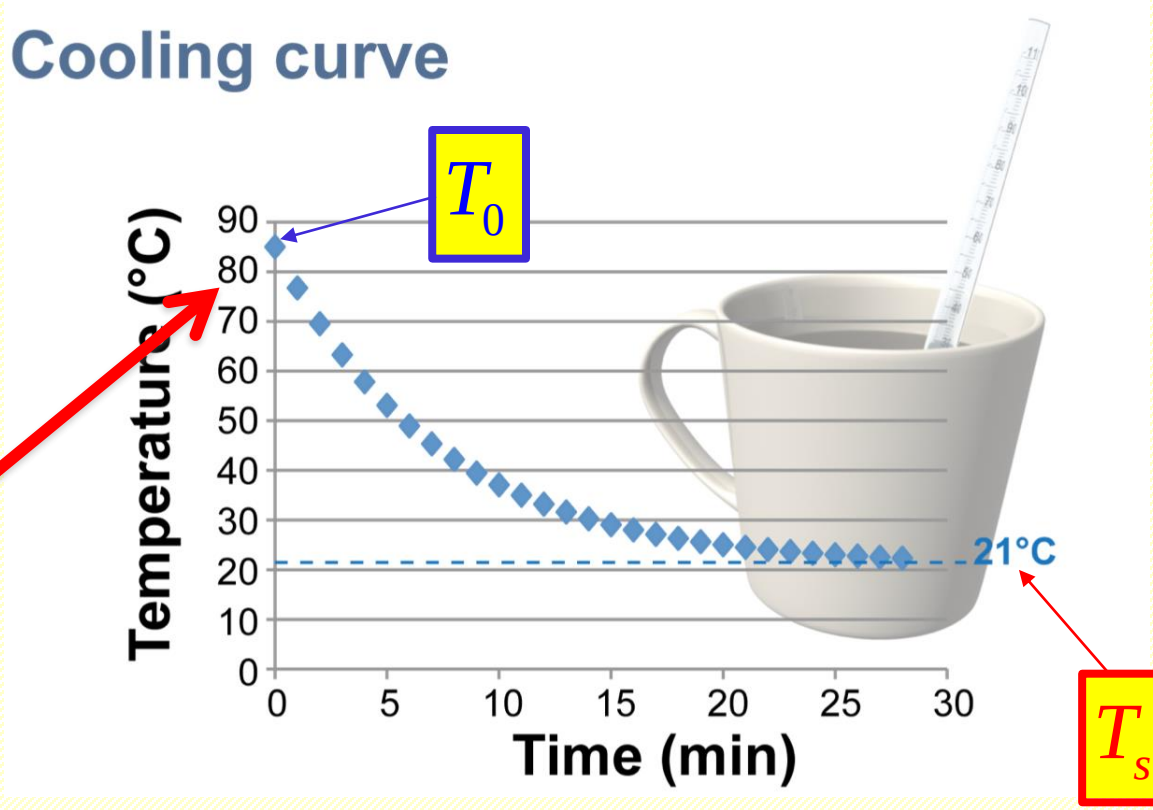
Courtesy: https://education.pasco.com/ebooks/static/FloridaPhysicsReview/BookInd-638.html

Newton's Law of Cooling (and Heating): Example for Everyone to Understand

Large temperature differences

A large temperature difference means rapid cooling.

When the coffee is much hotter than the air, its temperature drops **8°C in one minute**.



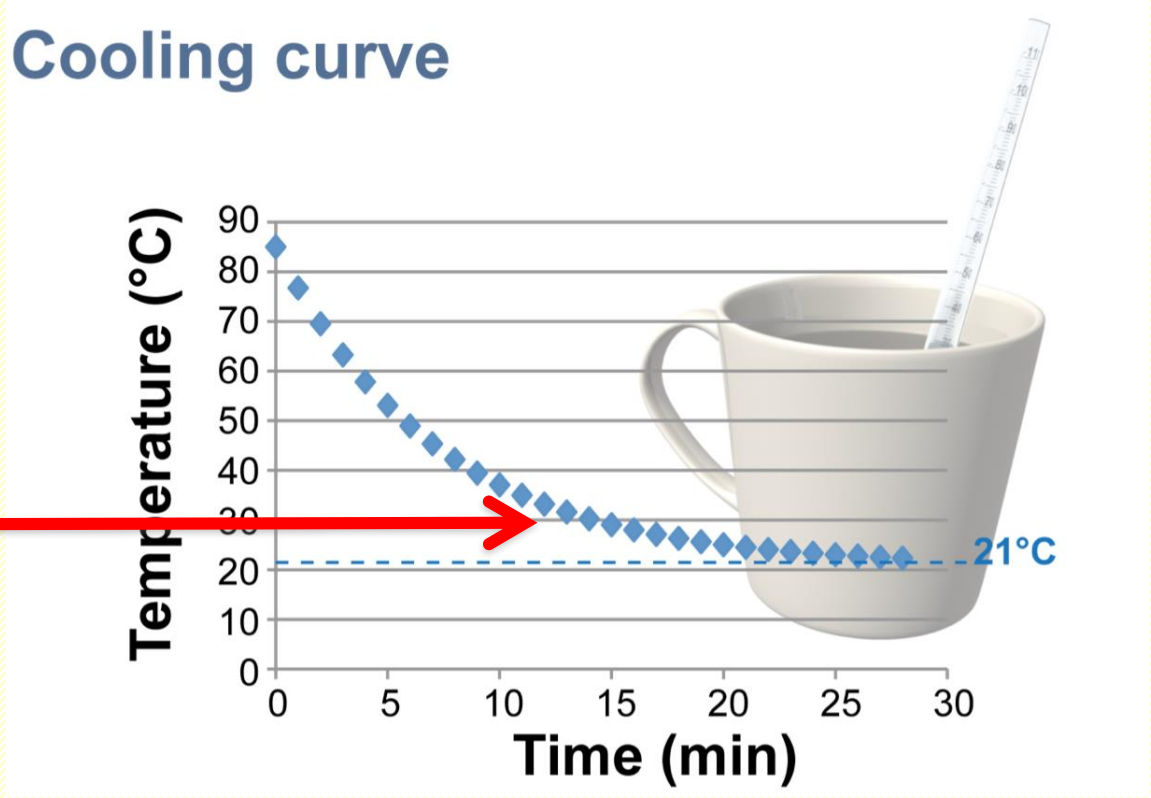
Courtesy: https://education.pasco.com/ebooks/static/FloridaPhysicsReview/BookInd-638.html

Newton's Law of Cooling (and Heating): Example for Everyone to Understand

Small temperature differences

A small temperature difference means slower cooling.

When the coffee has cooled to 30°C, the temperature changes by only **1.2°C per minute**.



Courtesy: https://education.pasco.com/ebooks/static/FloridaPhysicsReview/BookInd-638.html

Newton's Law of Cooling (and Heating):

$$\frac{dT}{dt} = k\,(T - T_s) \qquad \Rightarrow \qquad \frac{dT}{T - T_s} = k\,dt$$

$$\ln\left|T - T_s\right| = kt + \text{Constant}_1$$

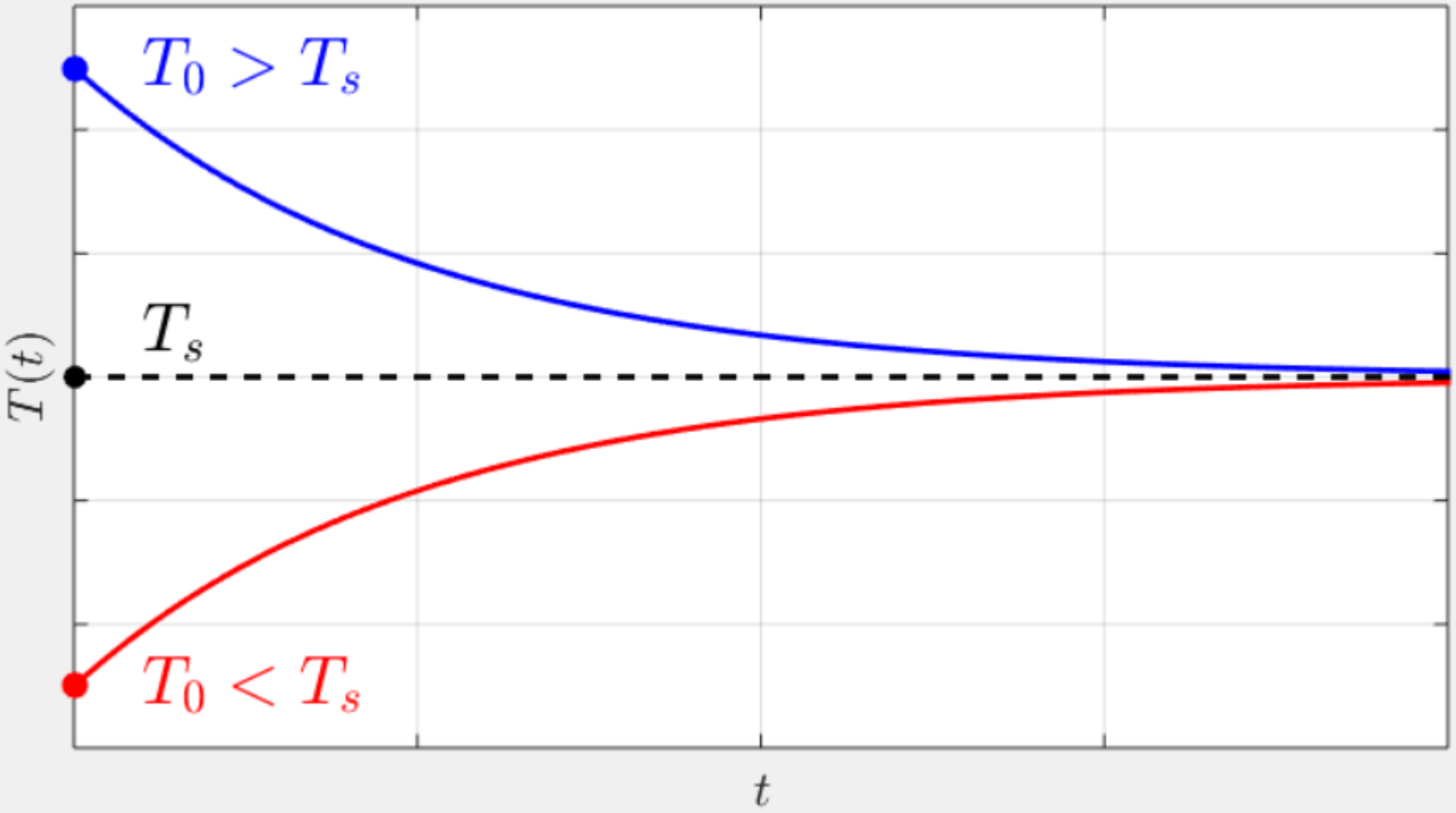
$$T - T_s = (\text{Constant}_2) \times e^{kt} \quad (\text{the } e^{\text{Constant}_1} \text{ and the } \pm \text{ all gets absorbed in the Constant}_2)$$

$$T = T_s + (\text{Constant}_2) \times e^{kt}$$

$$\text{For } t = 0: \quad T_0 = T_s + (\text{Constant}_2) \times e^{0} \quad \Rightarrow \quad (\text{Constant}_2) = T_0 - T_s$$

$$\boxed{T(t) = T_s + (T_0 - T_s) \times e^{kt}}$$

The figure below shows the general shape of $T(t)$ when $T_0 > T_s$ (i.e., in a **cooling** scenario) and when $T_0 < T_s$ (i.e., in a **heating** scenario):



LIMITATIONS:

The **Newton's Law of cooling** is applicable under the following **limitations**:

(1) This Law applies to cooling by convection and radiation and not by radiation alone. To achieve this condition the hot body is cooled in a uniform flow of air so than the radiation losses become small as compared to that due to forced convection.

(2) This Law is applicable in still air only for a temperature difference of about 20K or 30K but it is true for all temperature difference in conditions of forced convection of the air.

[*Reference: Gupta S.K. & Kumar A., Krishna's Engineering Physics. Volume II (Thermal Physics). Krishna Prakash Media, 2001, ISBN 81 87224 20 7. - p.2.56*]

NEWTON's LAW OF COOLING (and HEATING):

EXAMPLE

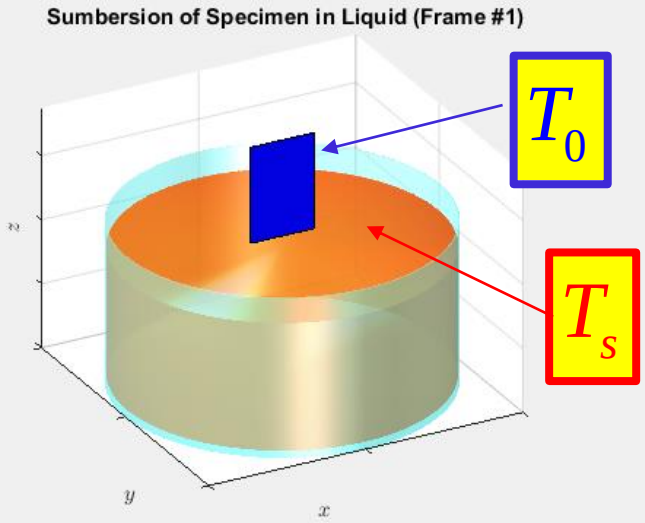
A small metal bar whose temperature is $30^{o}\,C$ is dropped into a container of $75^{o}\,C$ water.

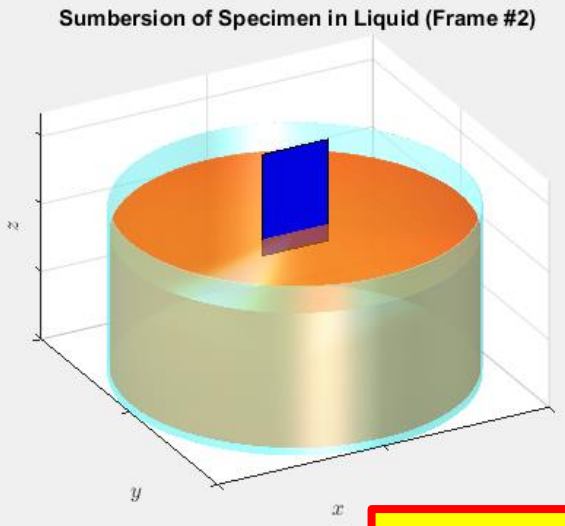
After 1 second the temperature of the bar has increased by $1^{o}\,C$ (i.e. up to $31^{o}\,C$).

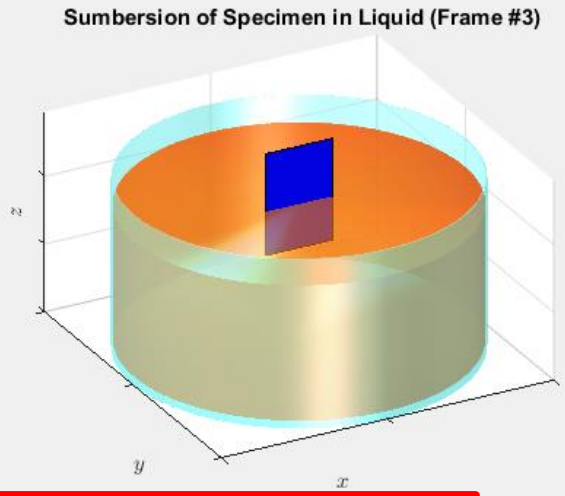
(a) How long will it take for the temperature of the bar to reach $70^{o}\,C$?

(b) 74◦C?

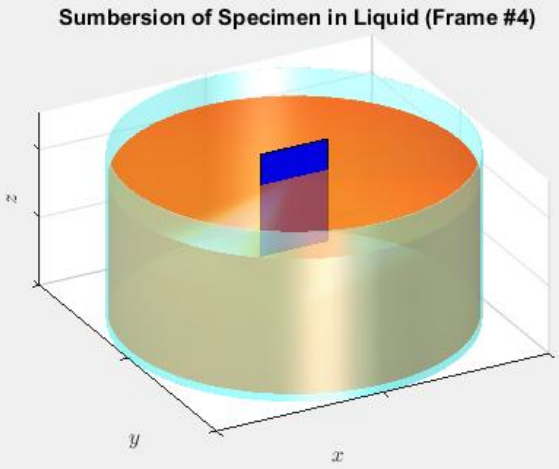
ILLUSTRATION of PROCESS by PMT with MATLAB's GRAPHICS

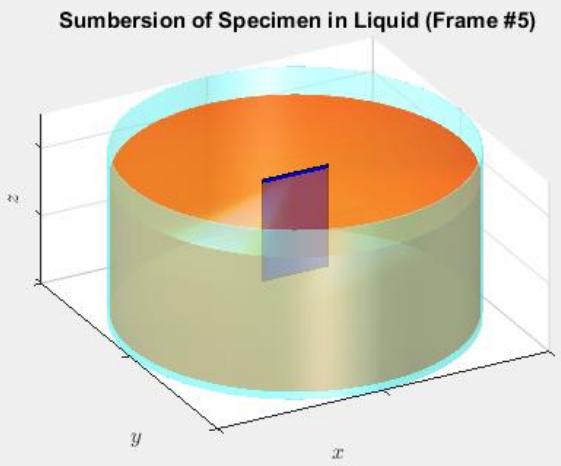


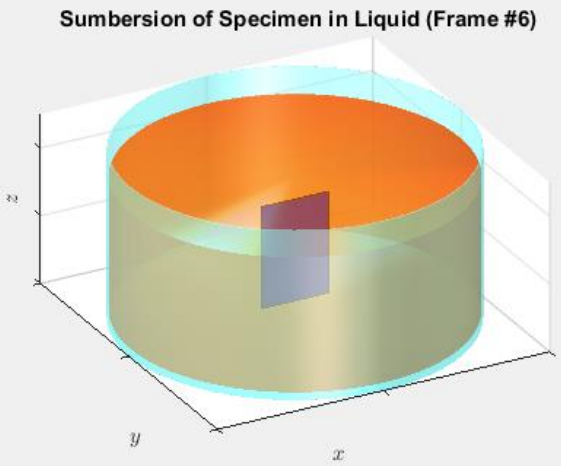




$$T(t) = T_s + (T_0 - T_s) \times e^{kt}$$







SOLUTION: (1) Determine value of *k* first

$$T(t) = T_s + (T_0 - T_s)e^{kt_1}$$

$$31 = 75 + (30 - 75)e^{k\times 1}$$

$$(75 - 31) = (75 - 30)e^{k}$$

$$e^{k} = \left(\frac{75-31}{75-30}\right) = \left(\frac{44}{45}\right)$$

$$k = \ln\left(\frac{44}{45}\right)$$

$$k = -0.0225$$

SOLUTION: (2) Simulating reach of 70°C

$$T(t) = T_s + (T_0 - T_s)\, e^{kt_{70}}$$

$$70 = 75 + (30 - 75)\, e^{kt_{70}}$$

$$(75 - 70) = (75 - 30)\, e^{kt_{70}}$$

$$e^{kt_{70}} = \left(\frac{75-70}{75-30}\right) = \frac{5}{45} = \frac{1}{9} \quad \Rightarrow \quad kt_{70} = \ln\left(\frac{1}{9}\right)$$

$$t_{70} = \frac{1}{k}\ln\left(\frac{1}{9}\right) = 97.7724 \text{ (sec)}$$

SOLUTION: (3) Simulating reach of 74°C

$$T(t) = T_s + (T_0 - T_s)\, e^{kt_{74}}$$

$$74 = 75 + (30 - 75)\, e^{kt_{74}}$$

$$(75 - 74) = (75 - 30)\, e^{kt_{74}}$$

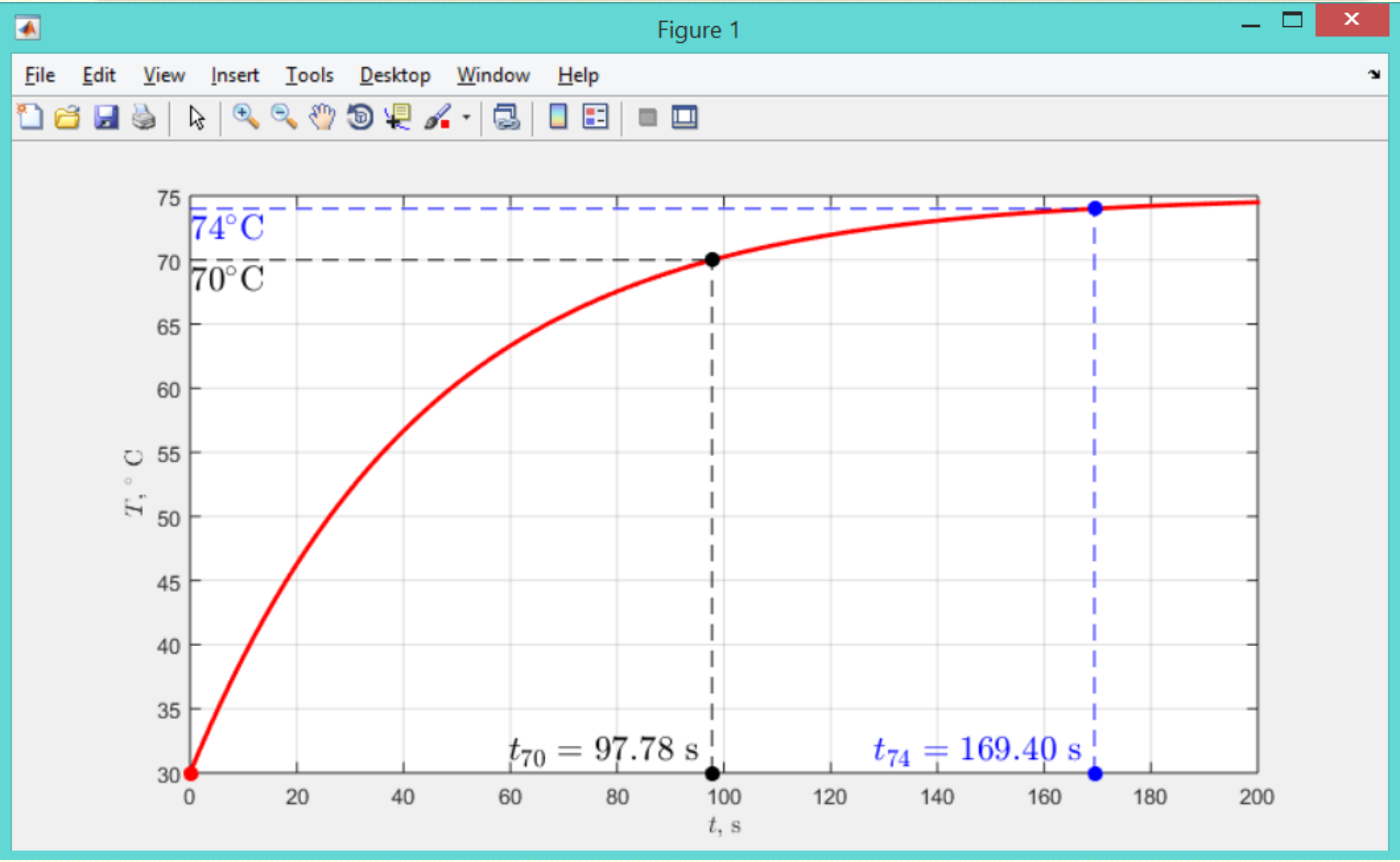
$$e^{kt_{74}} = \left(\frac{75-74}{75-30}\right) = \frac{1}{45} \quad \Rightarrow \quad kt_{74} = \ln\left(\frac{1}{45}\right)$$

$$t_{74} = \frac{1}{k}\ln\left(\frac{1}{45}\right) = 169.3894 \text{ (sec)}$$

Advanced MATLAB script for simulating process for the time range

```
%%
% Designed by Prof P.M.Trivailo (C)2020
tt=[0:0.01:1]*200;
k=log(44/45); T0=30; Ts=75;
syms T(t)
ode = diff(T,t) == k*(T-Ts)       % dT/dt= k(T-Ts)
cond = T(0) == T0; ySol(t) = dsolve(ode,cond)
TT=eval(subs(ySol,{t},{tt}));
figure; plot(tt,TT,'LineWidth',2,'Color','r'); hold on;
line('XData',tt(1),'YData',TT(1),'Marker','.','MarkerSize',24,'Color','r');
t70=interp1(TT,tt,70);
line('XData',t70,'YData',70,'Marker','.','MarkerSize',24,'Color','k');
line('XData',t70,'YData',T0,'Marker','.','MarkerSize',24,'Color','k');
line('XData',[1 1]*t70,'YData',[T0 70],'LineStyle','--','Color','k');
line('XData',[0 1]*t70,'YData',[1 1]*70,'LineStyle','--','Color','k');
text('String',sprintf('$t_{70}=%6.2f$ s ',t70),'Interpreter','LaTeX','Color','k','FontSize',16,...
    'Position',[t70 T0],'VerticalAlignment','bottom','HorizontalAlignment','right');
text('String','$70^\circ$C','Interpreter','LaTeX','Color','k','FontSize',16,...
    'Position',[0 70],'VerticalAlignment','top');
t74=interp1(TT,tt,74);
line('XData',t74,'YData',74,'Marker','.','MarkerSize',24,'Color','b');
line('XData',t74,'YData',T0,'Marker','.','MarkerSize',24,'Color','b');
line('XData',[1 1]*t74,'YData',[T0 74],'LineStyle','--','Color','b');
line('XData',[0 1]*t74,'YData',[1 1]*74,'LineStyle','--','Color','b');
text('String',sprintf('$t_{74}=%6.2f$ s ',t74),'Interpreter','LaTeX','Color','b','FontSize',16,...
    'Position',[t74 T0],'VerticalAlignment','bottom','HorizontalAlignment','right');
text('String','$74^\circ$C','Interpreter','LaTeX','Color','b','FontSize',16,...
    'Position',[0 74],'VerticalAlignment','top');
grid on; xlabel('$t$, s','Interpreter','LaTeX'); ylabel('$T$, $^\circ$ C','Interpreter','LaTeX');
set(gcf,'Position',[214   334   834   428]);
```

RESULTS: Simulation, using MATLAB



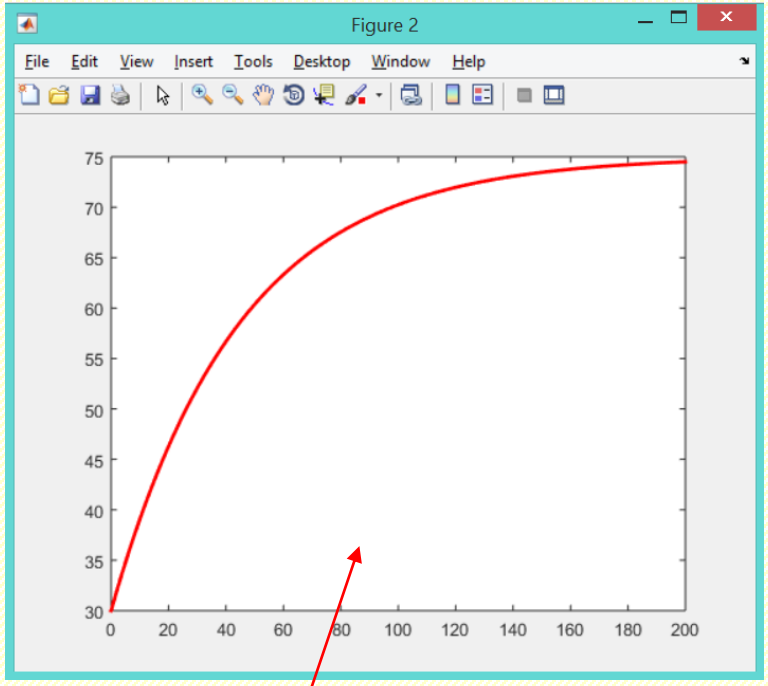
Comments of MATLAB script

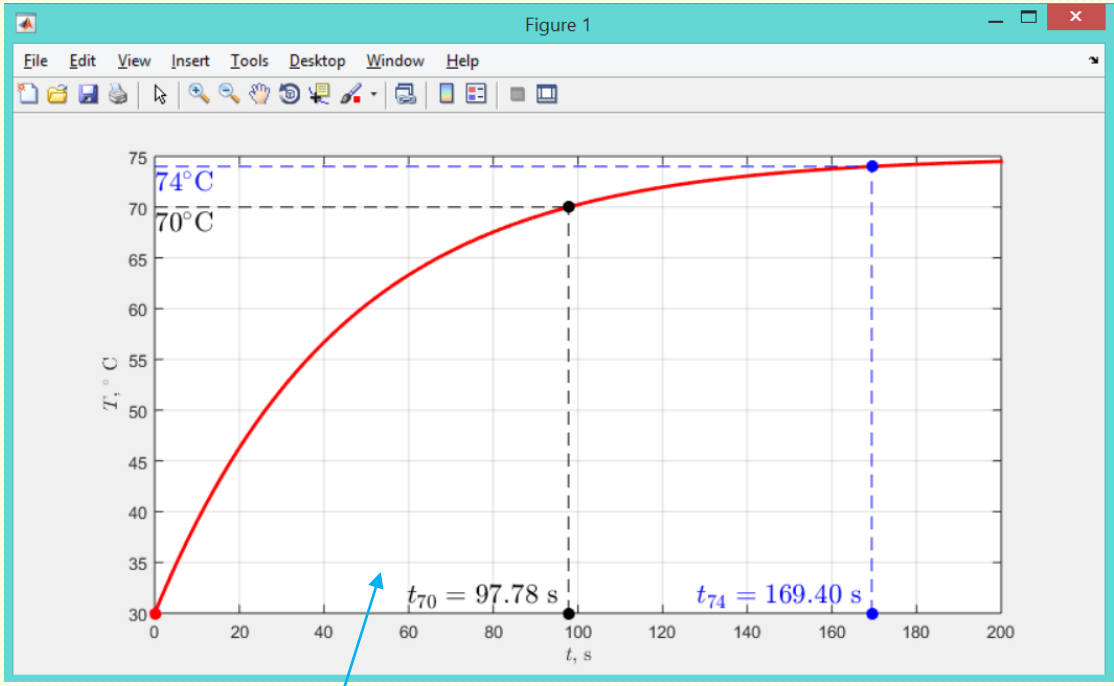
```
%%
% Designed by Prof P.M.Trivailo (C)2020
tt=[0:0.01:1]*200;
k=log(44/45); T0=30; Ts=75;
syms T(t)
ode = diff(T,t) == k*(T-Ts)       % dT/dt= k(T-Ts)
cond = T(0) == T0; ySol(t) = dsolve(ode,cond)
TT=eval(subs(ySol,{t},{tt}));
figure; plot(tt,TT,'LineWidth',2,'Color','r'); hold on;
line('XData',tt(1),'YData',TT(1),'Marker','.','MarkerSize',24,'Color','r');
t70=interp1(TT,tt,70);
line('XData',t70,'YData',70,'Marker','.','MarkerSize',24,'Color','k');
line('XData',t70,'YData',T0,'Marker','.','MarkerSize',24,'Color','k');
line('XData',[1 1]*t70,'YData',[T0 70],'LineStyle','--','Color','k');
line('XData',[0 1]*t70,'YData',[1 1]*70,'LineStyle','--','Color','k');
text('String',sprintf('$t_{70}=%6.2f$ s ',t70),'Interpreter','LaTeX','Color','k','FontSize',16,...
    'Position',[t70 T0],'VerticalAlignment','bottom','HorizontalAlignment','right');
text('String','$70^\circ$C','Interpreter','LaTeX','Color','k','FontSize',16,...
    'Position',[0 70],'VerticalAlignment','top');
t74=interp1(TT,tt,74);
line('XData',t74,'YData',74,'Marker','.','MarkerSize',24,'Color','b');
line('XData',t74,'YData',T0,'Marker','.','MarkerSize',24,'Color','b');
line('XData',[1 1]*t74,'YData',[T0 74],'LineStyle','--','Color','b');
line('XData',[0 1]*t74,'YData',[1 1]*74,'LineStyle','--','Color','b');
text('String',sprintf('$t_{74}=%6.2f$ s ',t74),'Interpreter','LaTeX','Color','b','FontSize',16,...
    'Position',[t74 T0],'VerticalAlignment','bottom','HorizontalAlignment','right');
text('String','$74^\circ$C','Interpreter','LaTeX','Color','b','FontSize',16,...
    'Position',[0 74],'VerticalAlignment','top');
grid on; xlabel('$t$, s','Interpreter','LaTeX'); ylabel('$T$, $^\circ$ C','Interpreter','LaTeX');
set(gcf,'Position',[214   334   834   428]);
```

"CORE" COMMANDS

"ANNOTATIONS"

COMPARE: “Core” & “Decorated” Plots





Plot, created with “CORE” commands only

Plot, created with “CORE” and “DECORATIONS” commands

!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

NUMERICAL SOLUTION, ILLUSTRATED WITH THE SAME EXAMPLE

SOLUTION: Numerical (!!!) Method

```
%% COOLING & HEATING EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2020
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
% Fundamental Law:  dT/dt = k(T-Ts);
%-------------------------------------%
T0 = 30; Ts=75;                       % initial & surroundings temperature
k = log(44/45);                       %
tmax = 200; t = [0:0.01:1]*tmax;      % regular time array

T_xdot_anonymous3 = @(t, T) k*(T-Ts);  % anonymous FUNCTION IS USED
[tt3,TT3] = ode45(T_xdot_anonymous3,t,T0);% call ODE

plot(tt3,TT3,'LineWidth',6,'Color','r','LineStyle','--');
grid on;
xlabel('$t$ [s]','Interpreter','LaTeX'); ylabel('$T(t)$ [$^\circ$]','Interpreter','LaTeX');
str = sprintf('Temperature: Time History ($$k$$=%4.3f, $$T_0$$=%g$$^\\circ C$$, $$T_s$$=%g$$^\\circ C$$)',k,T0,Ts)
title(str, 'Interpreter','LaTeX');
set(gca,'FontSize',16); set(gcf,'Position',[214   334   834   428]);
Texact = Ts +(T0-Ts)*exp(k*t);
hold on; plot(t,Texact,'LineWidth',2,'Color','b');
legend('Numerical Solution','Exact Analytical Solution','Location','SouthEast');
axis([0 tmax T0 Ts]);
```

SOLUTION: Numerical (!!!) Method

1. Input of the data

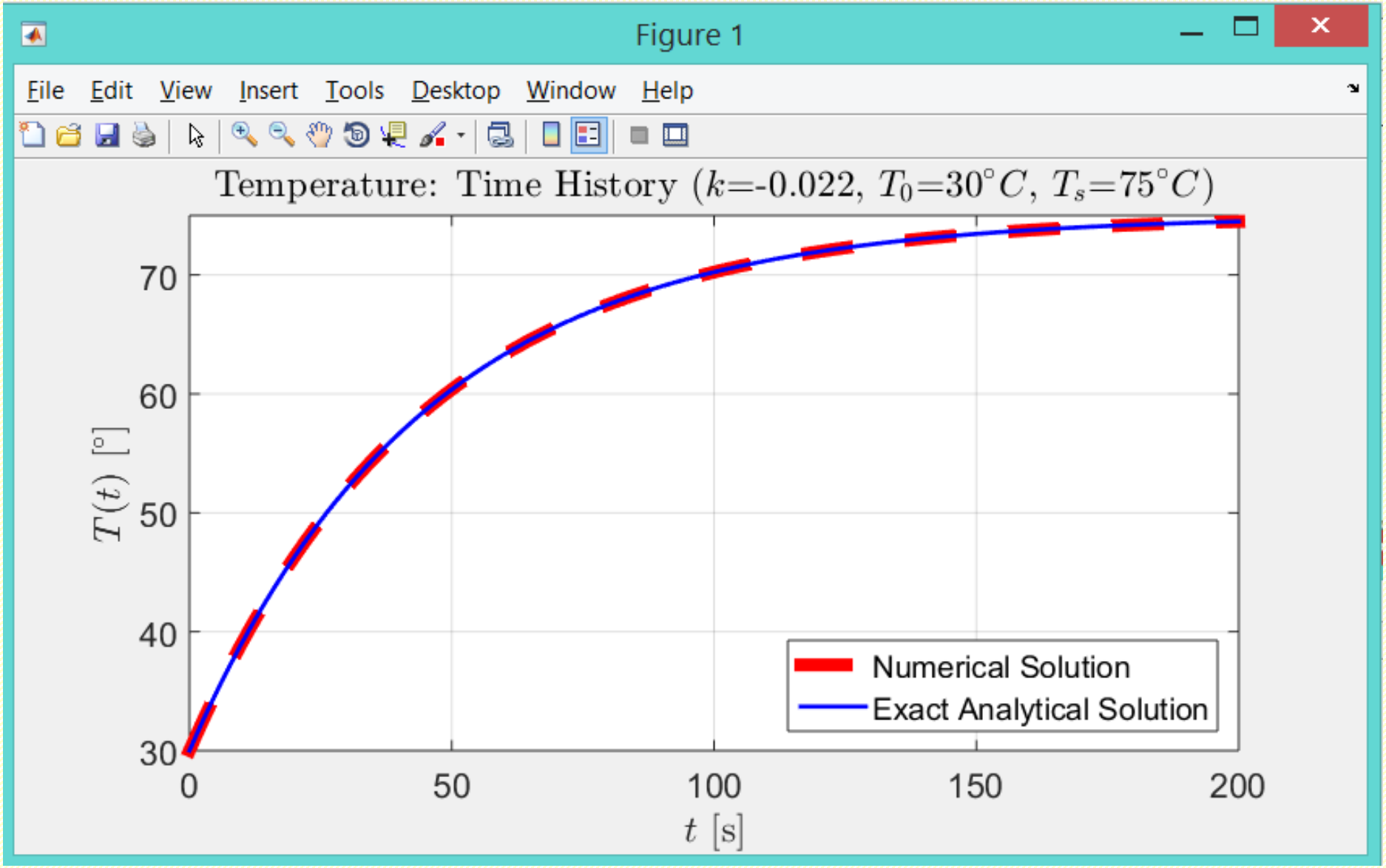
2. $$\frac{dT}{dt} = k(T - T_s)$$

```
%% COOLING & HEATING EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2020
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
% Fundamental Law:  dT/dt = k(T-Ts);
%-------------------------------------%
T0 = 30; Ts=75;                       % initial & surroundings temperature
k = log(44/45);                       %
tmax = 200; t = [0:0.01:1]*tmax;      % regular time array

T_xdot_anonymous3 = @(t, T) k*(T-Ts);  % anonymous FUNCTION IS USED
[tt3,TT3] = ode45(T_xdot_anonymous3,t,T0);% call ODE

plot(tt3,TT3,'LineWidth',6,'Color','r','LineStyle','--');
grid on;
xlabel('$t$ [s]','Interpreter','LaTeX'); ylabel('$T(t)$ [$^\circ$]','Interpreter','LaTeX');
str = sprintf('Temperature: Time History ($$k$$=%4.3f, $$T_0$$=%g$$^\\circ C$$, $$T_s$$=%g$$^\\circ C$$)',k,T0,Ts)
title(str, 'Interpreter','LaTeX');
set(gca,'FontSize',16); set(gcf,'Position',[214   334   834   428]);
Texact = Ts +(T0-Ts)*exp(k*t);
hold on; plot(t,Texact,'LineWidth',2,'Color','b');
legend('Numerical Solution','Exact Analytical Solution','Location
axis([0 tmax T0 Ts]);
```

4. Plotting results

3. Calling ode45 proc.!

Output of the MATLAB script



NEWTON's LAW OF COOLING:

PRACTICE EXAMPLE

You are given a very hot sample of metal, and wish to know its temperature.

You have a thermometer, but it only measures up to 200 ^{o}C and the metal is hotter than that!

You leave the metal in a room kept at 20 ^{o}C.

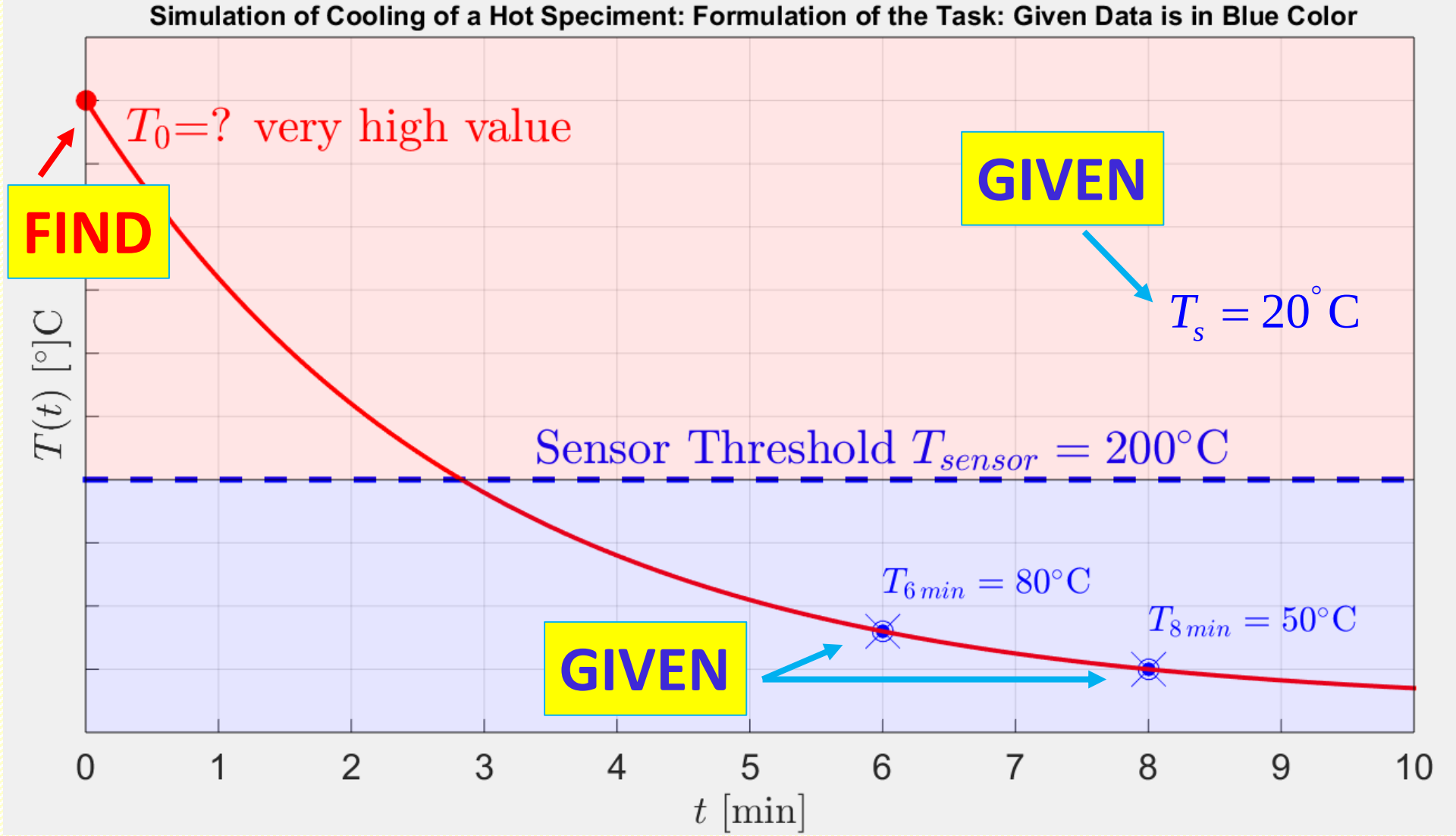
After six minutes it has cooled sufficiently that you can measure its temperature; it is 80 ^{o}C.

After another two minutes it is 50 ^{o}C.

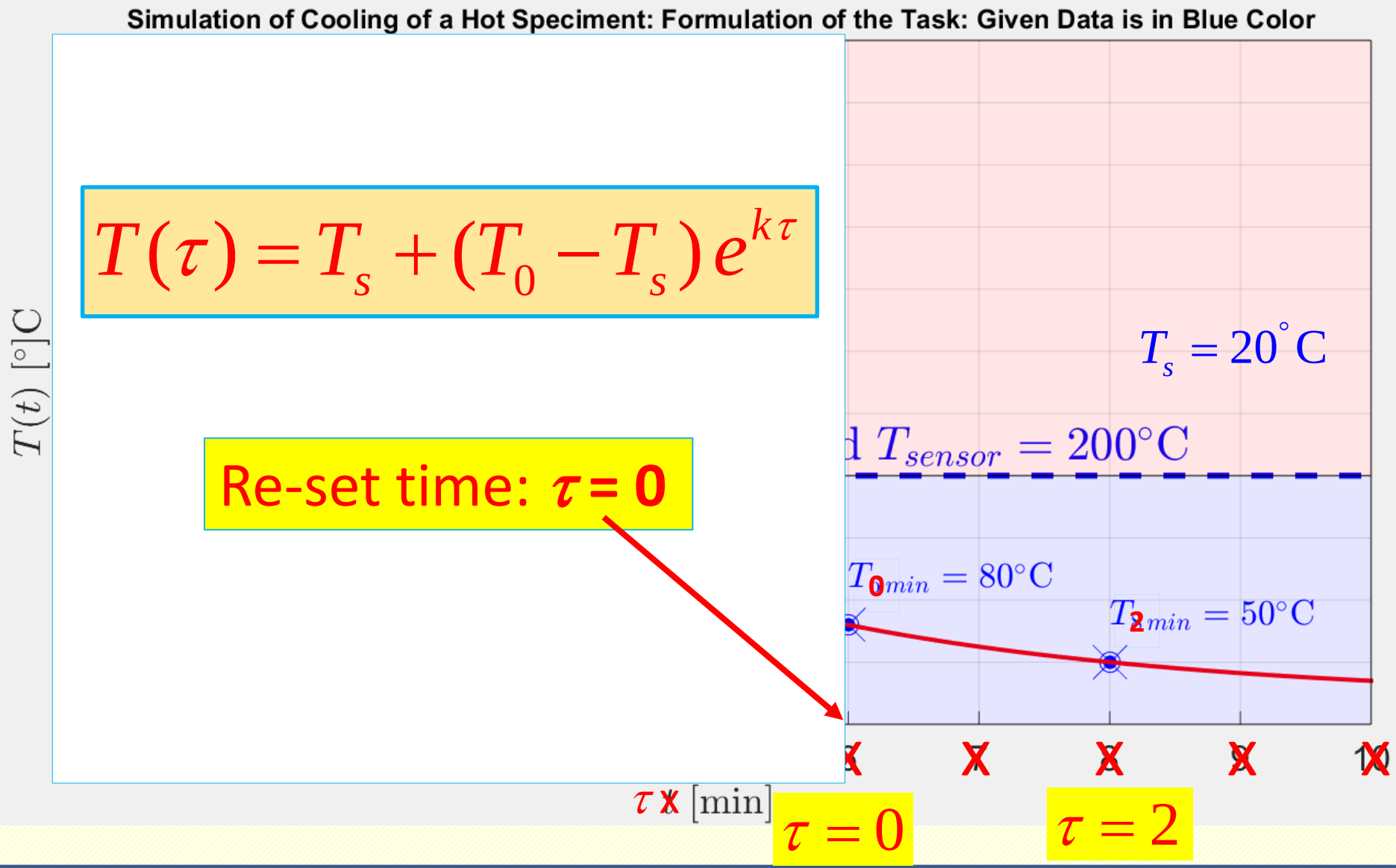
What was the initial temperature of the metal?

TASK FORMULATION: Illustrated with MATLAB

$T(t) = T_s + (T_0 - T_s)e^{kt}$ can not be immediately used, as T_0 is not known !



TASK FORMULATION: Illustrated with MATLAB



SOLUTION: Simulate cooling from 80°C to 50°C (happening during 2 minutes) by re-setting time at 6 min: $\tau = t-6$;

$$T(\tau) = T_s + (T_0 - T_s)\,e^{k\tau} \quad \text{where} \quad \tau = t - 6$$

$$50 = 20 + (80 - 20)\,e^{k\tau_2} \quad \Leftarrow \quad \text{for } \tau = 2$$

$$(50 - 20) = (80 - 20)\,e^{k\tau_2}$$

$$e^{k\tau_2} = \left(\frac{50-20}{80-20}\right) = \frac{30}{60} = \frac{1}{2} \quad \Rightarrow \quad k\tau_2 = \ln\left(\frac{1}{2}\right)$$

$$k = \frac{1}{\tau_2}\ln\left(\frac{1}{2}\right) = \frac{1}{2}\ln\left(\frac{1}{2}\right) = -0.3466 \text{ (1/min)}$$

Newton's Law of Cooling (and Heating):

By the way, note different temperature rate of change at different instants.

For t=6 min

$$\left.\frac{dT}{dt}\right|_{t=6\ \text{min}} = k(T - T_s) = -0.3466(80 - 20) = -20.7944\ (\text{deg/min})$$

Two minutes later t=8 min:

$$\left.\frac{dT}{dt}\right|_{t=8\ \text{min}} = k(T - T_s) = -0.3466(50 - 20) = -10.3972\ (\text{deg/min})$$

These are not our main focus, but interesting observations.

!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

NUMERICAL SOLUTION, using ODE45

MATLAB SUPPLEMENTARY "Try-1" script for simulating cooling from $T_0 = 200$

```
%%
% Designed by Prof P.M.Trivailo (C)2019

t=[0:0.01:1]*9;                          % simulate for 9 minutes
k=log(1/2)/2;                            % determine "k" from task info
Ts=20;

T_xdot_anonymous3 = @(t, T) k*(T-Ts); % anonymous FUNCTION

figure; grid on; hold on;

plot([6 8],[80 50],'rx','MarkerSize',24,'LineWidth',4); % plot given info

T0=200;
[tt3,TT3] = ode45(T_xdot_anonymous3,t,T0);% call ODE
plot(tt3,TT3,'LineWidth',2,'Color','b');

xlabel('t [min]'); ylabel('T(t) [deg C]');
title('Simulation of Cooling of a Hot Specimen for T0=200 deg C')
```

MATLAB SUPPLEMENTARY "Try-1" script for simulating cooling from $T_0 = 200$

```
%%
% Designed by Prof P.M.Trivailo (C)2019

t=[0:0.01:1]*9;
k=log(1/2)/2;
Ts=20;

T_xdot_anonymous3 = @(t, T) k*(T-Ts); % anonymous FUNCTION

figure; grid on; hold on;

plot([6 8],[80 50],'rx','MarkerSi

T0=200;
[tt3,TT3] = ode45(T_xdot_anonymous3,t,T0); % call ODE
plot(tt3,TT3,'LineWidth',2,'Color','b');

xlabel('t [min]'); ylabel('T(t) [deg C]');
title('Simulation of Cooling of a Hot Specimen for T0=200 deg C')
```

1. Input of the data

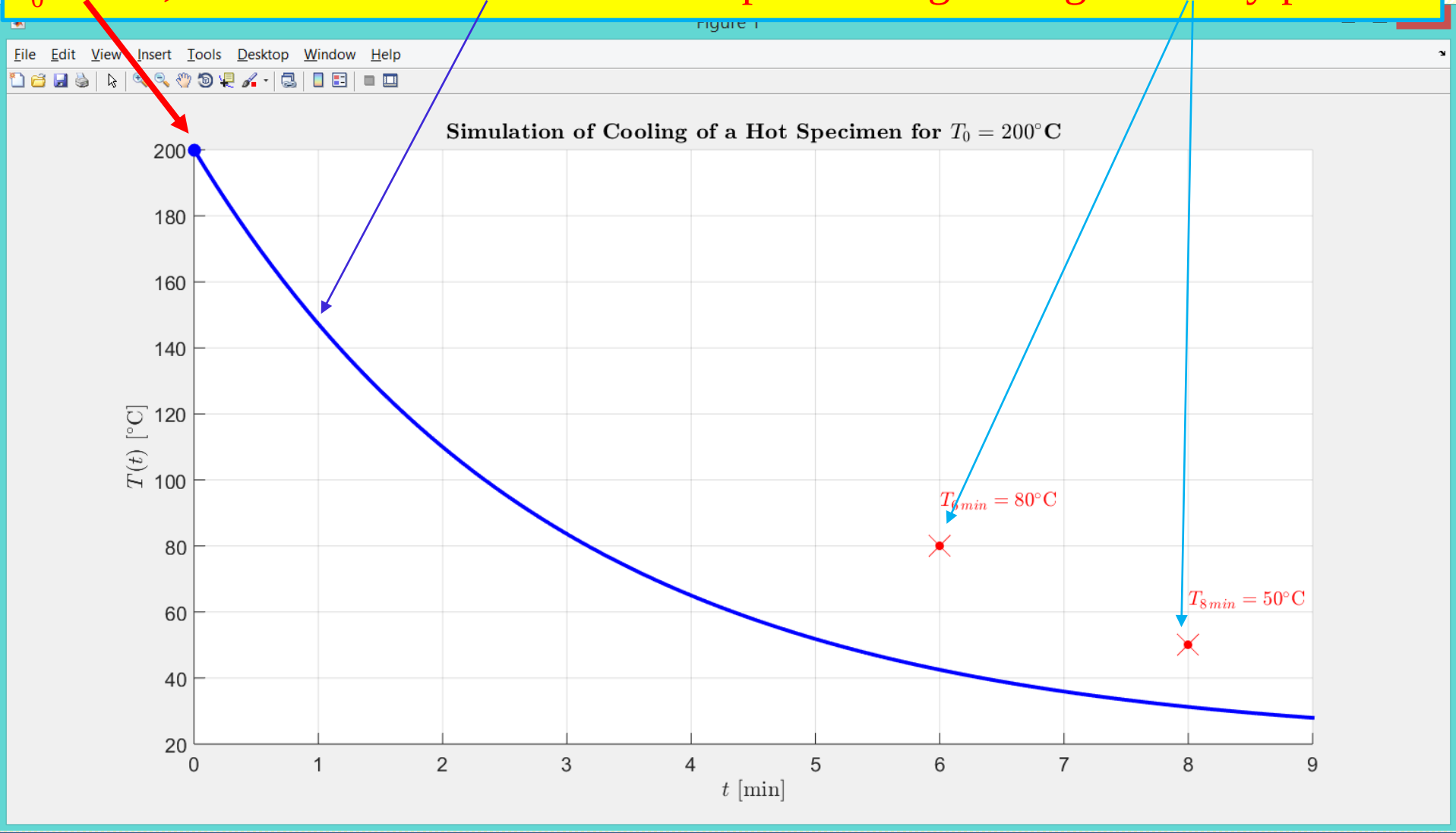
2. $$\frac{dT}{dt} = k(T - T_s)$$

3. Calling ode45 proc.!

4. Plotting results

"Try-1" Result: Simulation, using MATLAB

$T_0 \neq 200$, as simulation curve does not pass through two given "way-points" !

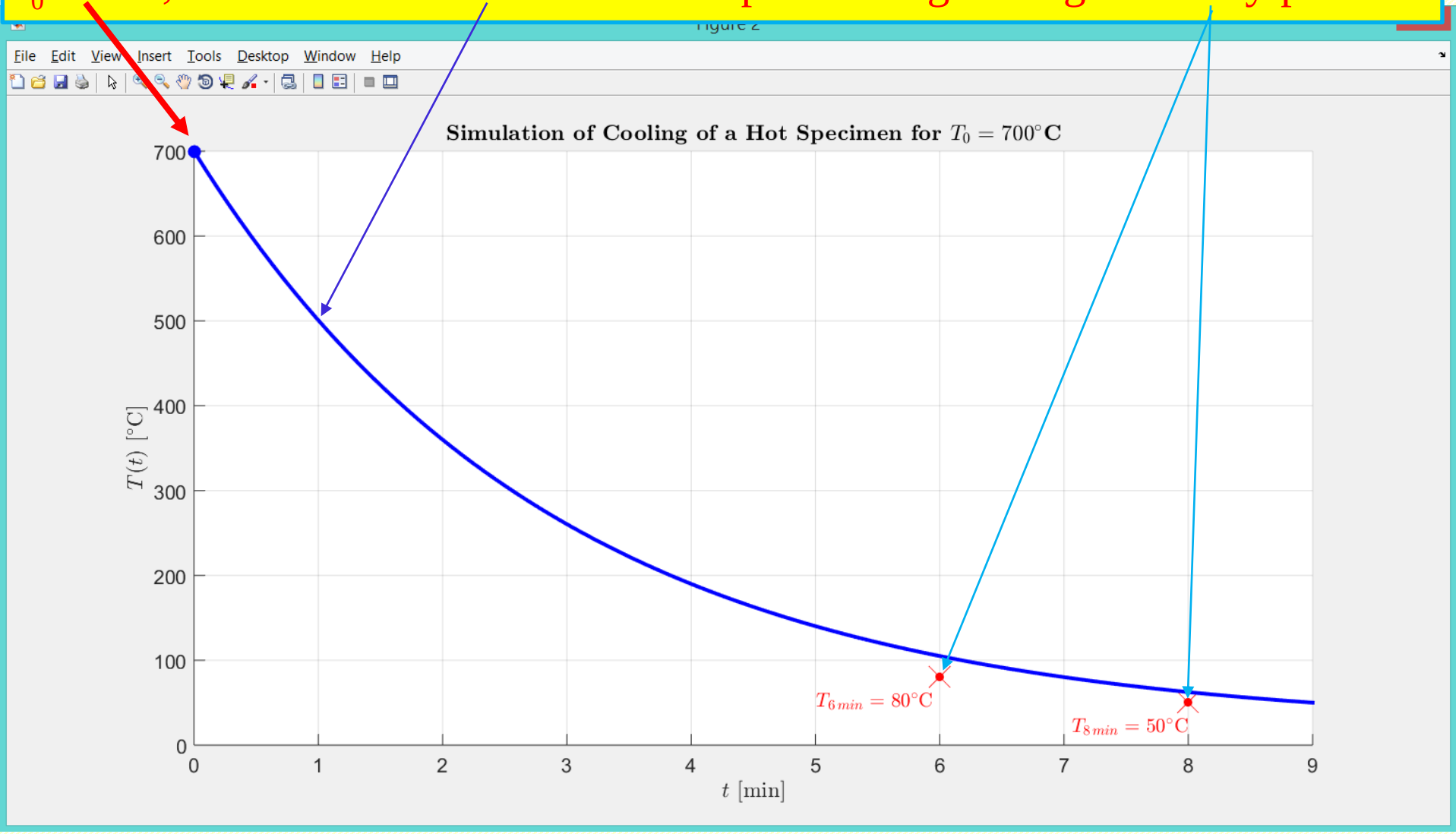


MATLAB SUPPLEMENTARY "Try-2" script for simulating cooling from $T_0 = 700$

```
%%
% Designed by Prof P.M.Trivailo (C)2019

t=[0:0.01:1]*9;                          % simulate for 9 minutes
k=log(1/2)/2;                            % determine "k" from task info
Ts=20;

T_xdot_anonymous3 = @(t, T) k*(T-Ts); % anonymous FUNCTION

figure; grid on; hold on;

plot([6 8],[80 50],'rx','MarkerSize',24,'LineWidth',4); % plot given info

T0=700;
[tt3,TT3] = ode45(T_xdot_anonymous3,t,T0);% call ODE
plot(tt3,TT3,'LineWidth',2,'Color','b');

xlabel('t [min]'); ylabel('T(t) [deg C]');
title('Simulation of Cooling of a Hot Specimen for T0=700 deg C')
```

"Try-2" Result: Simulation, using MATLAB

$T_0 \neq 700$, as simulation curve does not pass through two given "way-points" !

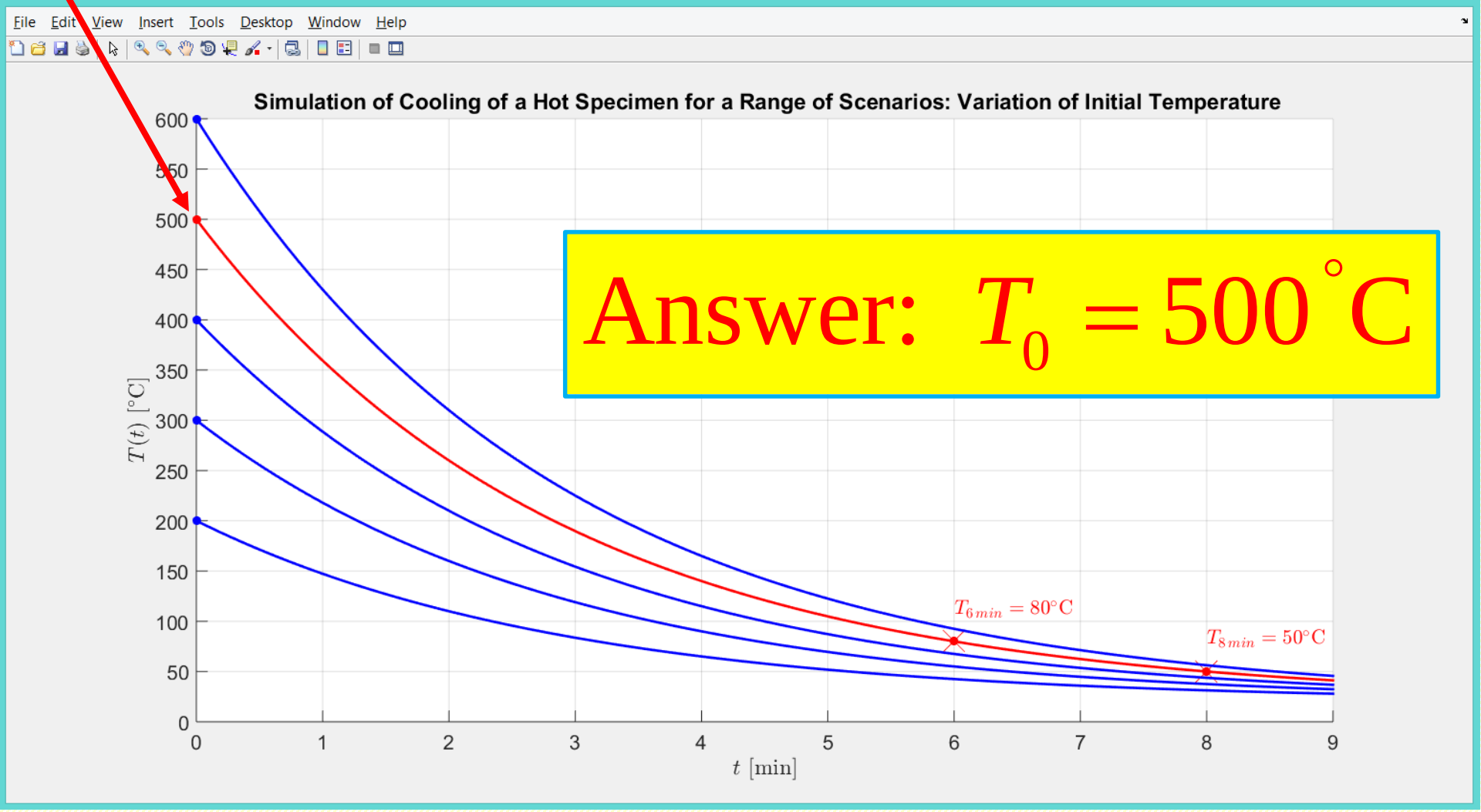


MATLAB MAIN script for simulating cooling for the T_0 range

```
%%
% Designed by Prof P.M.Trivailo (C)2019

t=[0:0.01:1]*9;                          % simulate for 9 minutes
k=log(1/2)/2;                            % determine "k" from task info

T_xdot_anonymous3 = @(t, T) k*(T-Ts); % anonymous FUNCTION

figure; grid on; hold on;

for T0=[200:100:600],
    [tt3,TT3] = ode45(T_xdot_anonymous3,t,T0);% call ODE
    plot(tt3,TT3,'LineWidth',2,'Color','b','LineStyle','-');
    plot(0,T0,'MarkerSize',24,'Color','b','Marker','.');
end

xlabel('t [min]'); ylabel('T(t) [deg C]','Interpreter','LaTeX');
```

RESULTS: Simulation, using MATLAB

Answer: $T_0 = 500$, as simulation curve passes through two given "way-points" !



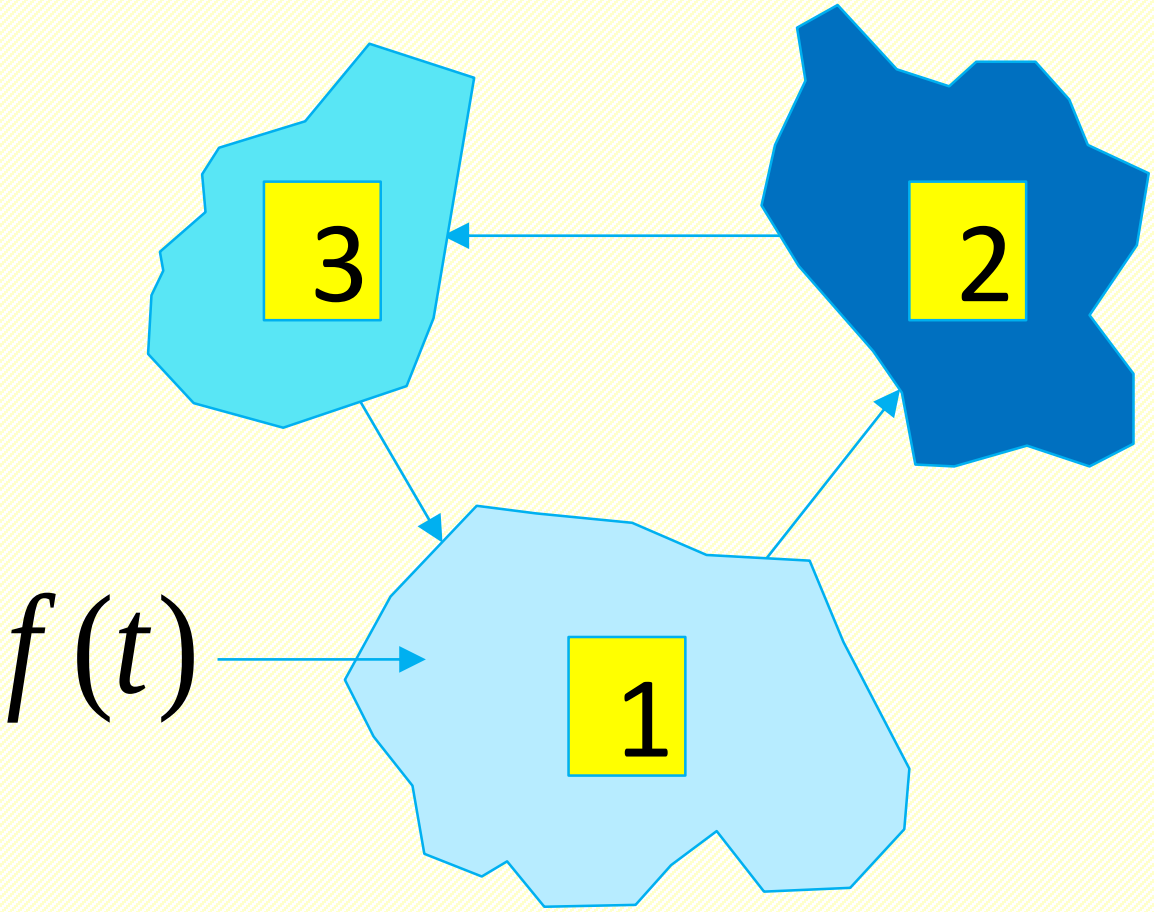
SYSTEM OF

FIRST ORDER

DIFFERENTIAL EQUATIONS:

POND POLLUTION EXAMPLE

Consider three ponds connected by streams.
The first pond has a pollution source, which spreads via the connecting streams to the other ponds.
The plan is to determine the amount of pollutant in each pond.



Three ponds 1, 2, 3 of volumes V_1, V_2, V_3 connected by streams. The pollution source *f(t)* is in pond 1.

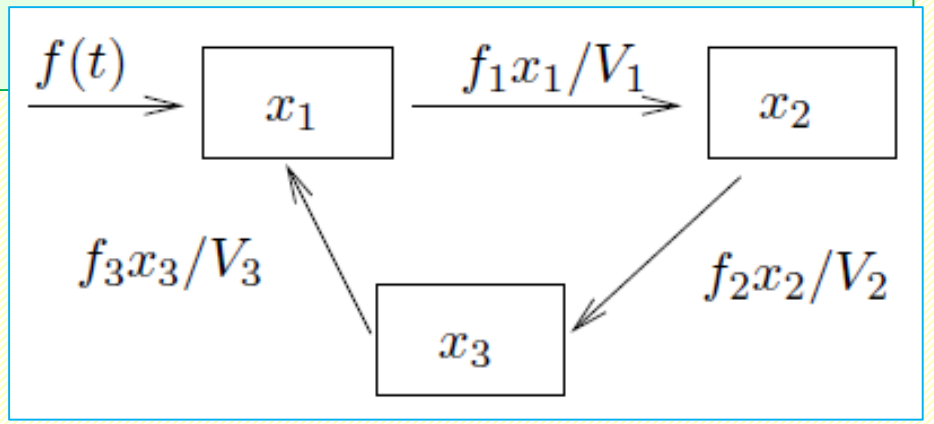
Symbol *f(t)* is the pollutant flow rate into pond 1 (kg/h).

- Symbols f_1, f_2, f_3 denote the pollutant flow rates out of ponds 1, 2, 3, respectively (m^3/h).
- It is assumed that the pollutant is well-mixed in each pond.
- The three ponds have volumes V_1, V_2, V_3 (m^3), which remain constant.
- Symbols $x_1(t), x_2(t), x_3(t)$ denote the amount (kg) of pollutant in ponds 1, 2, 3, respectively.

The pollutant flux is the flow rate times the pollutant concentration, e.g., pond 1 is emptied with flux f_1 times $x_1(t)/V_1$.

A compartment analysis is summarized in the following diagram.

The diagram plus compartment analysis gives the following differential equations:



$$\frac{dx_1(t)}{dt} = \frac{f_3}{V_3}x_3(t) - \frac{f_1}{V_1}x_1(t) + f(t)$$

$$\frac{dx_2(t)}{dt} = \frac{f_1}{V_1}x_1(t) - \frac{f_2}{V_2}x_2(t)$$

$$\frac{dx_3(t)}{dt} = \frac{f_2}{V_2}x_2(t) - \frac{f_3}{V_3}x_3(t)$$

For a specific numerical example, take $f_i/V_i = 0.03$, $1 \le i \le 3$, and let $f(t) = 3$ kg/h for the first 48 hours, thereafter $f(t) = 0$.

We expect due to uniform mixing that after a long time there will be 3*48 = 144 kg of pollutant uniformly deposited, which is 48 kg per pond.

Initially,
$x_1(0) = x_2(0) = x_3(0) = 0$,
if the ponds were pristine.

The specialized problem for the first 48 hours is solved numerically, using "ode45" MATLAB procedure.

$$\frac{dx_1(t)}{dt} = 0.03\,x_3(t) - 0.03\,x_1(t) + 3$$

$$\frac{dx_2(t)}{dt} = 0.03\,x_1(t) - 0.03\,x_2(t)$$

$$\frac{dx_3(t)}{dt} = 0.03\,x_2(t) - 0.03\,x_3(t)$$

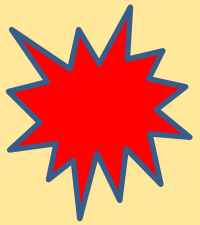
MATLAB:

SYMBOLIC SOLUTION

IT INVOLVES VERY COMPLEX EXPRESSIONS!
HOWEVER, DO NOT PANICK, PLEASE: THIS TECHNIQUE IS RATHER OPTIONAL IN THIS COURSE, AND IS TAKEN AS A CONTRACT TECHNIQUE FOR THE "ODE" COMPULSORY TECHNIQUE, DEMONSTRATED LATER

Advanced MATLAB script for the task (symbolic solution)

```
%% POLLUTED PONDS EXAMPLE: SYMBOLIC SOLUTION
% Designed by Prof P.M.Trivailo (C) 2020
%--------------------------------------
clear; close all; clc;
% Define matrices and the matrix equation.
syms x1(t) x2(t) x3(t)
r=[-1  0  1;...
    1 -1  0;...
    0  1 -1]*0.03;
D=[3; 0; 0];
X = [x1; x2; x3];

odes = diff(X) == r*X + D
% odes(t) =
%
%  diff(x1(t), t) == (3*x3(t))/100 - (3*x1(t))/100 + 3
%      diff(x2(t), t) == (3*x1(t))/100 - (3*x2(t))/100
%      diff(x3(t), t) == (3*x2(t))/100 - (3*x3(t))/100
```



Advanced MATLAB script (symbolic solution) - Continued

```
% Solve the matrix equation using "dsolve". Simplify the solution by
using the simplify function.
[x1Sol(t), x2Sol(t), x3Sol(t)] = dsolve(odes);
x1Sol(t) = simplify(x1Sol(t))
x2Sol(t) = simplify(x2Sol(t))
x3Sol(t) = simplify(x3Sol(t))

% x1Sol(t) =
% C23 + t - (C24*exp(-(9*t)/200)*cos((3*3^(1/2)*t)/200))/2 + (C25*exp(-
(9*t)/200)*sin((3*3^(1/2)*t)/200))/2 + (3^(1/2)*C25*exp(-
(9*t)/200)*cos((3*3^(1/2)*t)/200))/2 + (3^(1/2)*C24*exp(-
(9*t)/200)*sin((3*3^(1/2)*t)/200))/2 + 100/3
%
% x2Sol(t) =
% C23 + t - (C24*exp(-(9*t)/200)*cos((3*3^(1/2)*t)/200))/2 + (C25*exp(-
(9*t)/200)*sin((3*3^(1/2)*t)/200))/2 - (3^(1/2)*C25*exp(-
(9*t)/200)*cos((3*3^(1/2)*t)/200))/2 - (3^(1/2)*C24*exp(-
(9*t)/200)*sin((3*3^(1/2)*t)/200))/2
%
% x3Sol(t) =
% C23 + t + C24*exp(-(9*t)/200)*cos((3*3^(1/2)*t)/200) - C25*exp(-
(9*t)/200)*sin((3*3^(1/2)*t)/200) - 100/3
```

Note: symbolic solutions are given by very long expressions, which can not fit one line. Therefore, when I "cut-and-paste" them from MATLAB, being a one-line each, and inserting in the script as a commented one-line strings, they are taking a few lines on this PPT slide.

Advanced MATLAB script (symbolic solution) - Continued

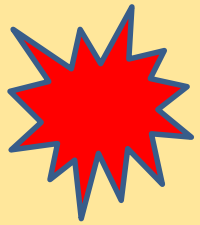
```
% The constants C23, C24 and C25 appear because no conditions are specified.
% Solve the system with the initial conditions x1(0)=0, x2(0)=0 and x3(0)=0.
% When specifying equations in matrix form, you must specify initial conditions in matrix form.
% "dsolve" finds values for the constants that satisfy these conditions.

C = X(0) == [0; 0; 0];
[x1Sol(t), x2Sol(t), x3Sol(t)] = dsolve(odes,C)

% x1Sol(t) =
% t - (100*cos((3*3^(1/2)*t)/200))/(3*exp(t)^(9/200)) +
(100*3^(1/2)*sin((3*3^(1/2)*t)/200))/(9*exp(t)^(9/200)) + 100/3
%
% x2Sol(t) =
% t - (200*3^(1/2)*sin((3*3^(1/2)*t)/200))/(9*exp(t)^(9/200))
%
% x3Sol(t) =
% t + (100*cos((3*3^(1/2)*t)/200))/(3*exp(t)^(9/200)) +
(100*3^(1/2)*sin((3*3^(1/2)*t)/200))/(9*exp(t)^(9/200)) - 100/3
```

Initial conditions are specified here as matrix.

Note: All solutions in symbolic form are found by MATLAB!!! Exciting !!!

Advanced MATLAB script: Commands only, without annotations

```
%% POLLUTED PONDS EXAMPLE: SYMBOLIC SOLUTION
% Designed by Prof P.M.Trivailo (C) 2020
%-------------------------------------
% Define matrices and the matrix equation.
syms x1(t) x2(t) x3(t)
r=[-1  0  1;  1 -1  0;  0  1 -1]*0.03;
D=[3; 0; 0]; X = [x1; x2; x3];
odes = diff(X) == r*X + D

% Solve the matrix equation using "dsolve".
[x1Sol(t), x2Sol(t), x3Sol(t)] = dsolve(odes);

% Solve the system with zero initial conditions.
C = X(0) == [0; 0; 0];
[x1Sol(t), x2Sol(t), x3Sol(t)] = dsolve(odes,C)
```



Symbolic solutions, derived by MATLAB

$$x_1(t) = t - \frac{100\cos\left(\frac{3\sqrt{3}\,t}{200}\right)}{3\left(e^t\right)^{\frac{9}{200}}} + \frac{100\sqrt{3}\sin\left(\frac{3\sqrt{3}\,t}{200}\right)}{9\left(e^t\right)^{\frac{9}{200}}} + \frac{100}{3}$$

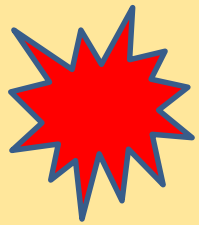
$$x_2(t) = t - \frac{200\sqrt{3}\sin\left(\frac{3\sqrt{3}\,t}{200}\right)}{9\left(e^t\right)^{\frac{9}{200}}}$$

QUITE IMPRESSIVE! IS NOT IT?

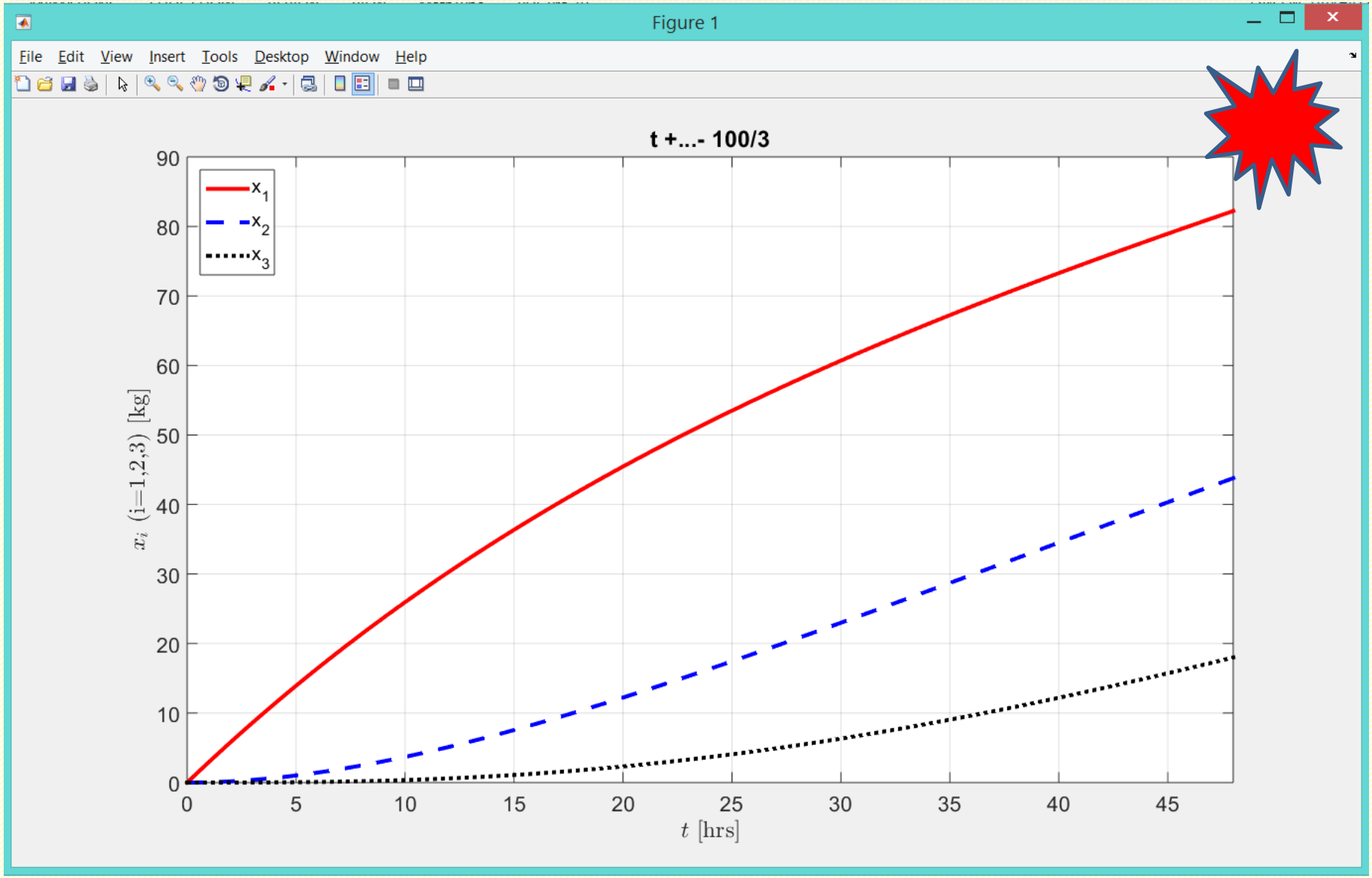
$$x_3(t) = t + \frac{100\cos\left(\frac{3\sqrt{3}\,t}{200}\right)}{3\left(e^t\right)^{\frac{9}{200}}} + \frac{100\sqrt{3}\sin\left(\frac{3\sqrt{3}\,t}{200}\right)}{9\left(e^t\right)^{\frac{9}{200}}} - \frac{100}{3}$$

Advanced MATLAB script (symbolic solution) - Continued

```
% Continuation of the script. It is split into parts for the PPT presentation only.
%-------------------------------------------------------------------
% Visualize the solution using fplot. Before R2016a, use ezplot instead.


%fplot(x1Sol)
h1=ezplot(x1Sol,[0 48 0 90]); set(h1,'LineStyle','-','Color','r');
hold on
%fplot(x2Sol)
h2=ezplot(x2Sol,[0 48 0 90]); set(h2,'LineStyle','--','Color','b');
%fplot(x3Sol)
h3=ezplot(x3Sol,[0 48 0 90]); set(h3,'LineStyle',':','Color','k');
set([h1, h2, h3],'LineWidth',3);
grid on
legend('x_1','x_2','x_3','Location','NorthWest');
xlabel('$t$ [hrs]','Interpreter','LaTeX');
ylabel('$x_i$ (i=1,2,3) [kg]','Interpreter','LaTeX');
set(gca,'FontSize',16);  set(gcf,'Position',[488 -90 1361 852]);
```



Output of the MATLAB script



!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

NUMERICAL SOLUTION, ILLUSTRATED WITH THE SAME EXAMPLE

MATLAB:

NUMERICAL SOLUTION

MATLAB script for numerical solution of the problem

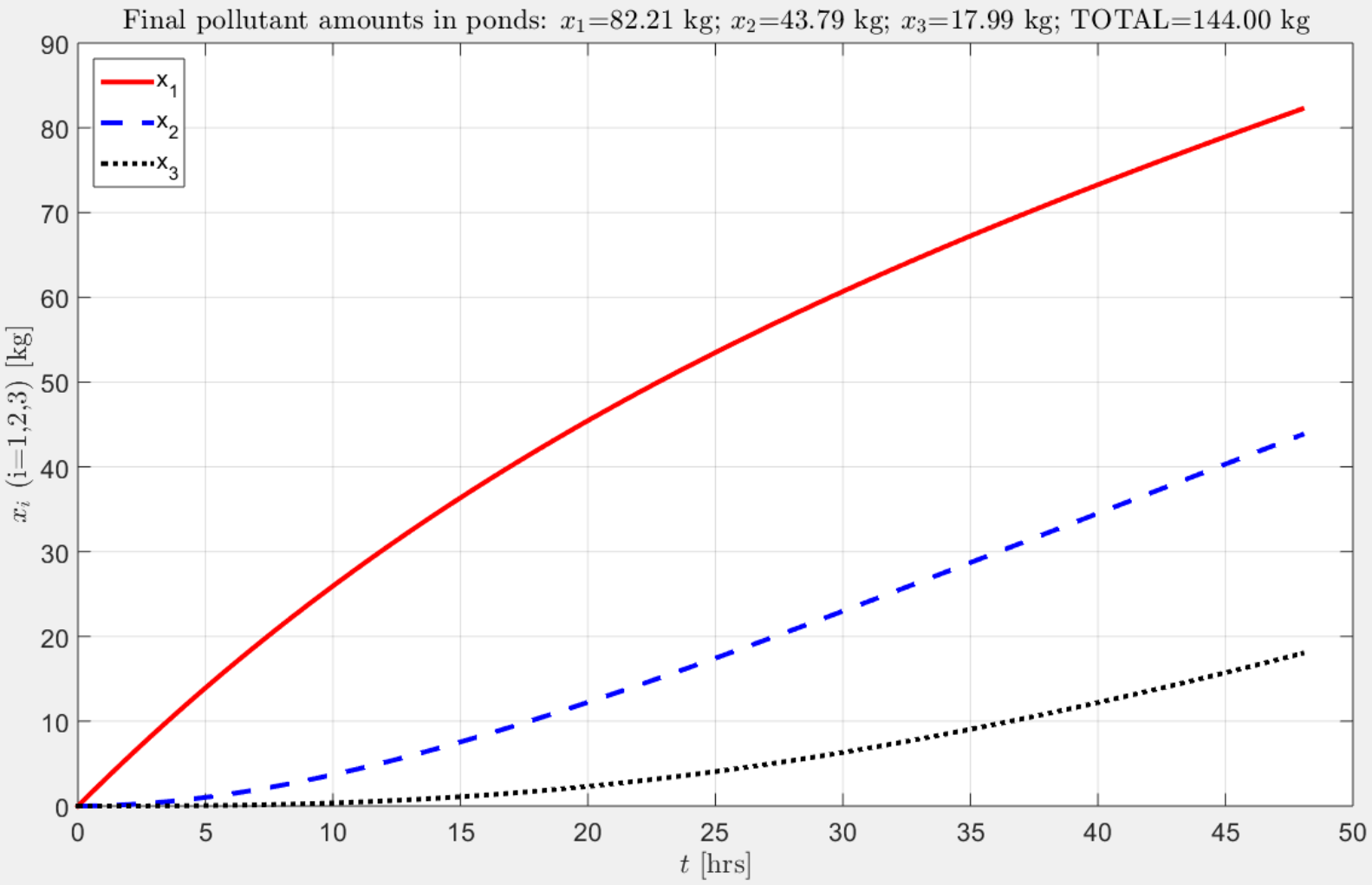
```
%% POLLUTED PONDS EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2020
%--------------------------------------
clear; close all; clc;
r=[-1  0  1;...
    1 -1  0;...
    0  1 -1]*0.03;
D=[3; 0; 0];
tmax=24*2; t=[0:0.01:1]*tmax; % hrs
lake_xdot_anonymous = @(t, x) (r*x+D);
[tt,zz]=ode45(lake_xdot_anonymous,t,[0;0;0]);
plot(tt,zz(:,1),'r-',tt,zz(:,2),'b--',tt,zz(:,3),'k:','LineWidth',3);
legend('x_1','x_2','x_3','Location','NorthWest');
xlabel('$t$ [hrs]','Interpreter','LaTeX');
ylabel('$x_i$ (i=1,2,3) [kg]','Interpreter','LaTeX');
grid on
str1='Final pollutant amounts in ponds: ';
str=sprintf('%s$$x_1$$=%5.2f kg; $$x_2$$=%5.2f kg; $$x_3$$=%5.2f kg; TOTAL=%6.2f kg',str1,zz(end,:),sum(zz(end,:)));
title(str,'Interpreter','LaTeX')
set(gca,'FontSize',16);  set(gcf,'Position',[488 -90 1361 852]);
```

CORE or "HEART" OF THE SCRIPT !!!

"DECORATIONS" commands

Output of the MATLAB script

File Edit View Insert Tools Desktop Window Help



LECTURE MATERIAL:

SECOND ORDER SYSTEMS

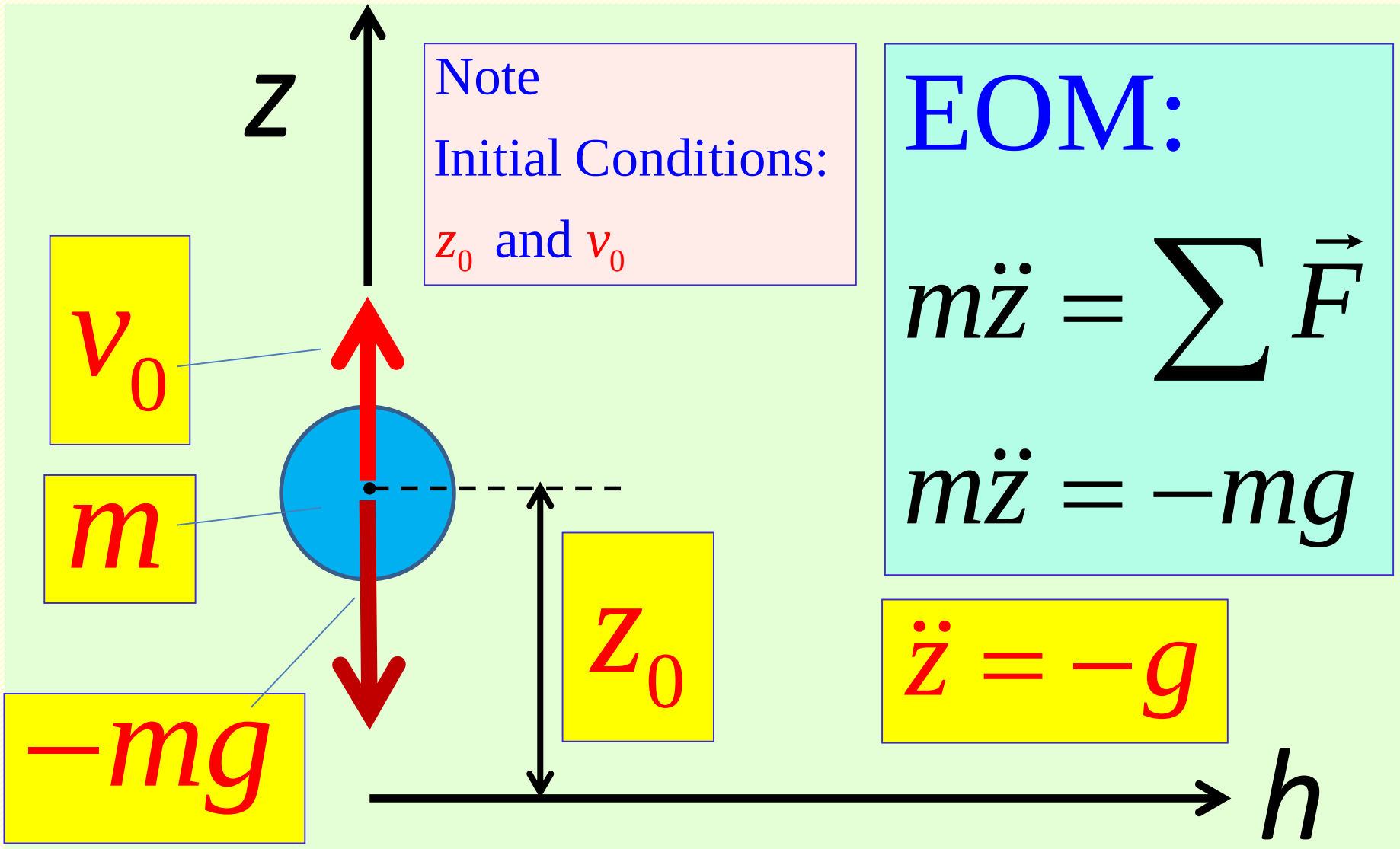
CASE STUDY: FALLING MASS

THEORY

SOLVING RESPONSES:

MATLAB and ODExxx

Modelling a Falling Mass



Note

Initial Conditions:

z_0 and v_0

EOM:

$$m\ddot{z} = \sum \vec{F}$$

$$m\ddot{z} = -mg$$

$$\ddot{z} = -g$$

SOLUTION: Analytical Method

Integration of this differential equation allows analytical solution:

$$\frac{d^2z}{dt^2} = -g; \quad \Rightarrow \quad \frac{dz}{dt} = -gt + C_1; \quad \Rightarrow \quad z = -\frac{gt^2}{2} + C_1 t + C_2$$

Application of Initial Conditions enables us to find the values of constants C_1 and C_2 :

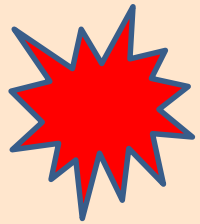
$$z\big|_{t=0} = z_0; \quad \Rightarrow \quad z_0 = C_2$$

$$\left.\frac{dz}{dt}\right|_{t=0} = v_0; \quad \Rightarrow \quad v_0 = C_1$$

Analytical exact solution: $z(t) = z_0 + v_0 t - \frac{gt^2}{2}$

SOLUTION: Analytical Method

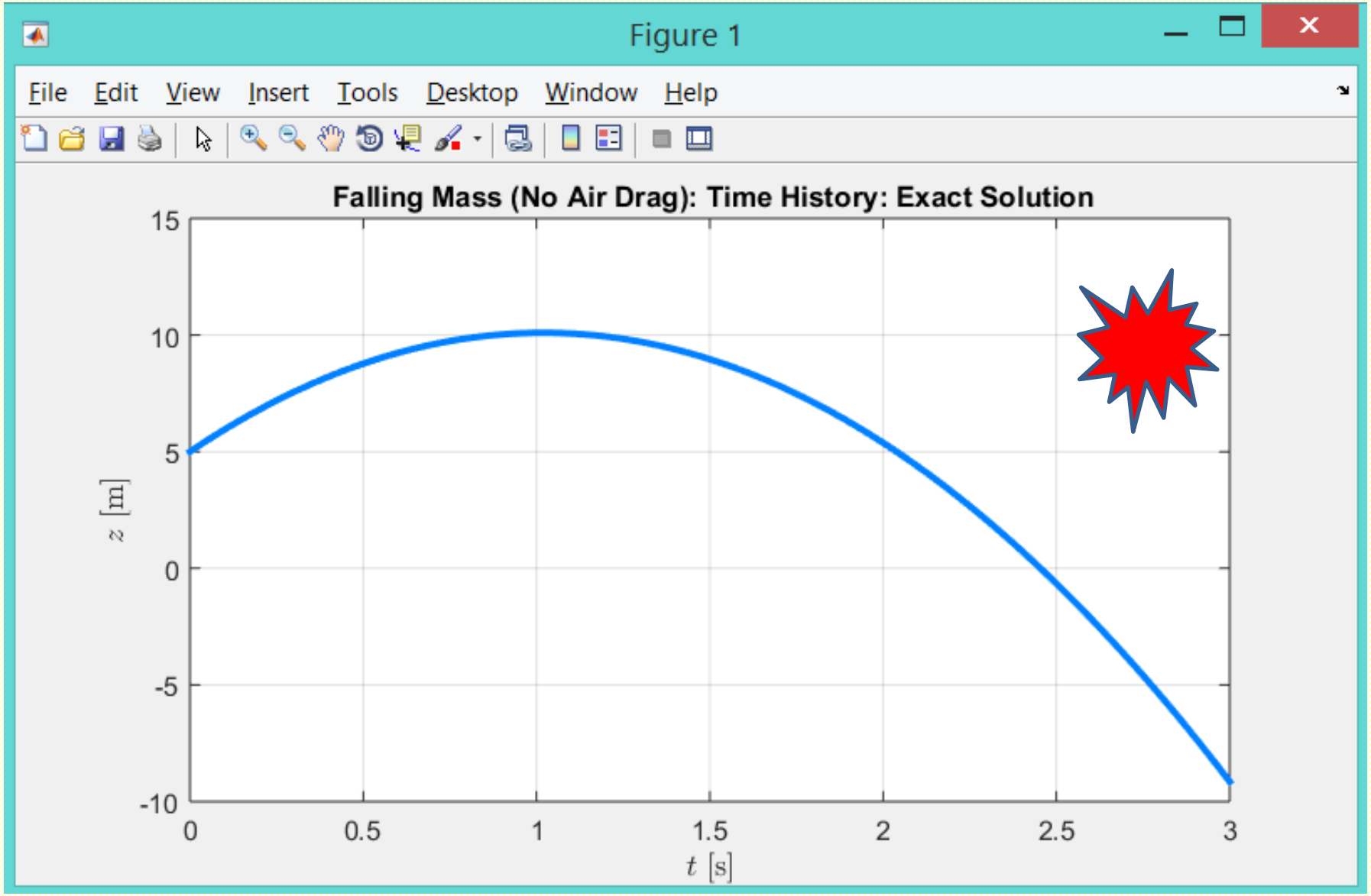
MATLAB SCRIPT (plotting results)



"OENG1116_falling_mass_NO_air_drag_EXACT.m" file

```
%% FALLING MASS (NO DRAG) EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2019
%-------------------------------------
        %                ^ z
% Initial Conditions:    |
v0=10;   % m/s           |
z0=5;    % m             |
tmax=3;  % s             +--                > t
t=[0:0.01:1]*tmax;
zz=z0+v0*t-g*t.^2/2;  % Note: pseudo-square operation !!!
plot(tt,zz,'LineWidth',3,'Color',[0 0.5 1]);
grid on;
xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$z$ [m]','Interpreter','LaTeX');
title('\bf Falling Mass (No Air Drag): Time History: Exact Solution');
set(gca, 'FontSize',12);
```

Output of the MATLAB script



!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

STATE-SPACE FORM
OF THE EQUATION OF MOTION

Modelling a Falling Mass: EOM

$$\ddot{z} = -g$$

Let us introduce new variables:

$$x_1 = z$$

$$x_2 = \dot{z}$$

then

$$\begin{cases} \dot{x}_1 = x_2 \end{cases}$$

Modelling a Falling Mass: EOM

$$\ddot{z} = -g$$

Also, with new variables:

$$x_1 = z$$

$$x_2 = \dot{z}$$

$$\begin{cases} \\ \dot{x}_2 = -g \end{cases}$$

EOM in the STATE-SPACE FORM

$$\ddot{z} = -g$$

Let us introduce new variables:

$$x_1 = z$$

$$x_2 = \dot{z}$$

and

then

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -g \end{cases}$$

!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

NUMERICAL SOLUTION, ILLUSTRATED WITH FALLING MASS EXAMPLE

SOLUTION: Numerical (!!!) Method

MATLAB SCRIPT (no annotations)

```
%% FALLING MASS (NO DRAG) EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2020
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
%----------------------------------------
        %                ^ z
% Initial Conditions:    |
v0=10;   % m/s           |
z0=5;    % m             |
tmax=3;  % s             +--------------> h
t=[0:0.01:1]*tmax;
f_mass_xdot_anonymous = @(t, x) ([x(2); -9.81]);
[tt,zz]=ode45(f_mass_xdot_anonymous,t,[z0; v0]);
plot(tt,zz(:,1),'LineWidth',3,'Color',[0 0.5 1]);
grid on; xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$z$ [m]','Interpreter','LaTeX');
title('\bf Falling Mass (No Air Drag): Time History');
set(gca, 'FontSize',12);
```

"OENG1116_falling_mass_NO_air_drag_ANONYMOUS.m" file

SOLUTION: Numerical (!!!) Method

"OENG1116_falling_mass_NO_air_drag_ANONYMOUS.m" file

```
%% FALLING MASS (NO DRAG) EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2020
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
%-----------------------------------
        %                  ^ z
% Initial Conditions:     |
v0=10;   % m/s            |
z0=5;    % m              |
tmax=3;  % s              +--------------> h
t=[0:0.01:1]*tmax;
f_mass_xdot_anonymous = @(t, x) ([x(2); -9.81]);
[tt,zz]=ode45(f_mass_xdot_anonymous,t,[z0; v0]);
plot(tt,zz(:,1),'LineWidth',3,'Color',[0 0.5 1]);
grid on; xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$z$ [m]','Interpreter','LaTeX');
title('\bf Falling Mass (No Air Drag): Time
set(gca, 'FontSize',12);
```

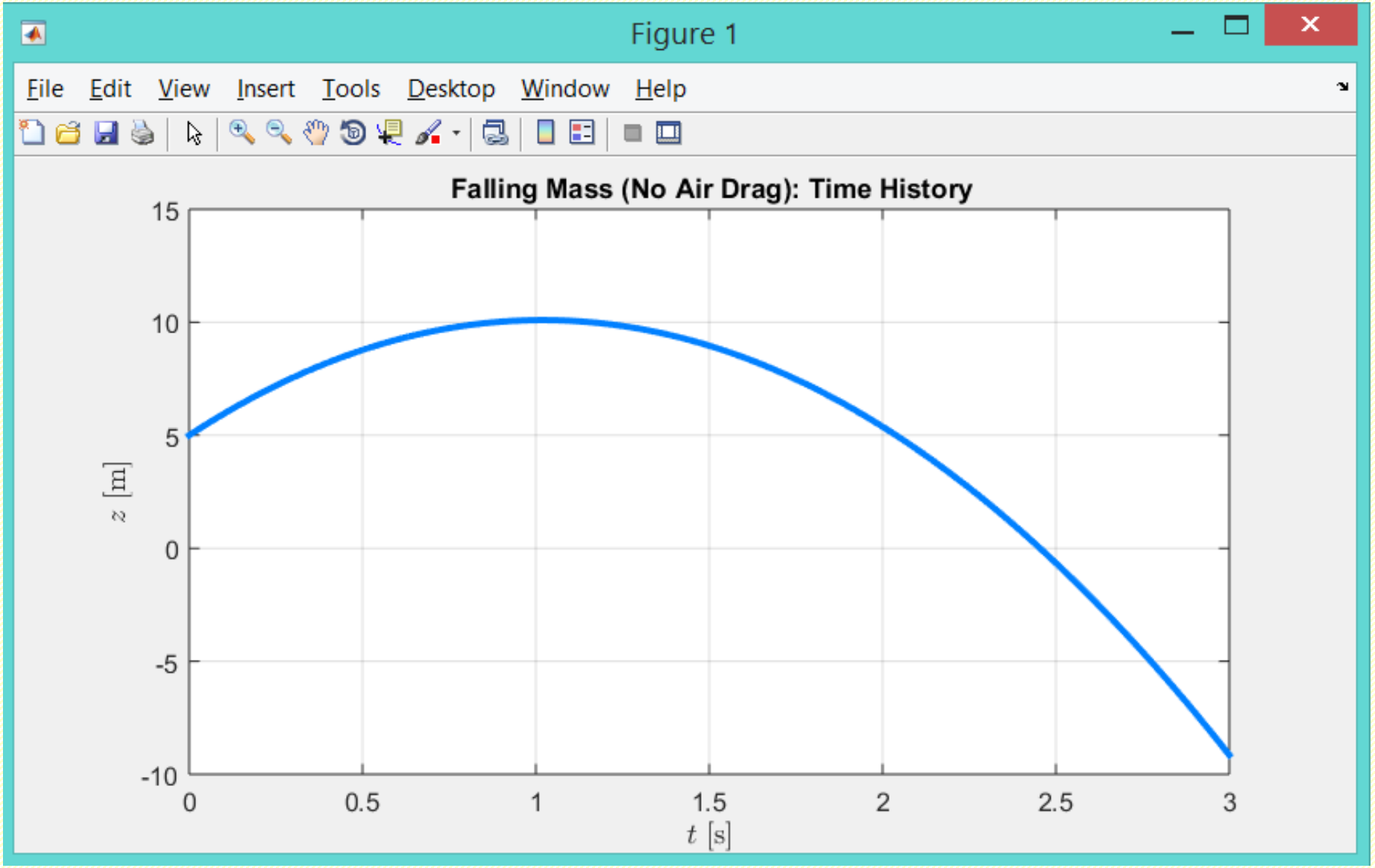
1. Input of the data

2. $$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -g \end{cases}$$

3. Calling ode45 procedure !

4. Plotting results

Output of the MATLAB script



MATLAB SCRIPT: 2 files solution

OENG1116_falling_mass.m

Main Program

```
%% FALLING MASS (NO DRAG) EXAMPLE
% Designed by Prof P.M.Trivailo (C)
%-----------------------------------
        %                ^ z
% Initial Conditions:    |
v0=10;   % m/s           |
z0=5;    % m             |
tmax=3; % s              +--------------> x
t=[0:0.01:1]*tmax;
[tt,zz]=ode45('falling_mass_xdot',t,[z0; v0]);
plot(tt,zz(:,1),'LineWidth',3,'Color',[0 0 1]);
grid on; xlabel('t [s]'); ylabel('z [m]');
```

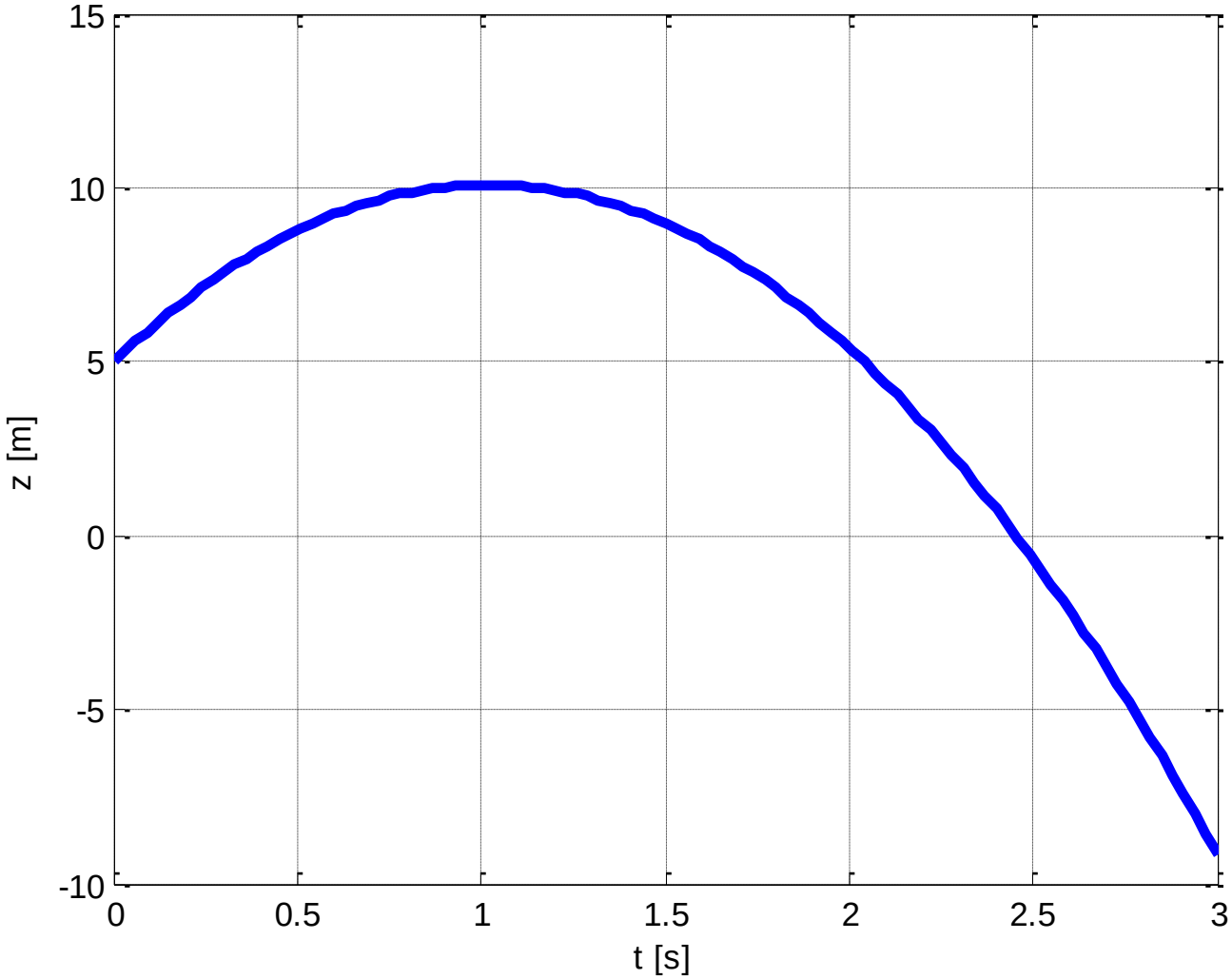
Must be the Same !!!!!!!!

falling_mass_xdot.m

```
function [xdot]= falling_mass_xdot(t,x)

xdot(1,1)=x(2);
xdot(2,1)=-9.81;
```

'Xdot' Function

RESULTS OF THE SIMULATION:



Discussion on the Script & Results

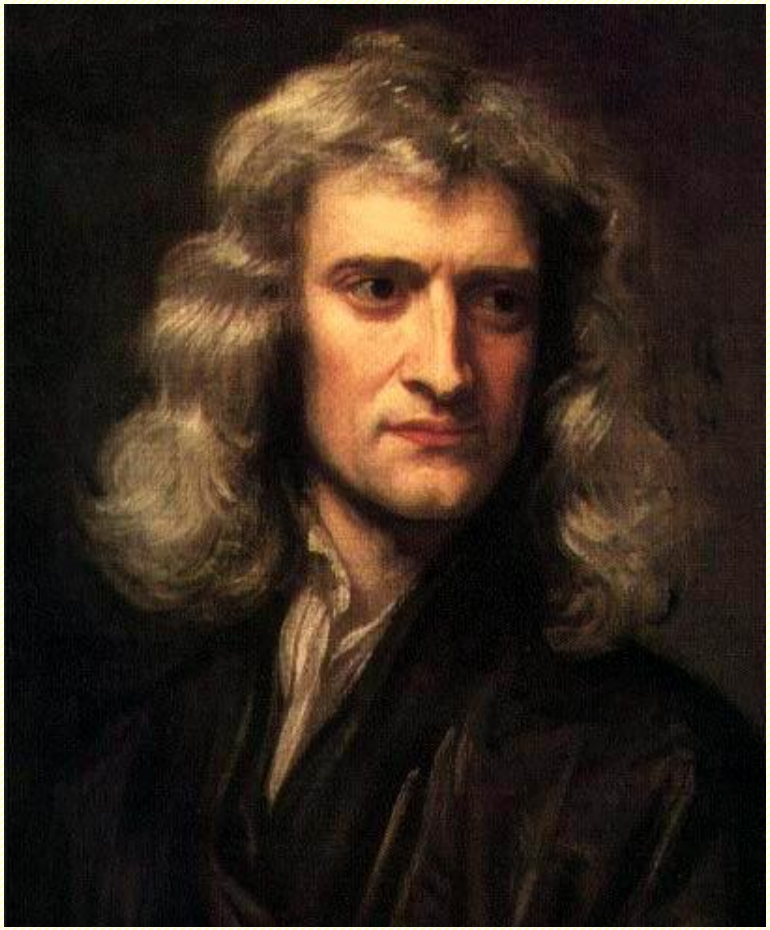
1. **NO DRAG !!!**
2. **Positive ‘z’ is UP**
3. Results (States) are in the ‘zz’ matrix.
4. Use ‘size(zz)’ to know the dimension.
5. ‘xdot’ Function **MUST** return a column vector!
6. Initial conditions are in the ‘column’ format.
7. Ask a question: WHEN the mass crosses the *z=0* level?

SYSTEMS, DESCRIBED WITH THE SECOND ORDER DIFFERENTIAL EQS

FUNDAMENTAL LAWS:

NEWTON's SECOND LAW EXAMPLE

ISAAC NEWTON



Sir **Isaac Newton** PRS (25 Dec. 1642 - 20 March 1727 [NS: 4 Jan. 1643 - 31 March 1727]) was an English physicist, mathematician, astronomer, natural philosopher, alchemist, and theologian, who has been ”considered by many to be the greatest and most influential scientist who ever lived.” His monograph *Philosophi Naturalis Principia Mathematica*, published in 1687, lays the foundations for most of classical mechanics. In this work, Newton described universal gravitation and the three laws of motion, which dominated the scientific view of the physical universe for the next three centuries.

Courtesy: Wikipedia https://en.wikipedia.org/wiki/Isaac_Newton

NEWTON'S LAWS of MOTION

Newton's First Law (also known as the Law of Inertia) states that an object at rest tends to stay at rest and that an object in uniform motion tends to stay in uniform motion unless acted upon by a net external force.

The meaning of this law is the existence of reference frames (called inertial frames) where objects not acted upon by forces move in uniform motion (in particular, they may be at rest).

Courtesy: Wikipedia https://en.wikipedia.org/wiki/Newton%27s_laws_of_motion

NEWTON'S LAWS of MOTION (Cont'd)

Newton's Second Law states that an applied force, ***F*** , on an object equals the rate of change of its momentum, ***p***, with time.

$$\sum \vec{\mathbf{F}} = \frac{d\vec{\mathbf{p}}}{dt} = \frac{d(m\vec{\mathbf{v}})}{dt}$$

$$\mathbf{F} = \frac{d\mathbf{p}}{dt} = \frac{d(m\mathbf{v})}{dt} = \mathbf{v}\frac{dm}{dt} + m\frac{d\mathbf{v}}{dt}$$

If applied to an object with constant mass

$$\frac{dm}{dt} = 0$$

the first term vanishes, and by substitution using the definition of acceleration, the equation can be written in the iconic form

$$\mathbf{F} = m\mathbf{a}$$

NEWTON'S LAWS of MOTION (Cont'd)

Newton's Third Law states that for every action there is an equal and opposite reaction. This means that any force exerted onto an object has a counterpart force that is exerted in the opposite direction back onto the first object.

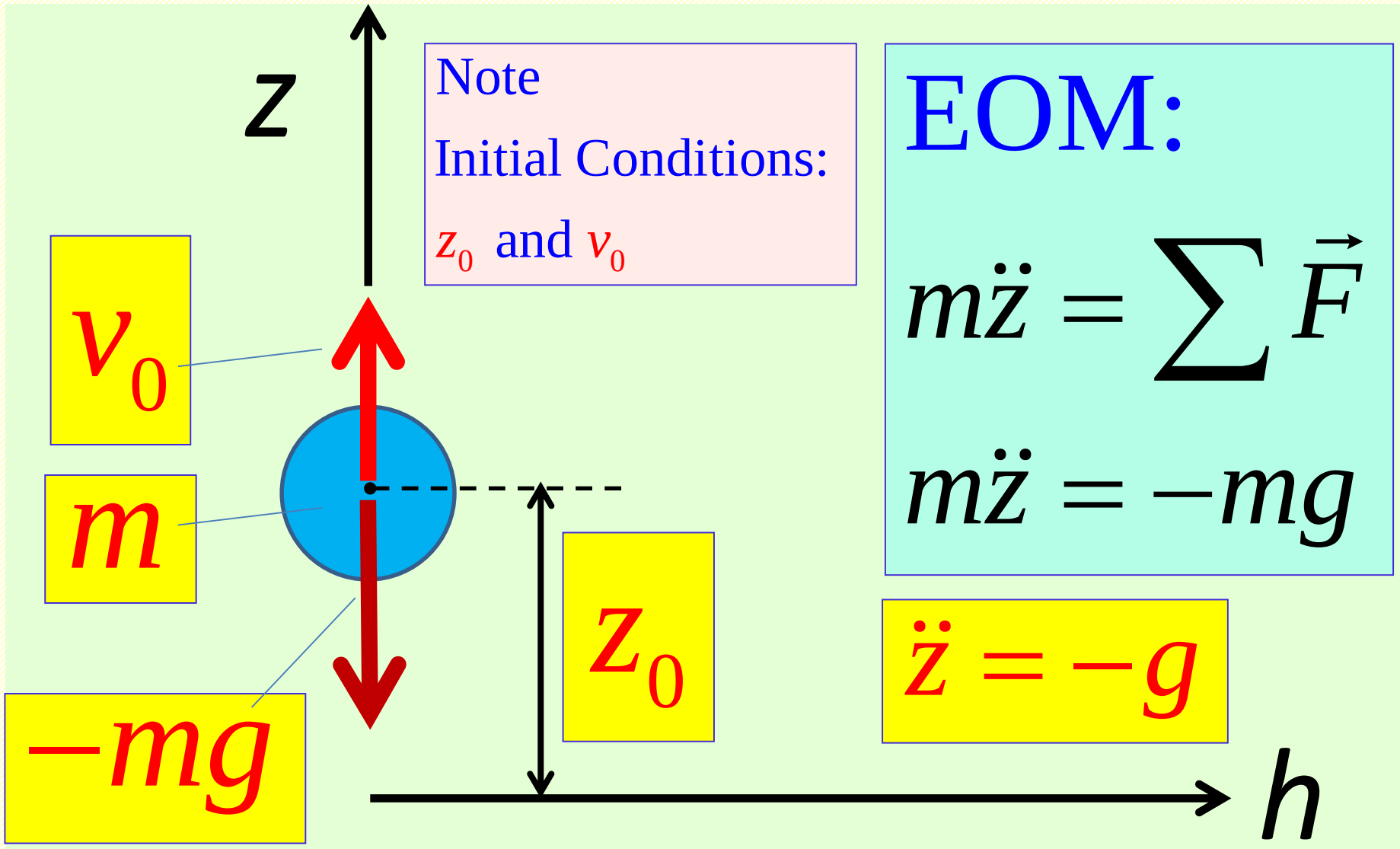
CASE STUDY: FALLING MASS

THEORY

SOLVING RESPONSES:

MATLAB and ODExxx

Modelling a Falling Mass



Note

Initial Conditions:

z_0 and v_0

EOM:

$$m\ddot{z} = \sum \vec{F}$$

$$m\ddot{z} = -mg$$

$$\ddot{z} = -g$$

SOLUTION: Analytical Method

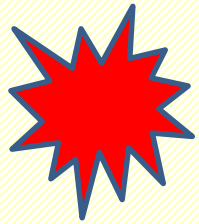
Integration of this differential equation allows analytical solution:

$$\frac{d^2z}{dt^2} = -g; \quad \Rightarrow \quad \frac{dz}{dt} = -gt + C_1; \quad \Rightarrow \quad z = -\frac{gt^2}{2} + C_1 t + C_2$$

Application of Initial Conditions enables us to find the values of constants C_1 and C_2 :

$$z\big|_{t=0} = z_0; \quad \Rightarrow \quad z_0 = C_2$$

$$\left.\frac{dz}{dt}\right|_{t=0} = v_0; \quad \Rightarrow \quad v_0 = C_1$$

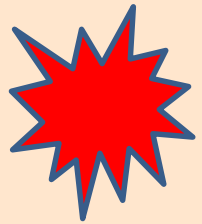


Analytical exact solution: $z(t) = z_0 + v_0 t - \frac{gt^2}{2}$

SOLUTION: Analytical Method

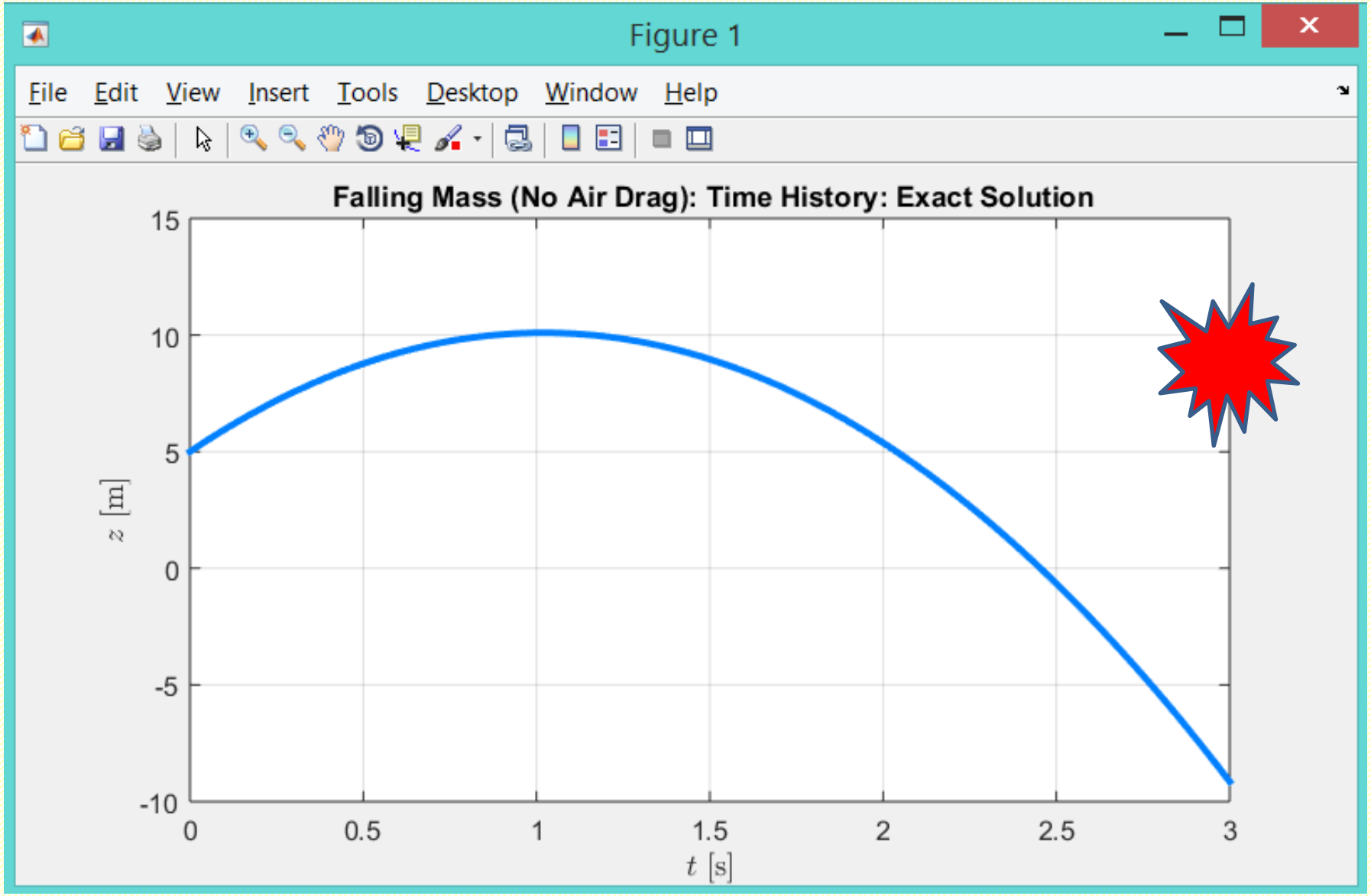
MATLAB SCRIPT (plotting results)

```
%% FALLING MASS (NO DRAG) EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2019
%-----------------------------------
        %                ^ z
% Initial Conditions:    |
v0=10;   % m/s           |
z0=5;    % m             |
tmax=3;  % s             +--                > n
t=[0:0.01:1]*tmax;
zz=z0+v0*t-g*t.^2/2;  % Note: pseudo-square operation !!!
plot(tt,zz,'LineWidth',3,'Color',[0 0.5 1]);
grid on;
xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$z$ [m]','Interpreter','LaTeX');
title('\bf Falling Mass (No Air Drag): Time History: Exact Solution');
set(gca, 'FontSize',12);
```



"OENG1116_falling_mass_NO_air_drag_EXACT.m" file

Output of the MATLAB script



!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

STATE-SPACE FORM
OF THE EQUATION OF MOTION

Modelling a Falling Mass: EOM

$$\ddot{z} = -g$$

Let us introduce new variables:

$$x_1 = z$$

$$x_2 = \dot{z}$$

then

$$\begin{cases} \dot{x}_1 = x_2 \end{cases}$$

Modelling a Falling Mass: EOM

$$\ddot{z} = -g$$

Also, with new variables:

$$x_1 = z$$

$$x_2 = \dot{z}$$

$$\begin{cases} \dot{x}_2 = -g \end{cases}$$

EOM in the STATE-SPACE FORM

$$\ddot{z} = -g$$

Let us introduce new variables:

$$x_1 = z$$

$$x_2 = \dot{z}$$

and

then

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -g \end{cases}$$

!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

NUMERICAL SOLUTION, ILLUSTRATED WITH FALLING MASS EXAMPLE

SOLUTION: Numerical (!!!) Method

MATLAB SCRIPT (no annotations)

```
%% FALLING MASS (NO DRAG) EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2019
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
%----------------------------------------
        %                ^ z
% Initial Conditions:    |
v0=10;   % m/s           |
z0=5;    % m             |
tmax=3;  % s             +--------------> h
t=[0:0.01:1]*tmax;
f_mass_xdot_anonymous = @(t, x) ([x(2); -9.81]);
[tt,zz]=ode45(f_mass_xdot_anonymous,t,[z0; v0]);
plot(tt,zz(:,1),'LineWidth',3,'Color',[0 0.5 1]);
grid on; xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$z$ [m]','Interpreter','LaTeX');
title('\bf Falling Mass (No Air Drag): Time History');
set(gca, 'FontSize',12);
```

"OENG1116_falling_mass_NO_air_drag_ANONYMOUS.m" file

SOLUTION: Numerical (!!!) Method

"OENG1116_falling_mass_NO_air_drag_ANONYMOUS.m" file

```
%% FALLING MASS (NO DRAG) EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2019
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
%-----------------------------------
        %                  ^ z
% Initial Conditions:     |
v0=10;   % m/s            |
z0=5;    % m              |
tmax=3;  % s              +--------------> h
t=[0:0.01:1]*tmax;
f_mass_xdot_anonymous = @(t, x) ([x(2); -9.81]);
[tt,zz]=ode45(f_mass_xdot_anonymous,t,[z0; v0]);
plot(tt,zz(:,1),'LineWidth',3,'Color',[0 0.5 1]);
grid on; xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$z$ [m]','Interpreter','LaTeX');
title('\bf Falling Mass (No Air Drag): Time
set(gca, 'FontSize',12);
```

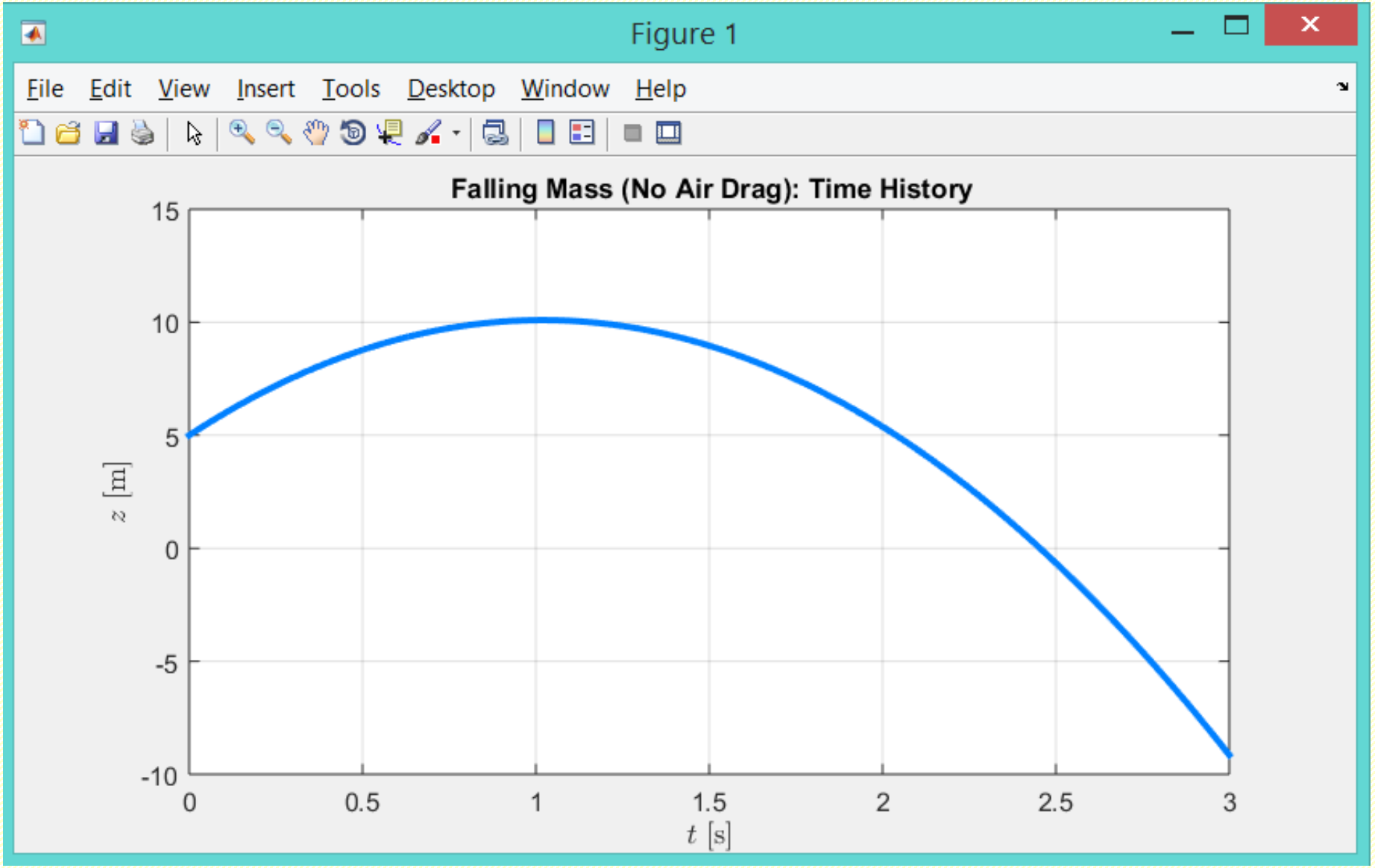
1. Input of the data

2. $$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -g \end{cases}$$

3. Calling ode45 procedure !

4. Plotting results

Output of the MATLAB script



MATLAB SCRIPT: 2 files solution

OENG1116_falling_mass.m

Main Program

```
%% FALLING MASS (NO DRAG) EXAMPLE
% Designed by Prof P.M.Trivailo (C)
%-----------------------------------
        %                ^ z
% Initial Conditions:    |
v0=10;   % m/s           |
z0=5;    % m             |
tmax=3; % s              +--------------> x
t=[0:0.01:1]*tmax;
[tt,zz]=ode45('falling_mass_xdot',t,[z0; v0]);
plot(tt,zz(:,1),'LineWidth',3,'Color',[0 0 1]);
grid on; xlabel('t [s]'); ylabel('z [m]');
```

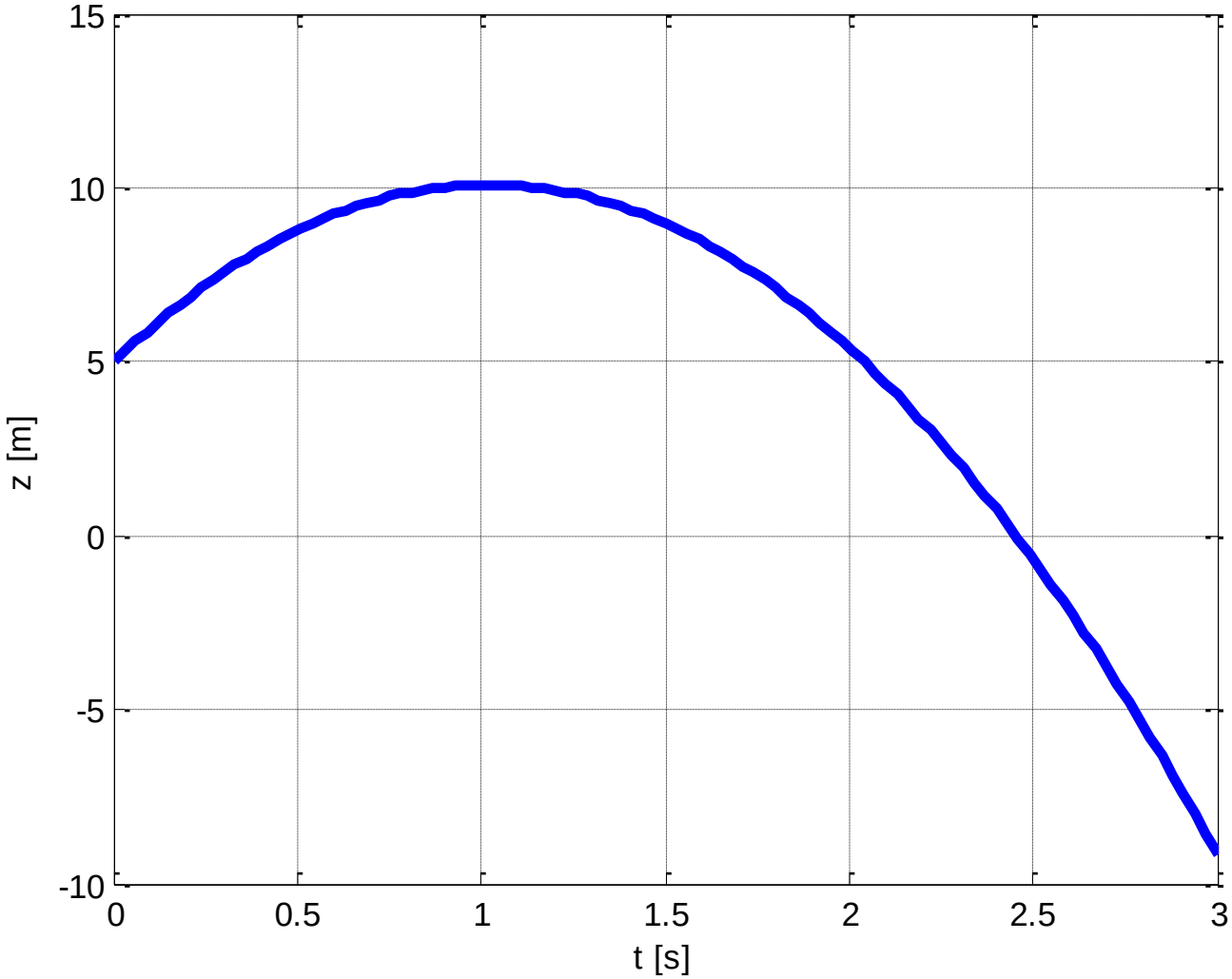
Must be the Same !!!!!!!!

falling_mass_xdot.m

```
function [xdot]= falling_mass_xdot(t,x)

xdot(1,1)=x(2);
xdot(2,1)=-9.81;
```

'Xdot' Function

RESULTS OF THE SIMULATION:



Discussion on the Script & Results

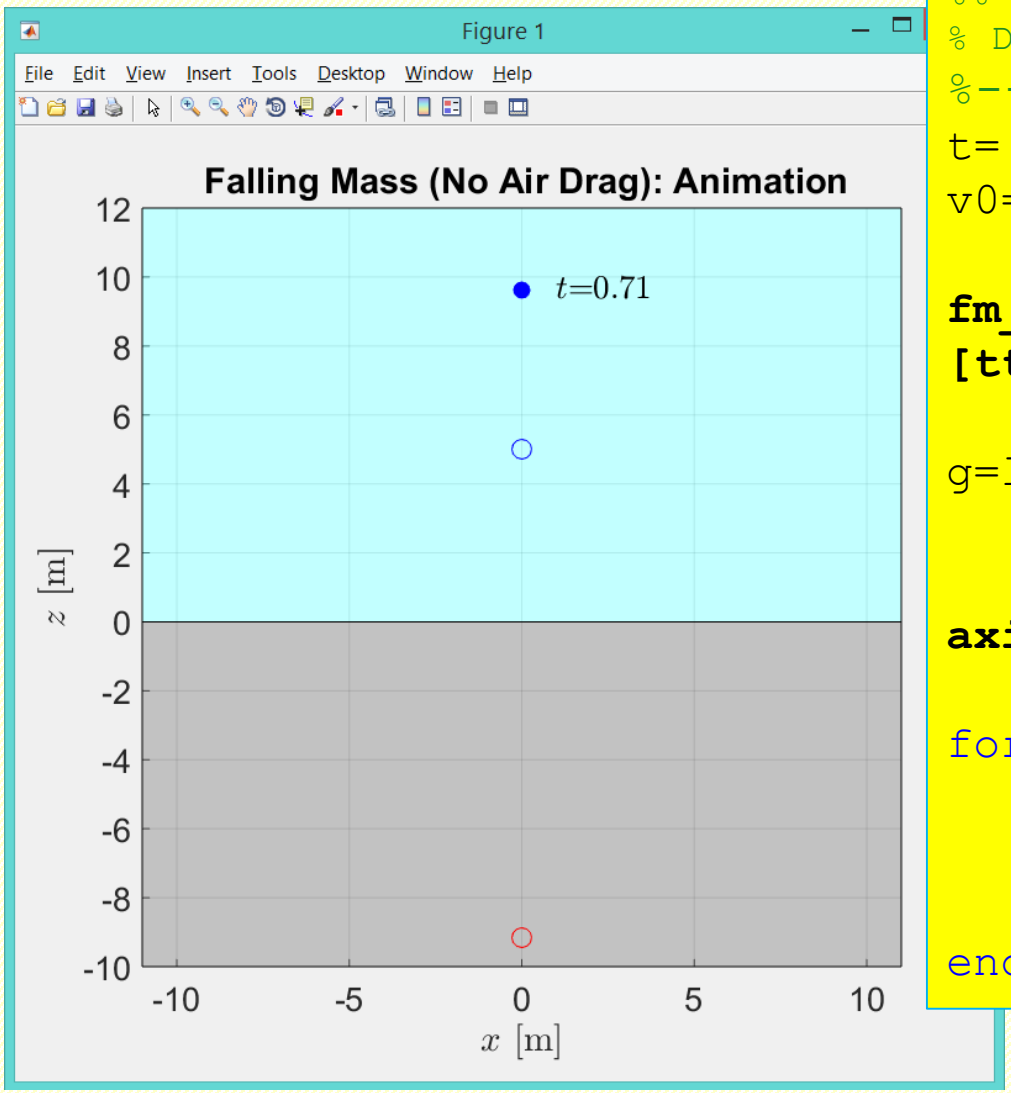
1. **NO DRAG !!!**
2. **Positive ‘z’ is UP**
3. Results (States) are in the ‘zz’ matrix.
4. Use ‘size(zz)’ to know the dimension.
5. ‘xdot’ Function **MUST** return a column vector!
6. Initial conditions are in the ‘column’ format.
7. Ask a question: WHEN the mass crosses the *z=0* level?

FALLING MASS:

ANIMATION EXAMPLE-1

(SINGLE MASS, i.e. 2 states task)

MATLAB: Animation Examples



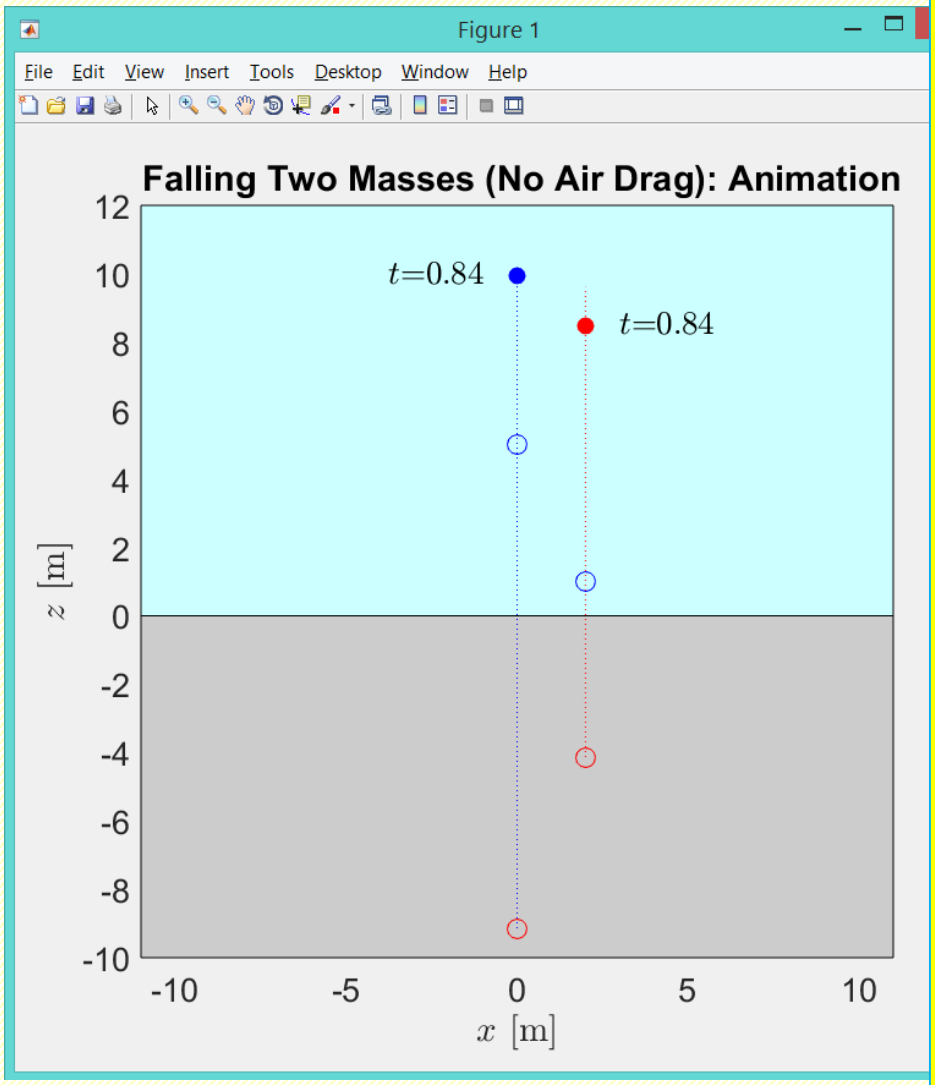

```
%% Basic solution/animation script:
% Designed by Prof P.M.Trivailo (C) 2019
%-----------------------------
t=[0:0.005:1]*3;
v0=10; z0=5;

fm_xdot = @(t, x) ([x(2); -9.81]);
[tt,zz]=ode45(fm_xdot,t,[z0; v0]);

g=line('XData',0,'YData',z0,…
    'Marker','.','MarkerSize',36);

axis([-11 11 -10 12]); grid on;

for i=1:length(tt)
    set(g,'YData',zz(i,1));
    drawnow;
    pause(0.01);
end
```

FALLING MASSES:

ANIMATION EXAMPLE-2

(TWO MASSES, i.e. 2x2 states task)

MATLAB: Animation Examples

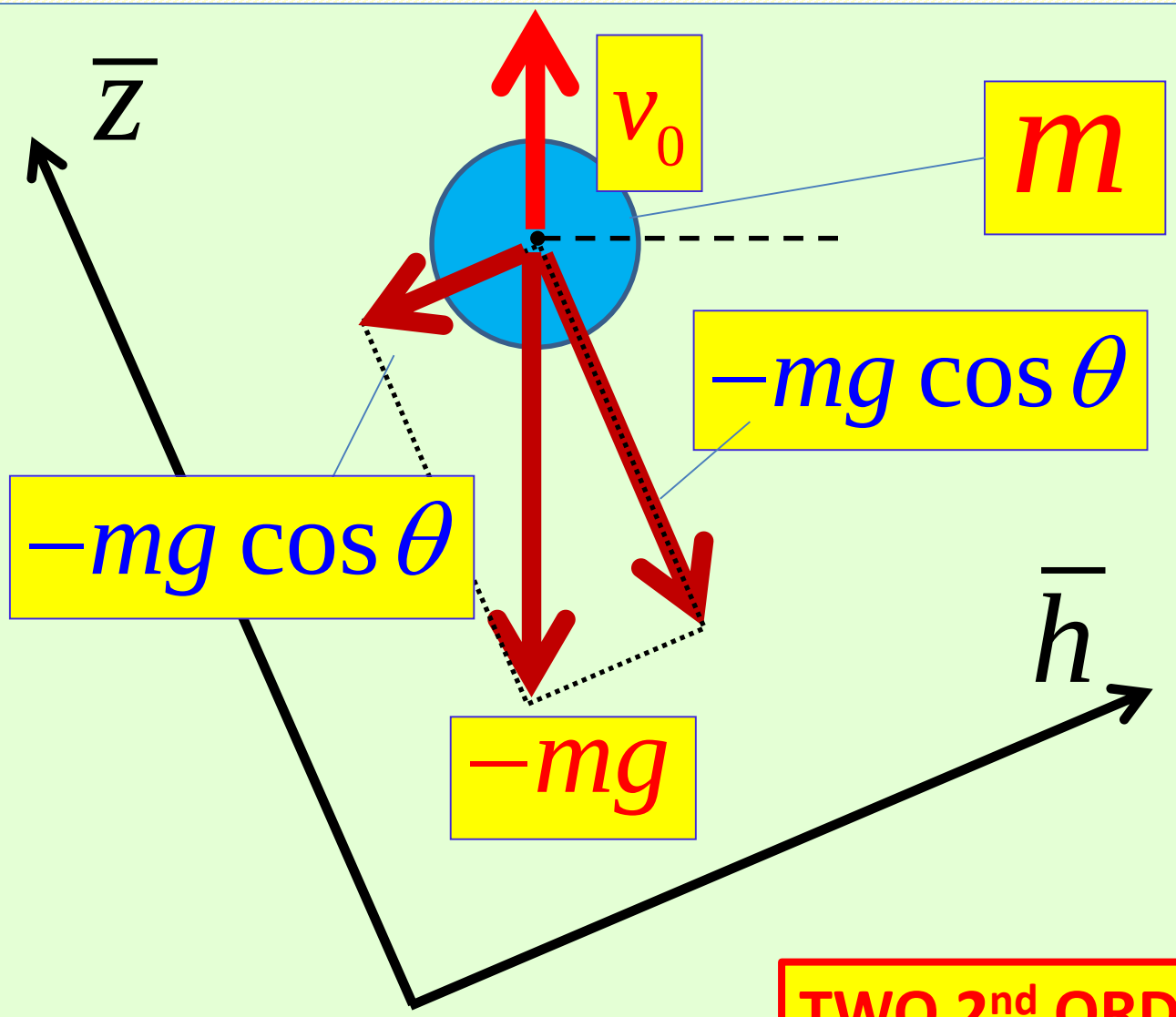



```
%% Basic solution/animation script:
% Designed by Prof P.M.Trivailo (C) 2019
%---------------------------
t=[0:0.005:1]*3;
v0=10; z0=5; v02=13; z02=1;

fm_xdot = @(t, x) ([x(2); -9.81]);

[tt,zz]=ode45(fm_xdot,t,[z0; v0]);
[tt2,zz2]=ode45(fm_xdot,t,[z02; v02]);

g=line('XData',0,'YData',z0,...
    'Marker','.','MarkerSize',36);
g2=line('XData',2,'YData',z0,...
'Marker','.','MarkerSize',36,'Color','r');

axis([-11 11 -10 12]); grid on;

for i=1:length(tt)
    set(g,'YData',zz(i,1));
    set(g2,'YData',zz2(i,1));
    drawnow;
    pause(0.01);
end
```

COORDINATE SYSTEMS:

CAN OTHER COORDINATE SYSTEM BE USED?

Modelling a Falling Mass: inclined CS!



EOM:

$$\begin{cases} m\ddot{\bar{z}} = \sum \vec{F}_{\bar{z}} \\ m\ddot{\bar{h}} = \sum \vec{F}_{\bar{h}} \end{cases}$$

$$\begin{cases} m\ddot{\bar{z}} = -mg\cos\theta \\ m\ddot{\bar{h}} = -mg\sin\theta \end{cases}$$

$$\begin{cases} \ddot{\bar{z}} = -g\cos\theta \\ \ddot{\bar{h}} = -g\sin\theta \end{cases}$$

TWO 2nd ORDER DIFF EQS!!!

MODELLING:

ONE-DOF SYSTEM EXAMPLE

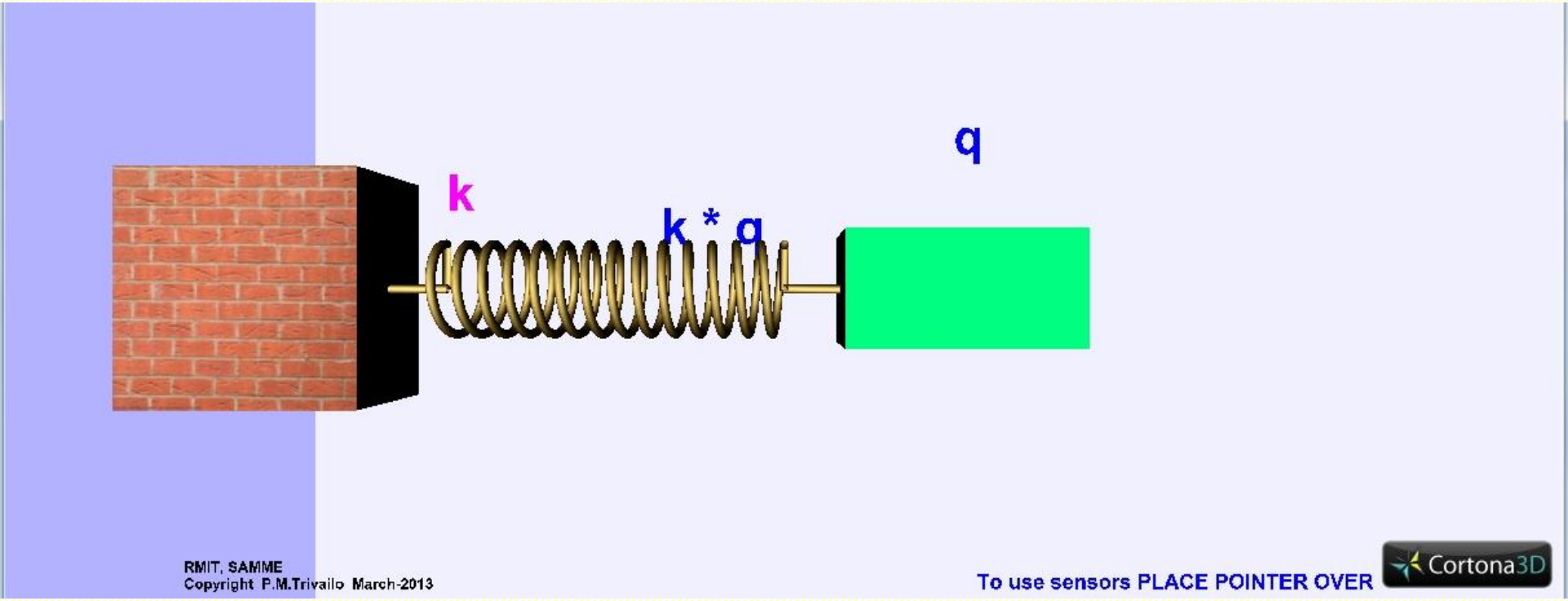
illustrated in Virtual Reality

One-DOF mass-spring system:

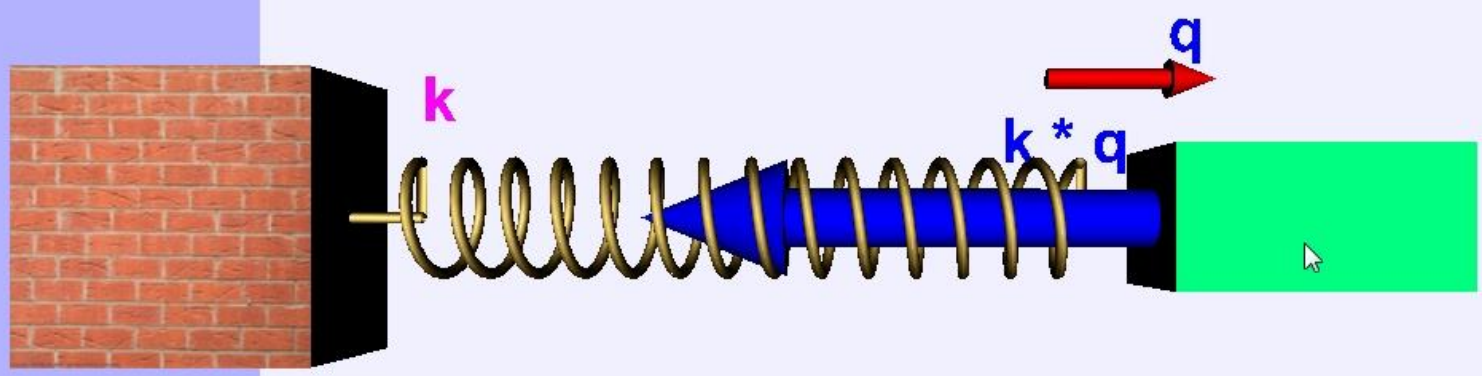
Derivation of EOM using Newton's Second Law & Free-Body Diagrams (FBDs).

1-DOF System: Static Equilibrium

VR Model



1-DOF System: Disturbed

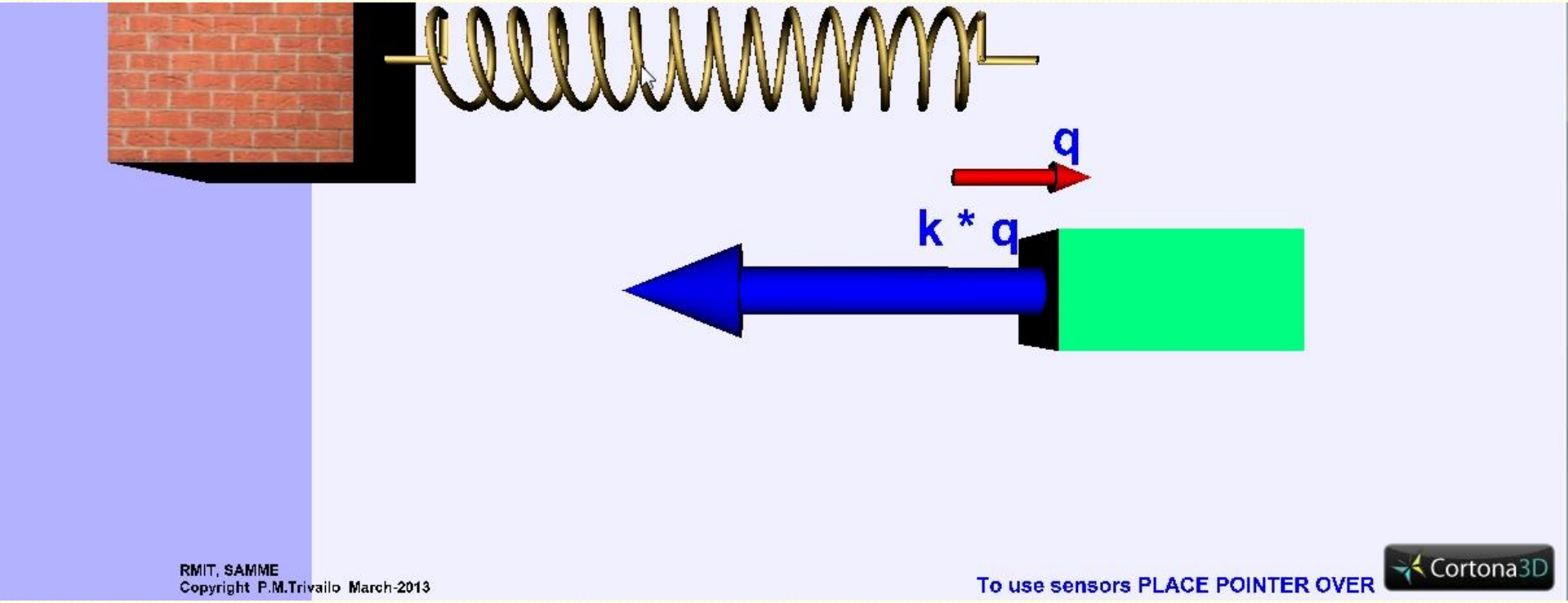


RMIT, SAMME
Copyright P.M.Trivailo March-2013

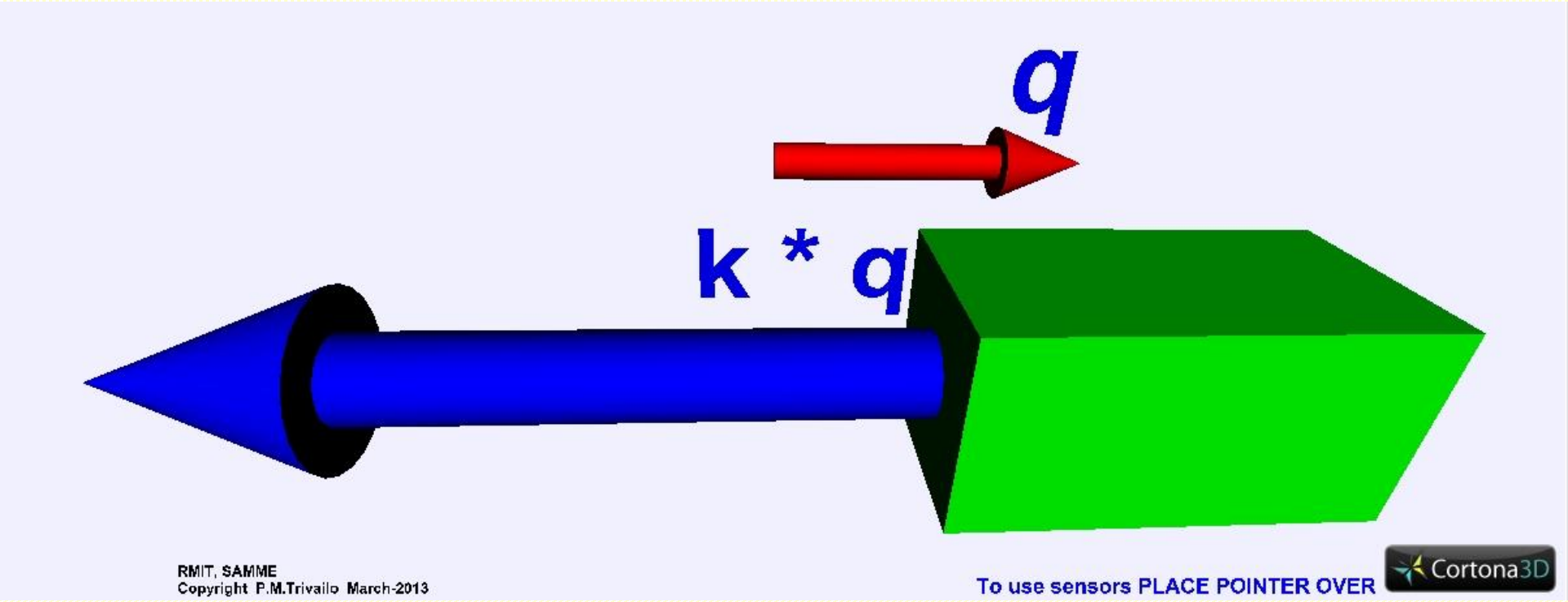
Press + Drag to CHANGE "q"

Cortona3D

1-DOF System: Constrains Removed



1-DOF System: Free-Body Diagram



$$m\mathbf{a} = \sum \mathbf{F} \quad \Rightarrow \quad m\ddot{q} = -kq$$

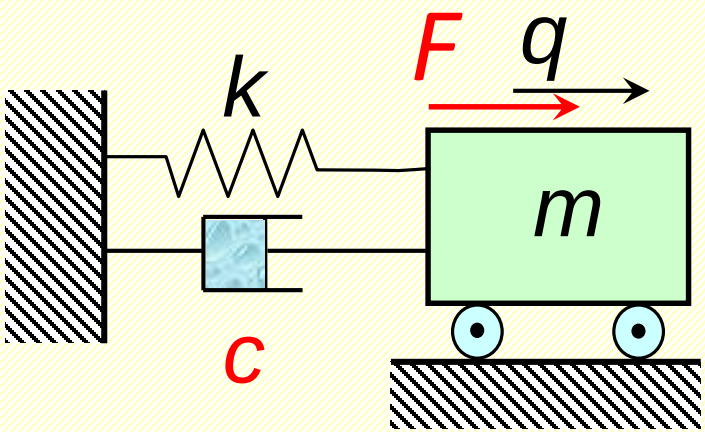
!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

NUMERICAL SOLUTION, ILLUSTRATED WITH MASS-SPRING EXAMPLE

1-DOF System: Free-Body Diagram



EXCITATION FORCE & DAMPING CAN ALSO BE ADDED.
RESULTANT Equation of Motion (EOM) IS:

$$m\ddot{q} + c\dot{q} + kq = F$$

LET US CONVERT IT IN THE STATE-SPACE FORM:

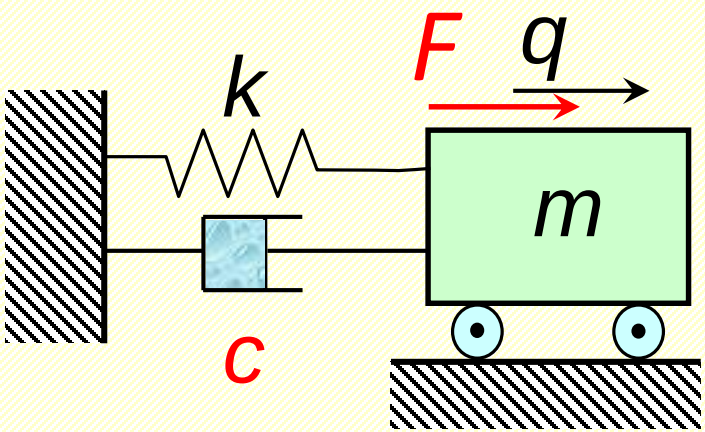
If $x_1 = q$ and $x_2 = \dot{q}$ then $\dot{x}_1 = x_2$.

Also, the EOM of the mass-damper-spring system can be written as:

$$\ddot{q} = -\frac{k}{m}q - \frac{c}{m}\dot{q} + \frac{F}{m} \quad \text{or} \quad \dot{x}_2 = -\frac{k}{m}x_1 - \frac{c}{m}x_2 + \frac{1}{m}F$$

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F$$

1-DOF System: Free-Body Diagram



EXCITATION FORCE & DAMPING CAN ALSO BE ADDED.
RESULTANT Equation of Motion (EOM) IS:

$$m\ddot{q} + c\dot{q} + kq = F$$

EOM IN THE STATE-SPACE MATRIX FORM:

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F$$

OR (vector format of the same):

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}, \quad \mathbf{x} = \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} q \\ \dot{q} \end{Bmatrix}, \quad A = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}$$

SOLUTION: Numerical (!!!) Method

MATLAB SCRIPT (no annotations)

```
%% MASS-SPRING-DAMPER SYSTEM EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2019
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
%---------------------------------------
m=10; k=100; c=10;
% Initial Conditions:
q0=3;          % m
q_dot0=15;     % m/s
tmax=6;        % s
t=[0:0.01:1]*tmax;
A=[0 1; -k/m -c/m]; B=[0; 1/m]; F=0;
mck_xdot_anonymous = @(t, x) A*x+B*F;
[tt,xx]=ode45(mck_xdot_anonymous,t,[q0; q_dot0]);
plot(tt,xx(:,1),'LineWidth',3,'Color',[0 0.5 1]);
grid on; xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$q$ [m]','Interpreter','LaTeX');
str1='\bf Free Vibration of Mass-Damper-Spring System: Time History ';
str2=sprintf('(m=%g kg; c=%g N*s/m; k=%g N/m)',m,c,k);
title([str1 str2]);
set(gca, 'FontSize',16); set(gcf,'Position',[240 -50 1200 750]);
```

SOLUTION: Numerical (!!!) Method

MATLAB SCRIPT (with annotations)

```
%% MASS-SPRING-DAMPER SYSTEM EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2019
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
%-----------------------------------
m=10; k=100; c=10;
% Initial Conditions:
q0=3;           % m
q_dot0=15;      % m/s
tmax=6;         % s
t=[0:0.01:1]*tmax;
A=[0 1; -k/m -c/m]; B=[0; 1/m]; F=0;
mck_xdot_anonymous = @(t, x) A*x+B*F;
[tt,xx]=ode45(mck_xdot_anonymous,t,[q0; q_dot0]);
plot(tt,xx(:,1),'LineWidth',3,'Color',[0 0.5 1]);
grid on; xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$q$ [m]','Interpreter','LaTeX');
str1='\bf Free Vibration of Mass-Damper-Spring System: T
str2=sprintf('(m=%g kg;
title([str1 str2]);
set(gca, 'FontSize',16)
```

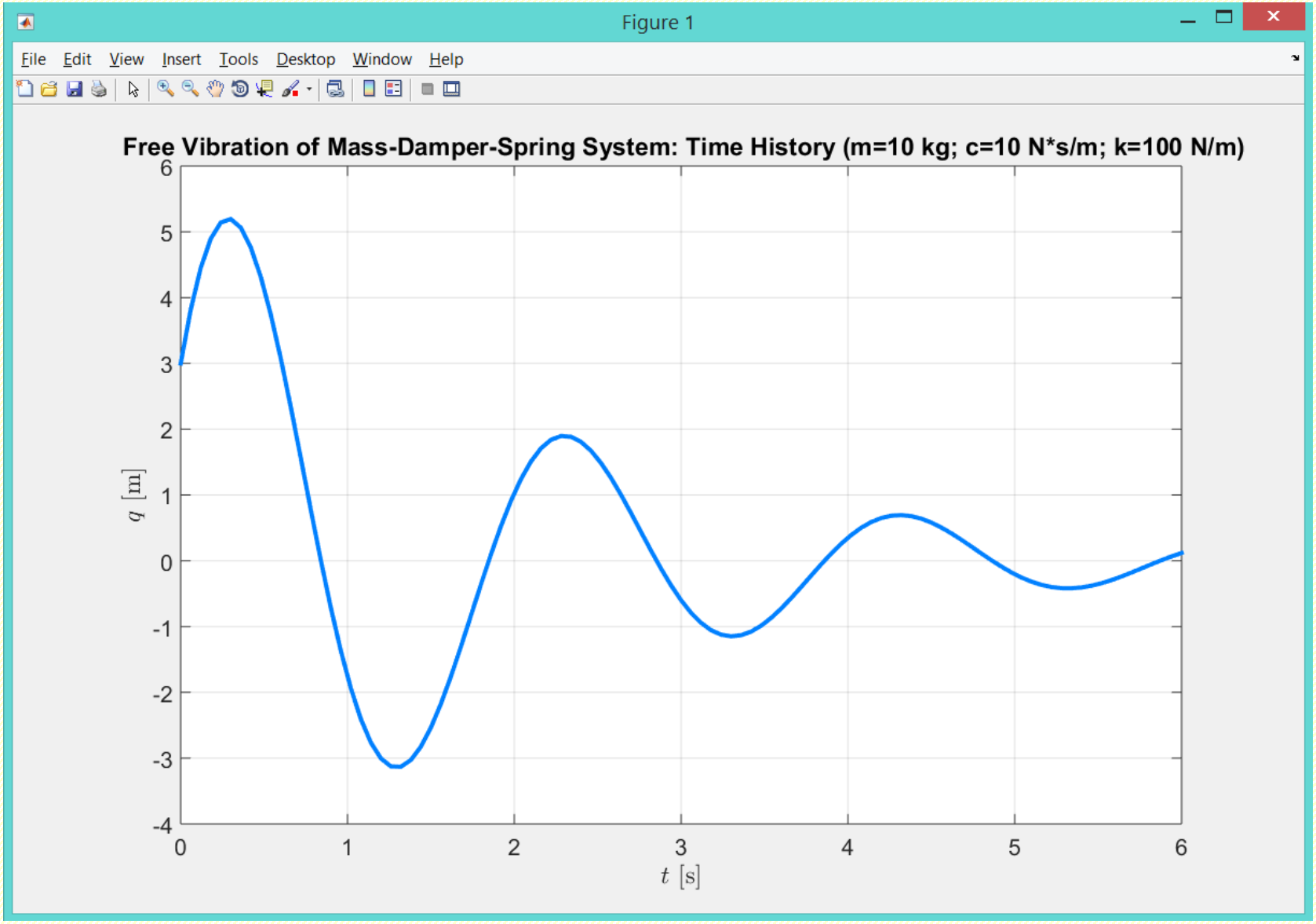
1. Input of the data

2. $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$

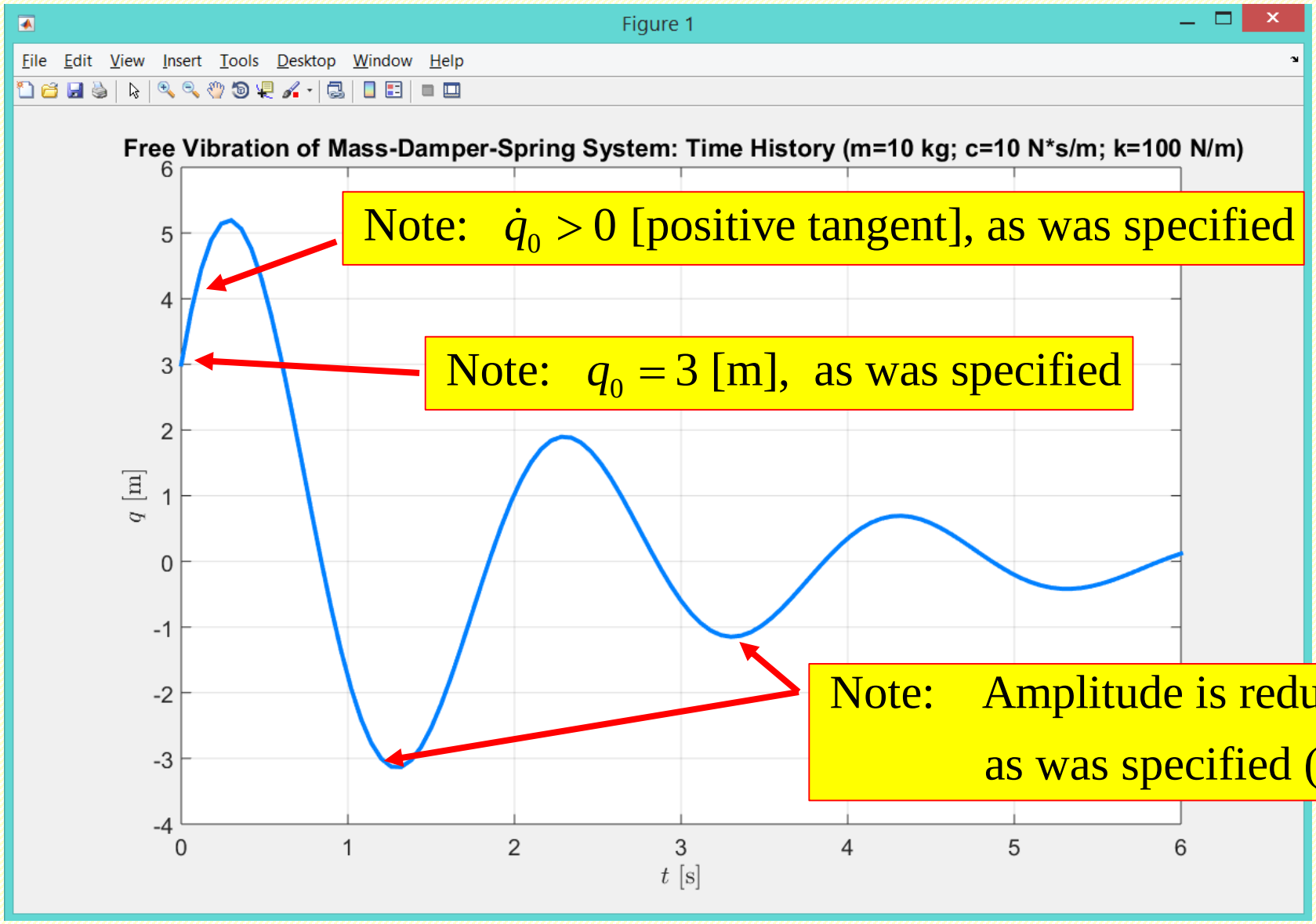
3. Calling ode45 procedure !

4. Plotting results

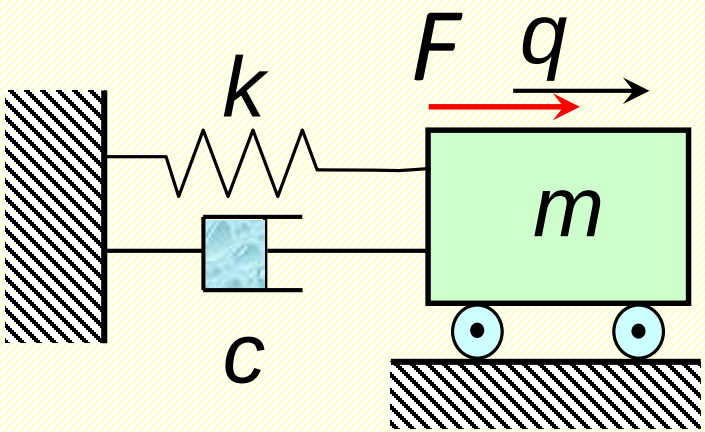
Output of the MATLAB script

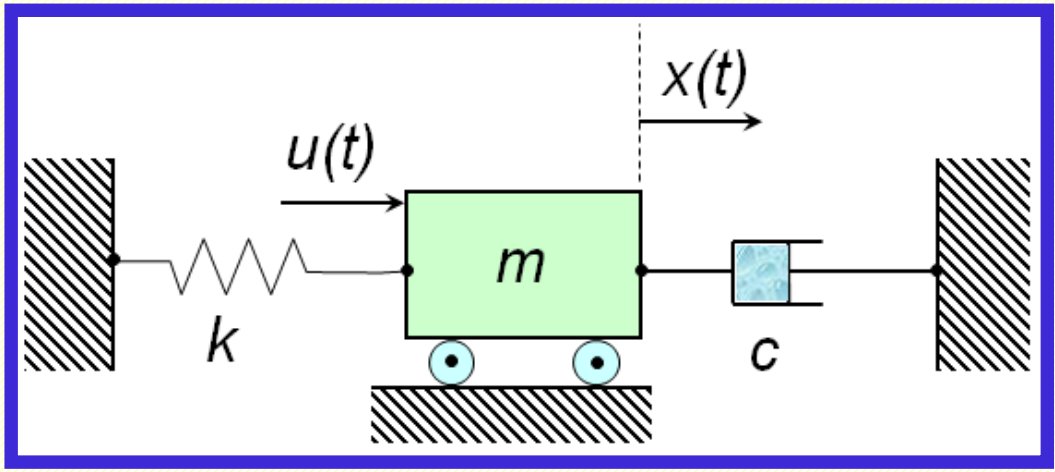


Output of the MATLAB script (annotated)



Some Comments:





1. Mass, damper, stiffness arrangements.
2. Adding springs in parallel/sequence & combination.
3. Initial Conditions – must be a vector!
4. Inspect dimensions of the results.
5. Which option to take: One-file or two-files?
6. Best choice: analytical, numerical and/or symbolic?

One-DOF mass-spring-damper system:

ADVANCED CASE: FORCED EXCITATION

SOLUTION: Numerical (!!!) Method

MATLAB SCRIPT (no annotations)

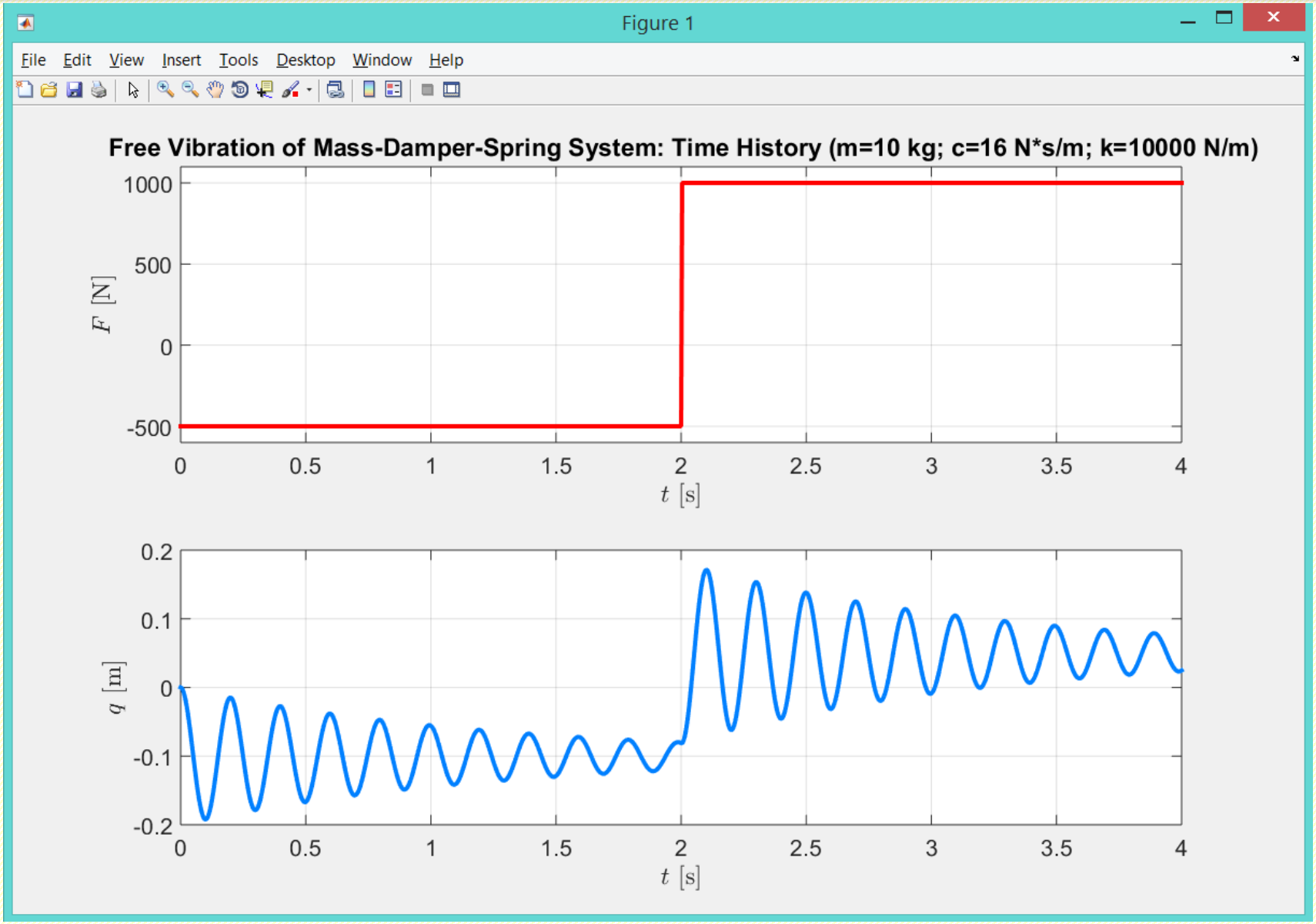
```
%% MASS-SPRING-DAMPER SYSTEM EXAMPLE: FORCED CASE
% Designed by Prof P.M.Trivailo (C) 2019
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
%--------------------------------------
clear; close all; clc;
m=10; k=10000; c=16;
% Initial Conditions:
q0=0;          % m
q_dot0=0;    % m/s
tmax=4;        % s
t=[0:0.001:1]*tmax;
A=[0 1; -k/m -c/m]; B=[0; 1/m];
mck_xdot_anonymous = @(t, x) A*x+B*(-1000*(t<=2)+500*(t>2));
F_anonymous = @(t) (-500*(t<=2)+1000*(t>2));
[tt,xx]=ode45(mck_xdot_anonymous,t,[q0; q_dot0]);
%--------------------
```

SOLUTION: Numerical (!!!) Method

MATLAB SCRIPT (no annotations)

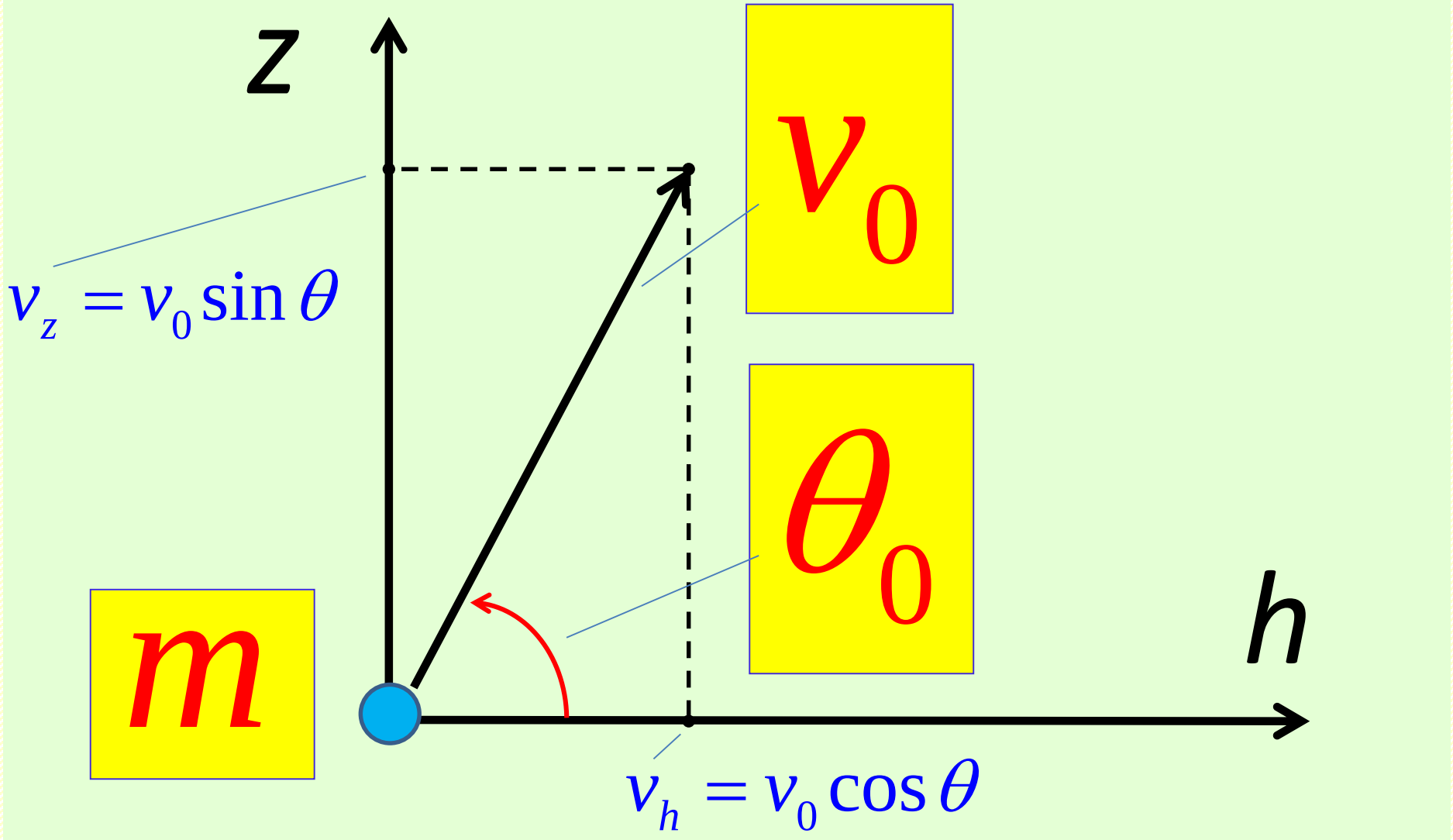
```
% Continuation of the script
%--------------------
subplot(2,1,1);
plot(t,F_anonymous(t),'LineWidth',3,'Color','r');
grid on; xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$F$ [N]','Interpreter','LaTeX');
axis([0 tmax -600 1100]);
set(gca, 'FontSize',16);
%--------------------
str1='\bf Free Vibration of Mass-Damper-Spring System: Time History ';
str2=sprintf('(m=%g kg; c=%g N*s/m; k=%g N/m)',m,c,k);
title([str1 str2]);
%--------------------
subplot(2,1,2);
plot(tt,xx(:,1),'LineWidth',3,'Color',[0 0.5 1]);
grid on; xlabel('$t$ [s]','Interpreter','LaTeX');
ylabel('$q$ [m]','Interpreter','LaTeX');
set(gca, 'FontSize',16);
set(gcf,'Position',[ 240  -50 1200  750]);
```

Output of the MATLAB script

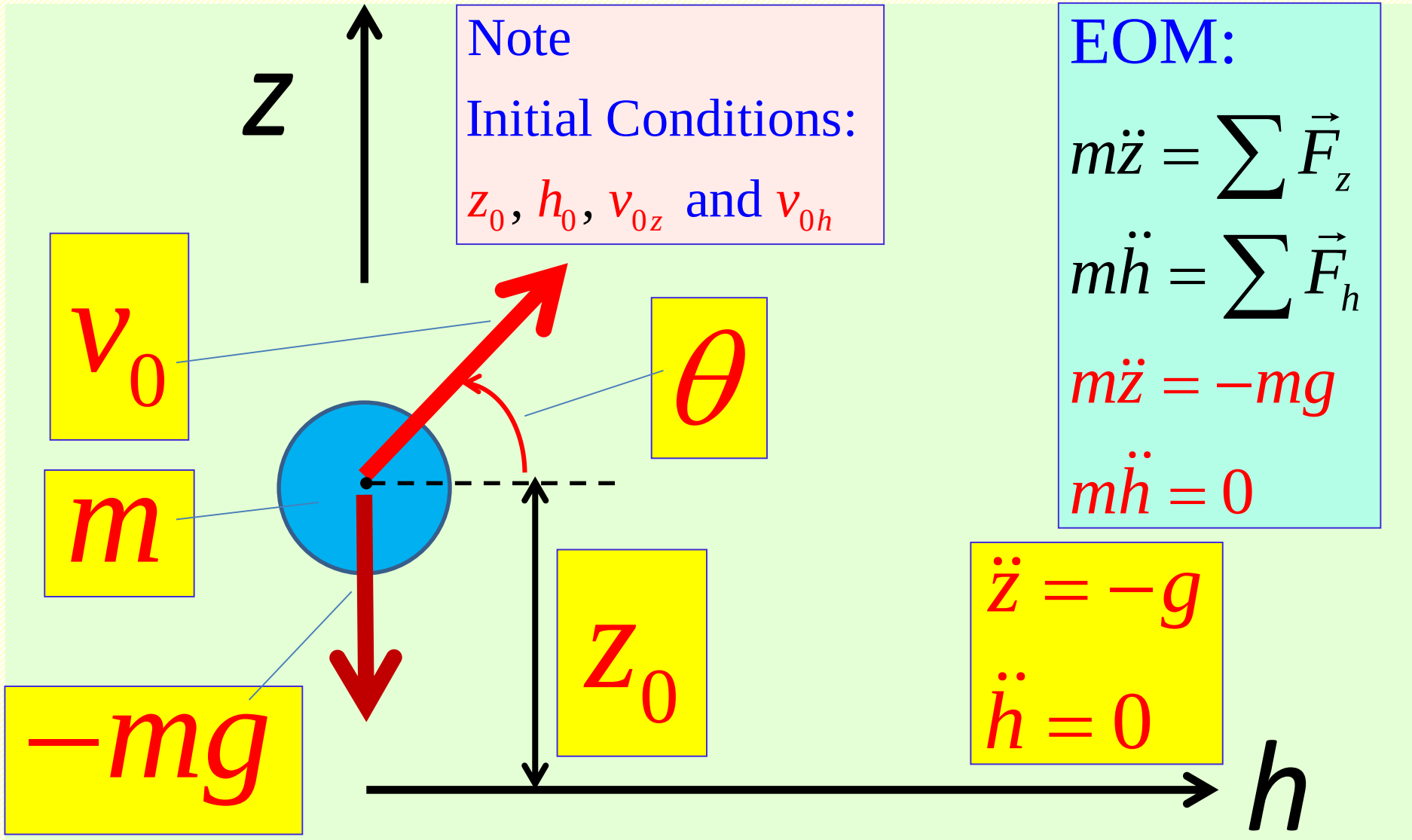


PROJECTILE MOTION: Numerical Solutions

Case Study: Definitions



Modelling a Projectile: No Air-Resistance



Modelling a Projectile: EOM

$$\ddot{z} = -g$$

Let us introduce new variables:

$$x_1 = z$$

$$x_2 = \dot{z}$$

then

$$\begin{cases} \dot{x}_1 = x_2 \end{cases}$$

Modelling a Projectile: EOM

$$\ddot{z} = -g$$

then

$$\dot{x}_2 = -g$$

Let us introduce new variables:

$$x_1 = z$$

$$x_2 = \dot{z}$$

Modelling a Projectile: EOM

$$\ddot{z} = -g$$

then

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

$$\dot{x}_2 = -g$$

then

Let us introduce new variables:

$$x_1 = z$$

$$x_2 = \dot{z}$$

Modelling a Projectile: EOM

$$\ddot{h} = 0$$

$$\begin{cases} \dot{x}_3 = x_4 \end{cases}$$

then

Let us introduce new variables:

$$x_3 = h$$

$$x_4 = \dot{h}$$

Modelling a Projectile: EOM

$$\ddot{h} = 0$$

then

$$\begin{cases} \\ \\ \\ \dot{x}_4 = 0 \end{cases}$$

Let us introduce new variables:

$$x_3 = h$$

$$x_4 = \dot{h}$$

Modelling a Projectile: EOM

$$\ddot{h} = 0$$

then

$$\begin{cases} \dot{x}_3 = x_4 \\ \dot{x}_4 = 0 \end{cases}$$

then

Let us introduce new variables:

$$x_3 = h$$

$$x_4 = \dot{h}$$

Modelling a Projectile: EOM

$$\ddot{z} = -g$$

$$\ddot{h} = 0$$

then

then

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

$$\dot{x}_2 = -g$$

$$\dot{x}_3 = x_4$$

$$\dot{x}_4 = 0$$

then

then

Let us introduce new variables:

$$x_1 = z$$

$$x_2 = \dot{z}$$

Let us introduce new variables:

$$x_3 = h$$

$$x_4 = \dot{h}$$

IMPORTANT COMMENTS:

$$\ddot{z} = -g$$

$$\ddot{h} = 0$$

Note: YOU SELECT ORDER OF STATES!

1. Always remember definition of states & their selected order !!!

IF (Example-1)

$$\begin{cases} x_1 = z \\ x_2 = \dot{z} \\ x_3 = h \\ x_4 = \dot{h} \end{cases}$$

2. Order of right hand terms depends upon the order of states !!!

THEN

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -g \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = 0 \end{cases}$$

IMPLICATIONS OF ORDER OF STATES:

Note: YOU SELECT ORDER OF STATES!

IF

(Example-1)

$$\begin{cases} x_1 = z \\ x_2 = \dot{z} \\ x_3 = h \\ x_4 = \dot{h} \end{cases}$$

(1) Vector of Initial Conditions should be:

THEN

$$\mathbf{x}_0 = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_{t=0} = \begin{bmatrix} z_0 \\ v_{0z} \\ h_0 \\ v_{0h} \end{bmatrix}$$

(2) Trajectory of the mass is given by $x_3 - x_1$ data (calculated by ode45)

IMPORTANT COMMENTS:

$$\ddot{z} = -g$$

$$\ddot{h} = 0$$

Note: YOU SELECT ORDER OF STATES!

1. Always remember definition of states & their selected order !!!

IF (Example-2)

$$\begin{cases} x_1 = z \\ x_2 = h \\ x_3 = \dot{z} \\ x_4 = \dot{h} \end{cases}$$

2. Order of right hand terms depends upon the order of states !!!

THEN

$$\begin{cases} \dot{x}_1 = x_3 \\ \dot{x}_2 = x_4 \\ \dot{x}_3 = -g \\ \dot{x}_4 = 0 \end{cases}$$

IMPLICATIONS OF ORDER OF STATES:

Note: YOU SELECT ORDER OF STATES!

IF

(Example-2)

$$\begin{cases} x_1 = z \\ x_2 = h \\ x_3 = \dot{z} \\ x_4 = \dot{h} \end{cases}$$

(1) Vector of Initial Conditions should be:

THEN

$$\mathbf{x}_0 = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_{t=0} = \begin{bmatrix} z_0 \\ h_0 \\ v_{0z} \\ v_{0h} \end{bmatrix}$$

(2) Trajectory of the mass is given by $x_2 - x_1$ data (calculated by ode45)

IMPORTANT COMMENTS:

$$\ddot{z} = -g$$

$$\ddot{h} = 0$$

Note: YOU SELECT ORDER OF STATES!

1. Always remember definition of states & their selected order !!!

IF (Example-3)

$$\begin{cases} x_1 = h \\ x_2 = z \\ x_3 = \dot{h} \\ x_4 = \dot{z} \end{cases}$$

2. Order of right hand terms depends upon the order of states !!!

THEN

$$\begin{cases} \dot{x}_1 = x_3 \\ \dot{x}_2 = x_4 \\ \dot{x}_3 = 0 \\ \dot{x}_4 = -g \end{cases}$$

IMPLICATIONS OF ORDER OF STATES:

Note: YOU SELECT ORDER OF STATES!

IF (Example-3)

$$\begin{cases} x_1 = h \\ x_2 = z \\ x_3 = \dot{h} \\ x_4 = \dot{z} \end{cases}$$

(1) Vector of Initial Conditions should be:

THEN

$$\mathbf{x}_0 = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_{t=0} = \begin{bmatrix} h_0 \\ z_0 \\ v_{0h} \\ v_{0z} \end{bmatrix}$$

(2) Trajectory of the mass is given by $x_1 - x_2$ data (calculated by ode45)

IMPORTANT COMMENTS:

$$\ddot{z} = -g$$

$$\ddot{h} = 0$$

Note: YOU SELECT ORDER OF STATES!

1. Always remember definition of states & their selected order !!!

IF (Example-4)

$$\begin{cases} x_1 = h \\ x_2 = \dot{h} \\ x_3 = z \\ x_4 = \dot{z} \end{cases}$$

2. Order of right hand terms depends upon the order of states !!!

THEN

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = 0 \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = -g \end{cases}$$

IMPLICATIONS OF ORDER OF STATES:

Note: YOU SELECT ORDER OF STATES!

IF (Example-4)

$$\begin{cases} x_1 = h \\ x_2 = \dot{h} \\ x_3 = z \\ x_4 = \dot{z} \end{cases}$$

(1) Vector of Initial Conditions should be:

THEN

$$\mathbf{x}_0 = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_{t=0} = \begin{bmatrix} h_0 \\ v_{0h} \\ z_0 \\ v_{0z} \end{bmatrix}$$

(2) Trajectory of the mass is given by $x_1 - x_3$ data (calculated by ode45)

!!!!!!!!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

NUMERICAL SOLUTION, ILLUSTRATED WITH PROJECTILE EXAMPLE

SOLUTION: Numerical (!!!) Method

MATLAB SCRIPT (no annotations)

```
%% PROJECTILE (NO DRAG) EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2019
th=45*pi/180; % red       ^ z
v0=10;        % m/s       |
z0=0; h0=0    % m         +---------------> h
tmax=2; t=[0:0.01:1]*tmax;
v0h=v0*cos(th); v0z=v0*sin(th);
pr_xdot = @(t, x) ([x(2); -9.81; x(4); 0]);
[tt,zz]=ode45(pr_xdot,t,[z0; v0z; h0; v0h]);
plot(zz(:,3), zz(:,1),'LineWidth',3,'Color',[0 0.5 1]);
grid on; axis equal; xlabel('$h$ [m]','Interpreter','LaTeX');
ylabel('$z$ [m]','Interpreter','LaTeX');
title('\bf Projectile (No Air Drag): Time History');
set(gca, 'FontSize',18); hold on;
g=line('XData',h0,'YData',z0,'Marker','.','MarkerSize',48);
for i=1:length(tt),
   set(g,'XData',zz(i,3),'YData',zz(i,1)); drawnow; pause(0.01);
end
```

SOLUTION: Numerical (!!!) Method

MATLAB SCRIPT (annotated)

```
%% PROJECTILE (NO DRAG) EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2019
th=45*pi/180; % red
v0=10;        % m/s
z0=0; h0=0    % m          +--------------> h
tmax=2; t=[0:0.01:1]*tmax;
v0h=v0*cos(th); v0z=v0*sin(th);
pr xdot = @(t, x) ([x(2); -9.81; x(4); 0]);
[tt,zz]=ode45(pr_xdot,t,[z0; v0z; h0; v0h]);
plot(zz(:,3), zz(:,1),'LineWidth',3,'Color',[0 0.5 1]);
grid on; axis equal; xlabel('$h$ [m]','Interpreter','LaTeX');
ylabel('$z$ [m]','Interpreter','LaTeX');
title('\bf Projectile (No Air Drag): Time Hi
set(gca, 'FontSize',18); hold on;
g=line('XData',h0,'YData',z0,'Marker','.','MarkerSize',48);
for i=1:length(tt)
   set(g,'XData',z
end
```

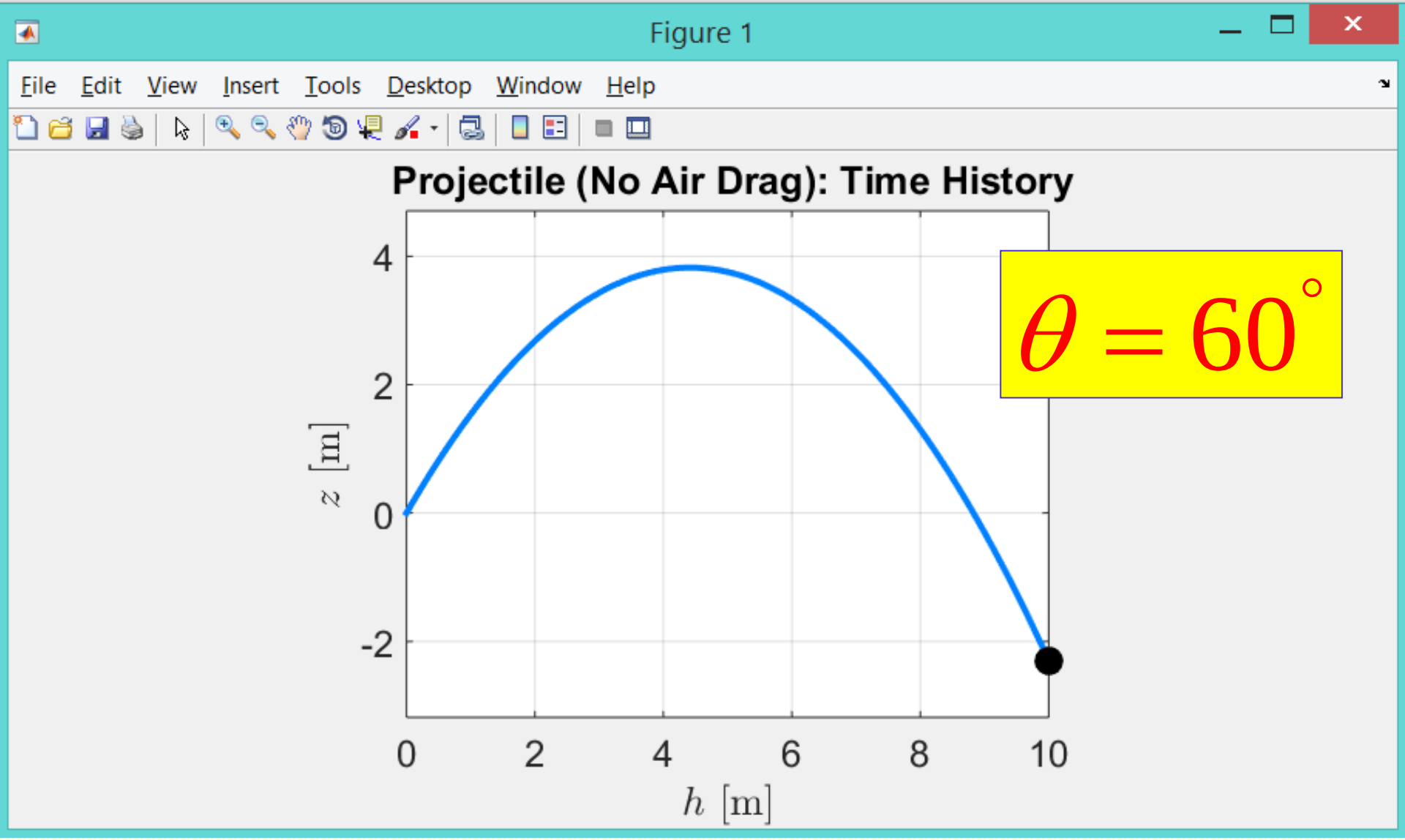
1. Input of the data

2. $$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -g \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = 0 \end{cases}$$

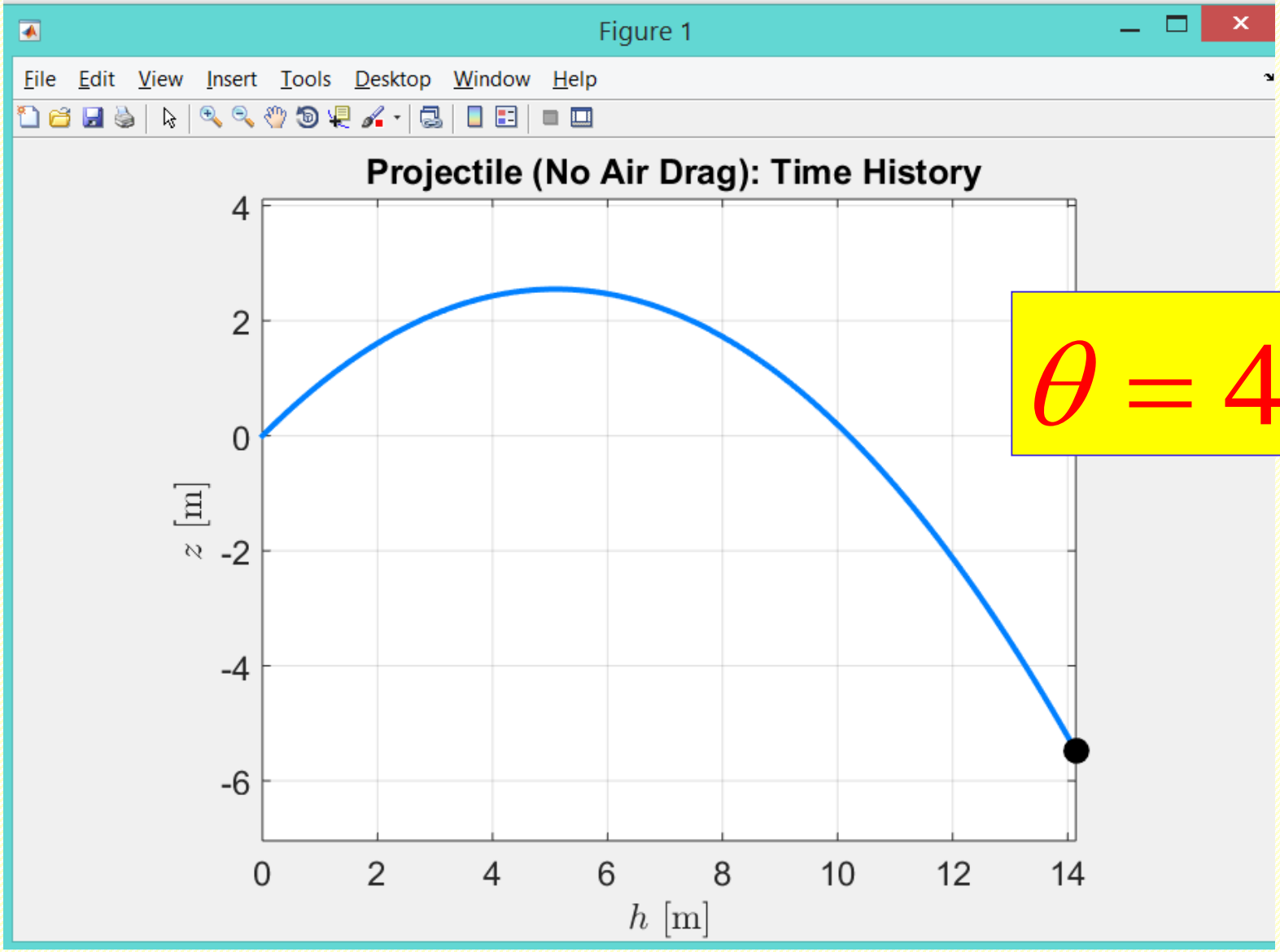
3. Calling ode45 procedure !

4. Plotting results

Output of the MATLAB script [0-2 sec]

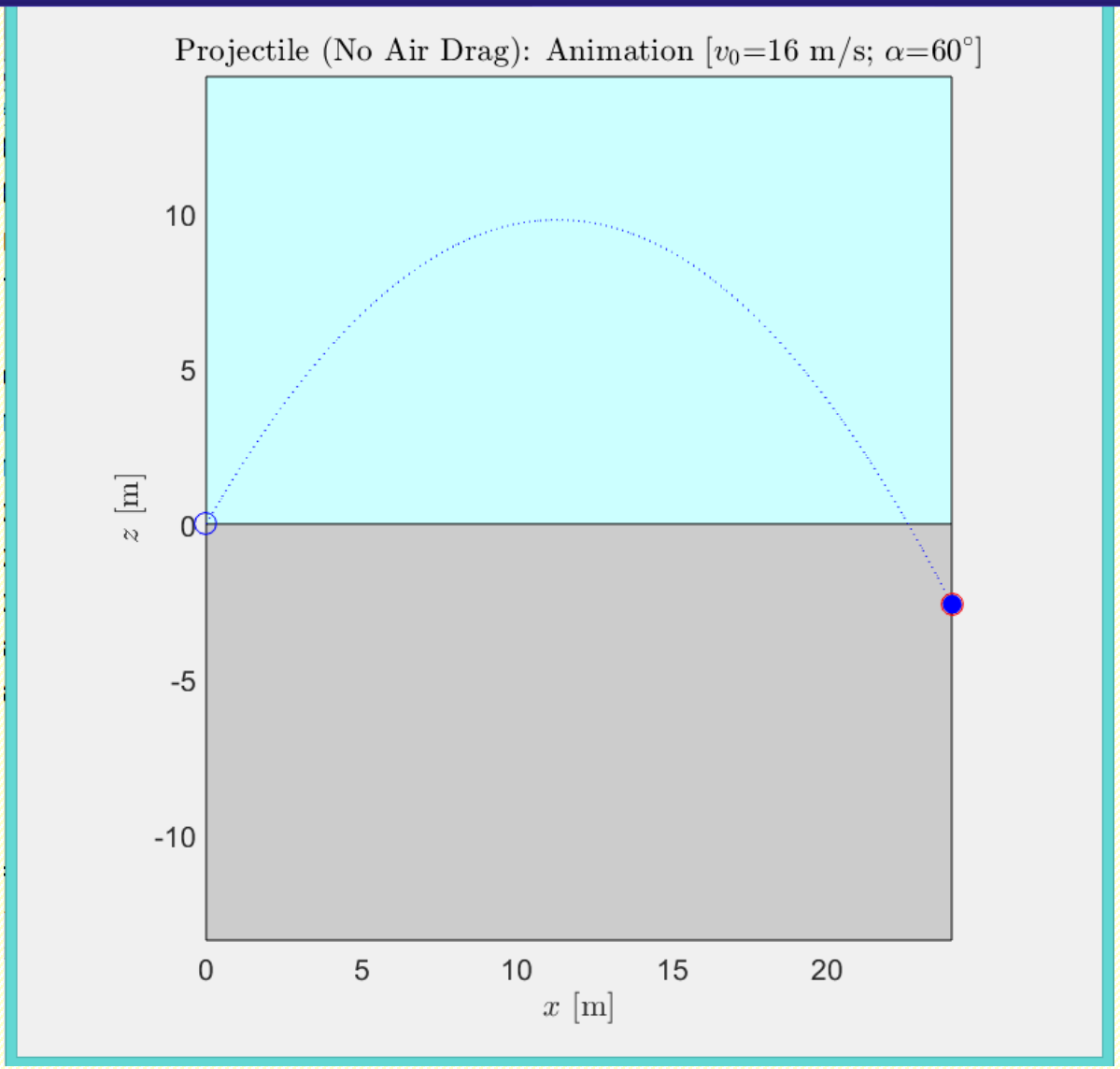


Output of the MATLAB script [0-2 sec]

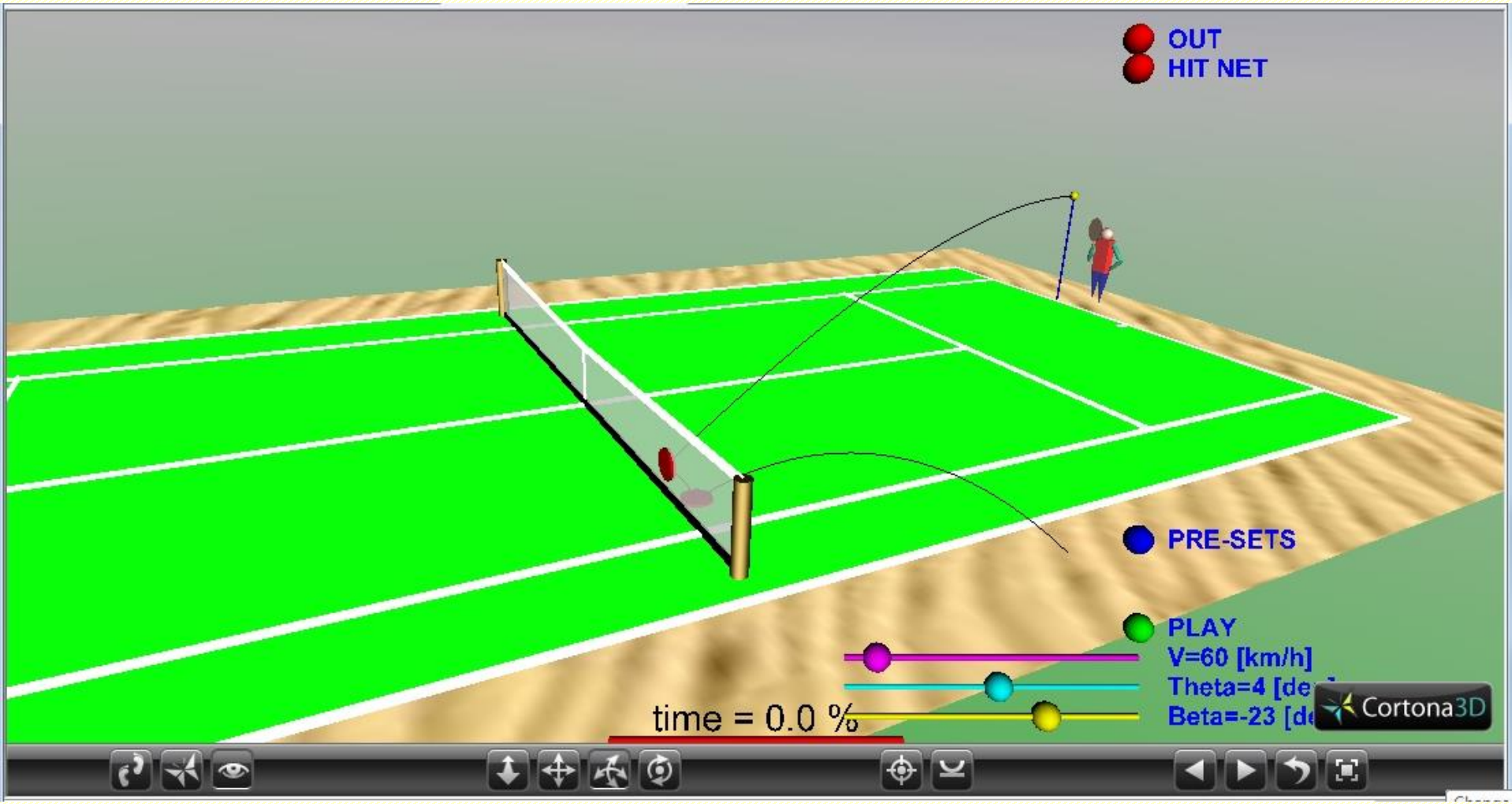


$\theta = 45^{\circ}$

Animation of the Projectile: Advanced Script



Tennis as Projectile Motion: DEMO in VR

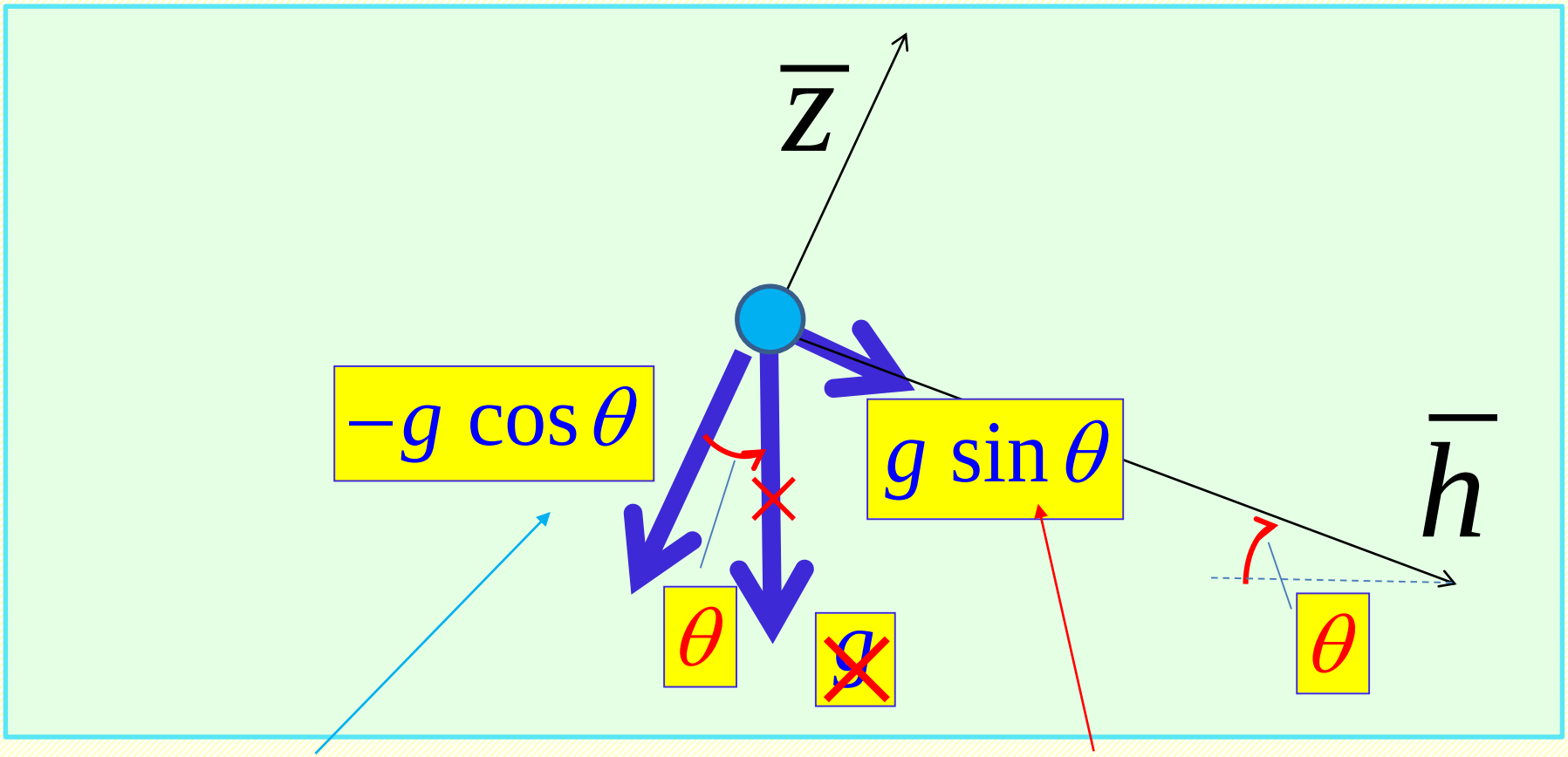


Tennis as Projectile Motion: DEMO in VR

PROJECTILE:

INCLINED CARTESIAN CORDINATE SYSTEM

Hint for Assignment-1: Incline



Contributes to acceleration along $\overline{Z}$

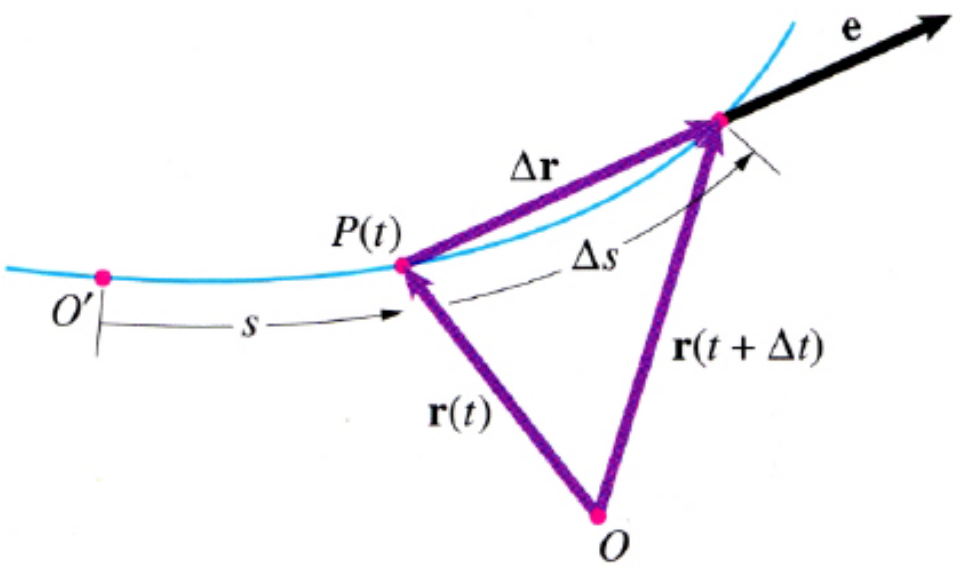
Contributes to acceleration along $\overline{h}$

PROJECTILE:

"NORMAL" & "TANGENTIAL" CORDINATE SYSTEMS

Normal & Tangential Coordinates

Curvilinear Motion: Normal & Tangential Coord



$$\mathbf{v} = v\mathbf{e}_t = \frac{ds}{dt}\mathbf{e}_t$$

$$\mathbf{a} = \frac{d\mathbf{v}}{dt} = \frac{dv}{dt}\mathbf{e}_t + v\frac{d\mathbf{e}_t}{dt}$$

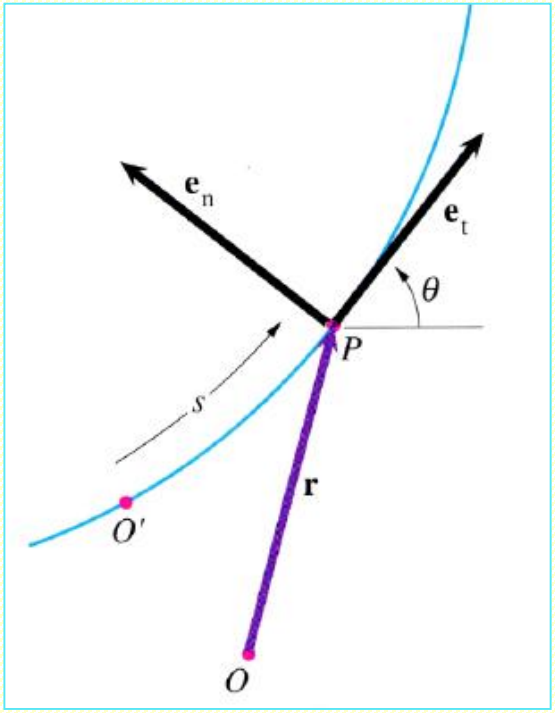
$$\frac{d\mathbf{e}_t}{dt} = \frac{d\theta}{dt}\mathbf{e}_n$$

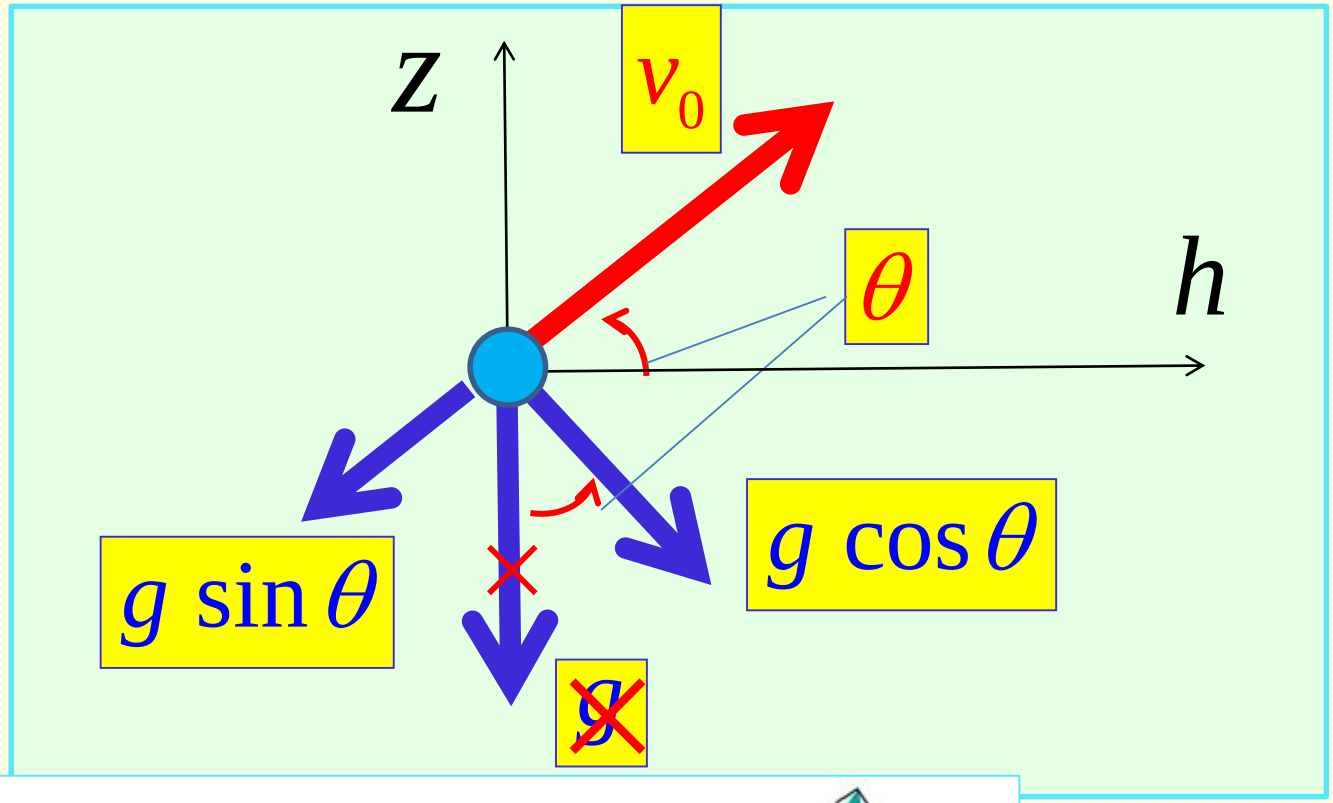
$$\mathbf{a} = \frac{dv}{dt}\mathbf{e}_t + v\frac{d\theta}{dt}\mathbf{e}_n =$$

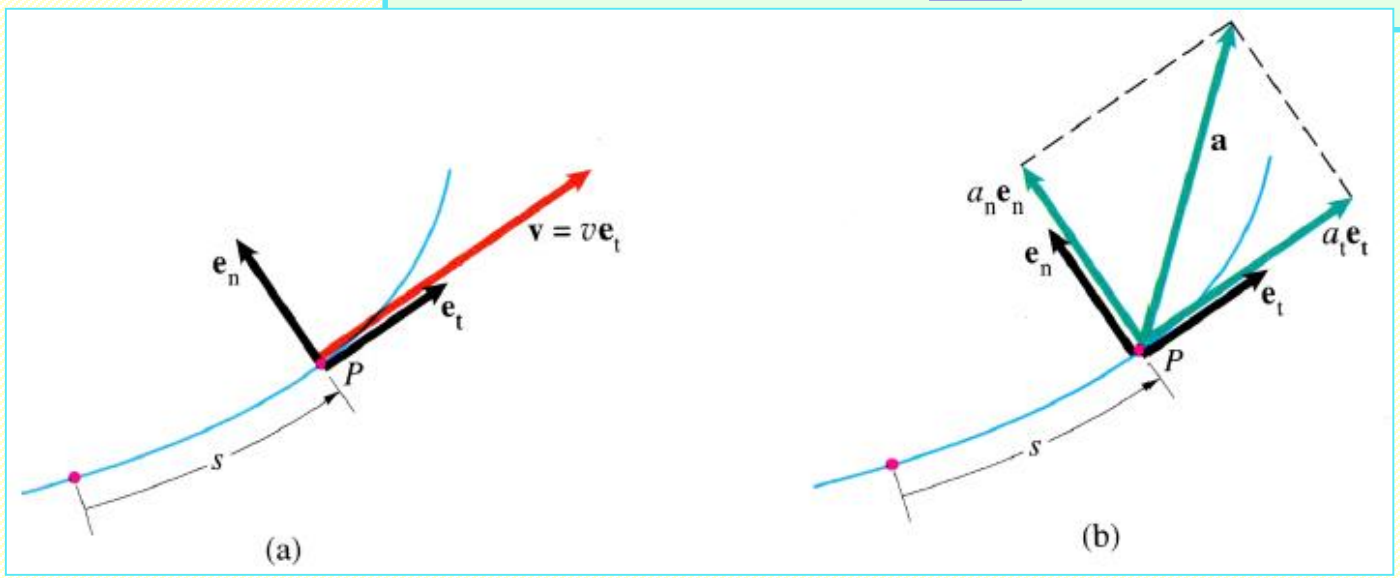
$$= \frac{dv}{dt}\mathbf{e}_t + \frac{v^2}{\rho}\mathbf{e}_n$$

Normal acceleration

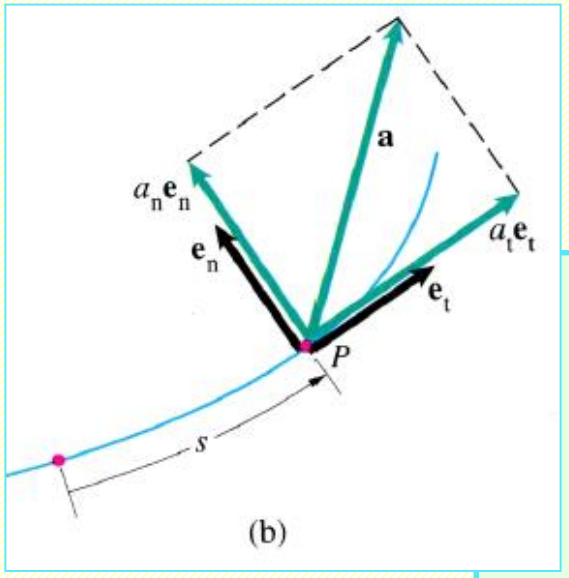
Tangential acceleration

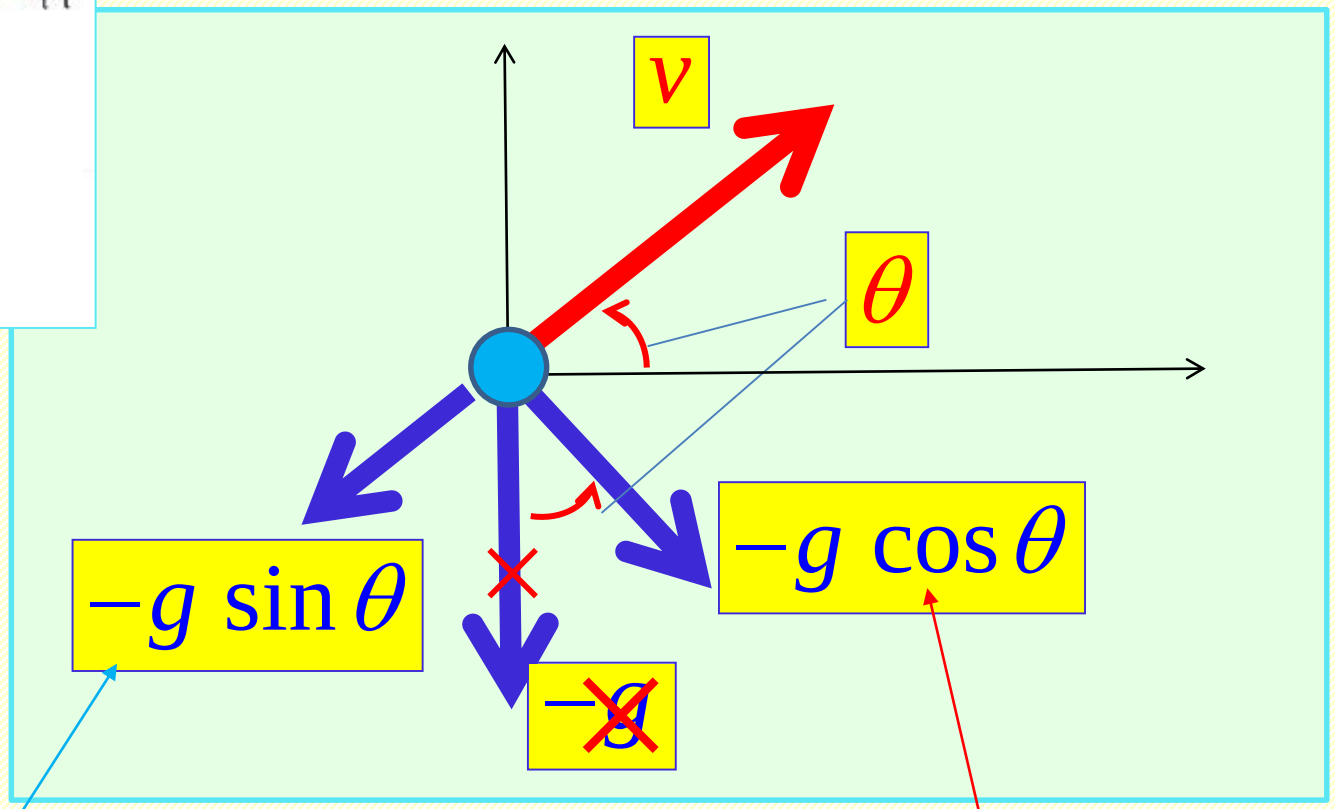

$\mathbf{e}_n$
$\mathbf{e}_t$
θ
P
s
$\mathbf{r}$
O'
O


Z
v_0
θ
h
$g \sin\theta$
$g \cos\theta$
g


$\mathbf{v} = v\mathbf{e}_t$
$\mathbf{e}_n$
$\mathbf{e}_t$
P
s
(a)
$\mathbf{a}$
$a_n\mathbf{e}_n$
$a_t\mathbf{e}_t$
$\mathbf{e}_n$
$\mathbf{e}_t$
P
s
(b)

Hint for Assignment-1: ρ





Contributes to tangential acceleration

$a_t = dv/dt$

Contributes to normal acceleration

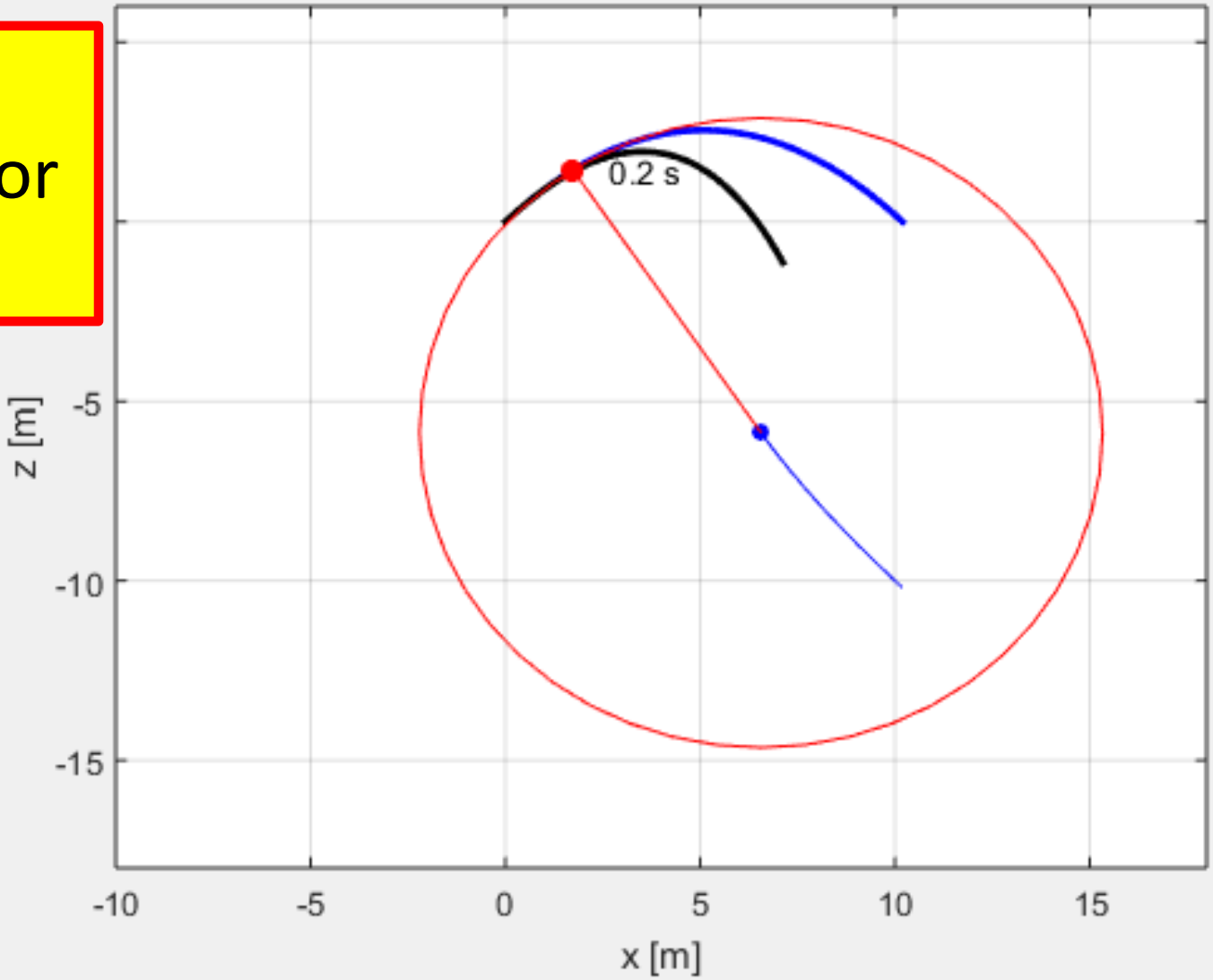
$|a_n| = v^2/\rho$

PROJECTILE:

"NORMAL" & "TANGENTIAL" CORDINATE SYSTEMS: PMT ANIMATION DEMO

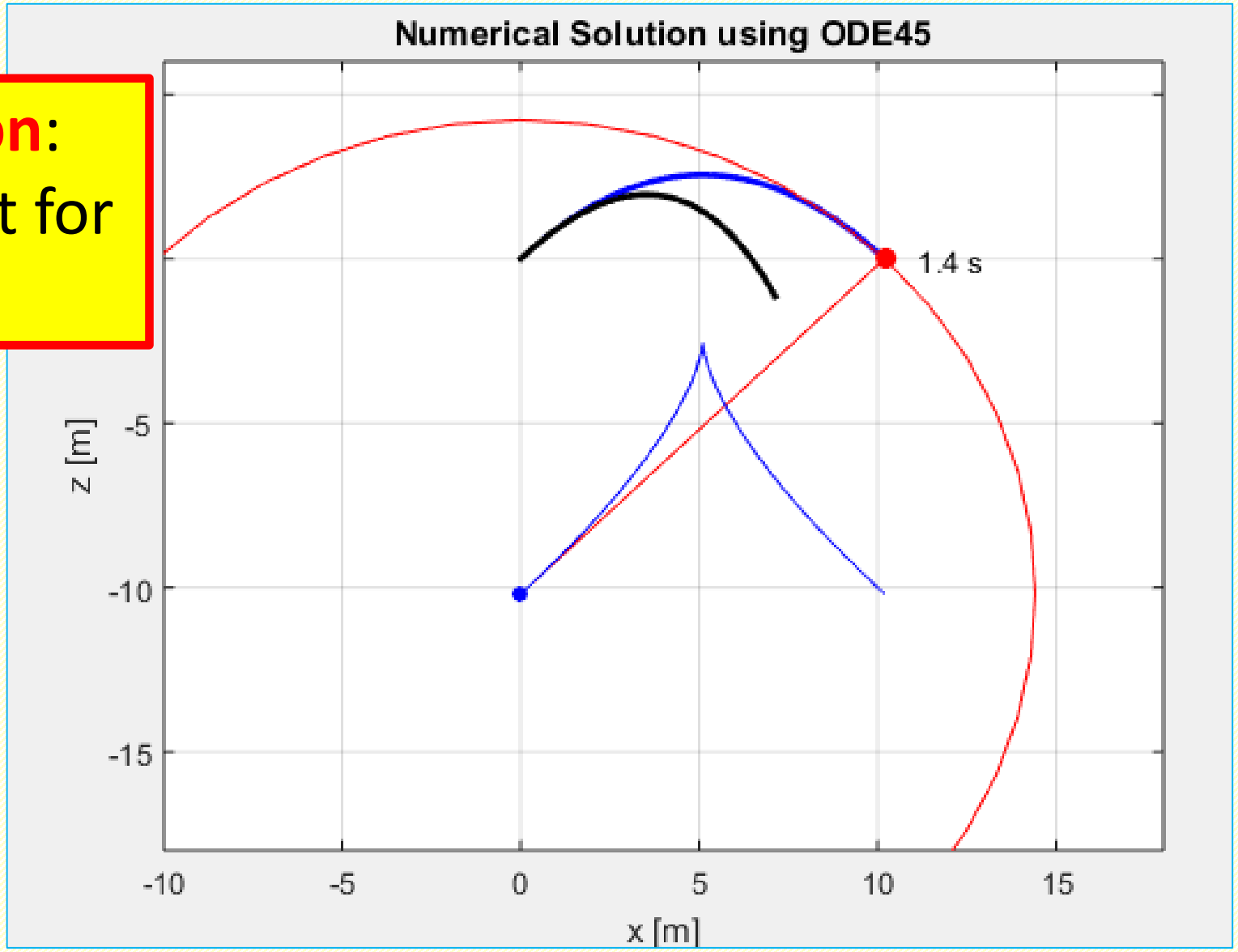
Instantaneous Centre of Rotation

Animation:
Snap shot for
t=0.2 s



Instantaneous Centre of Rotation

Animation:
Snap shot for
t=1.4 s



PROJECTILE:

TWO-FILES SOLUTION

(“Main” & “x_dot” Files)

IMPORTANT COMMENTS:

$\ddot{h} = 0$ $\ddot{z} = -g$

Note: Let Us SELECT FOLLOWING ORDER OF STATES:

1. We always remember definition of states & their selected order !!!

IF (Example-4)

$$\begin{cases} x_1 = h \\ x_2 = \dot{h} \\ x_3 = z \\ x_4 = \dot{z} \end{cases}$$

2. Order of right hand terms depends upon the order of states !!!

THEN

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = 0 \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = -g \end{cases}$$

IMPLICATIONS OF ORDER OF STATES:

Note: IF WE SELECT THE FOLLOWING ORDER OF STATES:

IF (Example-4)

$$\begin{cases} x_1 = h \\ x_2 = \dot{h} \\ x_3 = z \\ x_4 = \dot{z} \end{cases}$$

(1) Vector of Initial Conditions should be:

THEN

$$\mathbf{x}_0 = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_{t=0} = \begin{bmatrix} h_0 \\ v_{0h} \\ z_0 \\ v_{0z} \end{bmatrix}$$

(2) Trajectory of the mass is given by $x_1 - x_3$ data (calculated by ode45)

Solving Projectile Task, using ODE

```
1   %% PMT PROJECTILE SIMPLE EXAMPLE
2   % Designed by Prof P.M.Trivailo
3   % (c) 2013
4   close('all'); clear; clc;
5   global g m coeff
6
7   m=10;
8   g=9.81;
9   v0=10;
10  coeff=0.8; % F=coeff*Vsq;
11  theta=45; % deg
12
13  % Useful Calculation
14  theta=theta*pi/180;
15  vx0=v0*cos(theta); vz0=v0*sin(theta);
16  t_fin=(2*v0/g)*sin(theta);
17  tt=[0:0.01:1]*t_fin;
18  x0=[0; vx0; 0; vz0];
19
20  % Solving Task using MATLAB ODE Integration
21  [t_out,x_out] = ode45('projectile_xdot',tt,x0);
22  [t_out2,x_out2] = ode45('projectile_xdot_v_sq',tt,x0);
```

Solving Projectile Task, using ODE

Note:
This is an “x_dot” file for “no air-resistance” case!

```
1   function [x_dot] = projectile_xdot(t,x)
2   global g
3
4   % x(1) - x;
5   % x(2) - vx;
6   % x(3) - y;
7   % x(4) - vy;
8   %---------------
9   x_dot(1,1) = x(2);
10  x_dot(2,1) = 0;
11  x_dot(3,1) = x(4);
12  x_dot(4,1) = -g;
```

PROJECTILE:

APPLICATIONS & DEMOS

ADVANCED EXAMPLE: Projectile Towards Static Cable

with prompting photos

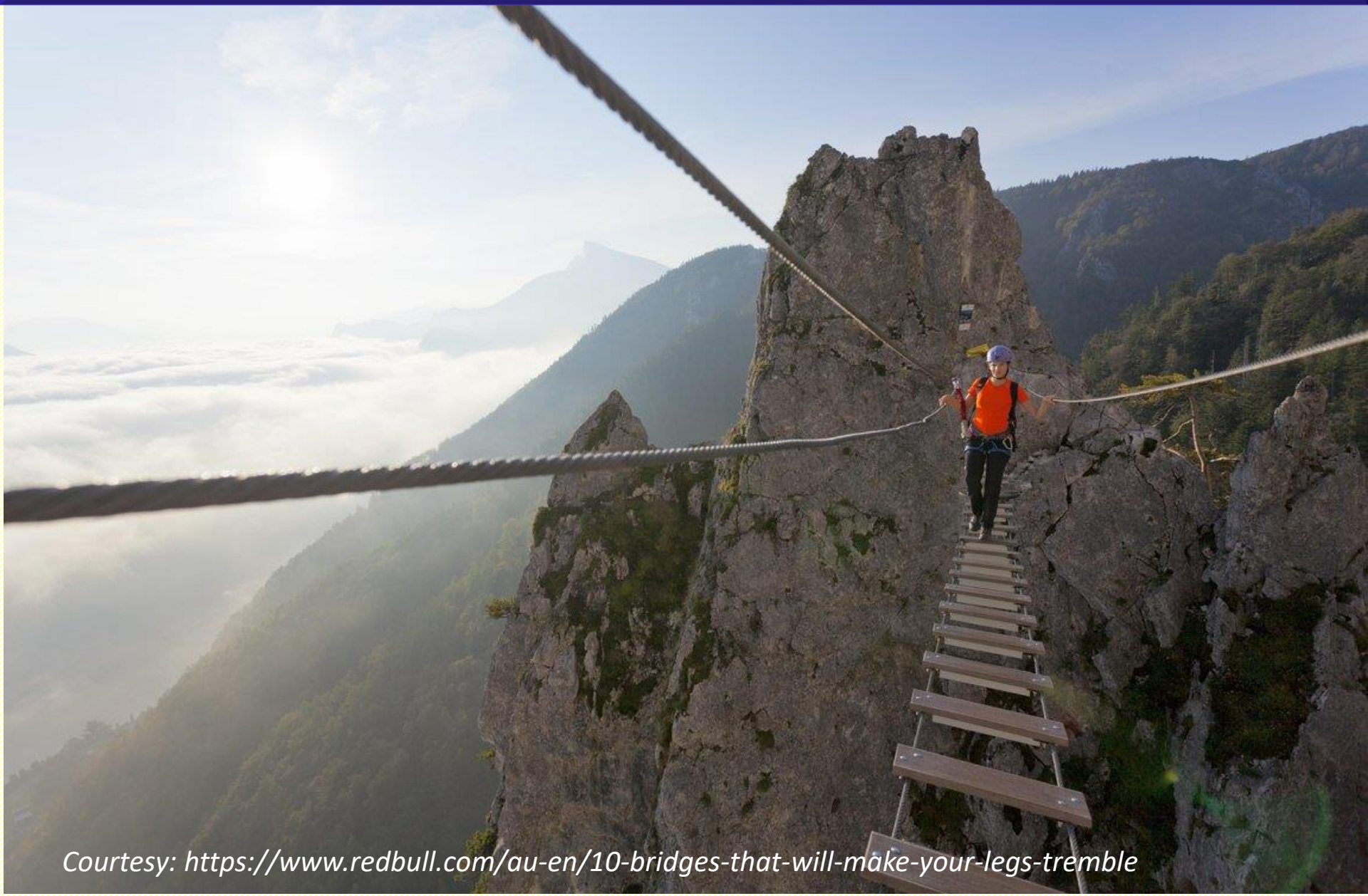
Cable System: Man, crossing Mekong River, China, on a high wire

Courtesy: https://www.redbull.com/au-en/10-bridges-that-will-make-your-legs-tremble

Cable System: Sky walking, Mt Nimbus, USA

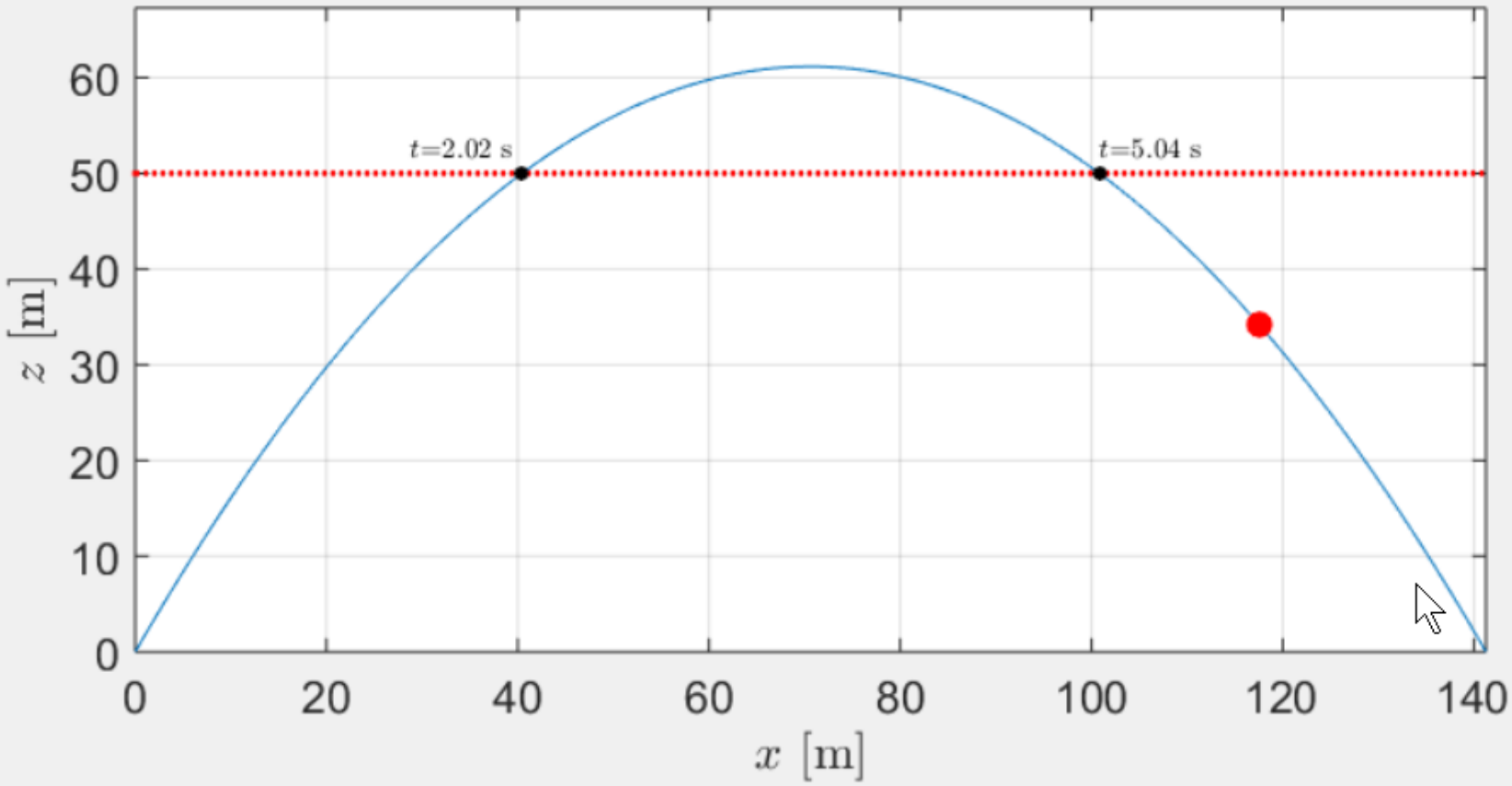
Courtesy: https://www.redbull.com/au-en/10-bridges-that-will-make-your-legs-tremble

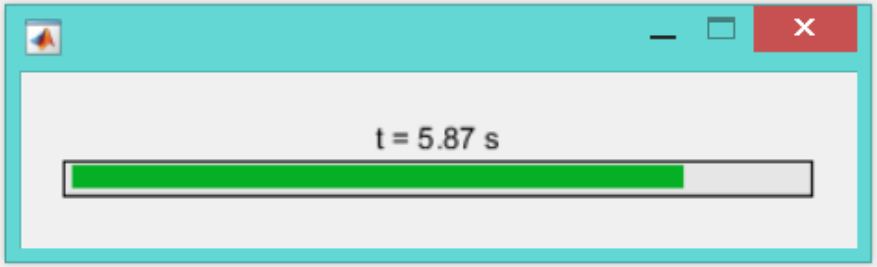
Cable System: High rope bridge in Austria



Courtesy: https://www.redbull.com/au-en/10-bridges-that-will-make-your-legs-tremble

ANIMATED SIMULATION (MATLAB):

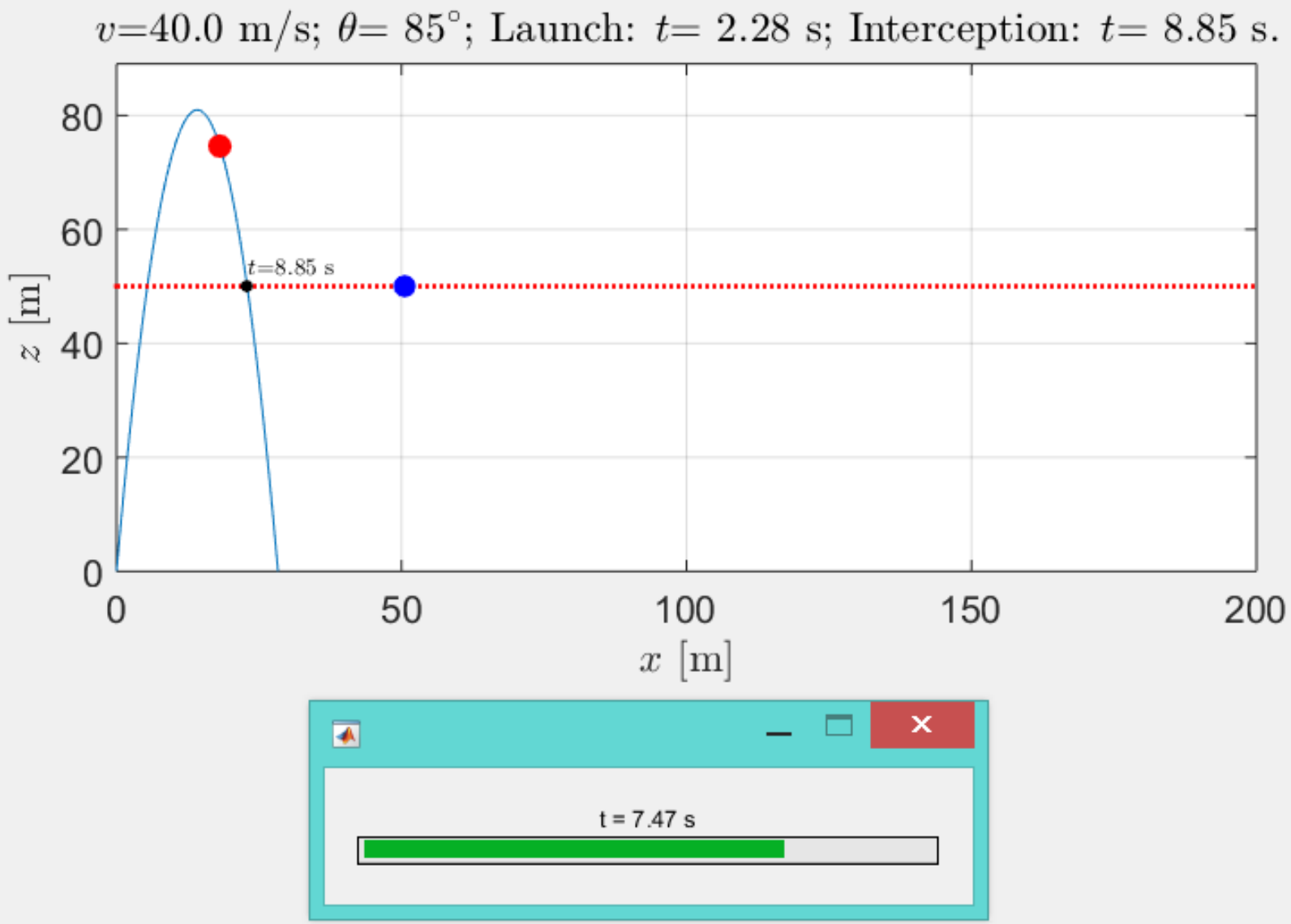




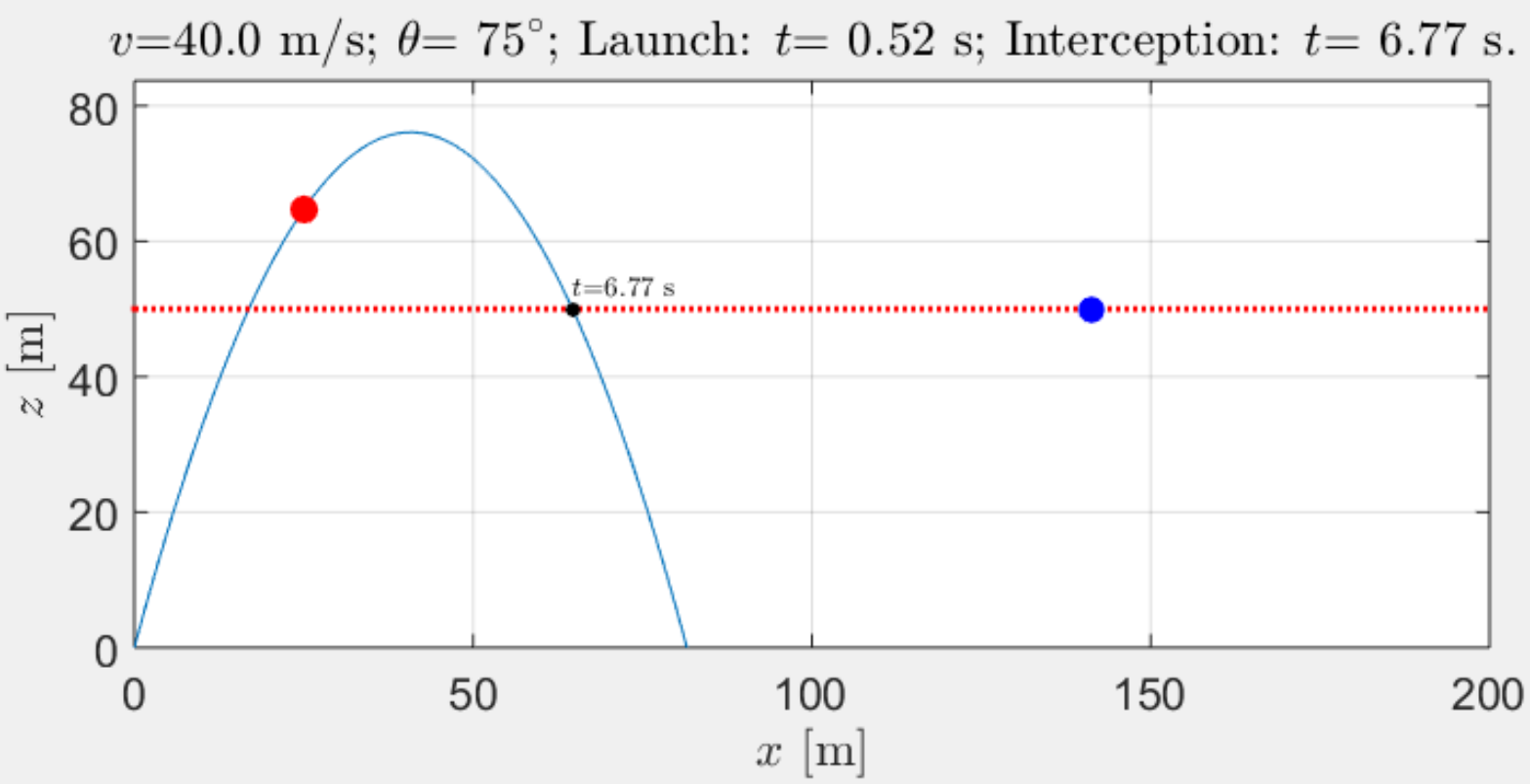
ADVANCED EXAMPLE: Projectile Towards Moving Drone

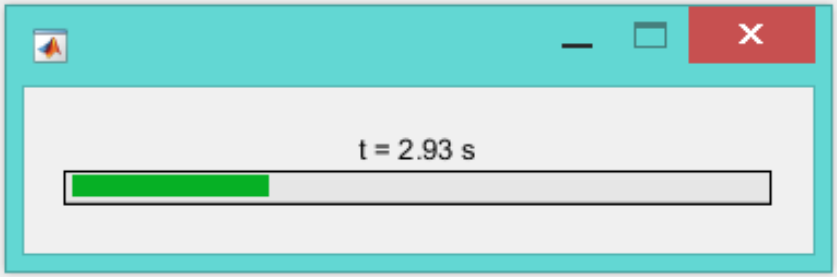
with prompting photos

ANIMATED SIMULATION (MATLAB):

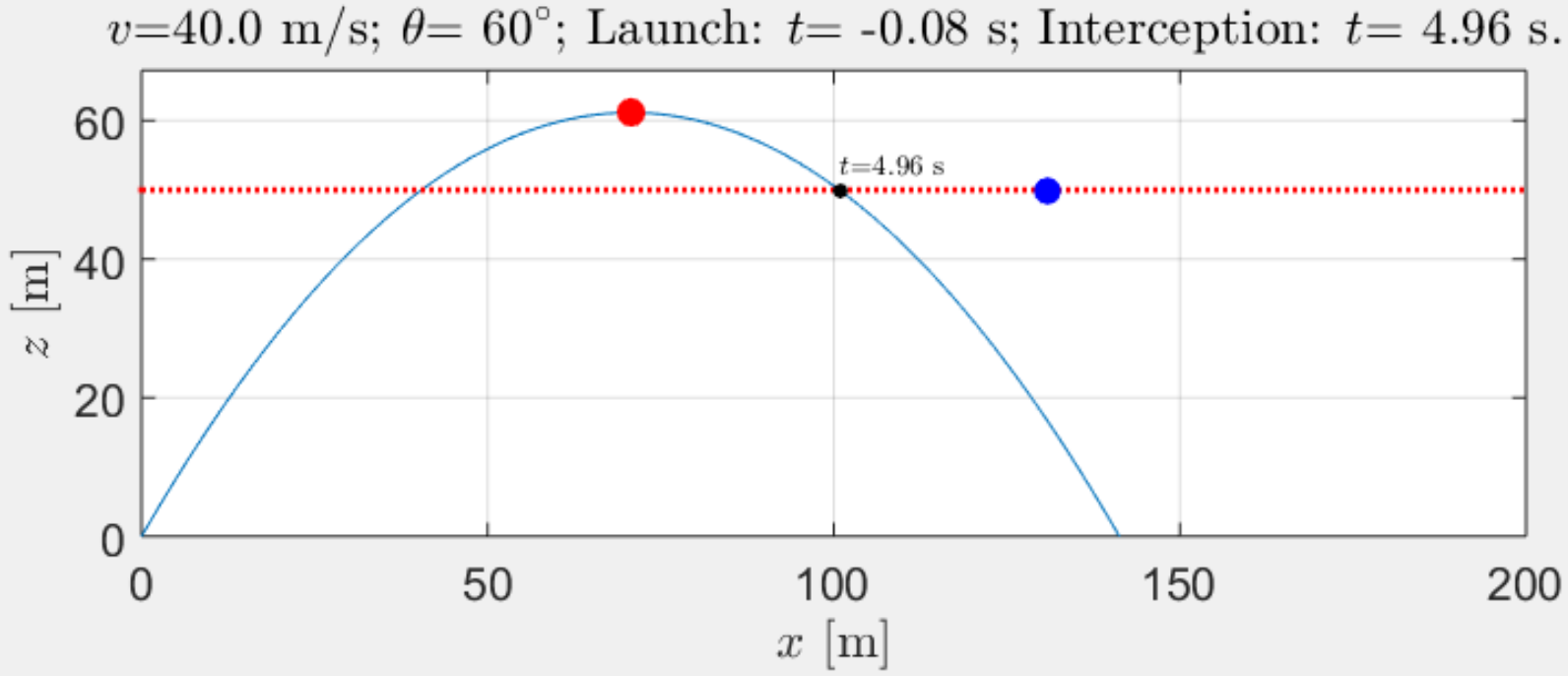


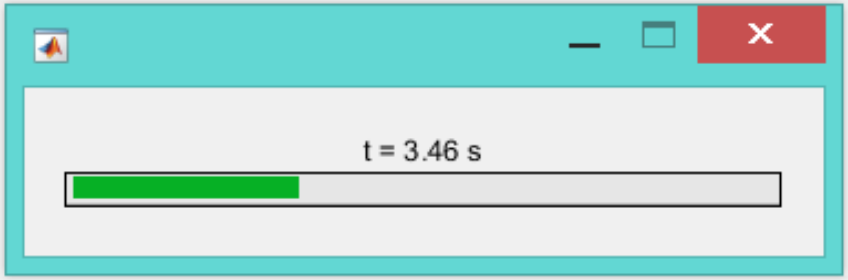
ANIMATED SIMULATION (MATLAB):





ANIMATED SIMULATION (MATLAB):

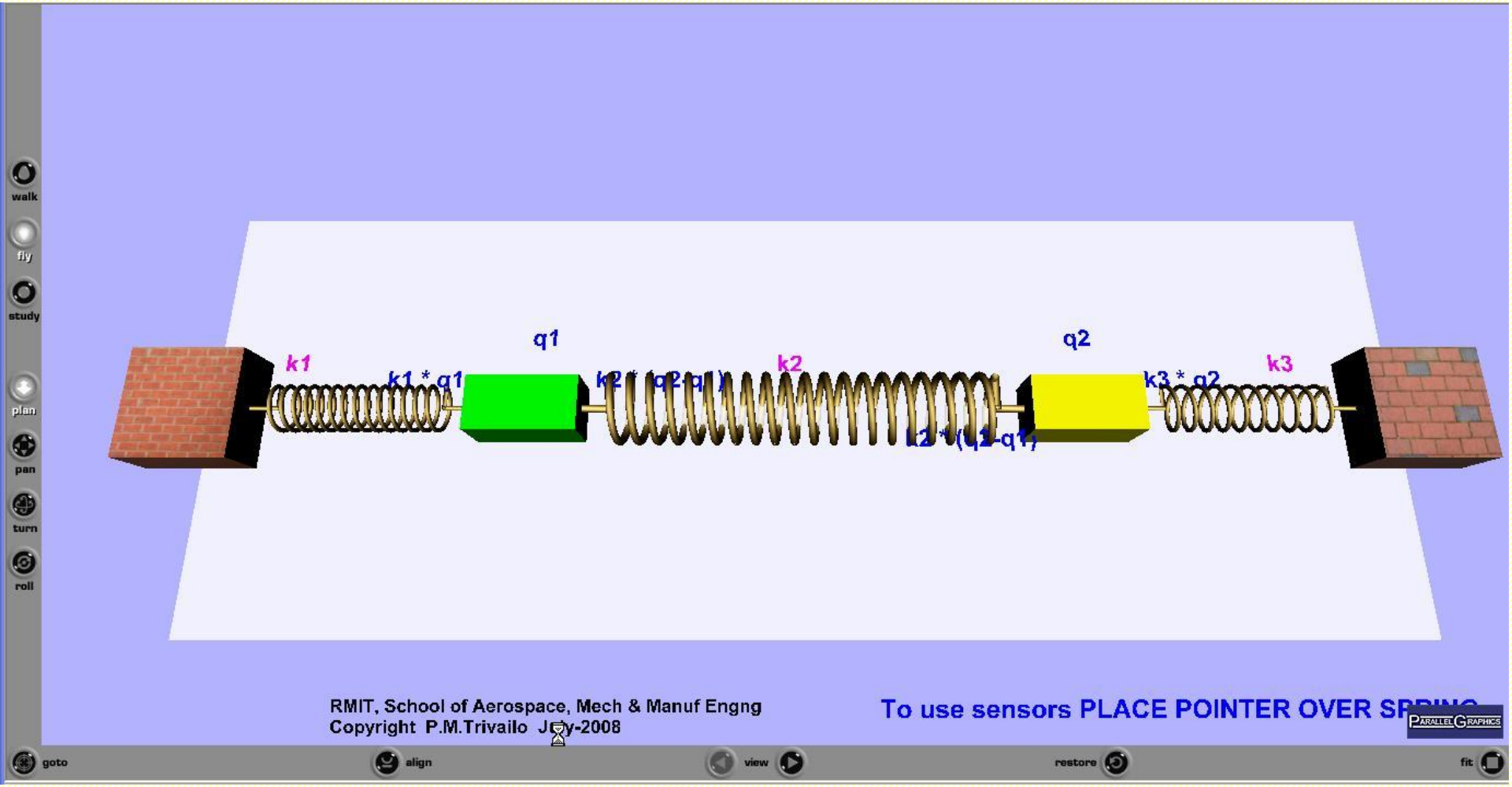


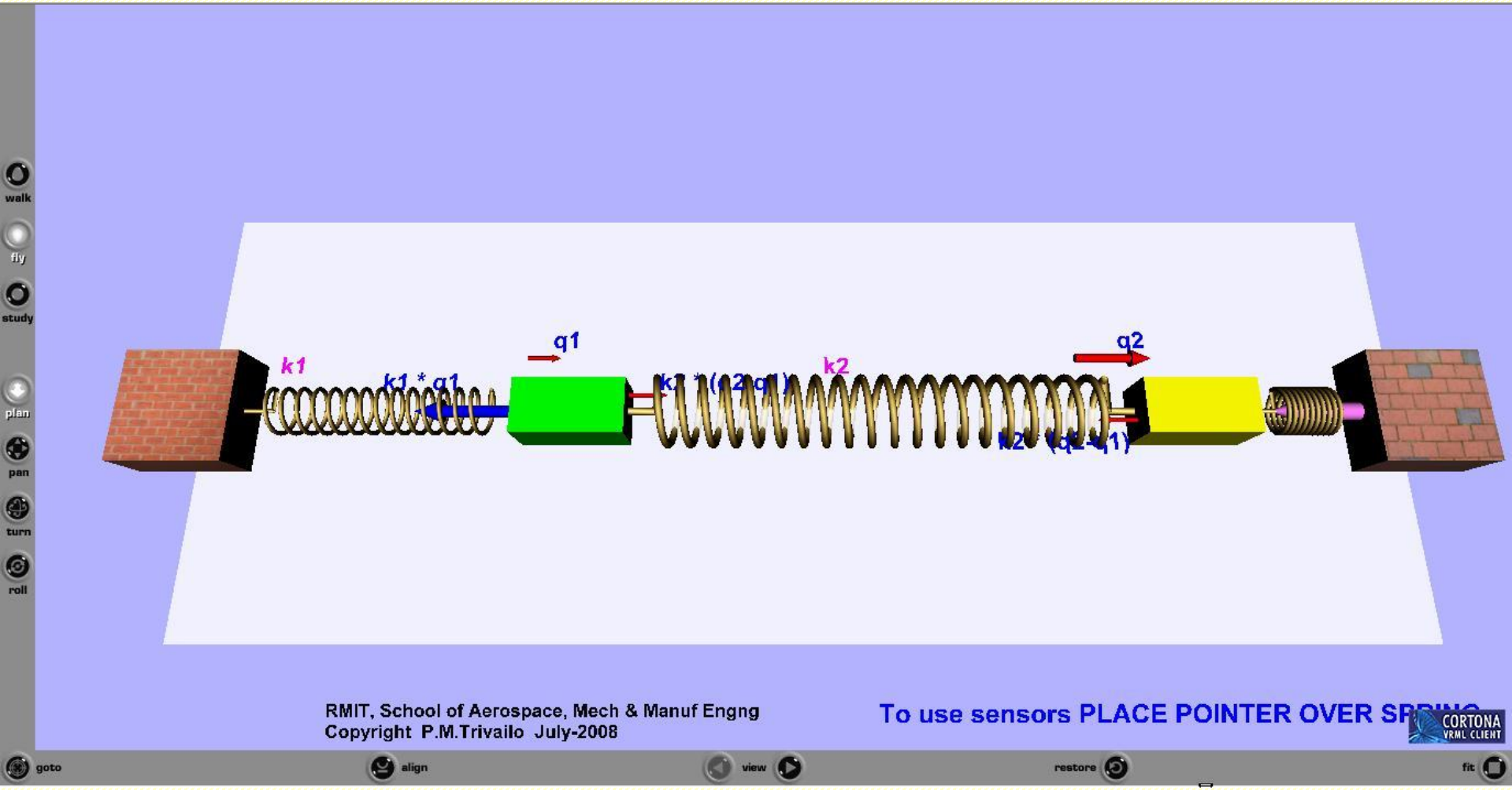


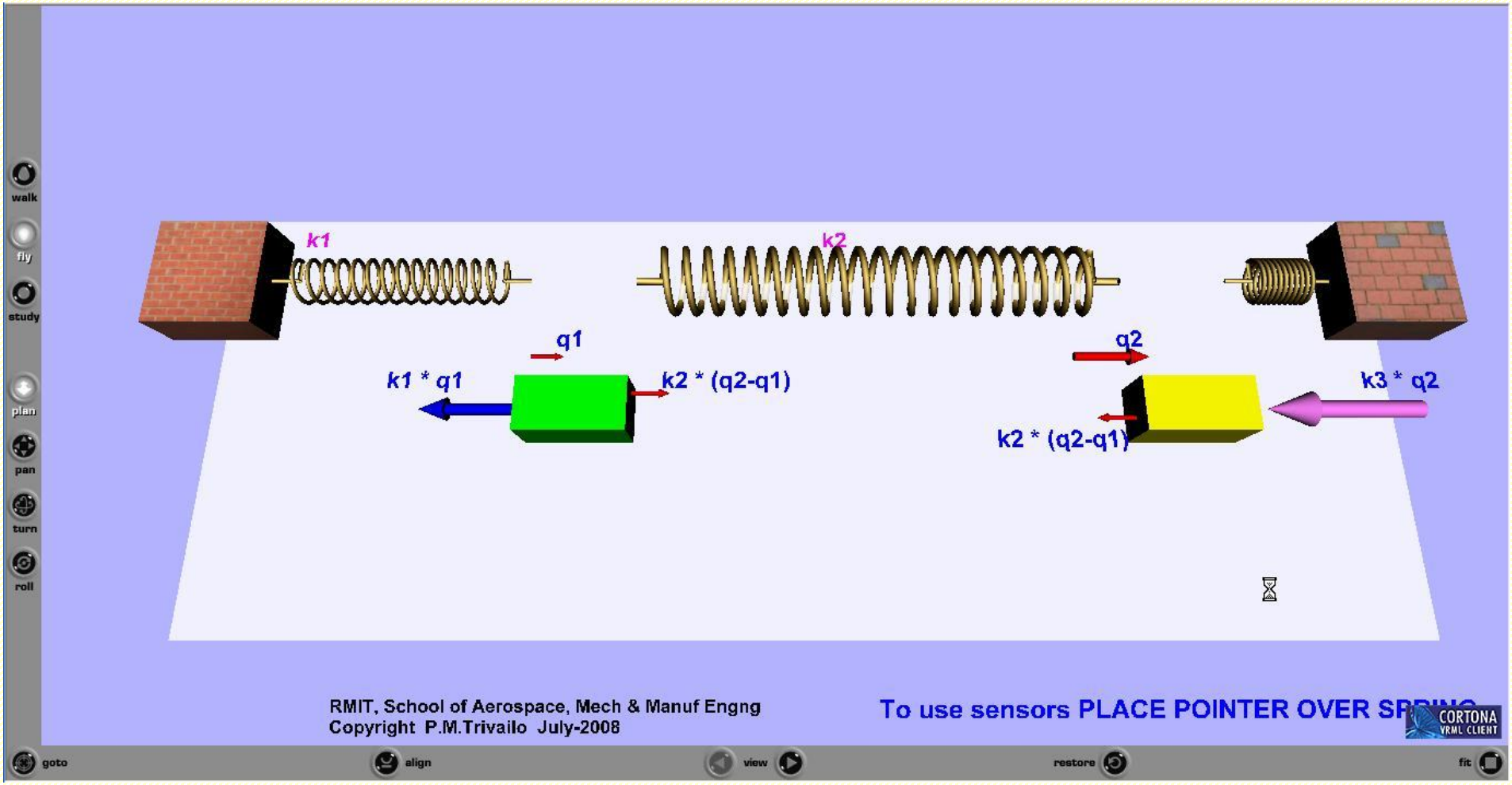
Multi-DOF mass-spring systems:

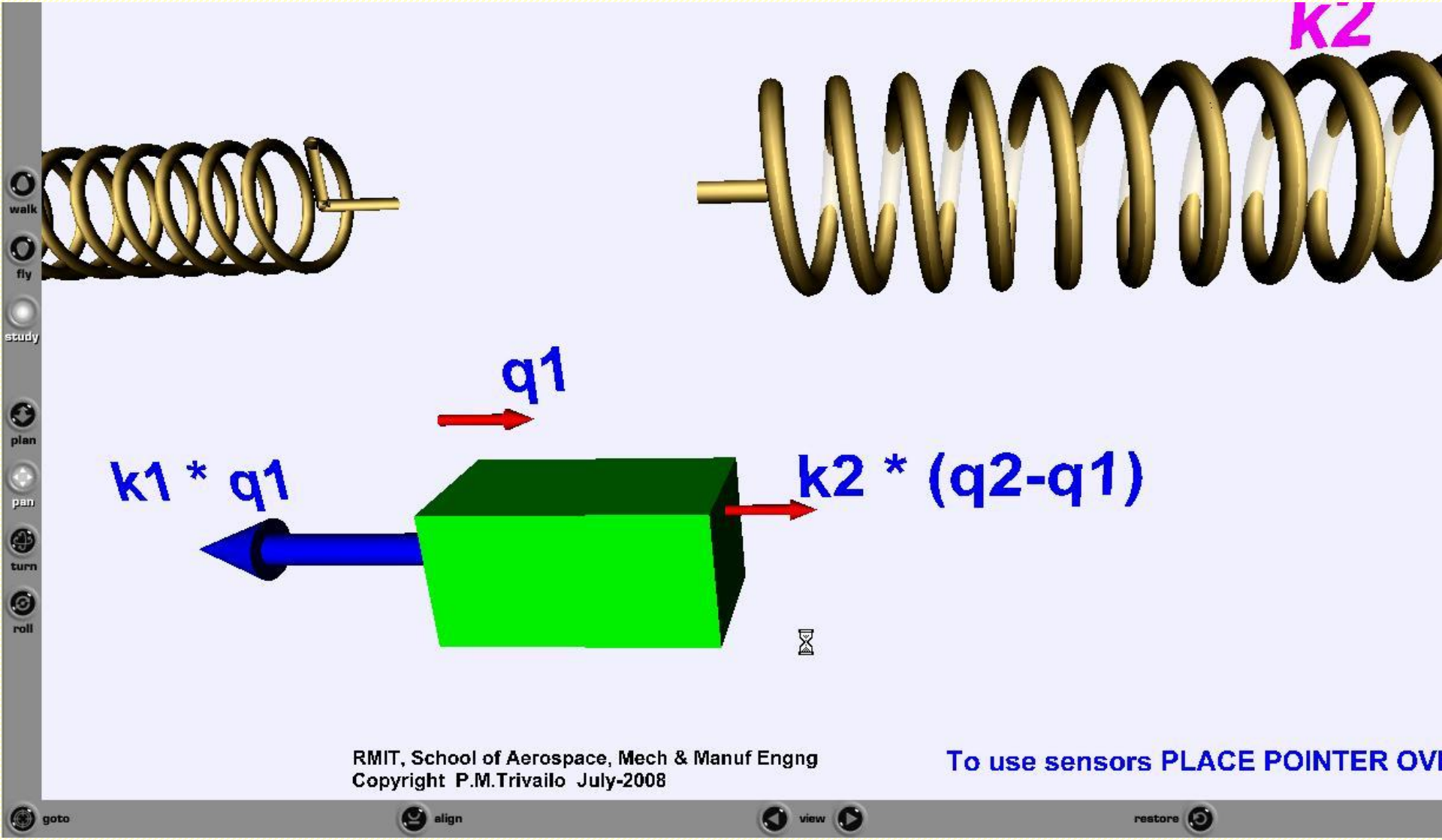
Derivation of EOM using Newton's Second Law & Free-Body Diagrams (FBDs).

Demo in Virtual Reality

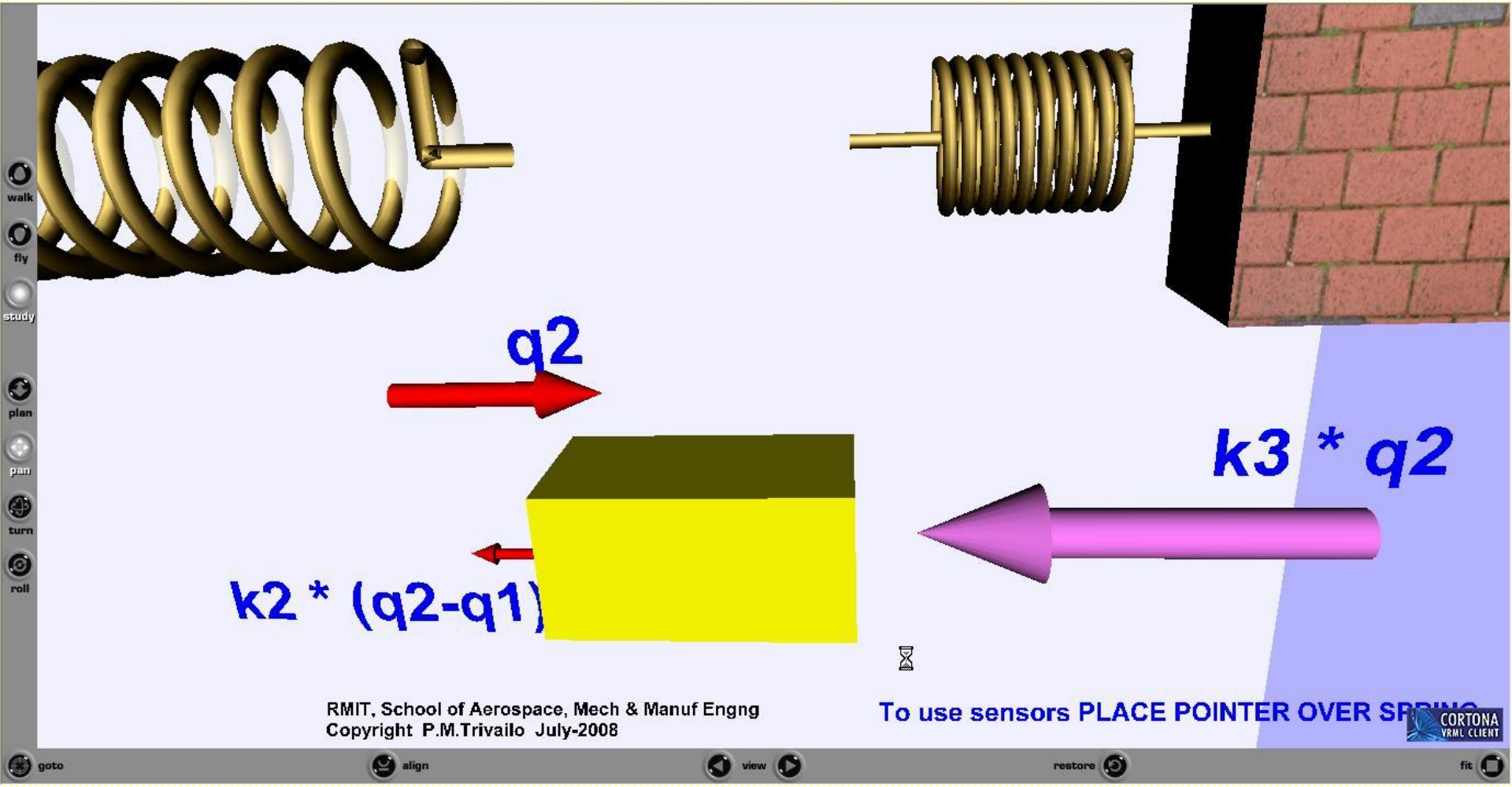



walk
fly
study
plan
pan
turn
roll
k1
k1 * q1
q1
k2
q2
RMIT, School of Aerospace, Mech & Manuf Engng
Copyright P.M.Trivailo July-2008
To use sensors PLACE POINTER OVER SP
CORTONA VRML CLIENT
goto
align
view
restore
fit

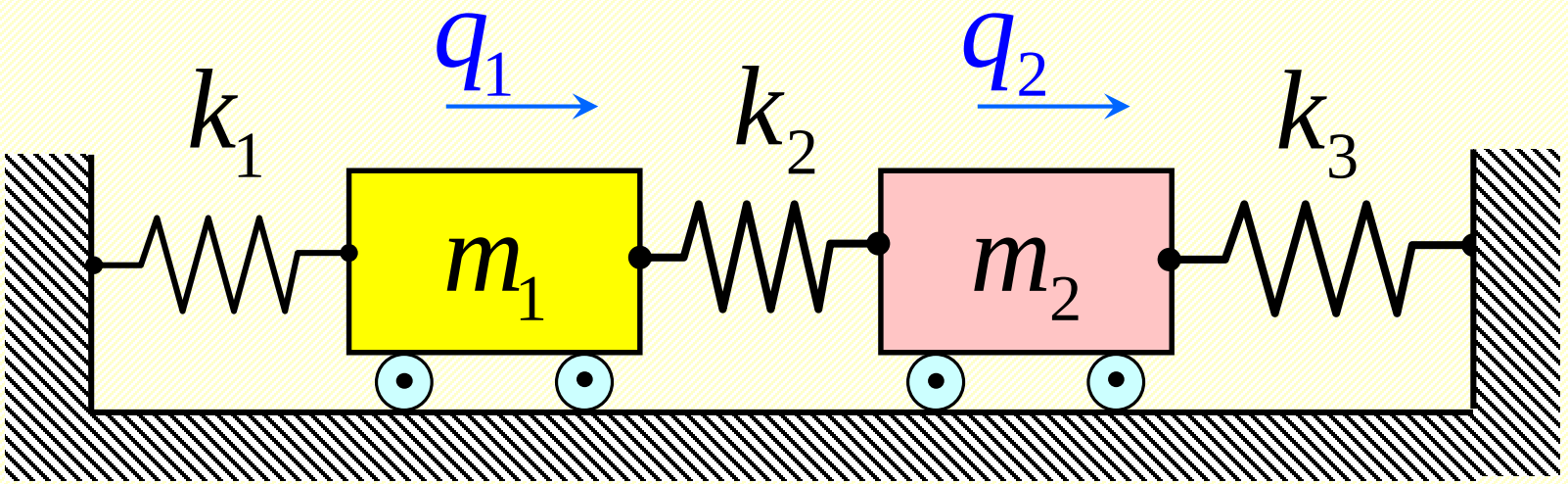

walk
fly
study
plan
pan
turn
roll
k1
k2
q1
k1 * q1
k2 * (q2-q1)
q2
k3 * q2
k2 * (q2-q1)
RMIT, School of Aerospace, Mech & Manuf Engng
Copyright P.M.Trivailo July-2008
To use sensors PLACE POINTER OVER SP
CORTONA VRML CLIENT
goto
align
view
restore
fit



FBD-1: $m_1 \ddot{q}_1 = -k_1 q_1 + k_2 (q_2 - q_1)$



FBD-2: $m_2 \ddot{q}_2 = -k_2(q_2 - q_1) - k_3 q_2$



$$\begin{cases} m_1\ddot{q}_1 + (k_1 + k_2)q_1 - k_2 q_2 = 0 \\ m_2\ddot{q}_2 - k_2 q_1 + (k_2 + k_3)q_2 = 0 \end{cases}$$

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \begin{Bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{Bmatrix} + \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 + k_3 \end{bmatrix} \begin{Bmatrix} q_1 \\ q_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$

$$[m] = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}; \quad [k] = \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 + k_3 \end{bmatrix}$$

EQUATIONS IN STATE-SPACE FORM

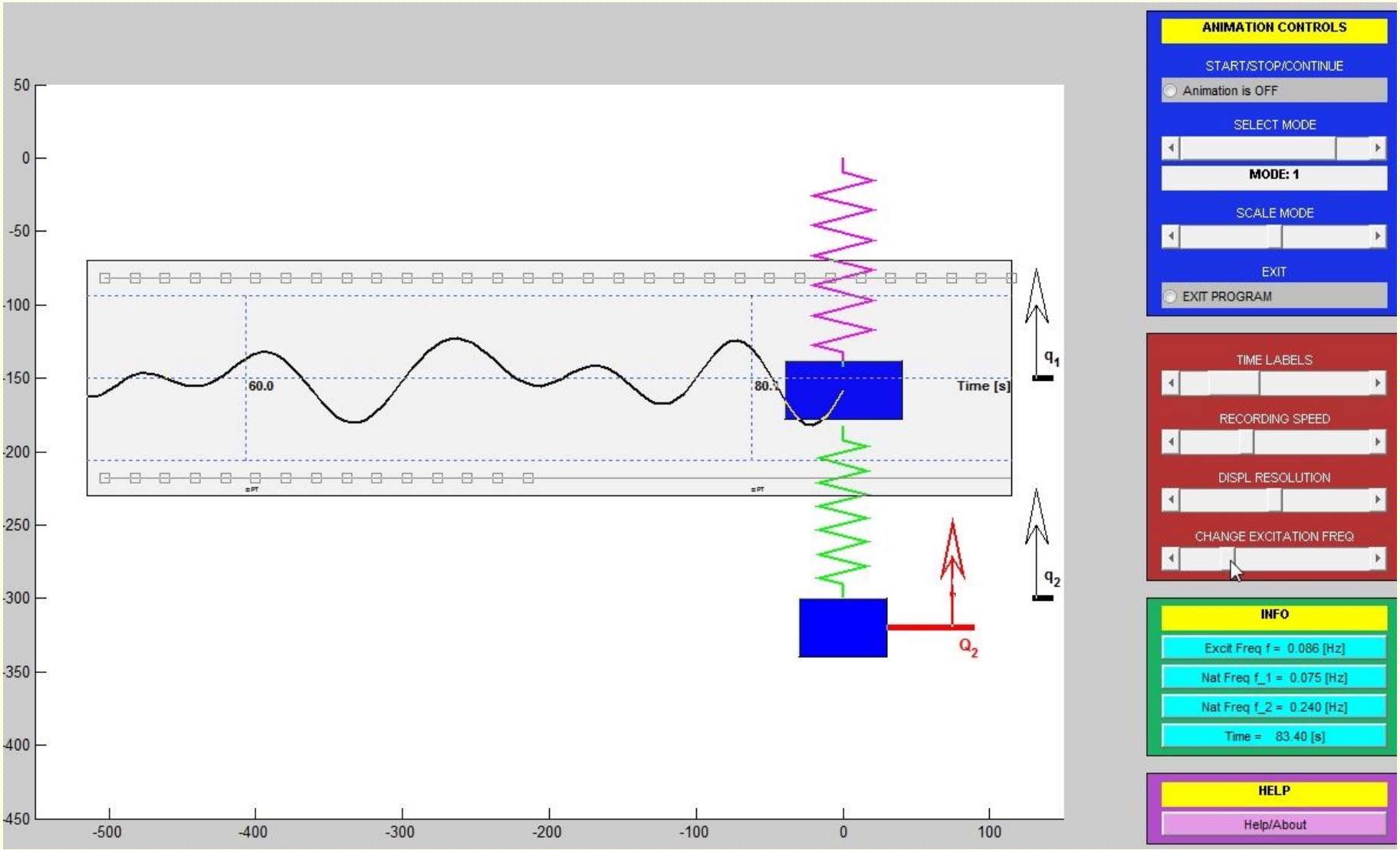
$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

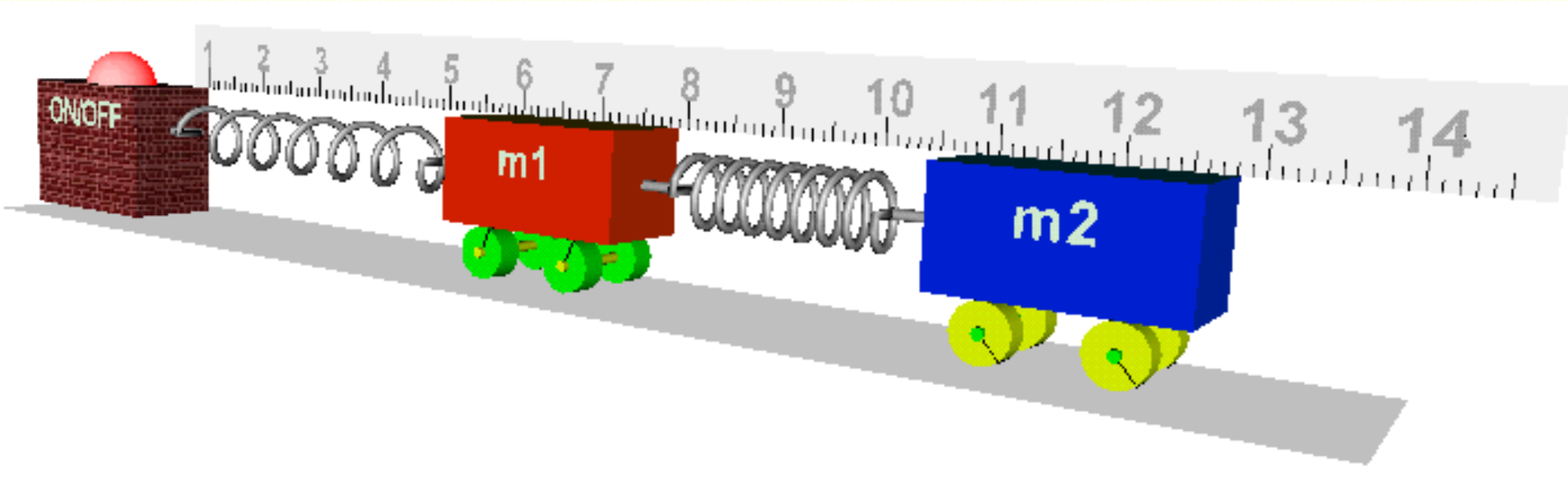
$$A = \left[\begin{array}{c|c} [0] & [1] \\ \hline -[m]^{-1} * [k] & -[m]^{-1} * [c] \end{array}\right]; \quad B = \left[\begin{array}{c} [0] \\ \hline [m]^{-1} \end{array}\right]$$

SOLVING RESPONSES FOR MULTI-DOF SYSTEMS:

MATLAB and ODExxx


ANIMATION CONTROLS
START/STOP/CONTINUE
Animation is OFF
SELECT MODE
MODE: 1
SCALE MODE
EXIT
EXIT PROGRAM
TIME LABELS
RECORDING SPEED
DISPL RESOLUTION
CHANGE EXCITATION FREQ
INFO
Excit Freq f = 0.086 [Hz]
Nat Freq f_1 = 0.075 [Hz]
Nat Freq f_2 = 0.240 [Hz]
Time = 83.40 [s]
HELP
Help/About
60.0
Time [s]
q1
q2
Q2
50
0
-50
-100
-150
-200
-250
-300
-350
-400
-450
-500
-400
-300
-200
-100
0
100

Virtual Laboratory



Virtual Laboratory

Solve Eigenvalue Problem using MATLAB

MATLAB IN VIBRATION ANALYSIS

MATLAB Command to Calculate

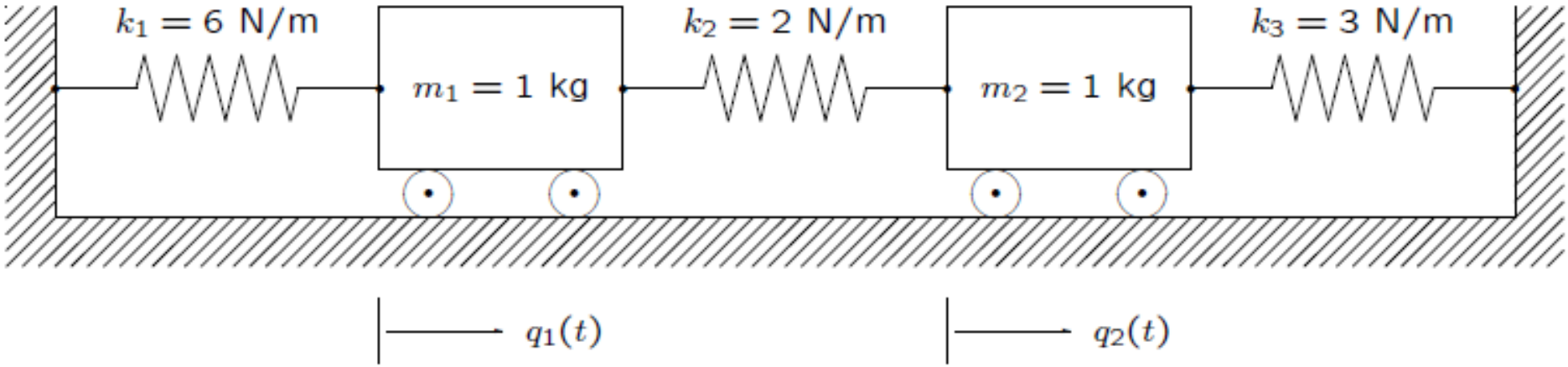
Eigenvalues and Eigenvectors of the System:

$$[U, D] = eig(\ inv(m)\ *\ k\ )$$

or

$$[U, D] = eig(\ k\ ,\ m\ )$$

Numerical Example

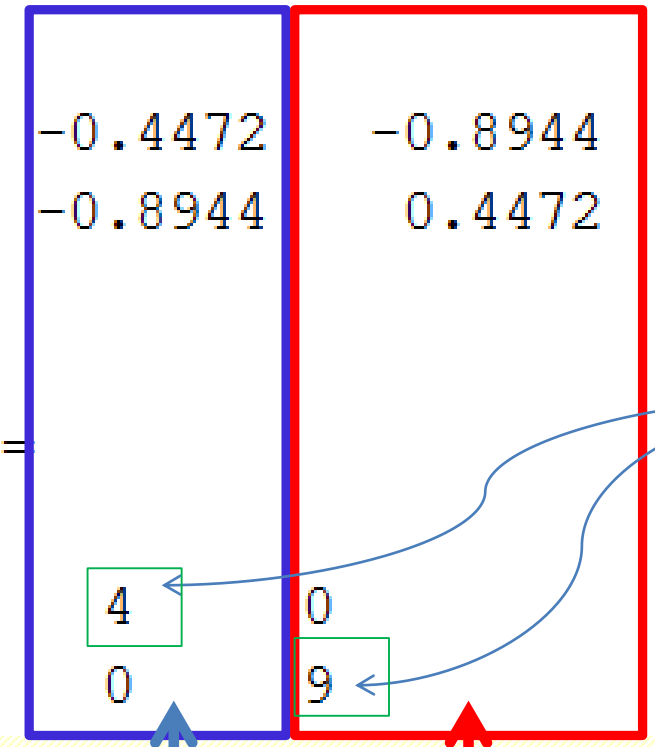


$$[m] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad [k] = \begin{bmatrix} 8 & -2 \\ -2 & 5 \end{bmatrix};$$

```
m1=1; m2=1; k1=6; k2=2; k3=3;
m=[1 0; 0 1];
k=[k1+k2 -k2; -k2 k2+k3];
[U,D] = eig(k,m);
```

Command Window

```
>> m1=1; m2=1; k1=6; k2=2; k3=3;
>> m=[m1 0; 0 m2]; k=[k1+k2 -k2; -k2 k2+k3];
>> [U,D]=eig(k,m)

U =

   -0.4472   -0.8944
   -0.8944    0.4472


D =

    4     0
    0     9
```



EIGENVECTORS (or 'MODE SHAPES')

EIGENVALUES

MODE-1

MODE-2

END OF LECTURE-2a SLIDES