

RMIT University

OENG-1116: MODELLING & SIMUALTION

OF ENGINEERING SYSTEMS

Week 2: Part-2

Simplified Introduction into FEM.

Simulation of Multi-DOF Systems.

Simulation of Non-Linear 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**”:

The screenshot shows the RMIT Canvas course materials page for course ID 49609. The left sidebar includes icons for Account, Dashboard, Courses (selected), Calendar, Inbox (1), Arc, Commons, and Help. The main content area features a dark blue header box with the text "Course Welcome and Orientation" and a subtitle: "Get started with this course here and stay up to date with the support that is available." Below this are six boxes representing course weeks:

- Weeks 1-2:** Modelling of Deterministic Systems, using ODE. (Blue border)
- Weeks 3-4:** Solving ODE, using MATLAB & SIMULINK. (Green border)
- Weeks 5-6:** Modelling of Engineering Systems using FEM (Structural Dynamics Applications).
- Weeks 7-8:** Non-Deterministic Systems. Introduction into Neural Networks.
- Weeks 9-10:** Introduction into Supervised Machine Learning and Support Vector Machine (SVM).
- Weeks 11-12:** Non-Linear Regression.

At the bottom right is an orange "Assessments" box with the text "See the complete list here".

CANVAS Lecture Notes :

On the **Canvas**, under “Home” -> “Weeks 1 and 2”:



Weeks 3-4: Solving ODE, using MATLAB & SIMULINK.

Introduction to the weeks 3-4

You will be provided with a comprehensive review of the relevant useful capabilities of MATLAB and SiMULINK environments. In particular, methods of solving ordinary differential equations (ODE) will be presented in details. Review of advanced graphics and animation techniques will be illustrated on the examples, most relevant to the Assignment-1.

Links to the learning objectives

This module will contribute to following Course Learning Objectives:

- Ability to characterise a given engineering system in terms of its essential elements, that is, purpose, parameters, constraints, performance requirements, subsystems, interconnections and environmental context.
- Ability to develop a modelling strategy for a real world engineering system, which considers prediction and evaluation against design criteria, and integrates any required sub-system models.

Purpose

By completing the essential set text reading and reviewing the recorded lecture to ensure you have understood the key concepts, you will be able to complete work on Assignment 1.

∴ Note: RMIT Log-in is essential, i.e. you can access the following documents, if you are currently logged in into your RMIT Google account, and NOT private account (like johnny1234@gmail.com).

* [Week-3 Lecture Notes](#)

* [Week-4 Lecture Notes](#)

ASSIGNMENT-1, Pt1

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

Deadline for A1, Pt1:

The deadline for the submission of the
Individual Variants Assignment-1, Pt1 (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 (OENG1116) will
be sent via your Student Email.

**PLEASE, IMMEDIATELY REPORT TO ME (via email)
IF YOU DID NOT RECEIVE THIS EMAIL FROM ME.**

COMMENTS ON FINDING Assignment-1 GOOGLE FORM

Delete Spam Block ...

OENG1116 Assignment-1 2019



Google Forms <forms-receipts-noreply@google.com>
Sun 24/03/2019 01:01
Pavel Trivailo ▾

Google Forms

- Inside this email, there is an “**EDIT RESPONSE**” link:

- To find this email in your mailbox, you can use search for “**OENG1116 Assignment-1 2019**” string.

Thanks for filling in [OENG1116 Assignment-1 2019](#)

Here's what we've received from you:

[EDIT RESPONSE](#)

OENG1116 Assignment-1 2019

(1) This Individual Assignment is due on Friday, 05-April-2019, 23:59. Submit electronically.

(2) Total weighting of this Assignment: 35%. Attempt ALL tasks. Questions have equal 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.

Your email address (pavel.trivailo@rmit.edu.au) was recorded when you submitted this form.

SELECT YOUR NAME FROM THE LIST BELOW:

Important: (1) Students are listed alphabetically, with Surname shown first; (2) Submit only your own Assignment and never register someone's Assignment. If assignments would be mixed up! Penalties may be applied to the offenders.

COMMENTS ON FINDING Assignment GOOGLE FORM

- To find this email in your mailbox, you can use search for “**Google Forms**” string:

A screenshot of the Microsoft Outlook inbox interface. A red arrow points from the text "search for ‘Google Forms’ string:" in the slide content to the search bar at the top of the inbox. The search bar contains the text "google forms". Below the search bar, the inbox lists several items: New message, Delete, Archive, Spam, Move to, Categorise, and more. The inbox section shows categories like Junk Email, Drafts, Sent Items, Deleted Items, and Archive, with counts of 121, 62, 745, 138, and 0 respectively. The main pane displays search results under "Results" and "Top results". One result is highlighted: "Google Forms" from "OENG1116 Assignment-1 2019" dated Sun 24/03/2019 01:01, sent by Pavel Trivailo. To the right of the inbox, there is a large purple "Google Forms" button.

COMMENTS ON FINDING Assignment GOOGLE FORM

The screenshot shows the Microsoft Outlook inbox interface. On the left, there's a sidebar with navigation icons and a list of folders: Inbox (38278), Junk Email (121), Drafts (62), Sent Items (745), Deleted Items (138), and Archive. The main area displays two recent emails from "Google Forms". Both emails have the subject "OENG1116 Assignment-1 2019" and the body text "Thanks for filling in OENG1116 Assignment-1 2019 ...". A yellow callout box with a black border and a blue arrow points from the search bar at the top ("OENG1116 Assignment-1 2019") to the subject line of the first email.

Outlook

← OENG1116 Assignment-1 2019

+ New message

Inbox 38278

Junk Email 121

Drafts 62

Sent Items 745

Deleted Items 138

Archive

2019_OE... 79

_IAC_2005_Fuk...

Delete all Mark all as read

- You can also search for “**OENG1116 Assignment-1 2019**” string:

Google Forms

OENG1116 Assignment-1 2019

Thanks for filling in OENG1116 Assignment-1 2019 ...

Sun 24/03

Inbox

Google Forms

OENG1116 Assignment-1 2019

Thanks for filling in OENG1116 Assignment-1 2019 ...

Sun 24/03

Inbox

KNOWN ISSUES: Missing Google emails

- There were several cases reported in other Course, when the confirmation email was filtered out by the University anti-spam system.

A screenshot of a Google Mail interface showing a quarantine notification. The search bar at the top contains the text "RMIT Postmaster". Below the search bar, there is a red box highlighting the "RMIT Postmaster" entry in the dropdown menu. The main content area shows an email from "RMIT Postmaster <postmaster-do-not-reply@rmit.edu.au>" to the user. The subject line is "Email quarantine notification for pavel.trivailo@rmit.edu.au". The email body contains instructions for handling the quarantined message, including options to Release, Block, or Permit. A table at the bottom lists the message details: From, Subject, Date, Reason, and buttons for Release, Block, and Permit.

From	Subject	Date	Reason	Release	Block	Permit
forms-receipts-noreply@google.com		2018-03-02 09:55	Spam Policy	Release	Block	Permit

- You can search for these using "**RMIT Postmaster**"

A1 : 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:

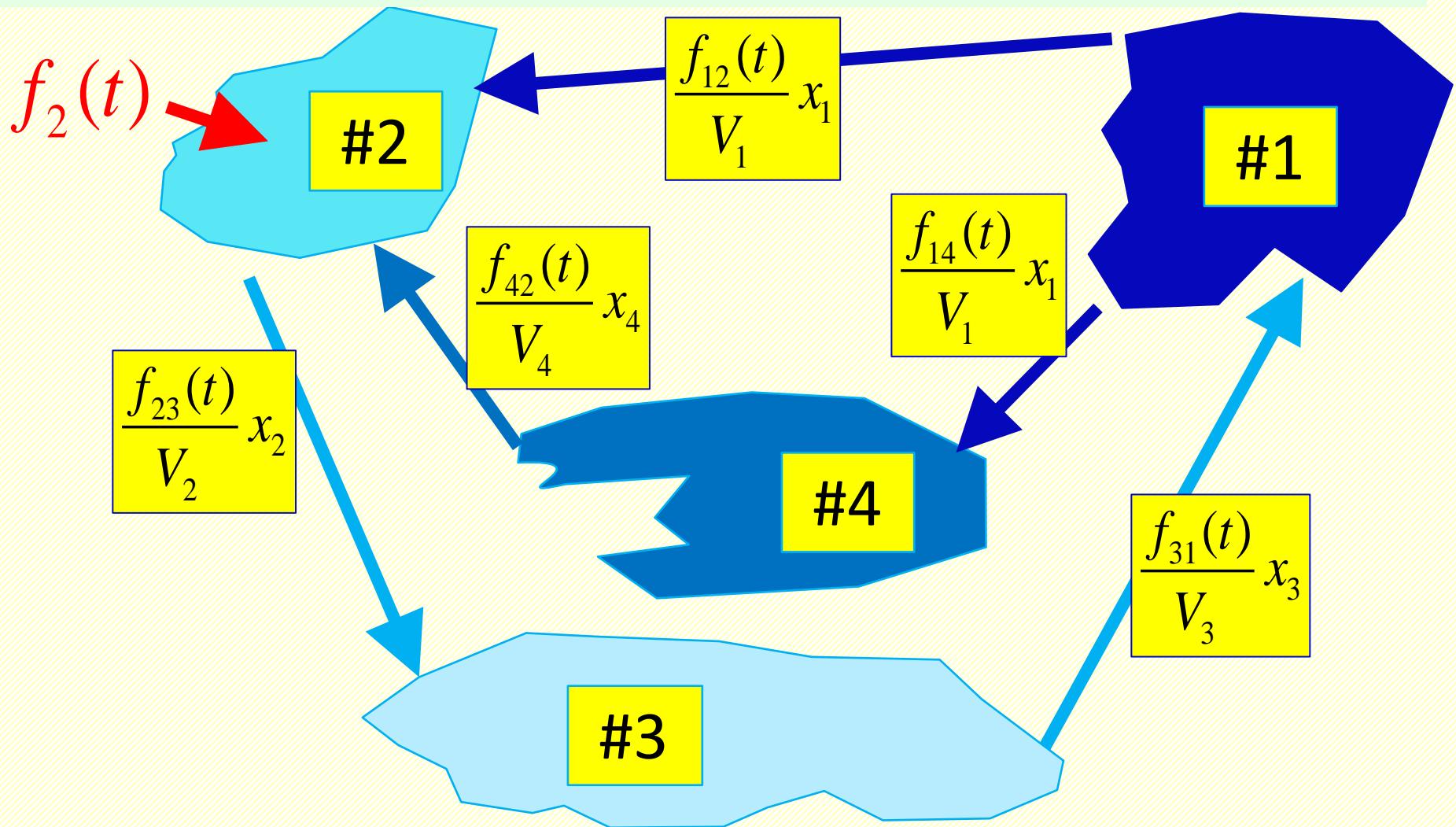
- (1) This Individual Assignment is due on Friday, 03-April-2020, 23:59. Submit electronically.
- (2) Total weighting of this Assign-1, Pt1: 20%. Attempt ALL tasks. Questions have shown values.
- (3) **You can only assume successful submission, if you (immediately after submission) receive confirmation 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, Pt2 Assist: MATLAB MINI-TUTORIAL: (2) LAKES: 1st order system; ode45

Consider four ponds connected by streams.

The second pond has a pollution source $f_2=2$ kg/h, which spreads via the connecting streams to the other ponds.

Plot the amount of pollutant in each pond as a function of time.



Symbol $f(t)$ is the pollutant flow rate into pond 2 (kg/h).

- Symbols f_1, f_2, f_3, f_4 denote the pollutant flow rates out of ponds 1, 2, 3, 4 respectively (m^3/h).
- It is assumed that the pollutant is well-mixed in each pond.
- The ponds have volumes V_1, V_2, V_3, V_4 (m^3), which remain constant.
- Symbols $x_1(t), x_2(t), x_3(t), x_4(t)$ denote the amount (kg) of pollutant in ponds 1, 2, 3, 4 respectively.

The pollutant flux is the flow rate times the pollutant concentration, e.g., pond 1 is emptied with flux f_{12} times $x_1(t)/V_1$ plus flux f_{14} times $x_1(t)/V_1$.

The diagram plus compartment analysis gives the following differential equations:

$$\frac{dx_1(t)}{dt} = -\frac{f_{12}}{V_1} x_1(t) - \frac{f_{14}}{V_1} x_1(t) + \frac{f_3}{V_3} x_3(t)$$

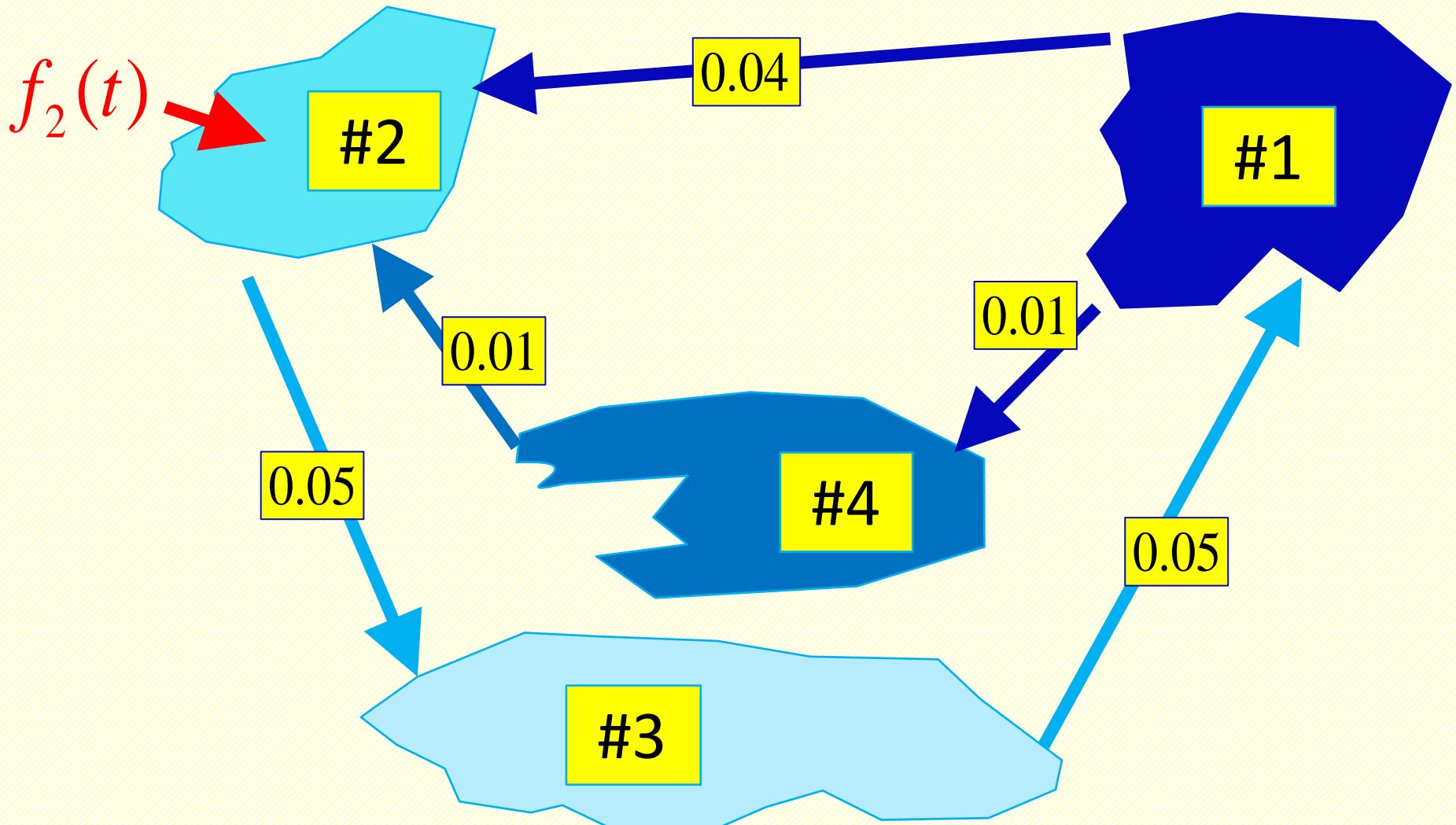
$$\frac{dx_2(t)}{dt} = \frac{f_{12}}{V_1} x_1(t) - \frac{f_{23}}{V_2} x_2(t) + \frac{f_{42}}{V_4} x_4(t) + f_2(t)$$

$$\frac{dx_3(t)}{dt} = \frac{f_{23}}{V_2} x_2(t) - \frac{f_{31}}{V_3} x_3(t)$$

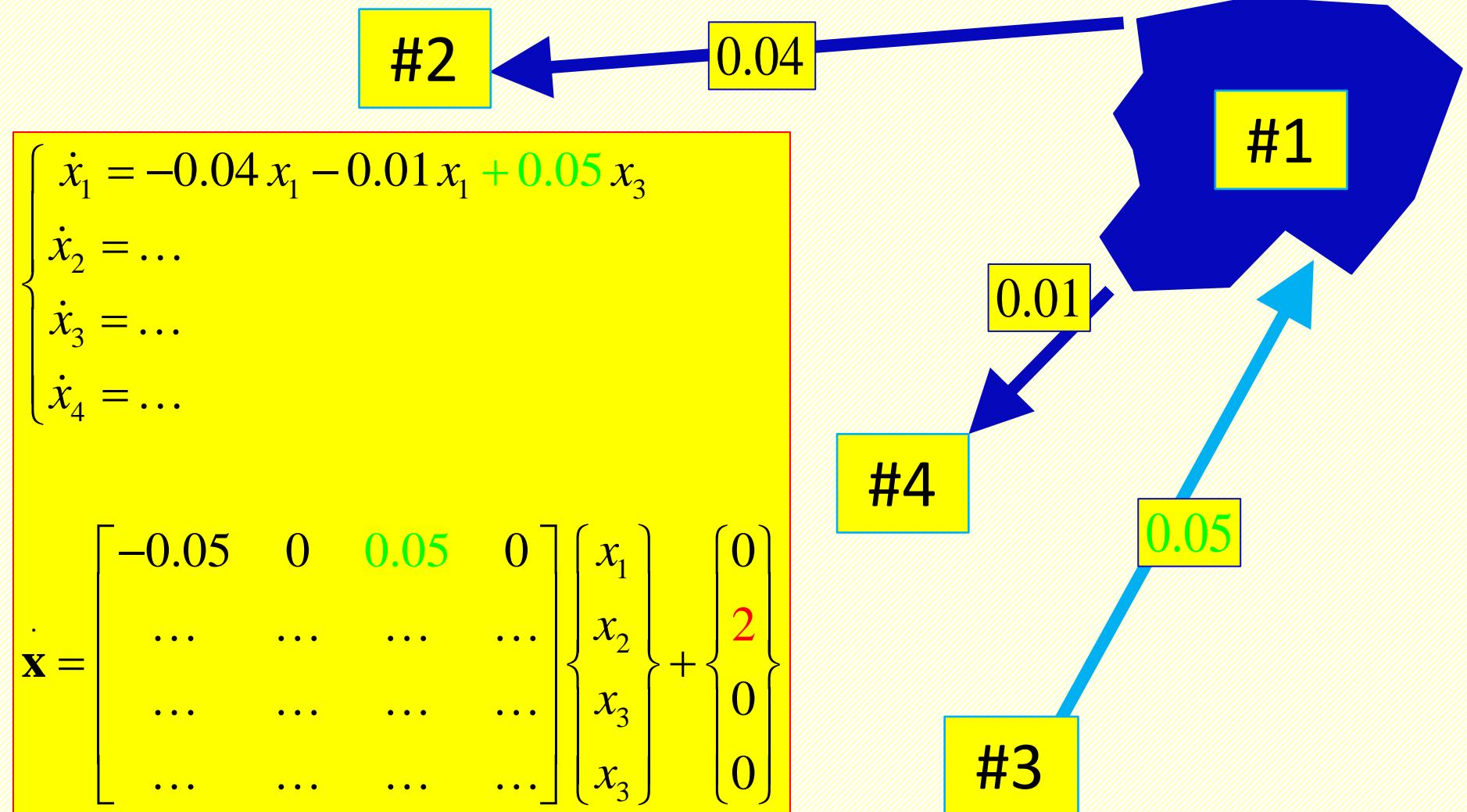
$$\frac{dx_4(t)}{dt} = \frac{f_{14}}{V_1} x_1(t) - \frac{f_{41}}{V_4} x_4(t)$$

For a specific numerical example, take $f_{14}/V_1 = f_{42}/V_4 = 0.01$, $f_{12}/V_1 = 0.04$, $f_{23}/V_2 = f_{31}/V_3 = 0.05$ [all in 1/h] and let $f_2(t) = 2 \text{ kg/h}$ for the first 48 hours, thereafter $f(t) = 0$.

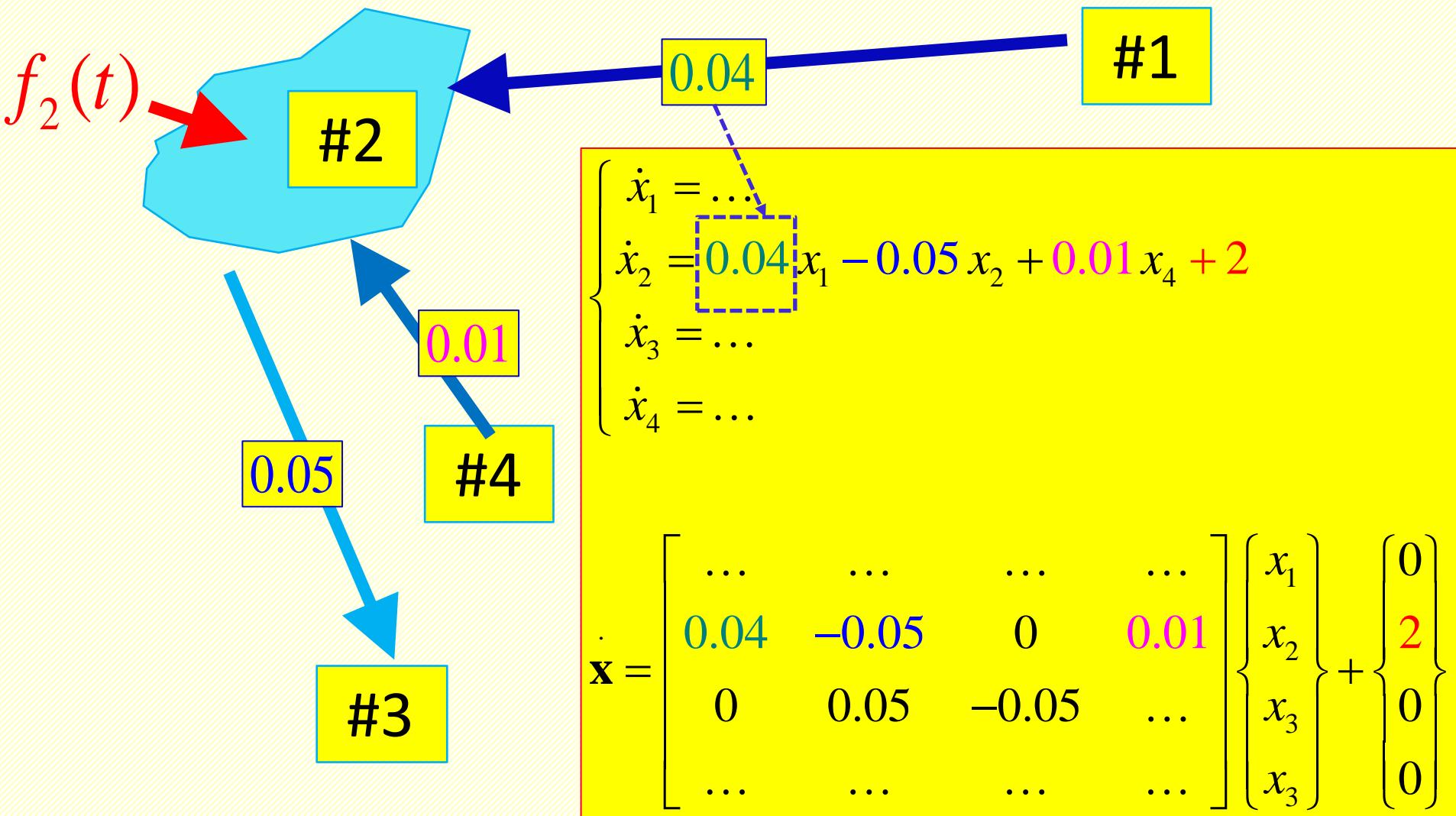
We expect due to uniform mixing that after a long time there will be $2*48 = 96 \text{ kg}$ of pollutant uniformly deposited. Initially, $x_1(0) = x_2(0) = x_3(0) = x_4(0) = 0$, i.e. the ponds were pristine.



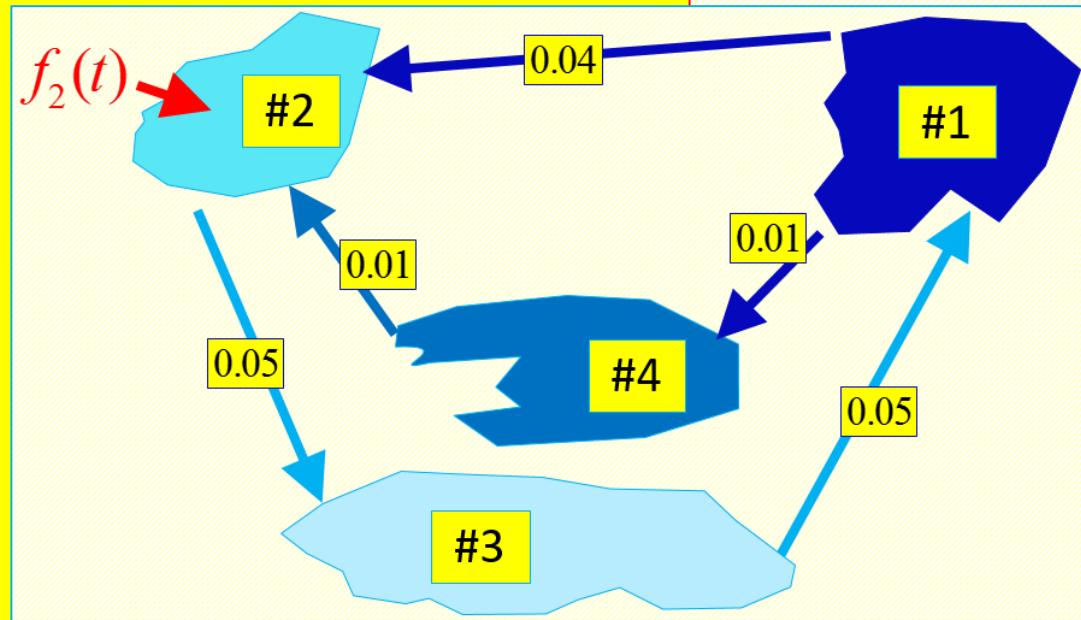
Consider Pond-1 only to set 1st Equation:



Consider Pond-2 only to set 2nd Equation:



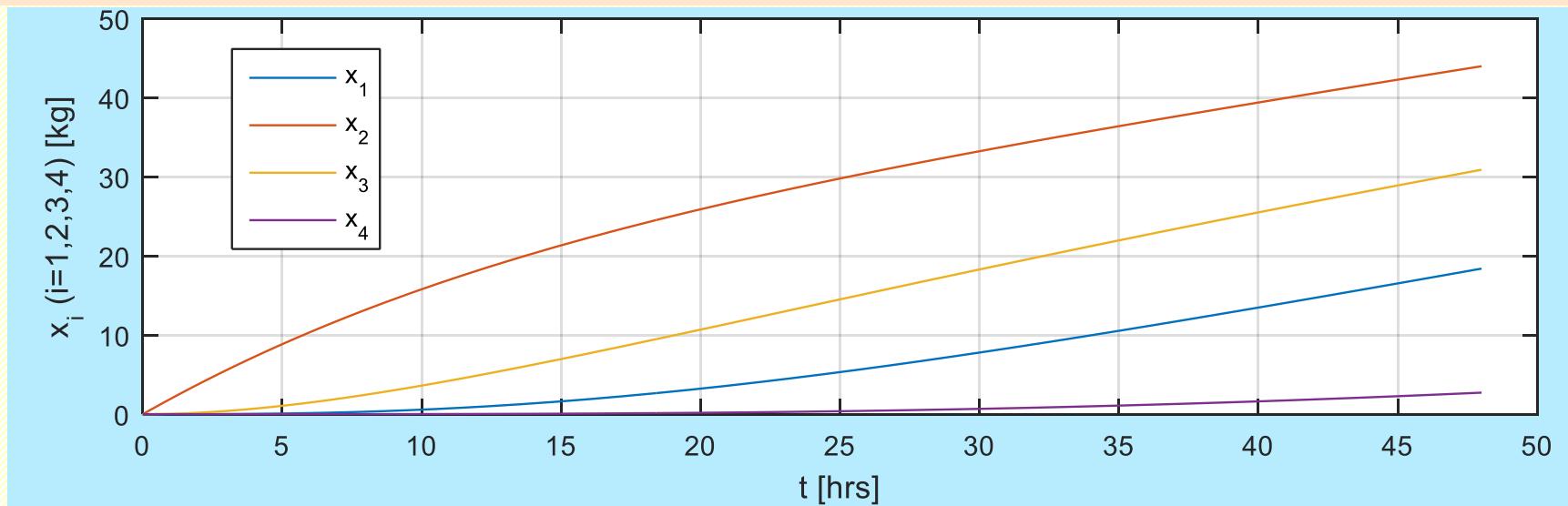
$$\begin{cases} \dot{x}_1 = -0.04x_1 - 0.01x_1 + 0.05x_3 \\ \dot{x}_2 = 0.04x_1 - 0.05x_2 + 0.01x_4 + 2 \\ \dot{x}_3 = 0.05x_2 - 0.05x_3 \\ \dot{x}_4 = 0.01x_1 - 0.01x_4 \end{cases}$$



$$\dot{\mathbf{x}} = \begin{bmatrix} -0.05 & 0 & 0.05 & 0 \\ 0.04 & -0.05 & 0 & 0.01 \\ 0 & 0.05 & -0.05 & 0 \\ 0.01 & 0 & 0 & -0.01 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 0 \\ 2 \\ 0 \\ 0 \end{pmatrix}$$

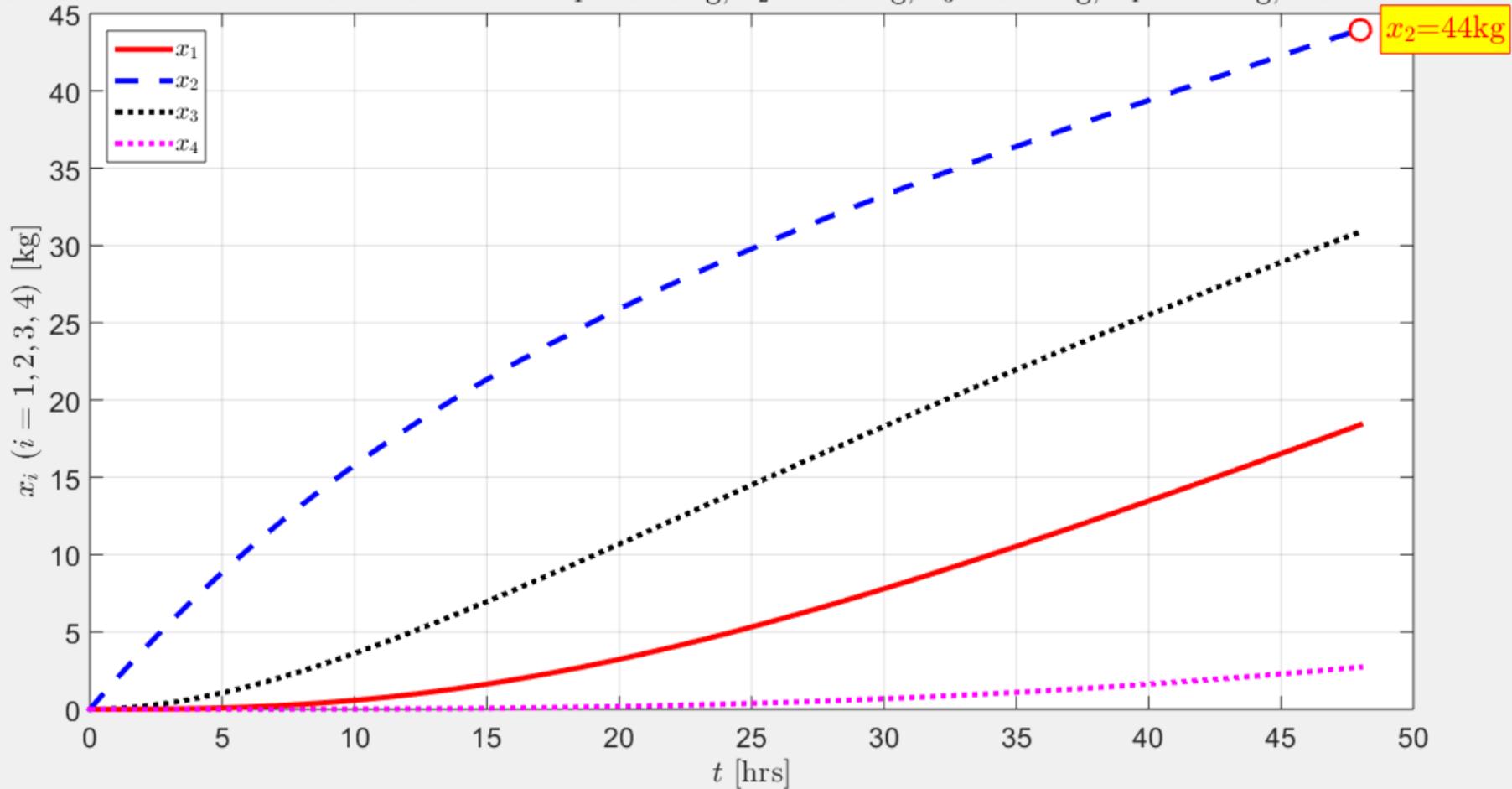
MATLAB SCRIPT (no annotations)

```
%% Designed by Prof Pavel M.Trivailo (C) 2020
% LAKES: Simple Script
clear; close all; clc;
r=[ -5 0 5 0; 4 -5 0 1 ; 0 5 -5 0; 1 0 0 -1]*0.01;
D=[0; 2; 0; 0]; tmax=24*2; % hrs
xd = @(t, x) r*x+D;
[tt,zz]=ode45(xd, [0 tmax], [0;0;0;0]);
plot(tt,zz); grid on; legend('x_1','x_2','x_3','x_4');
xl=xlabel('t [hrs]'); yl=ylabel('x_i (i=1,2,3,4) [kg]');
```



MATLAB annotated plot

TASK-0A: EXAMPLE IN CLASS: $t=48$ h: $x_1=18.40$ kg; $x_2=43.98$ kg; $x_3=30.90$ kg; $x_4= 2.73$ kg; TOTAL= 96.00 kg



Extra Task-1: find x_2 for $t=24$ hrs

Command Window

New to MATLAB? See resources for [Getting Started](#).

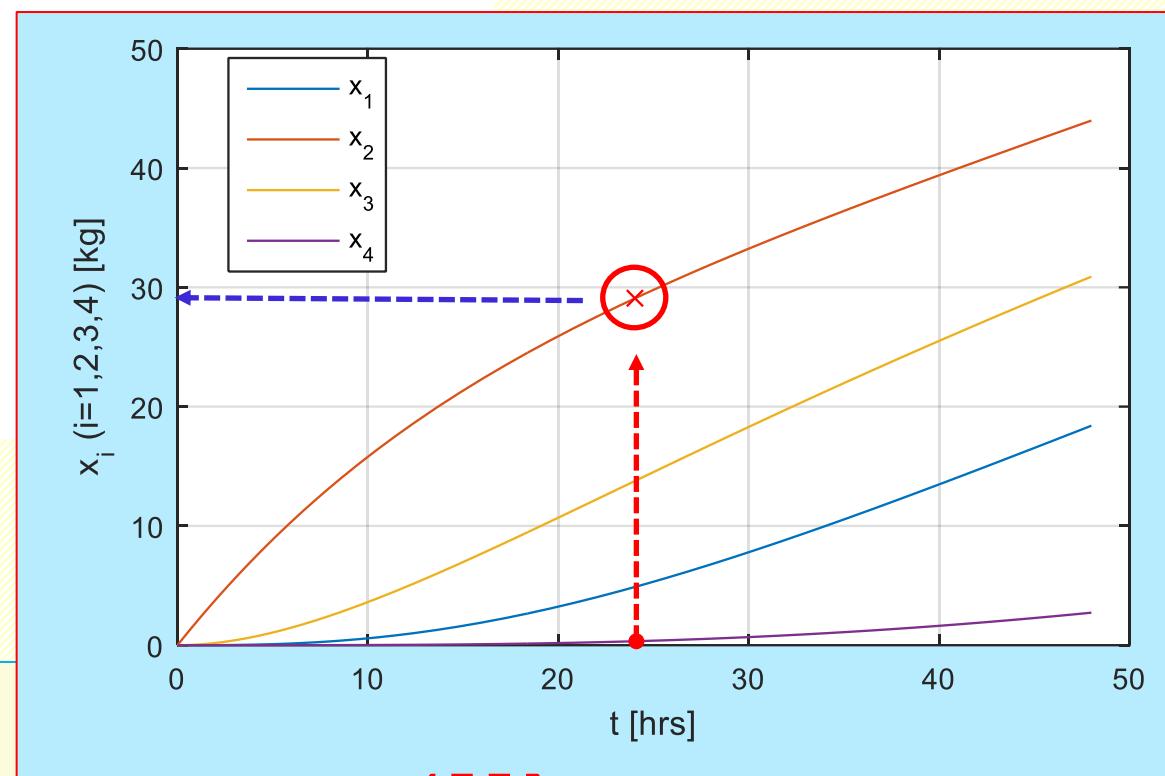
```
>> interp1(tt, xx(:, 2), 24)
```

ans =

29.0509

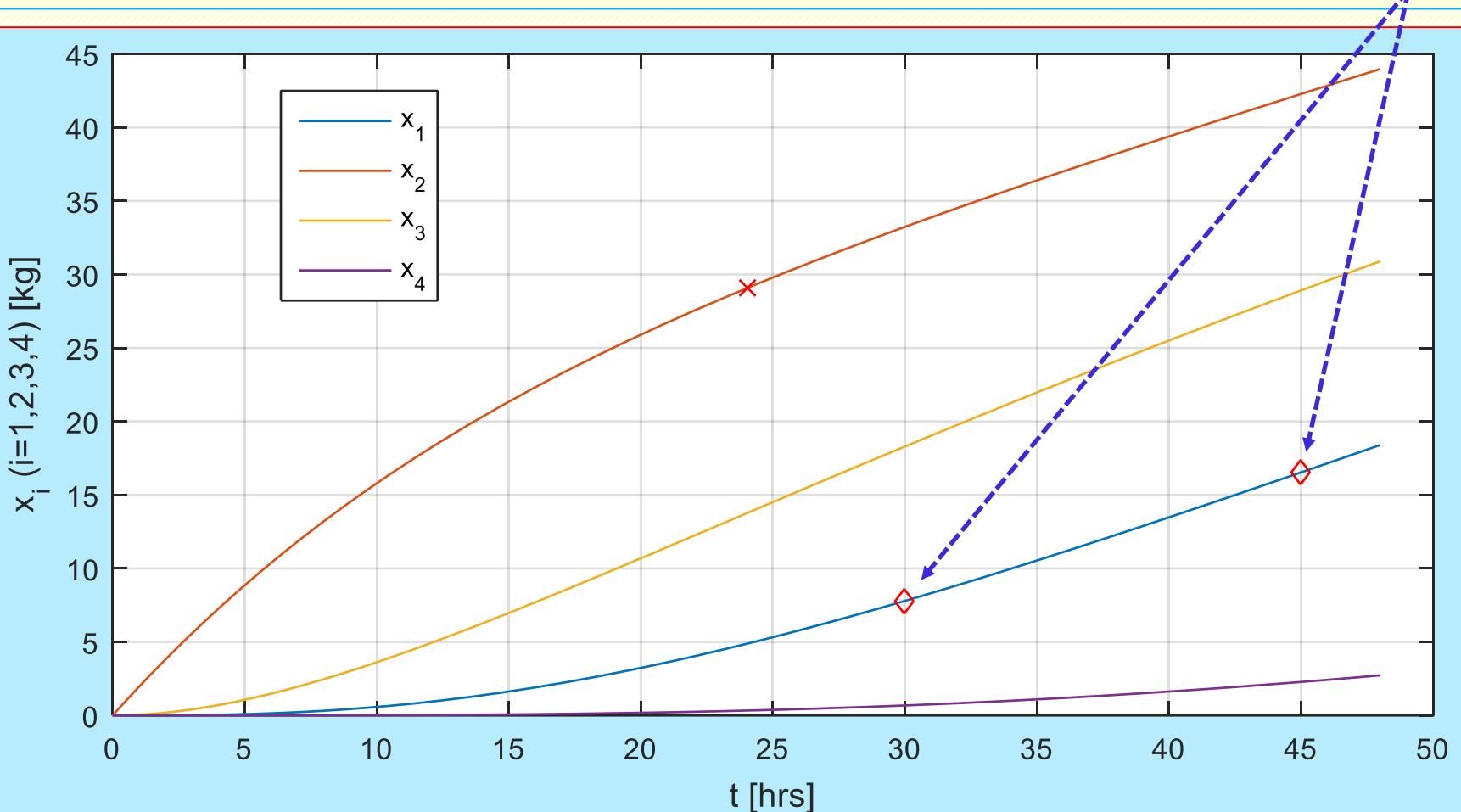
hold on;

```
plot(24, interp1(tt, xx(:, 2), 24), 'rx');
```



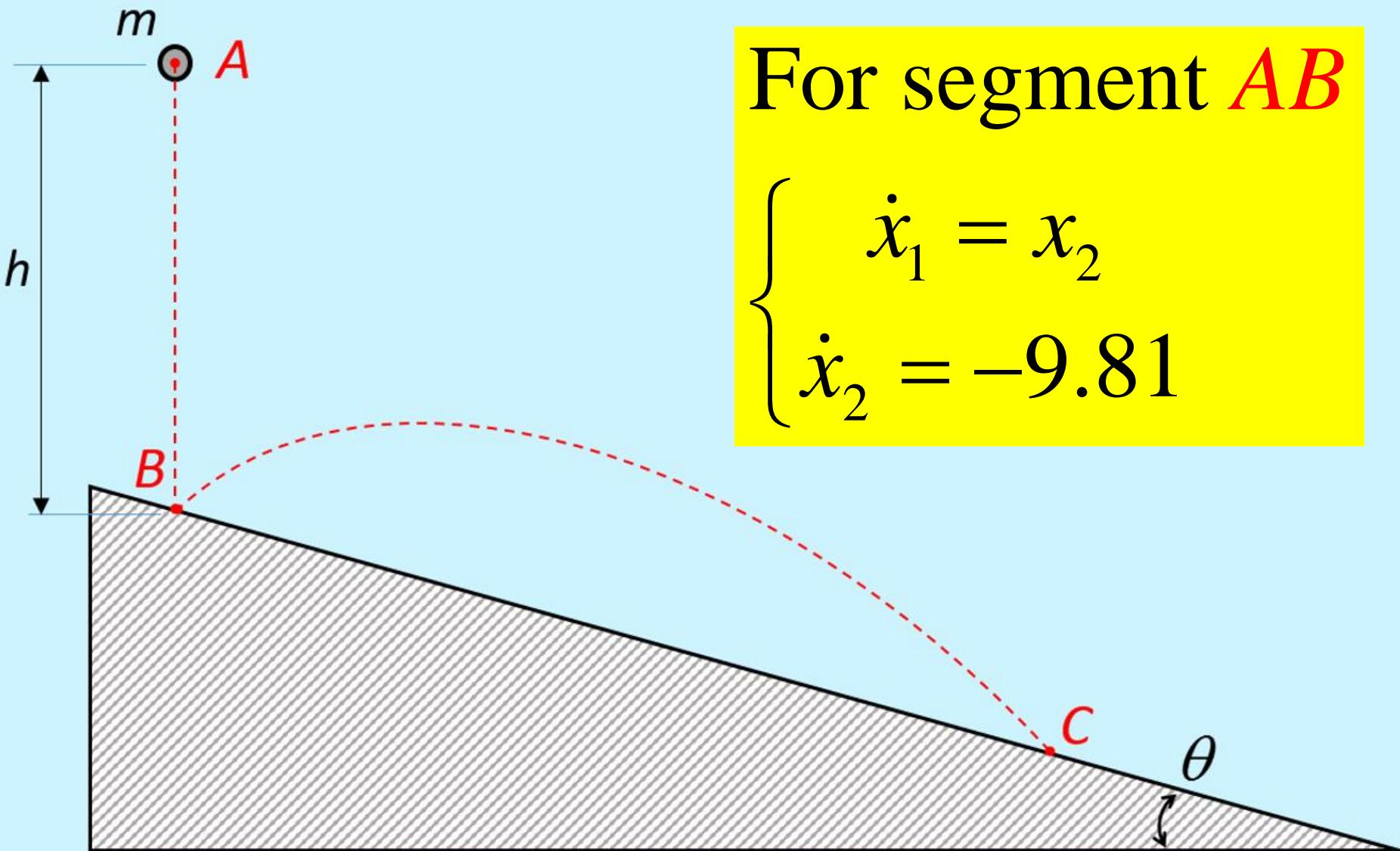
Extra Task-2: find x_1 for $t=[30 \ 45]$ hrs

```
p2=plot([30 45], interp1(tt, xx(:,1), [30 45]), 'rd');
```



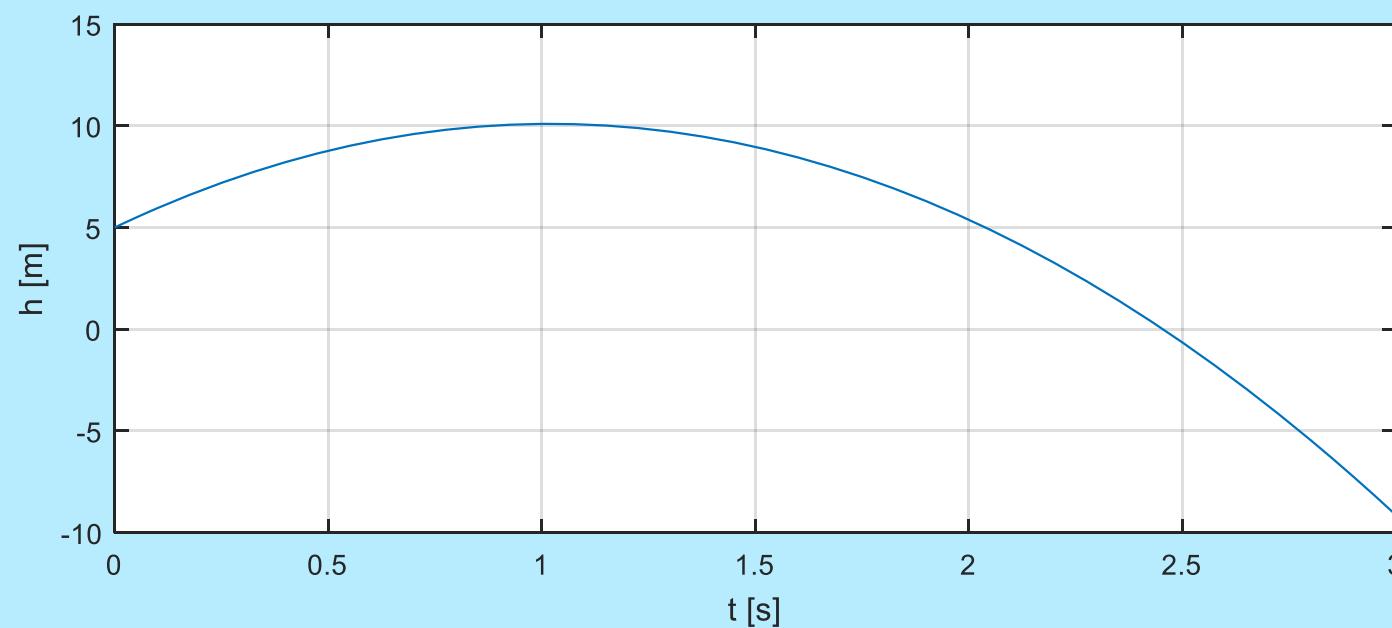
A1, Pt2 Assist: MATLAB MINI-TUTORIAL: (2) FALLING MASS: 2nd order system; ode45

T3 in Assignment-1



MATLAB SCRIPT (no annotations)

```
%% Designed by Prof Pavel M.Trivailo (C) 2019
% Falling mass: simple script
clear; close all; clc;
f = @(t,x) [x(2); -9.81];
[tt,xx]=ode45(f, [0 3], [5 10]);
plot(tt,xx(:,1));
xlabel('t [s]'); ylabel('h [m]'); grid on
```



Extra Task-1: find t when $h=0$

Command Window

New to MATLAB? See resources for [Getting Started](#).

```
>> find(xx(:, 1) < 0)
```

ans =

37

38

39

40

41

42

43

44

45

fx

Command Window

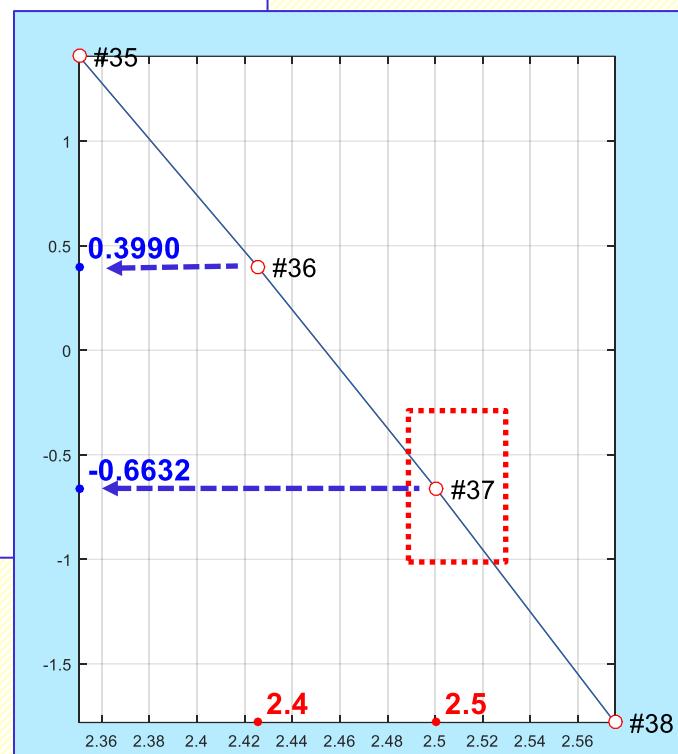
New to MATLAB? See resources for [Getting Started](#).

```
>> xx(36:37, 1)
```

ans =

0.3990

-0.6632



Extra Task-1: find t when $h=0$

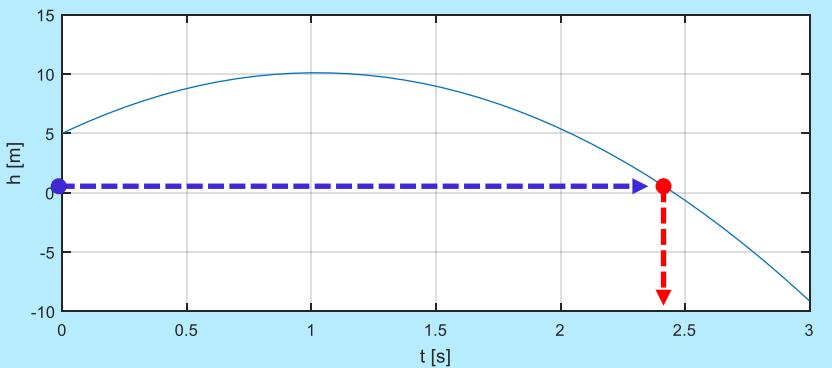
Command Window

New to MATLAB? See resources for [Getting Started](#).

```
>> t_B=interp1(xx(idx1:idx2), tt(idx1:idx2), 0)
```

```
t_B =
```

```
2.4536
```



```
>> sprintf('Time, when mass hits the ground is: %6.4f s', t_B)
```

```
ans =
```

```
Time, when mass hits the ground is: 2.4536 s
```

A-1

HINTS:

Previous & New Examples

Task in Assignment-1: Optics Analogy for Bouncing

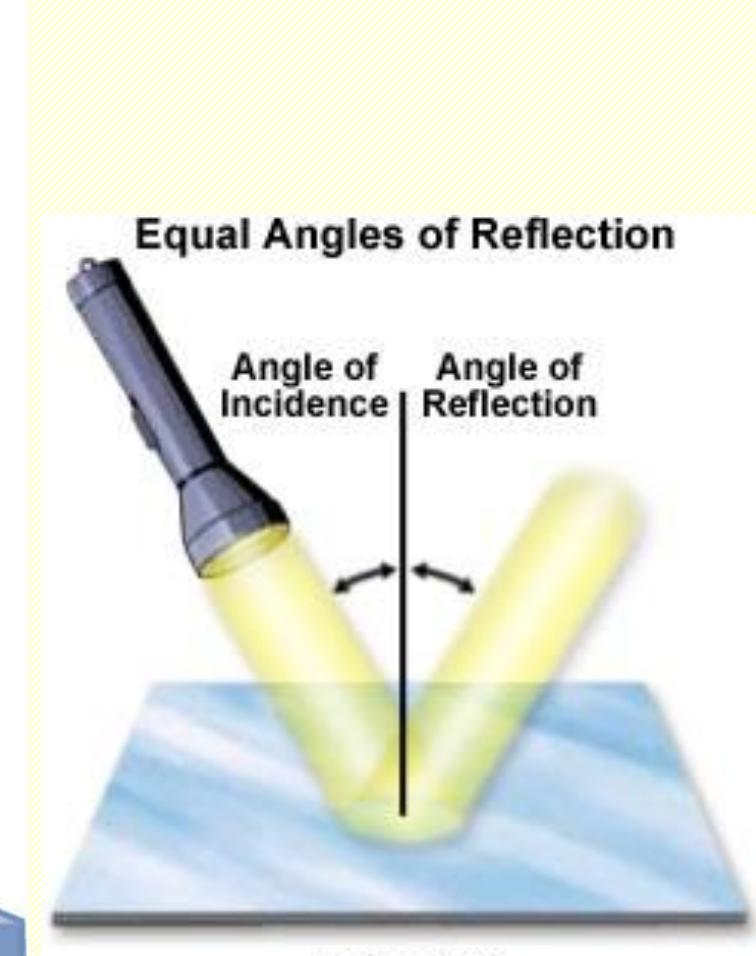
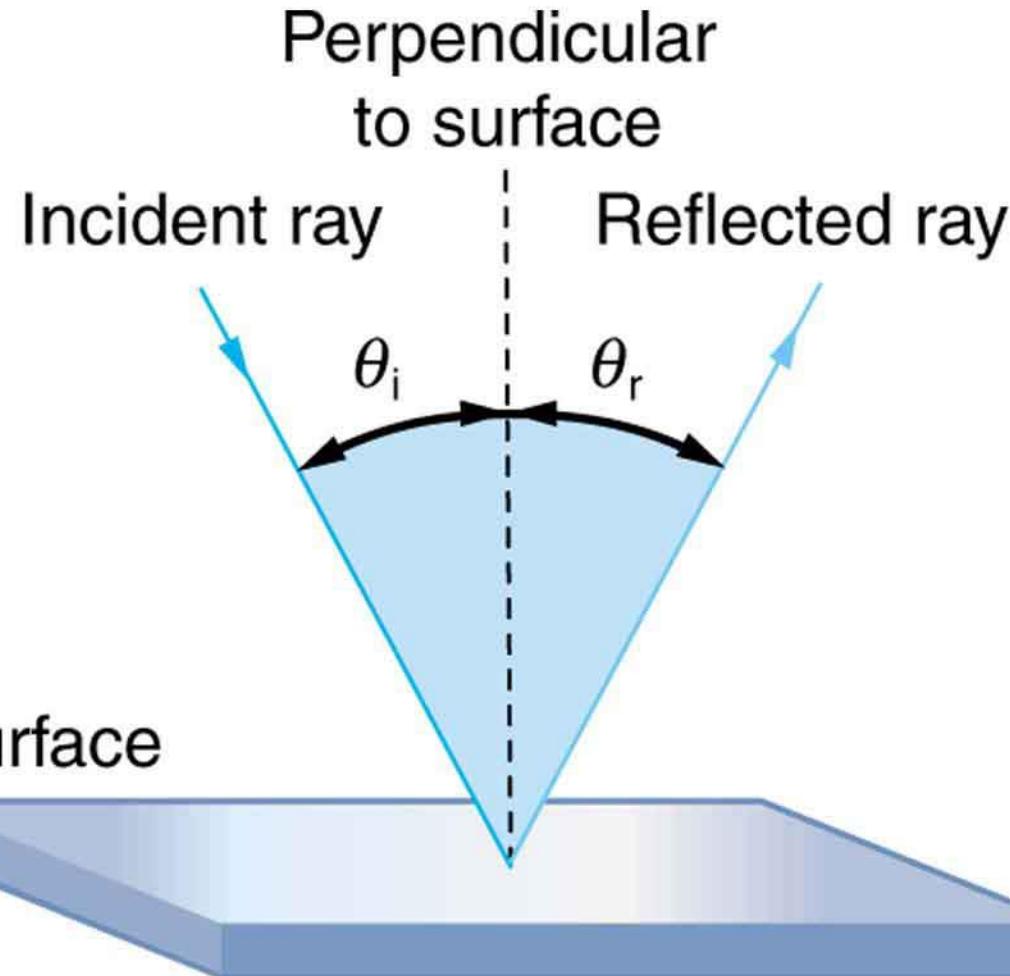


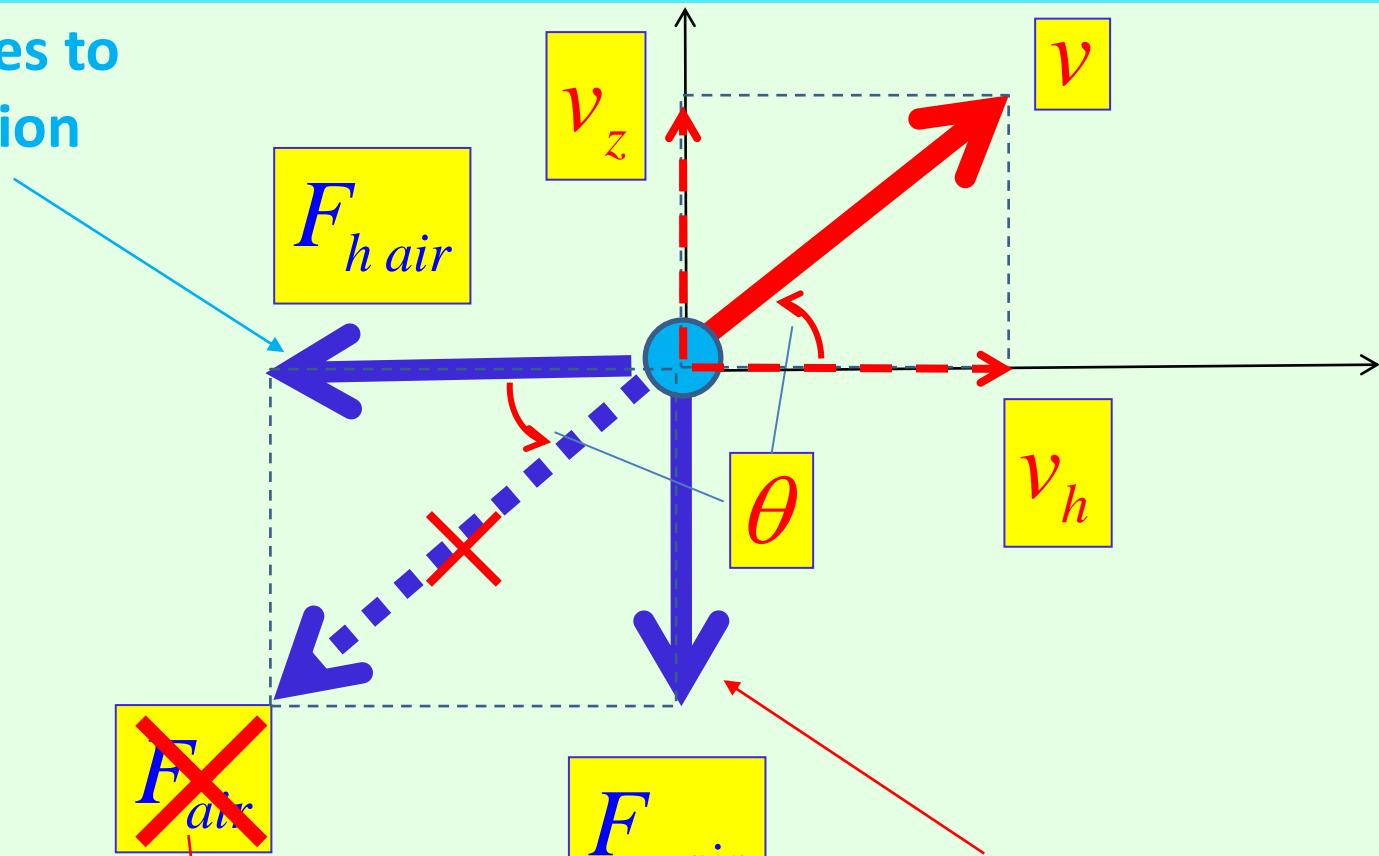
Figure 2

PROJECTILE: INCLUDING AIR RESISTANCE

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

Hint for Assignment-1: Air Resistance

Contributes to
“*h*” equation



Contributes to
“*z*” equation

$$F_{air} = C_d v^2 = C_d (v_h^2 + v_z^2)$$

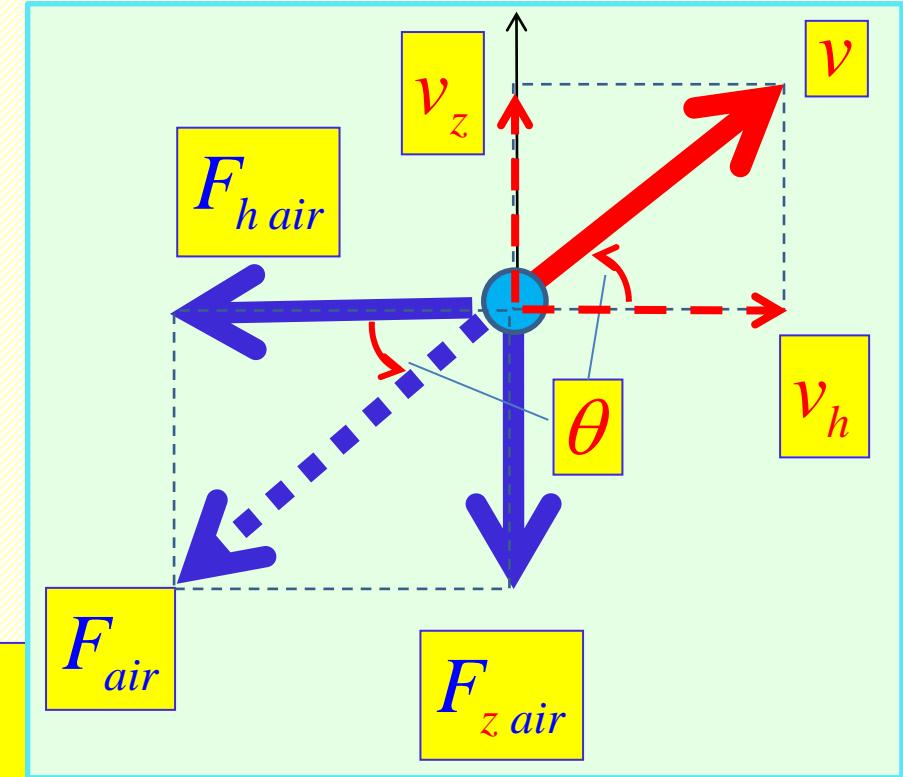
Hint for Assignment-1: Air Resistance

**IMPORTANT
ANALYTICAL
DERIVATIONS!!!**

$$F_{air} = C_d v^2;$$

$$F_{z\ air} = F_{air} \sin \theta = F_{air} \frac{v_z}{v} = C_d v^2 \frac{v_z}{v} = C_d v_z v = C_d v_z \sqrt{v_h^2 + v_z^2};$$

$$F_{h\ air} = F_{air} \cos \theta = F_{air} \frac{v_h}{v} = C_d v^2 \frac{v_h}{v} = C_d v_h v = C_d v_h \sqrt{v_h^2 + v_z^2};$$



MATLAB SCRIPT (no annotations)

```
%% PROJECTILE (WITH AIR RESISTANCE) EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2019
% Feature: ALL COMMANDS ARE IN ONE FILE!!!
%
m=10; coeff=0.6; % F=coeff*Vsq;
th=45*pi/180; % rad      ^ z
v0=10;          % m/s      |
z0=0; h0=0       % m          +-----> h
g=9.81;
t=[0:0.01:1]*1.3;
v0h=v0*cos(th); v0z=v0*sin(th);

figure; grid on; hold on; axis equal;
pr_xdot = @(t, x) ([x(2); -9.81-coeff*x(2)*sqrt(x(2)^2+x(4)^2)/m; ...
    x(4); -coeff*x(4)*sqrt(x(2)^2+x(4)^2)/m]);
[tt,zz]=ode45(pr_xdot,t,[z0; v0z; h0; v0h]);
plot(zz(:,3), zz(:,1), 'LineWidth',3, 'Color',[0 0.5 1]);
xlabel('$h$ [m]', 'Interpreter', 'LaTeX');
ylabel('$z$ [m]', 'Interpreter', 'LaTeX');
title('bf Projectile (With Air Drag): Time History');
set(gca, 'FontSize',18);
set(gcf, 'Position', [20 80 1270 680]);

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
```

Definition of states:

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

MATLAB SCRIPT (with annotations)

```
%% PROJECTILE (WITH AIR RESISTANCE) EXAMPLE  
% Designed by Prof P.M.Trivailo (C) 2019  
% Feature: ALL COMMANDS ARE IN ONE FILE!!!  
%-----  
m=10; coeff=0.6; % F=coeff*Vsq;  
th=45*pi/180; % rad ^ z  
v0=10; % m/s  
z0=0; h0=0 % m  
g=9.81;  
t=[0:0.01:1]*1.3;  
v0h=v0*cos(th); v0z=v0*sin(th);
```

figure; grid on; hold on; axis equal;

```
pr_xdot = @(t, x) ([x(2); -9.81-coeff*x(2)*sqrt(x(2)^2+x(4)^2)/m; ...  
x(4); -coeff*x(4)*sqrt(x(2)^2+x(4)^2)/m]);
```

```
[tt,zz]=ode45(pr_xdot,t,[z0; v0z; h0; v0h]);
```

```
plot(zz(:,3), zz(:,1), 'LineWidth',3, 'Color',[0 0.5 1]);  
xlabel('$h$ [m]', 'Interpreter', 'LaTeX');  
ylabel('$z$ [m]', 'Interpreter', 'LaTeX');  
title('Projectile (With Air Drag): Time History'),  
set(gca, 'FontSize',18);  
set(gcf, 'Position', [20 80 1270 680]);
```

```
g=line('XData',h0, 'YData',z0, 'Marker', ' ', 'MarkerSize', 48);  
for i=1:length(tt),  
    set(g, 'XData',zz  
end
```

2.

EOM

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -g - \frac{1}{m} cv^2 \sin \theta \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = -\frac{1}{m} cv^2 \cos \theta \end{cases}$$

Input of the data

1.

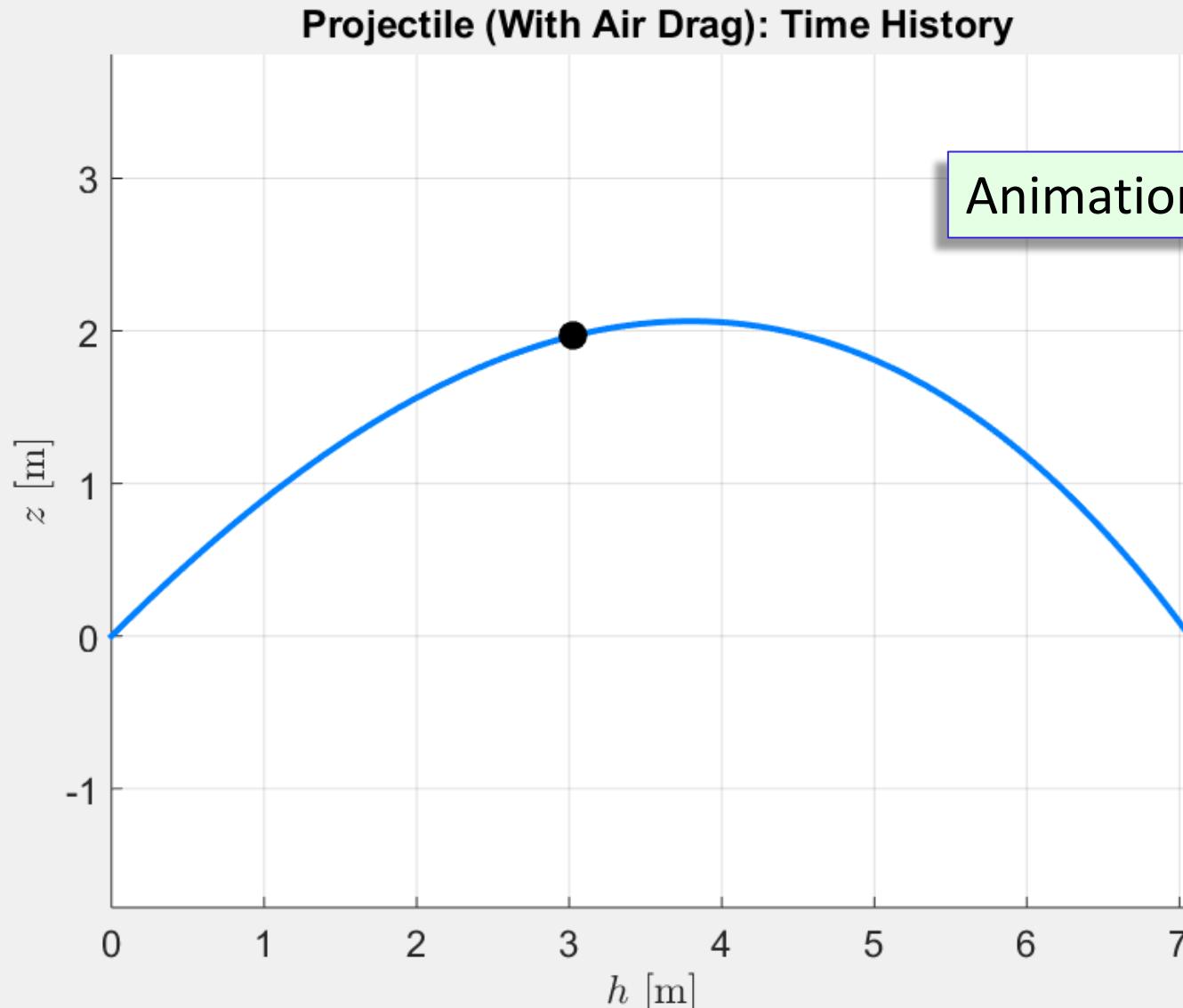
4.

Plotting results

3.

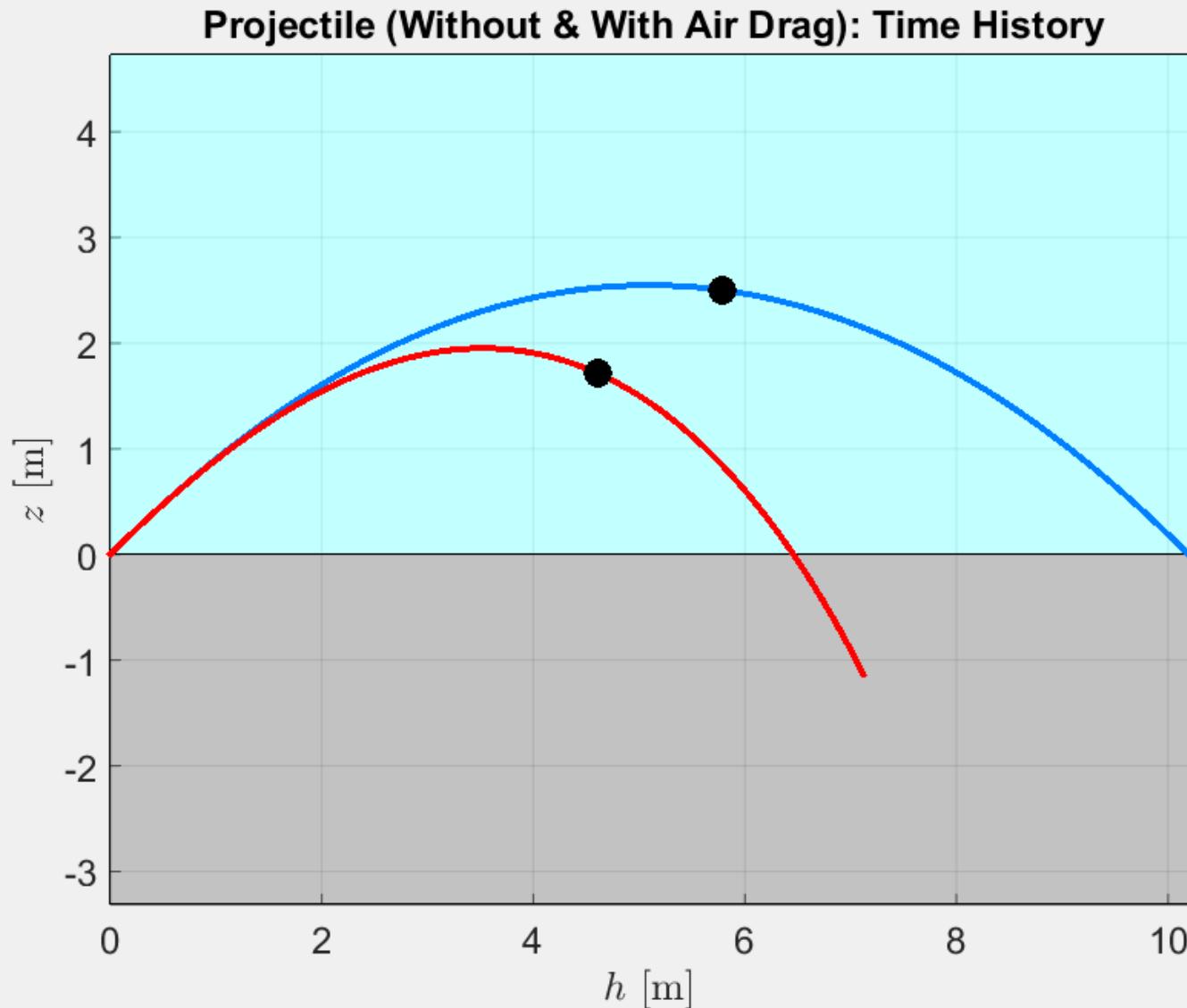
Calling `ode45` procedure !

Projectile: Trajectory for Air-Resistance Case

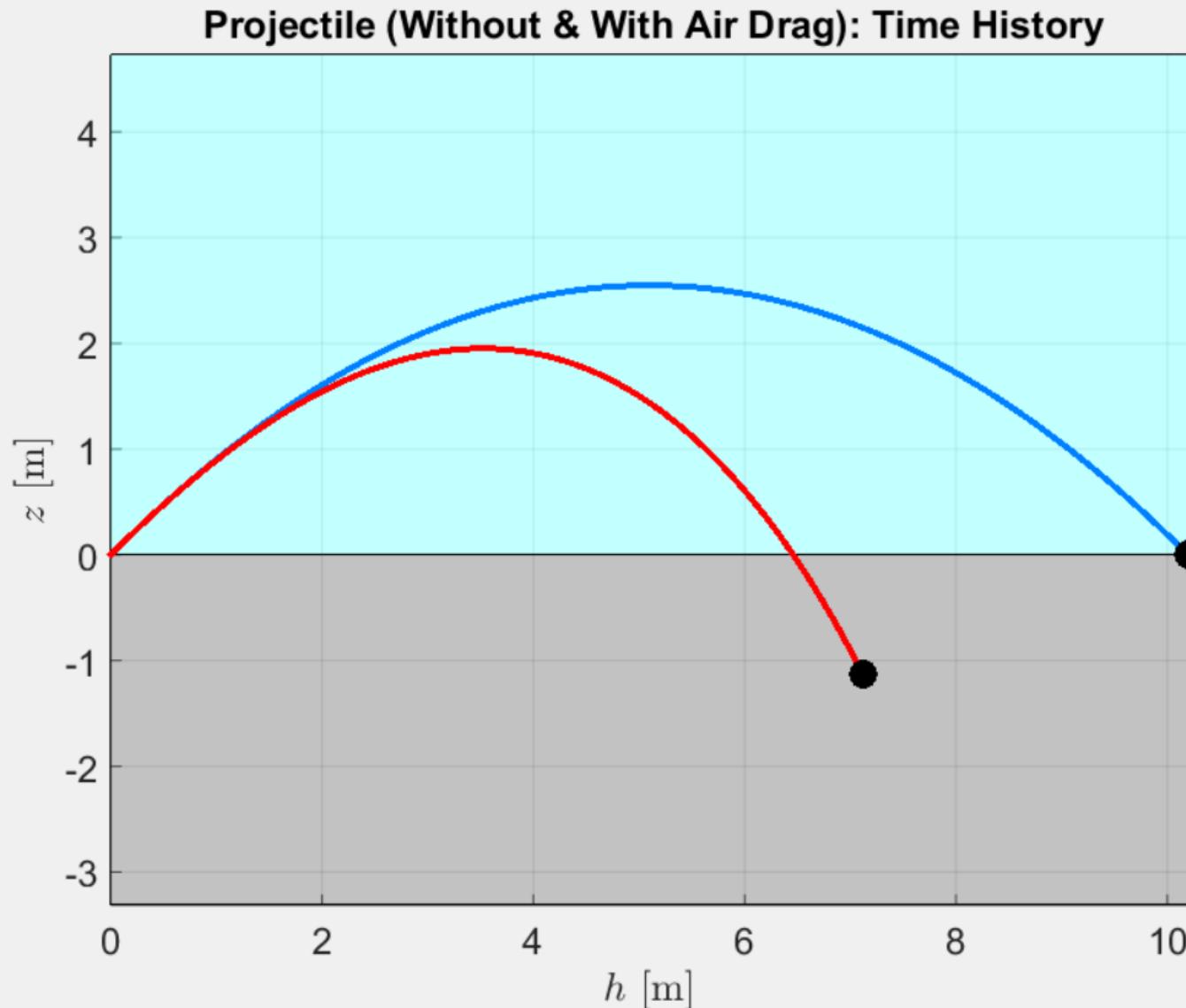


Animation in Class

Air-resistance versus No-air-resistance Cases



Air-resistance versus No-air-resistance Cases

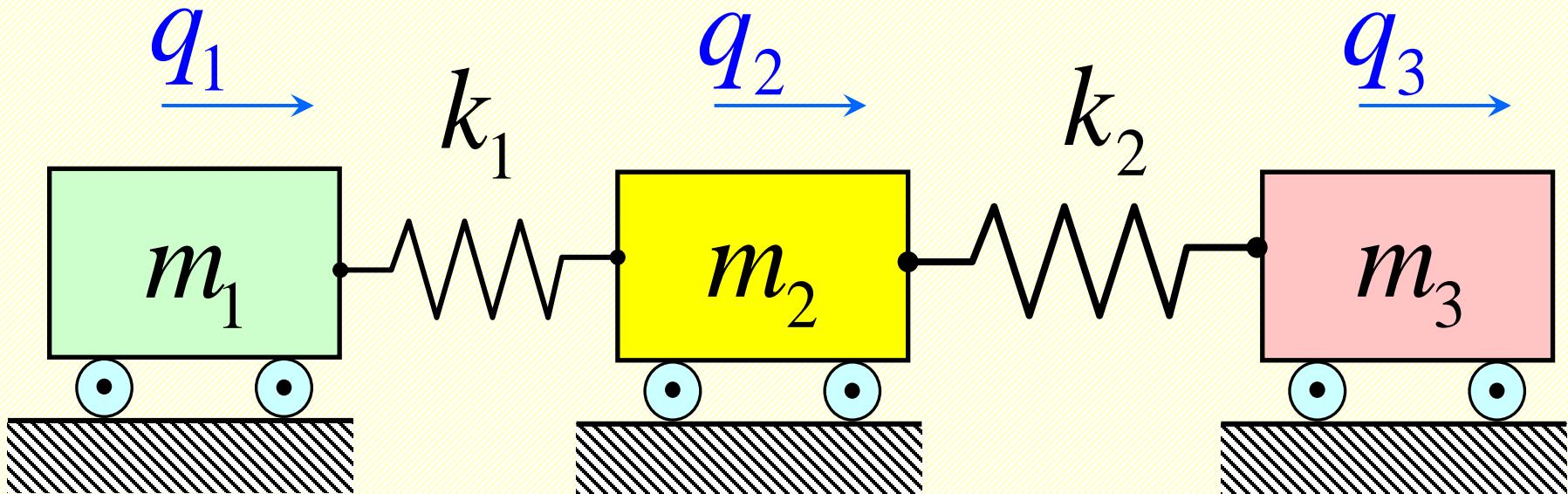


Simplified Introduction into FEM: Mass-Spring System Examples

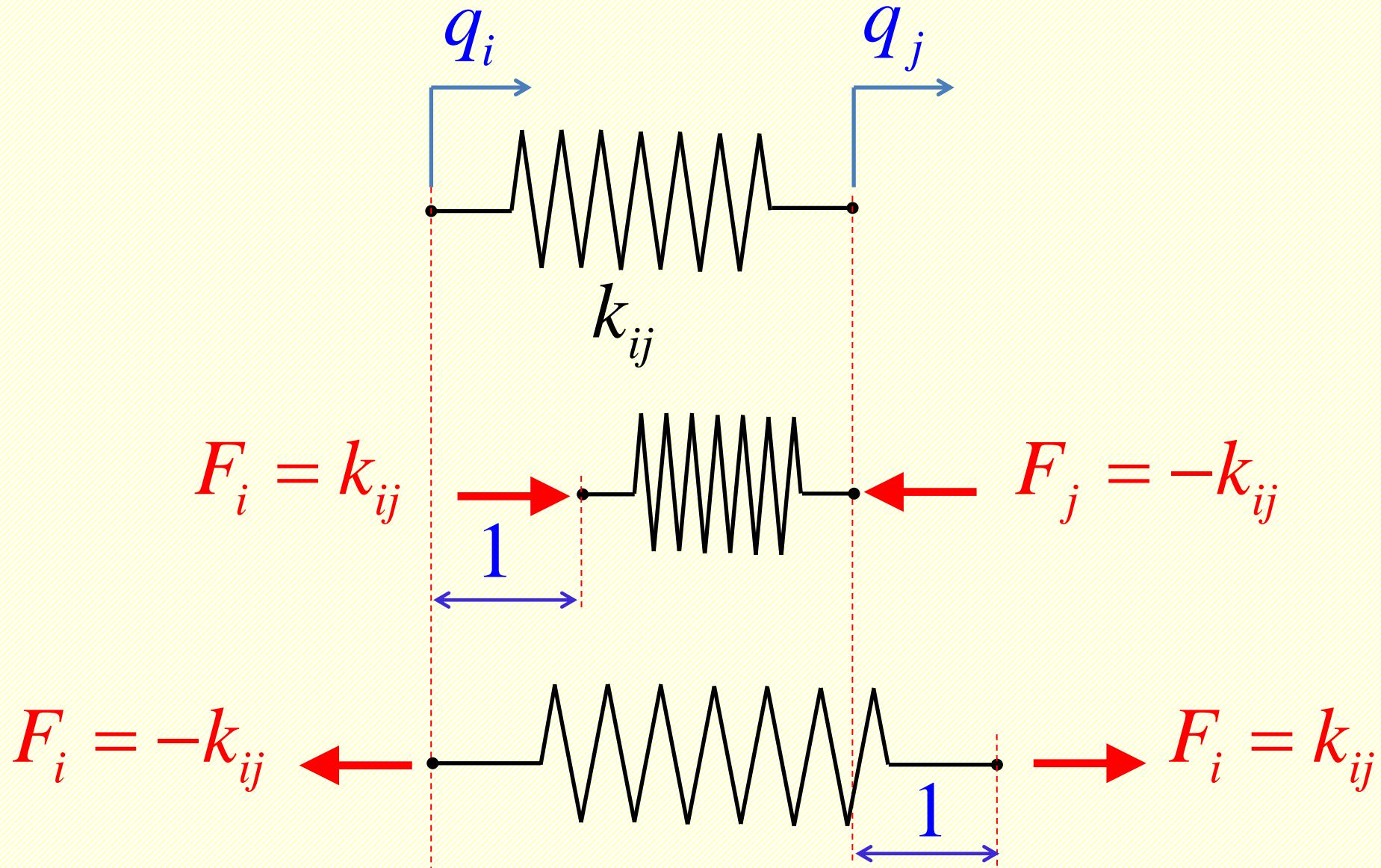
Multi-DOF mass-spring systems: Derivation of EOM using an FEM approach: assembling the global stiffness matrix.

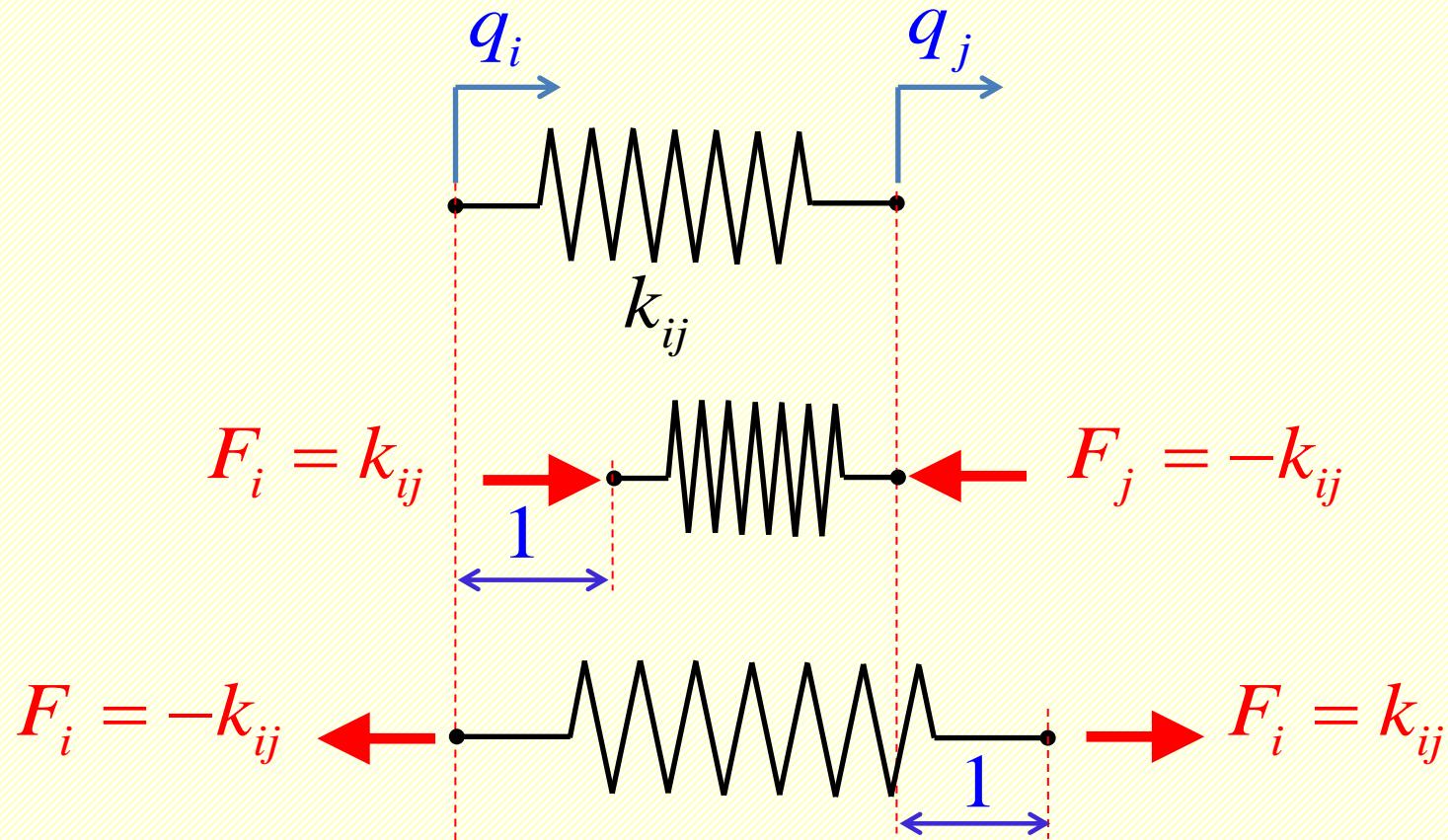
Introduction into FEM: Free Boundary Case

Case-1 Study: 3-DOF System



Element Stiffness Matrix: Derivation



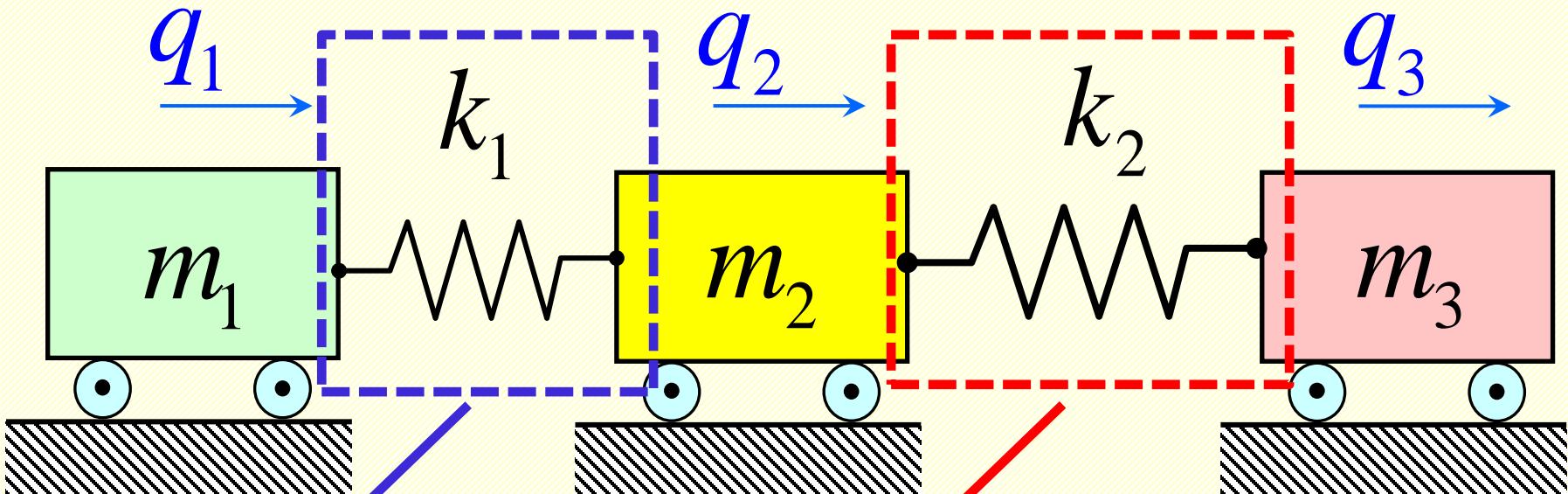


$$\begin{bmatrix} k_{ij} & -k_{ij} \\ -k_{ij} & k_{ij} \end{bmatrix} \begin{Bmatrix} q_i \\ q_j \end{Bmatrix} = \begin{Bmatrix} F_i \\ F_j \end{Bmatrix}$$

$$\begin{bmatrix} k_{ij} & -k_{ij} \\ -k_{ij} & k_{ij} \end{bmatrix} \begin{Bmatrix} q_i \\ q_j \end{Bmatrix} = \begin{Bmatrix} F_i \\ F_j \end{Bmatrix}$$

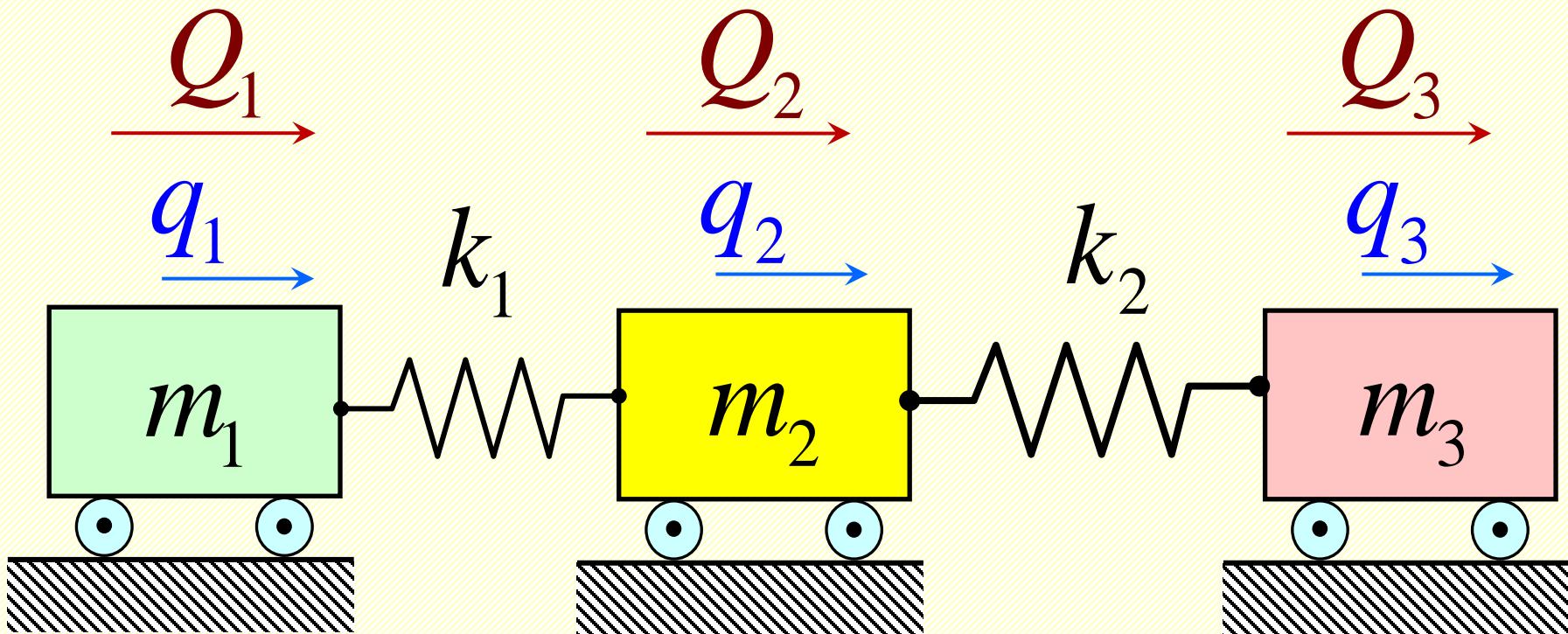
$$[k_{\text{FElement } (ij-\text{DOF})}] = \begin{bmatrix} k_{ij} & -k_{ij} \\ -k_{ij} & k_{ij} \end{bmatrix}$$

Case-1 Study: Assembling Global $[k]$



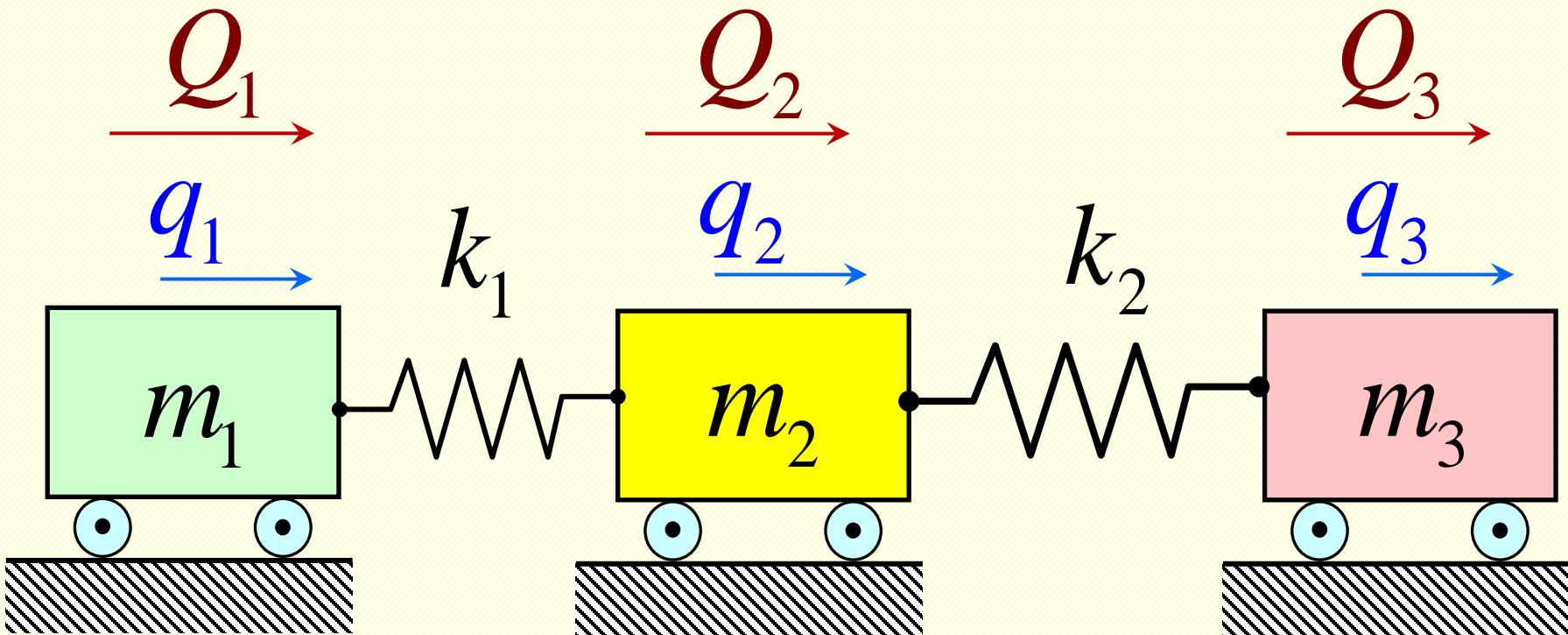
$$\begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} = \begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix}$$

Case-1 Study: Stiffness Static Equation



$$\begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} q_1 \\ q_2 \\ q_3 \end{Bmatrix} = \begin{Bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{Bmatrix}$$

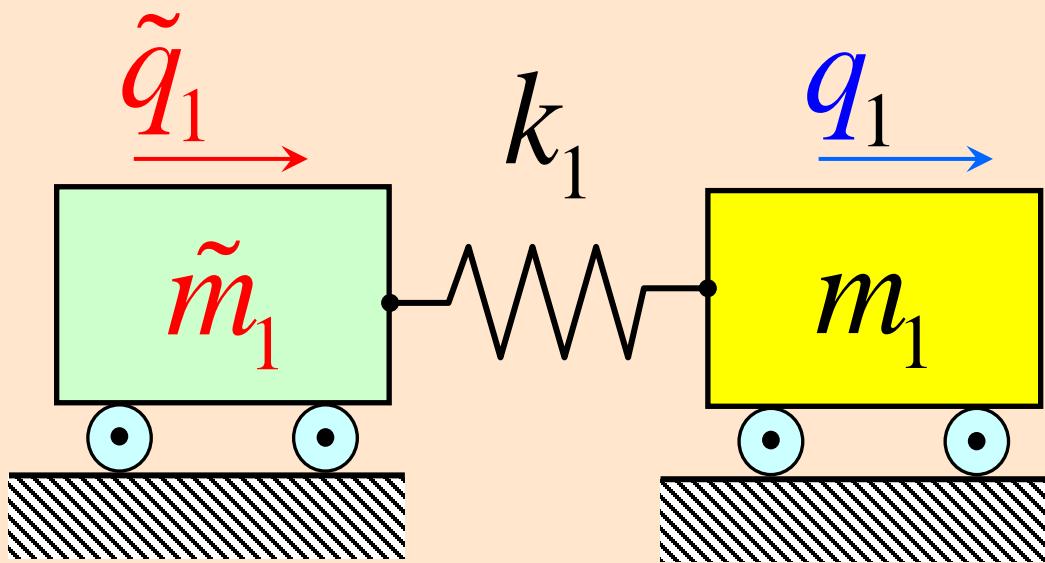
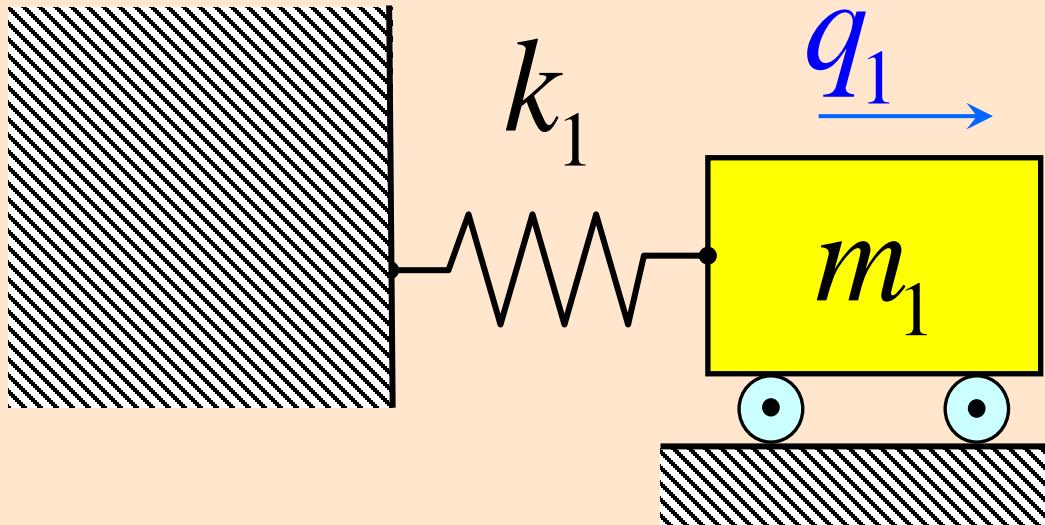
Case-1 Study: Equations of Motion



$$\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \begin{Bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{Bmatrix} + \begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} q_1 \\ q_2 \\ q_3 \end{Bmatrix} = \begin{Bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{Bmatrix}$$

Introduction into FEM: Fixed Boundary Case, 1-mass Example

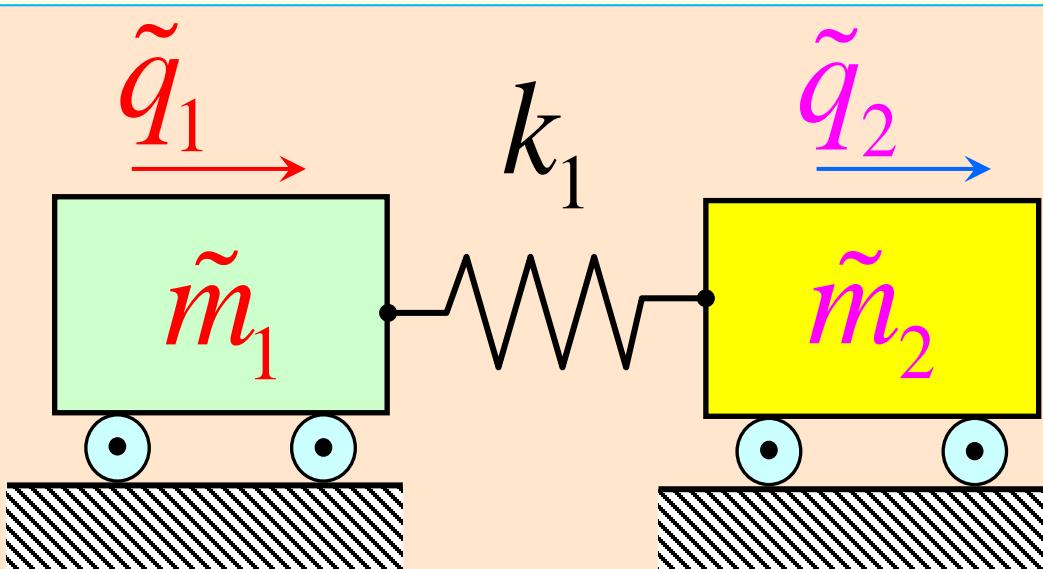
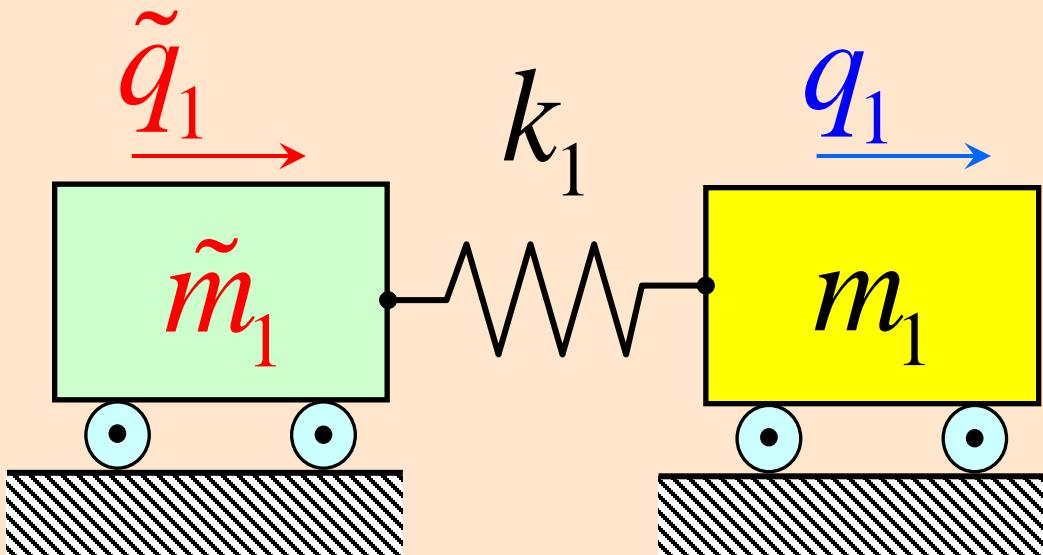
Original & Supplementary Systems



To model 1-mass and 1-spring case (as in Assignment, for N=1),

we start with a **SUPPLEMENTARY** system, having **ONE** spring and **TWO** masses, by adding to the **ORIGINAL** system a **DUMMY** mass, replicating left wall.

Rename Supplementary Parameters

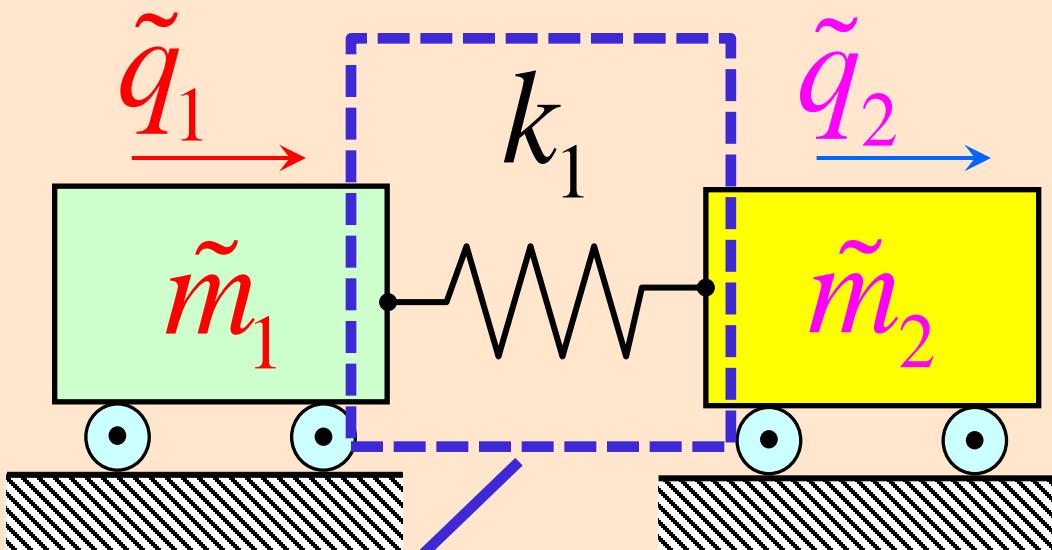


In the **SUPPLEMENTARY** system

**we then rename notations
(this is done for consistency and
programming convenience):**

q_1 is renamed to \tilde{q}_2
 m_1 is renamed to \tilde{m}_2

Assembling Stiffness Matrix



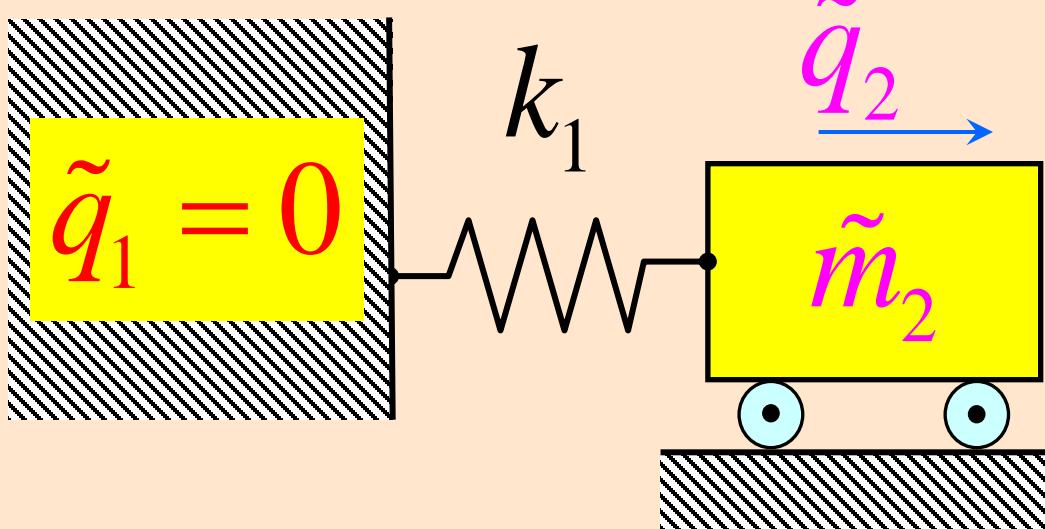
$$[\tilde{K}] = \begin{bmatrix} k_1 & -k_1 \\ -k_1 & k_1 \end{bmatrix}$$

GLOBAL STIFFNESS MATRIX for SUPPLEMENTARY System in this case is equal to the elementary stiffness matrix:

=

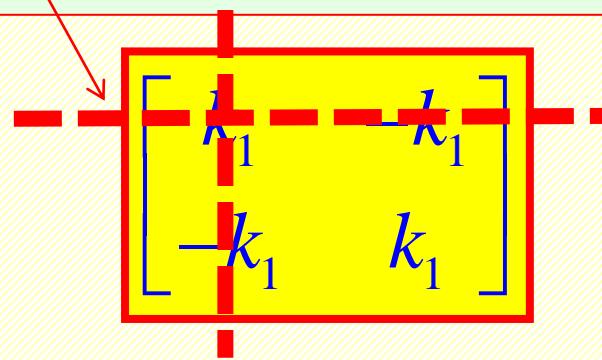
$$\begin{bmatrix} k_1 & -k_1 \\ -k_1 & k_1 \end{bmatrix}$$

Constraining Supplementary System



Now we are returning back to the ORIGINAL System, replacing the dummy mass in the SUPPLEMENTARY system with a wall.

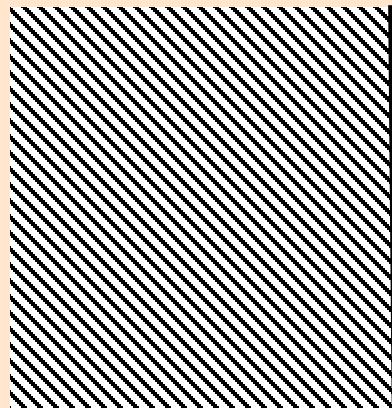
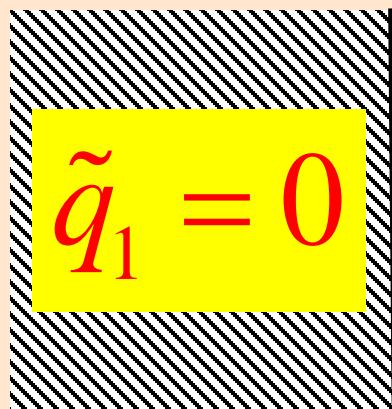
Mathematically, this is equivalent to setting $\tilde{q}_1 = 0$, which means that in the global matrix of the system first row and column can be crossed out:



reducing GLOBAL stiffness matrix to one number, k_1

$$[k_1]$$

Renaming Supplementary Parameters



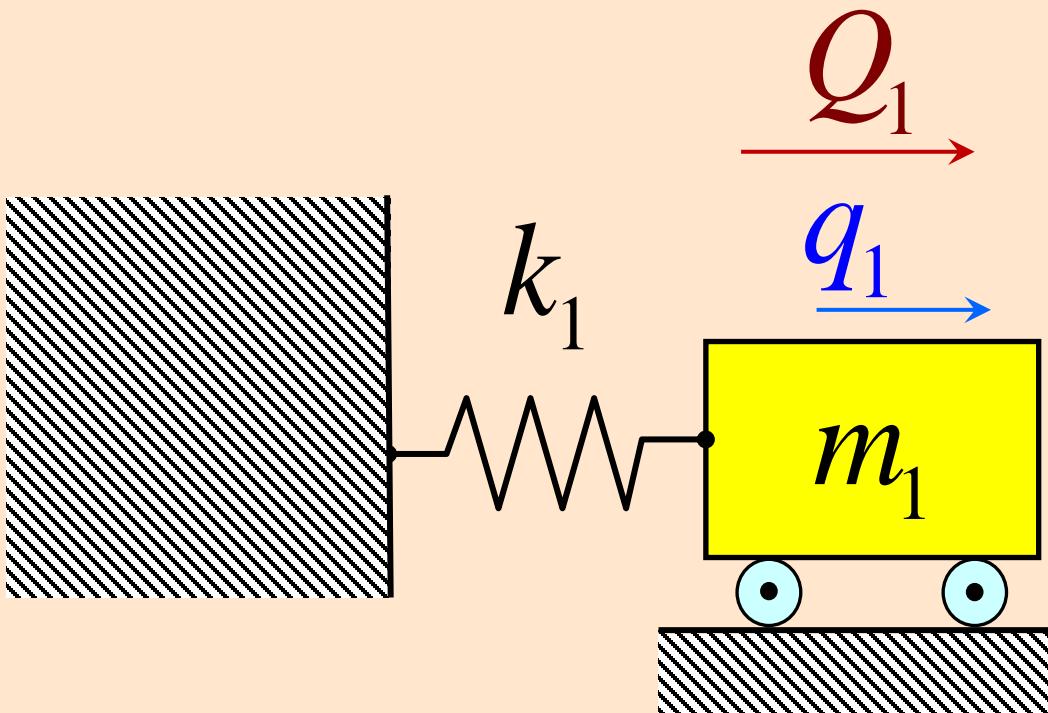
Let us rename notations,
returning back to the original
notations:

\tilde{q}_2 is renamed to q_1
 \tilde{m}_2 is renamed to m_1

**SOLUTION: System's
GLOBAL stiffness matrix is:**

$[k_1]$

Equation of Motion for Original Case

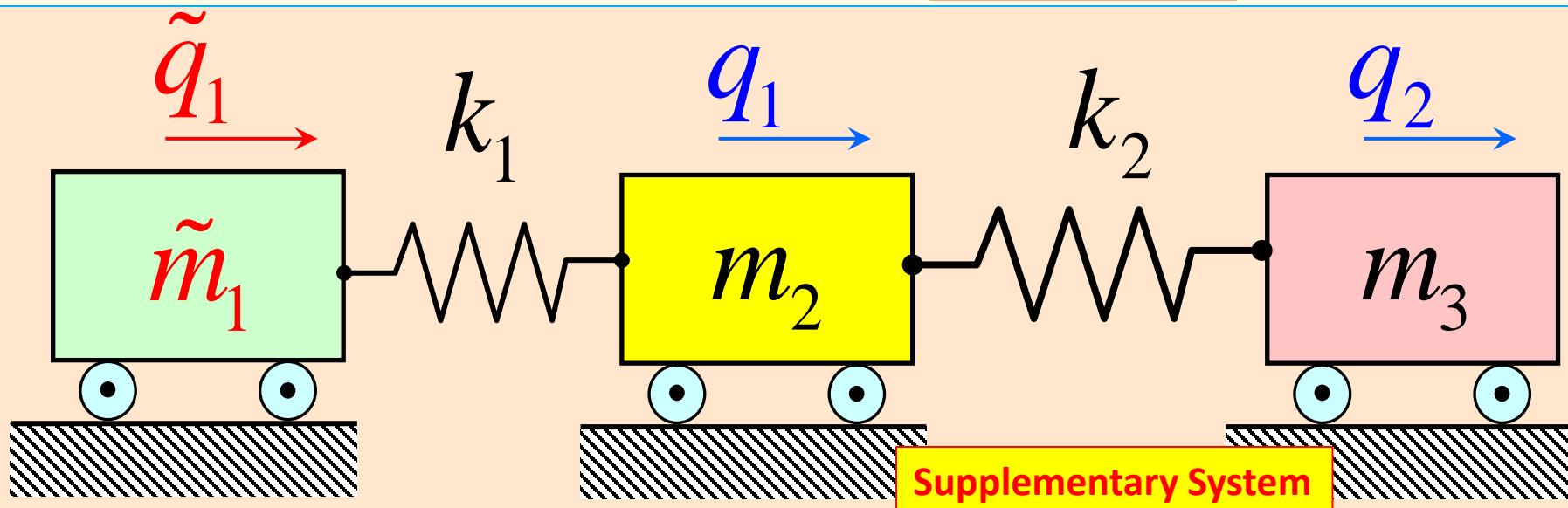
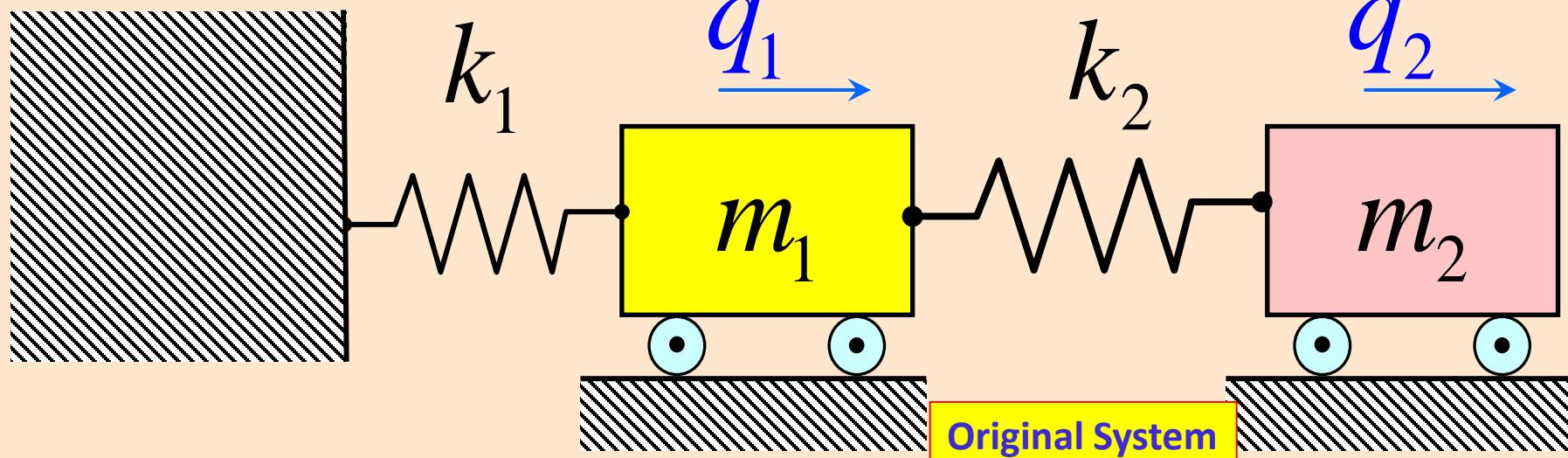


$$m_1 \ddot{q}_1 + k_1 q_1 = Q_1$$

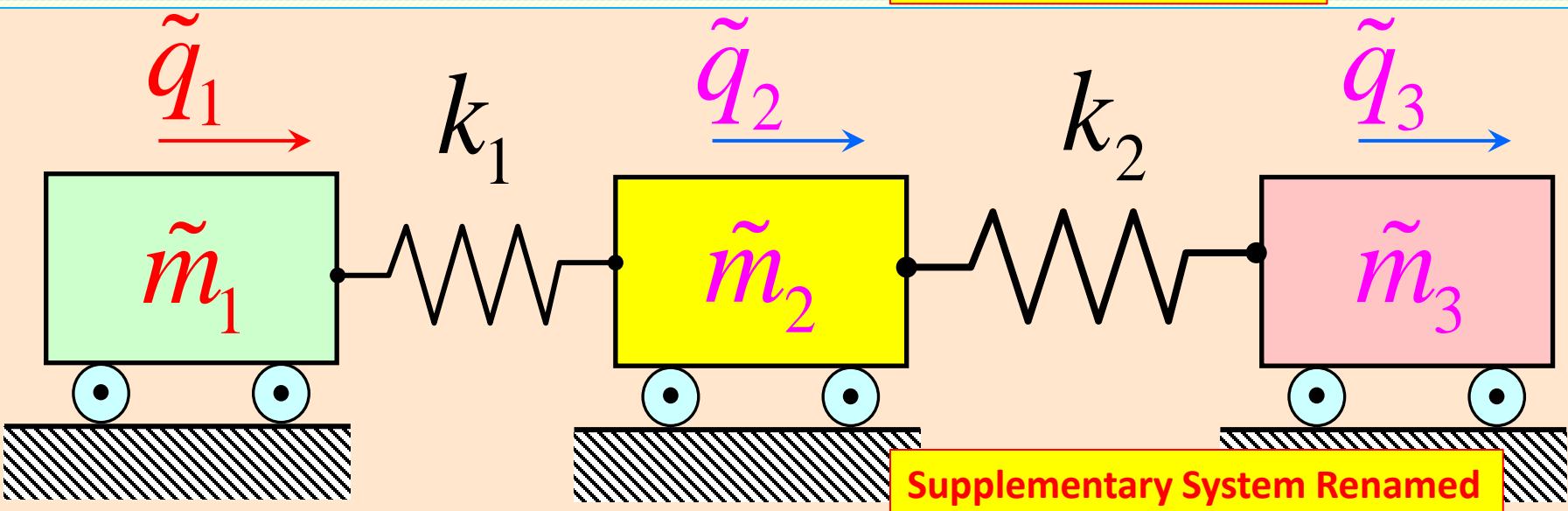
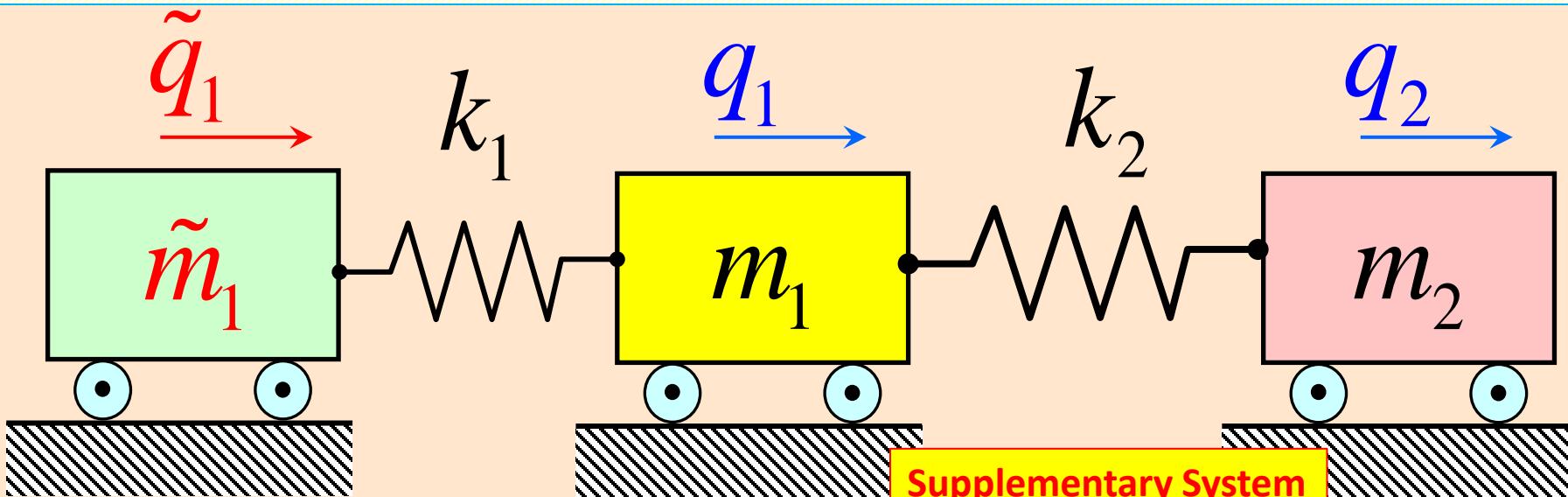
EOM for the Original System, derived by FEM

Introduction into FEM: Fixed Bounday Case, 2-masses Example

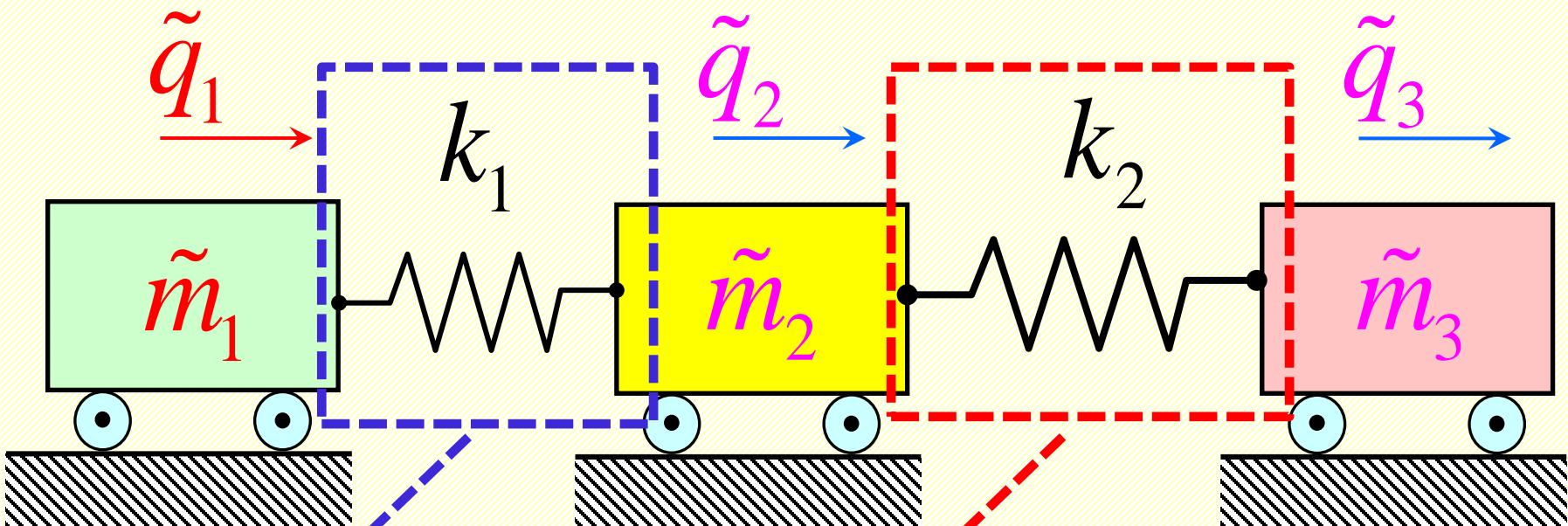
Original & Supplementary Systems



Rename Supplementary Parameters



Assembling Stiffness Matrix

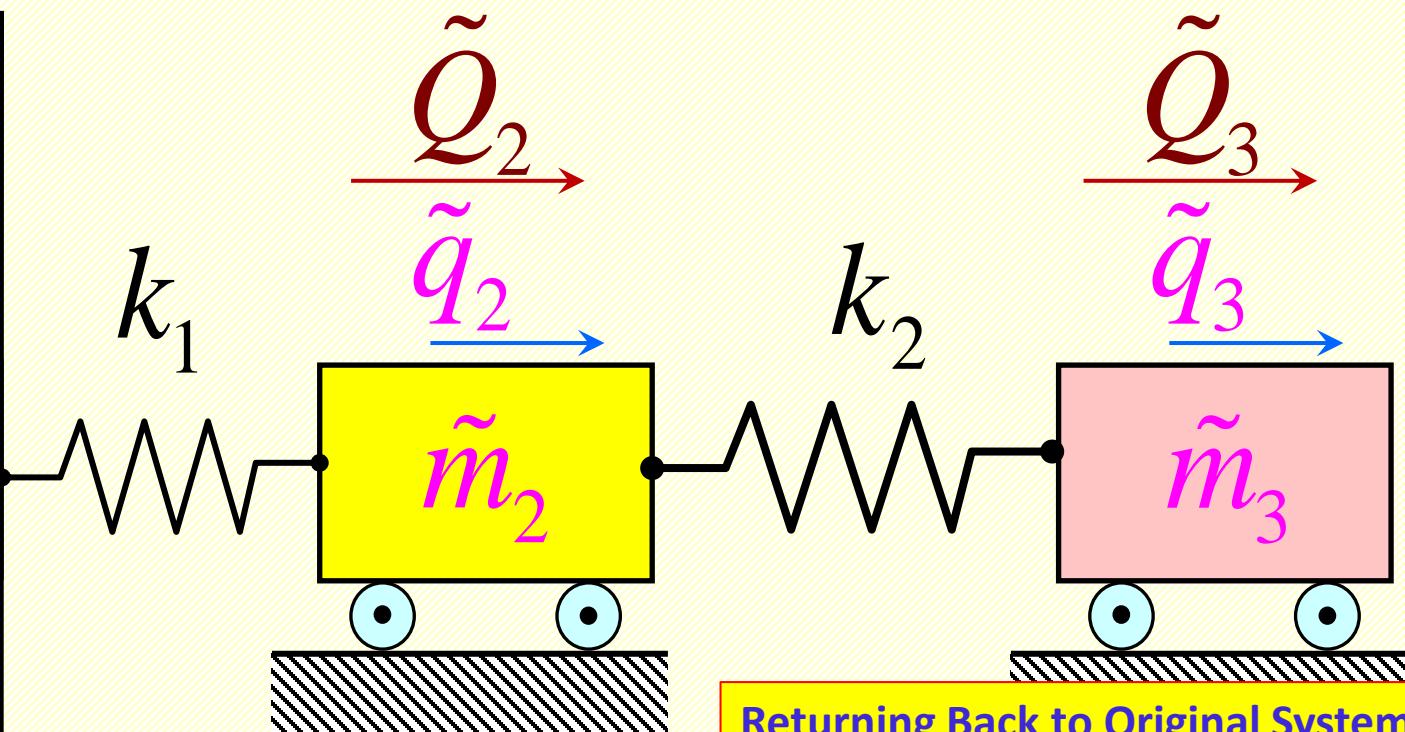


Supplementary System

$$\begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 & 0 \\ 0 & 0 & 0 \end{bmatrix} +
 \begin{bmatrix} 0 & 0 & 0 \\ 0 & k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} =
 \begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix}$$

Constraining Supplementary System

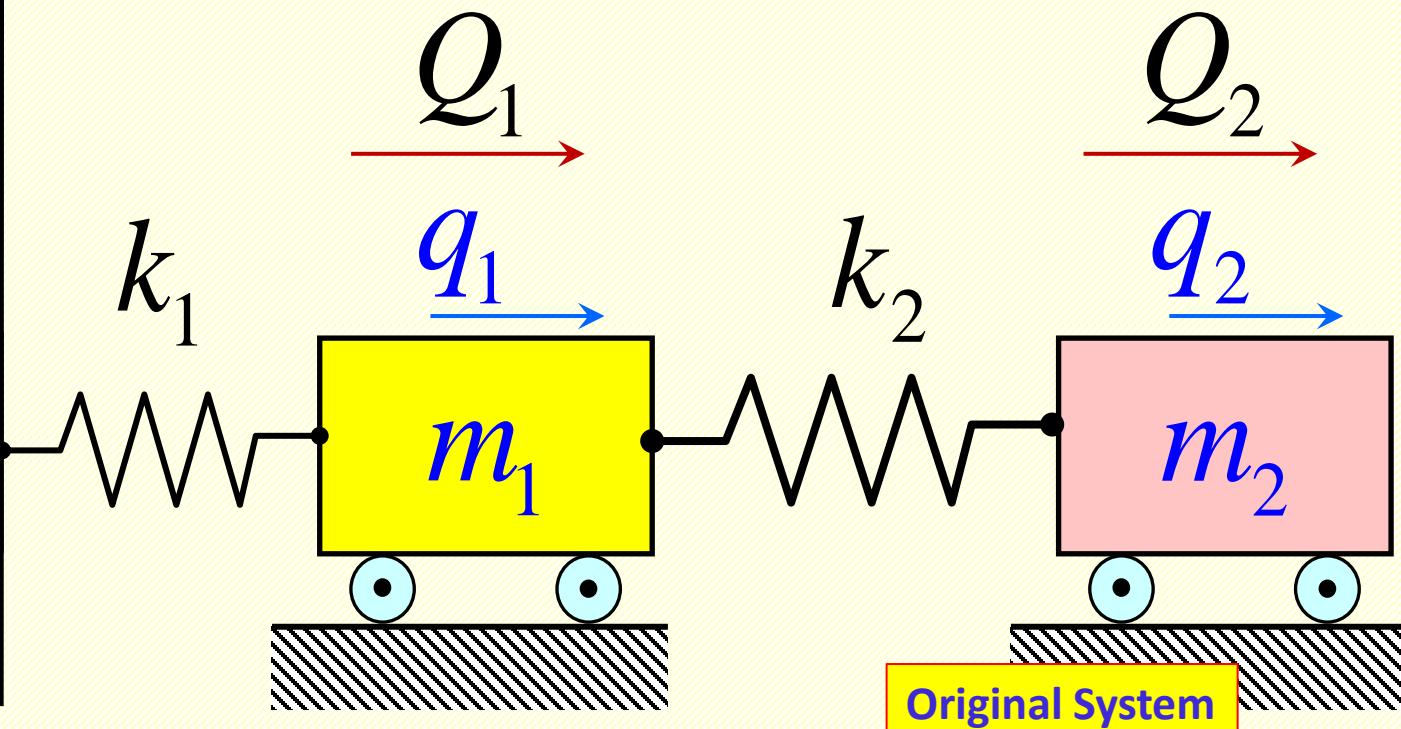
$$\tilde{q}_1 = 0$$



$$\begin{bmatrix} -\tilde{m}_1 & 0 & 0 \\ 0 & \tilde{m}_2 & 0 \\ 0 & 0 & \tilde{m}_3 \end{bmatrix} \begin{Bmatrix} \ddot{\tilde{q}}_1 \\ \ddot{\tilde{q}}_2 \\ \ddot{\tilde{q}}_3 \end{Bmatrix} + \begin{bmatrix} \cancel{k}_1 & \cancel{k}_1 & 0 \\ -\cancel{k}_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} \tilde{q}_1 \\ \tilde{q}_2 \\ \tilde{q}_3 \end{Bmatrix} = \begin{Bmatrix} \tilde{Q}_1 \\ \tilde{Q}_2 \\ \tilde{Q}_3 \end{Bmatrix}$$

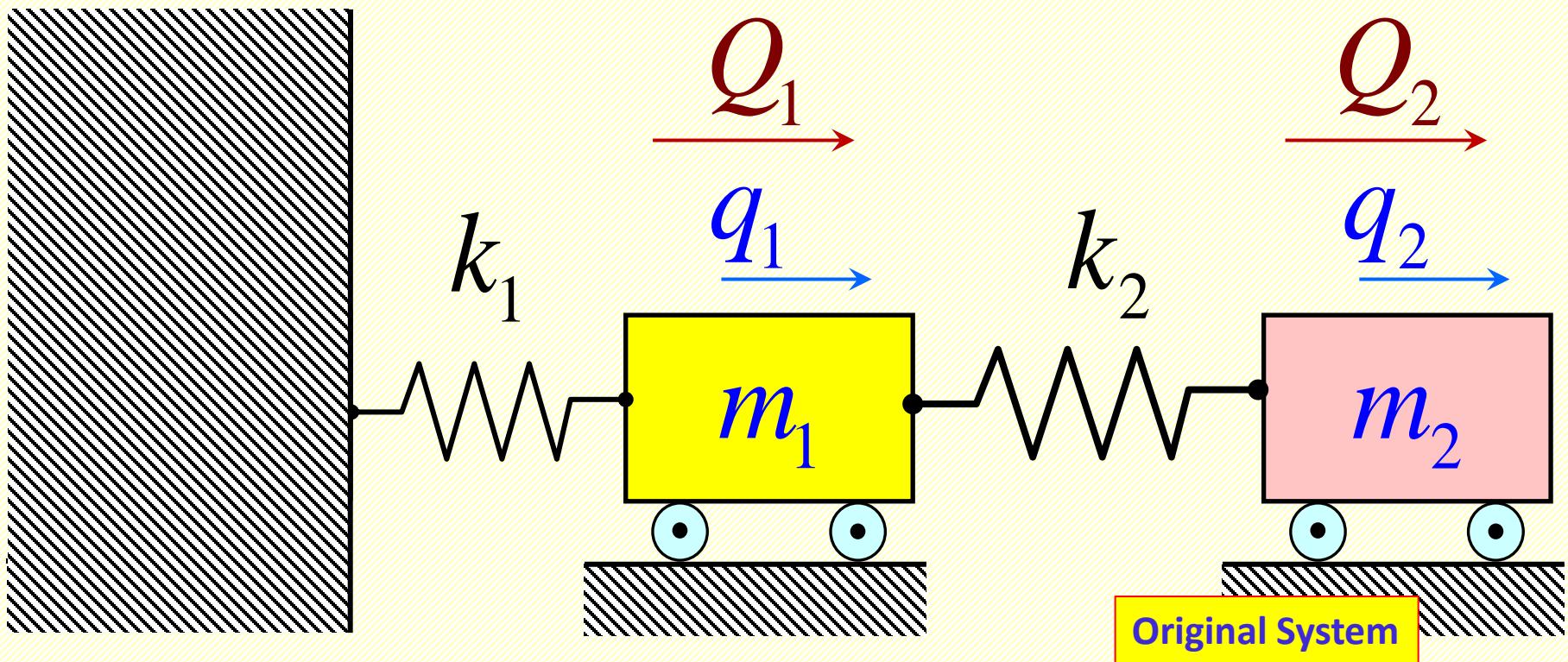
Renaming Supplementary Parameters

$$\tilde{q}_1 = 0$$



$$\begin{bmatrix} -\tilde{m}_1 & 0 & 0 \\ 0 & m_1 & 0 \\ 0 & 0 & m_2 \end{bmatrix} \begin{Bmatrix} \ddot{\tilde{q}}_1 \\ \ddot{q}_1 \\ \ddot{q}_2 \end{Bmatrix} + \begin{bmatrix} \tilde{k}_1 & k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} \tilde{q}_1 \\ q_1 \\ q_2 \end{Bmatrix} = \begin{Bmatrix} \tilde{Q}_1 \\ Q_1 \\ Q_2 \end{Bmatrix}$$

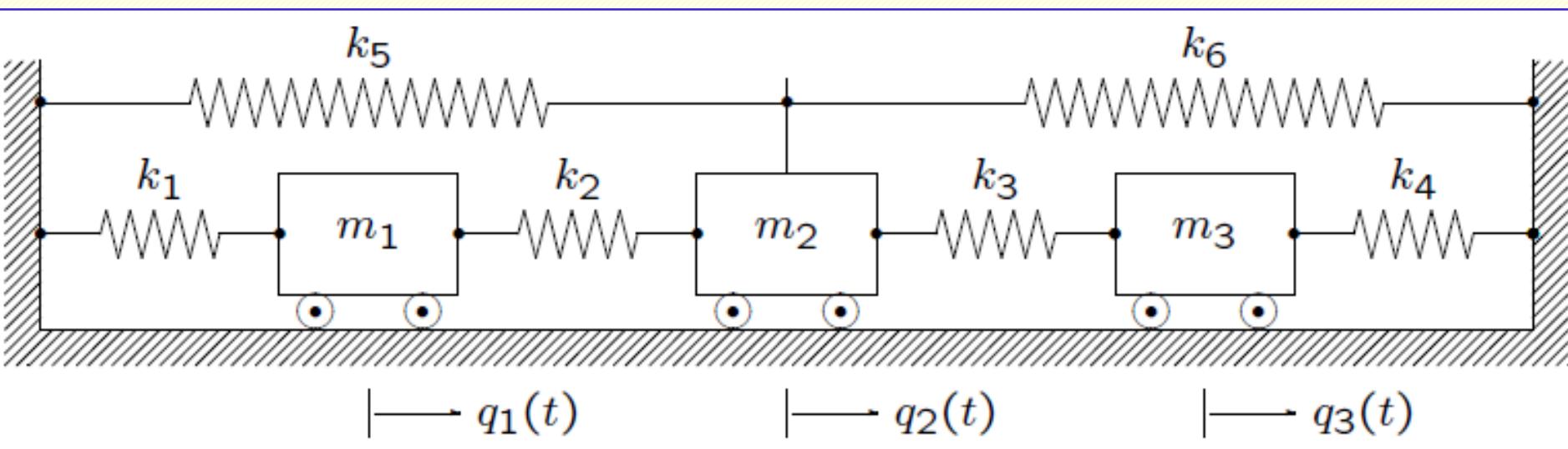
Equation of Motion for Original Case



$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \begin{Bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{Bmatrix} + \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} q_1 \\ q_2 \end{Bmatrix} = \begin{Bmatrix} Q_1 \\ Q_2 \end{Bmatrix}$$

Introduction into FEM: Homework Examples: try to derive

Case-3 Study: $[m]$ & $[k]$ matrices



$$[m] = \begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix};$$

$$[k] = \begin{bmatrix} k_1 + k_2 & -k_2 & 0 \\ -k_2 & k_2 + k_3 + k_5 + k_6 & -k_3 \\ 0 & -k_3 & k_3 + k_4 \end{bmatrix}$$

Case-3 Study: EOM with FBDs

$$\begin{cases} m_1 \ddot{q}_1 = -k_1 q_1 - k_2 (q_1 - q_2), \\ m_2 \ddot{q}_2 = -k_5 q_2 + k_2 (q_1 - q_2) - k_3 (q_2 - q_3) - k_6 q_2, \\ m_3 \ddot{q}_3 = -k_4 q_3 + k_3 (q_2 - q_3). \end{cases}$$

$$+ \begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \begin{Bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{Bmatrix} + \\ + \begin{bmatrix} k_1 + k_2 & -k_2 & 0 \\ -k_2 & k_2 + k_3 + k_5 + k_6 & -k_3 \\ 0 & -k_3 & k_3 + k_4 \end{bmatrix} \begin{Bmatrix} q_1 \\ q_2 \\ q_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}.$$

FEM:

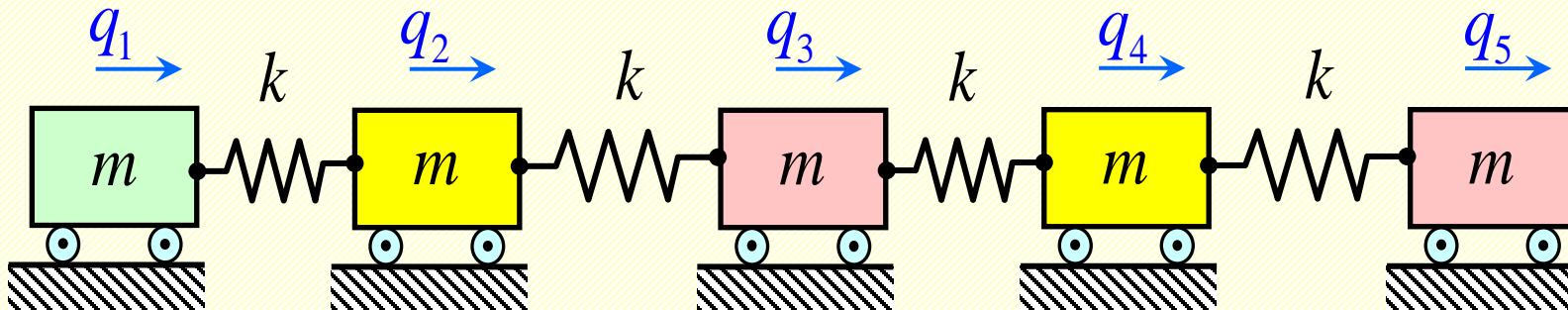
CHAIN MASS-SPRING SYSTEMS

Method-1

**Modelling an UNCONSTRAINED
Supplementary System First and
then Removing Constrained DOFs**

A-1: 5-DOF “Mock-Up” System

This is a SUPPLEMENTARY SYSTEM (not exactly as in A1), with NO CONSTRAINTS, where left wall is replaced with a left mass with added Degree-of-Freedom



Creation of the GLOBAL MASS MATRIX of the SUPPLEMENTARY system is simple:

$$[M] = \begin{bmatrix} m & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 \\ 0 & 0 & 0 & m & 0 \\ 0 & 0 & 0 & 0 & m \end{bmatrix}$$

N is a number of FEs;

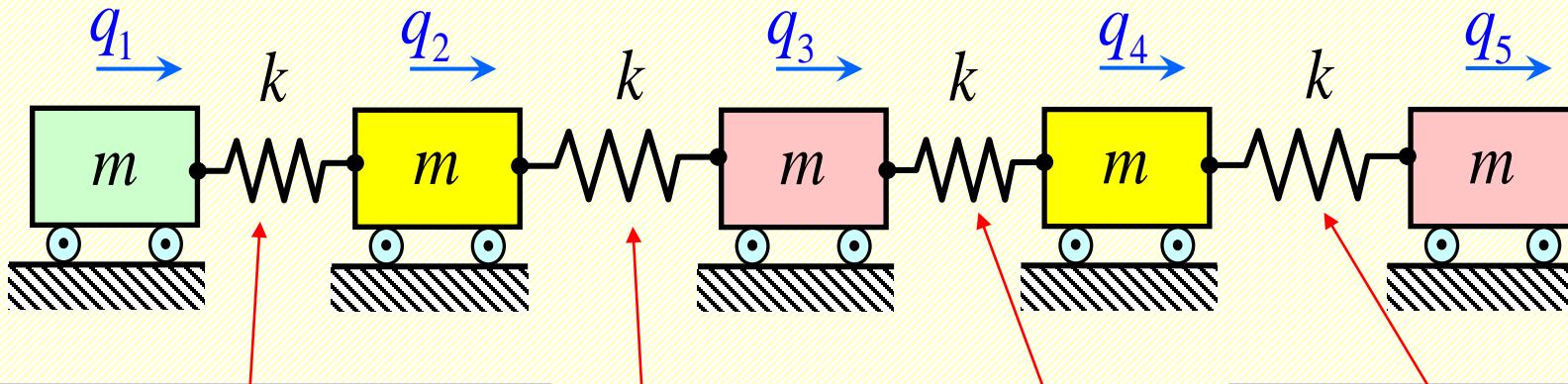
In MATLAB: $N=4$; $m=7$; $M=\text{eye}(N+1)*m;$

N+1 is a number of DOFs

here **N** is the number of springs,
hence FINITE ELEMENTS!

A-1: 5-DOF “Mock-Up” System

Creation of the GLOBAL STIFFNESS MATRIX can be done, using FEM:



$$\begin{array}{c} DOF \quad 1 \quad 2 \\ [k_{12}] = \begin{matrix} 1 & \begin{bmatrix} k & -k \\ -k & k \end{bmatrix} \\ 2 & \end{matrix} \end{array}$$

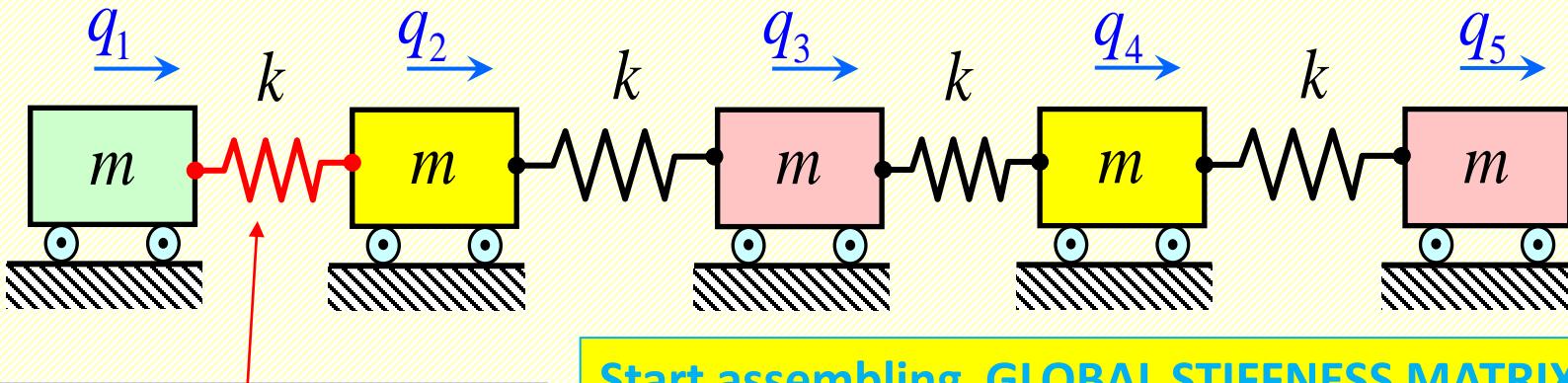
$$\begin{array}{c} DOF \quad 4 \quad 5 \\ [k_{45}] = \begin{matrix} 4 & \begin{bmatrix} k & -k \\ -k & k \end{bmatrix} \\ 5 & \end{matrix} \end{array}$$

$$\begin{array}{c} DOF \quad 2 \quad 3 \\ [k_{23}] = \begin{matrix} 2 & \begin{bmatrix} k & -k \\ -k & k \end{bmatrix} \\ 3 & \end{matrix} \end{array}$$

$$\begin{array}{c} DOF \quad 3 \quad 4 \\ [k_{34}] = \begin{matrix} 3 & \begin{bmatrix} k & -k \\ -k & k \end{bmatrix} \\ 4 & \end{matrix} \end{array}$$

Assembling [K]: FE-1 contribution

Creation of the GLOBAL STIFFNESS MATRIX can be done, using FEM:



DOF 1 2

$$[k_{12}] = \begin{bmatrix} 1 & \begin{bmatrix} k & -k \\ -k & k \end{bmatrix} \\ 2 & \end{bmatrix}$$

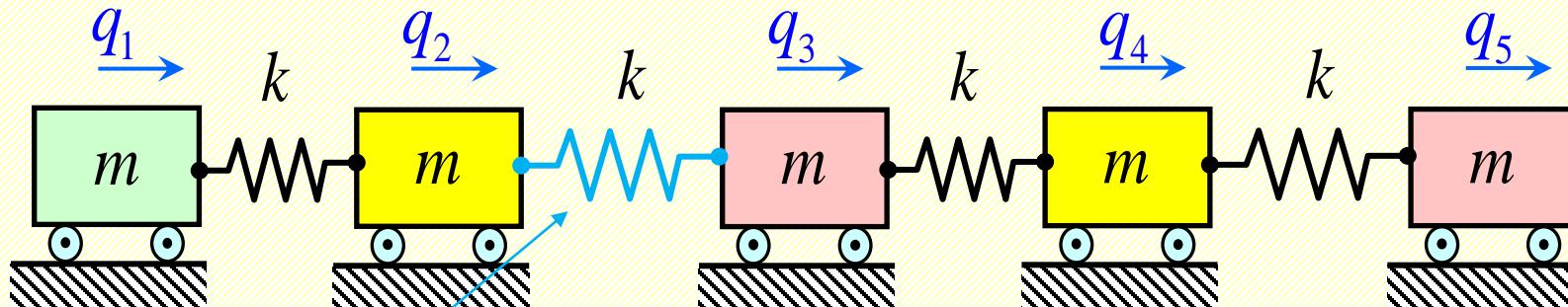
Start assembling GLOBAL STIFFNESS MATRIX: FE #1:

<i>DOF</i>	1	2	3	4	5
1	k	$-k$	0	0	0
2	$-k$	k	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

$$[K_{12}] =$$

Assembling [K]: FE-2 contribution

Creation of the GLOBAL STIFFNESS MATRIX can be done, using FEM:



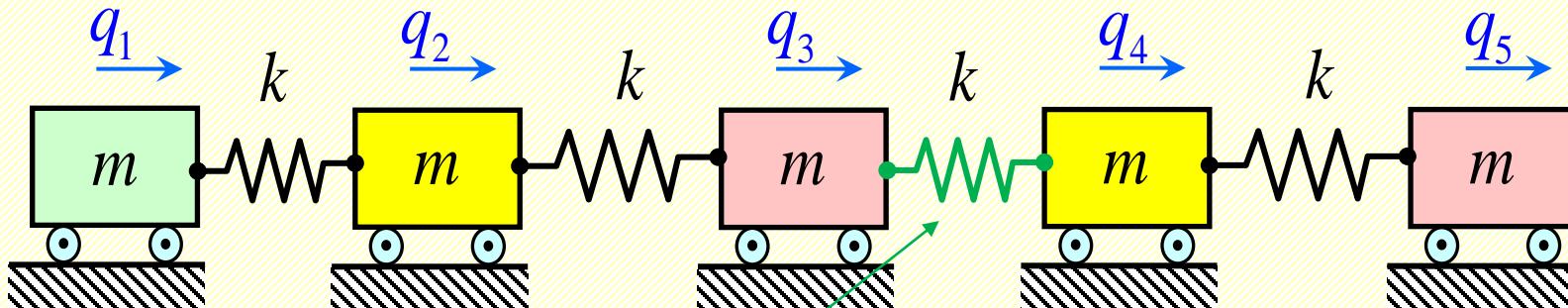
Continue assembling GLOBAL STIFFNESS MATRIX: FE #2:

$$DOF \quad 2 \quad 3$$
$$[k_{23}] = \begin{matrix} 2 & \begin{bmatrix} k & -k \\ -k & k \end{bmatrix} \\ 3 & \end{matrix}$$

$$DOF \quad 1 \quad 2 \quad 3 \quad 4 \quad 5$$
$$[K_{23}] = \begin{matrix} 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & \boxed{k} & -k & 0 & 0 \\ 3 & 0 & -k & k & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Assembling [K]: FE-3 contribution

Creation of the GLOBAL STIFFNESS MATRIX can be done, using FEM:



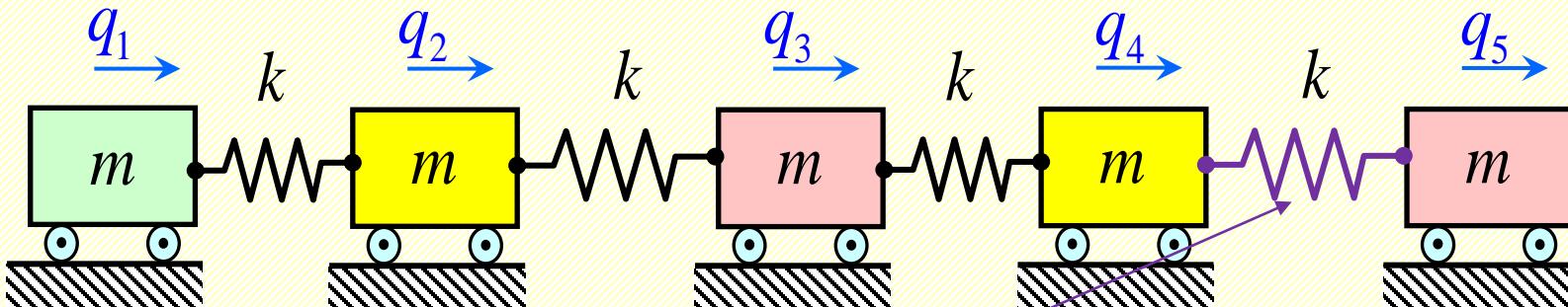
Continue assembling GLOBAL STIFFNESS MATRIX: FE #3:

$$[k_{34}] = \begin{matrix} DOF & 3 & 4 \\ 3 & k & -k \\ 4 & -k & k \end{matrix}$$

DOF	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	k	-k	0
4	0	0	-k	k	0
5	0	0	0	0	0

Assembling [K]: FE-4 contribution

Creation of the GLOBAL STIFFNESS MATRIX can be done, using FEM:



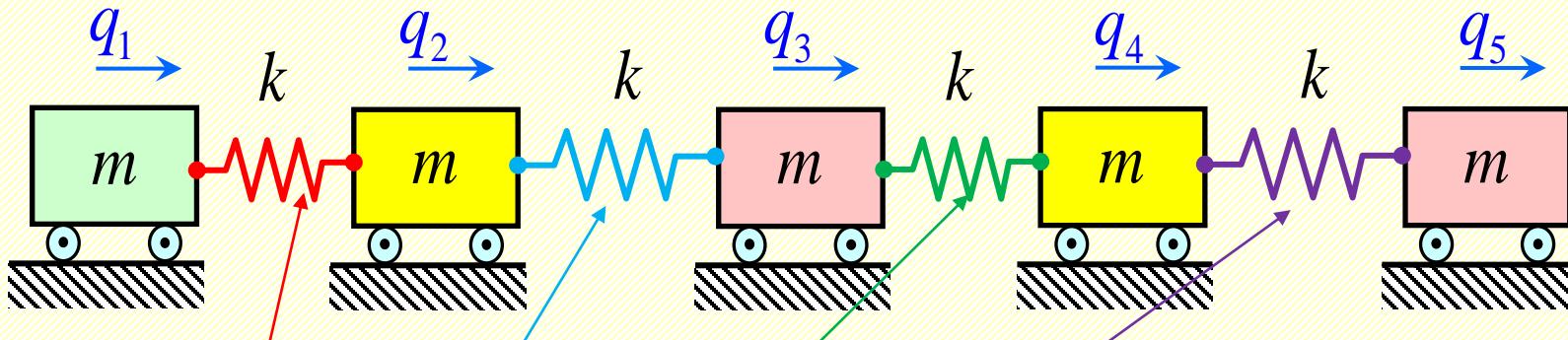
Continue assembling GLOBAL STIFFNESS MATRIX: FE #4:

$$[k_{45}] = \begin{matrix} DOF & 4 & 5 \\ 4 & k & -k \\ 5 & -k & k \end{matrix}$$

DOF	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	k	$-k$
5	0	0	0	$-k$	k

Assembling $[K]$: FEM-4 contribution

Creation of the GLOBAL STIFFNESS MATRIX can be done, using FEM:

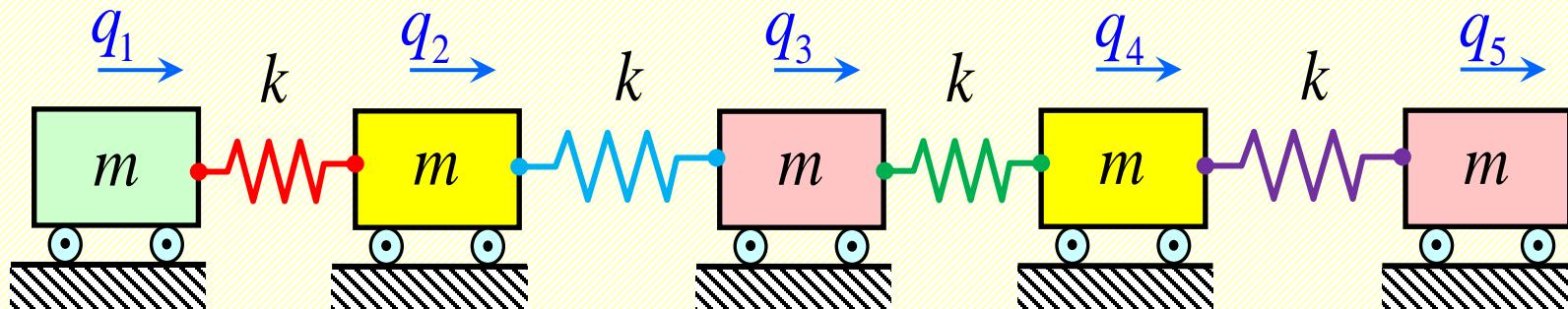


Assembling GLOBAL STIFFNESS MATRIX: ALL FE CONTRIBUTIONS ARE ADDED:

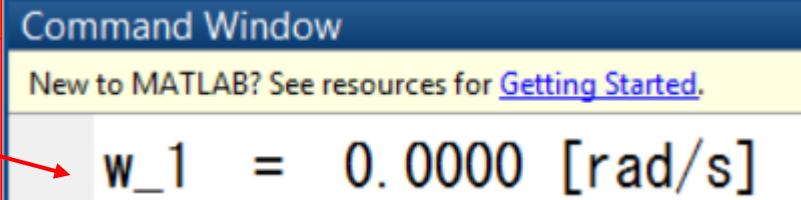
$$[K] = [K_{12}] + [K_{23}] + [K_{34}] + [K_{45}]$$

$$[K] = \begin{bmatrix} k & -k & 0 & 0 & 0 \\ -k & 2k & -k & 0 & 0 \\ 0 & -k & 2k & -k & 0 \\ 0 & 0 & -k & 2k & -k \\ 0 & 0 & 0 & -k & k \end{bmatrix}$$

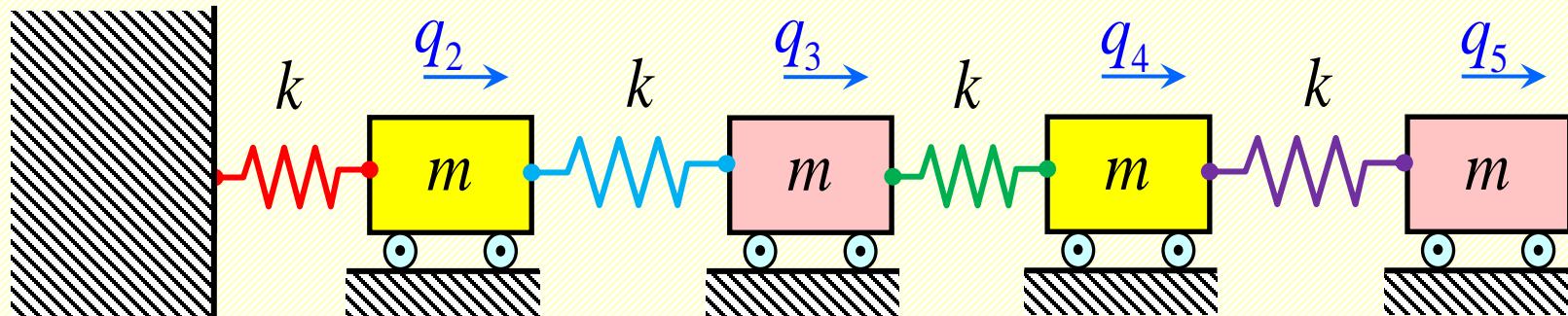
MATLAB Script



```
% SUPPLEMENTARY PROBLEM : ALL MASSES ARE FREE !!!  
% Designed by Prof P.M.Trivailo (C)2020  
N=4; m=7; k=500; % N is a number of springs (= number of FEs)  
  
MM=eye(N+1)*m; % GLOBAL MASS MATRIX - ready  
ke=[k -k; -k k]; % ELEMENT STIFFNESS MATRIX  
KK=zeros(N+1,N+1); % PROVISION FOR GLOBAL STIFFNESS MATRIX  
for ii=1:N  
    KK(ii:ii+1, ii:ii+1)= KK(ii:ii+1, ii:ii+1)+ke;  
end  
  
[U2,D2]=eig(KK,MM); % SOLVING EIGENVALUE PROBLEM  
disp(sprintf('w_1 = %7.4f [rad/s]',sqrt(D2(1,1))))  
commandwindow
```



MATLAB Script



```
%% CONSTRAINED SYSTEM
% Designed by Prof P.M.Trivailo (C)2020
N=4; m=7; k=500;

MM=eye(N+1)*m;          % GLOBAL MASS MATRIX - ready
ke=[k -k; -k k];        % ELEMENT STIFFNESS MATRIX
KK=zeros(N+1,N+1);      % PROVISION FOR GLOBAL STIFFNESS MATRIX
for ii=1:N
    KK(ii:ii+1, ii:ii+1)= KK(ii:ii+1, ii:ii+1)+ke;
end
KK(1,:)=[]; KK(:,1)=[]; MM(1,:)=[]; MM(:,1)=[]; % ADD CONSTRAINING WALL!

[U,D]=eig(KK,MM); % SOLVING EIGENVALUE PROBLEM
disp(sprintf('w_1 = %7.4f [rad/s]',sqrt(D(1,1))))
commandwindow
```

First frequency is NOT ZERO, because system IS CONSTRAINED

Command Window

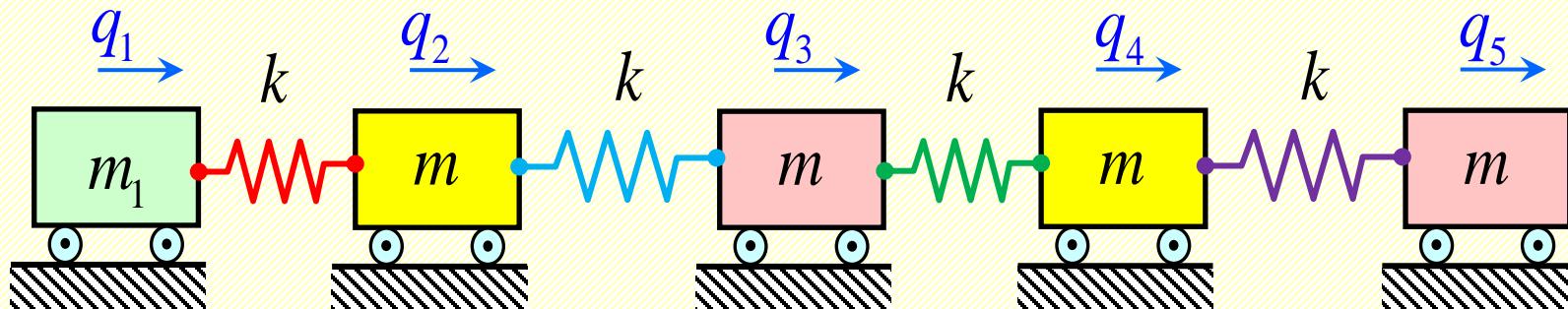
New to MATLAB? See resources for [Getting Started](#).

w_1 = 2.9352 [rad/s]

Method-2

Modelling of a constraining Wall with a Very Heavy Mass

MATLAB Script



```
% SOLUTION-2 : MODELLING WALL CONSTRAINT AS A HEAVY EXTRA ADDED MASS !!!  
% Designed by Prof P.M.Trivailo (C)2020  
N=4; k=500;  
K=zeros(N+1,N+1); ke=[1 -1; -1 1]*k;  
  
for i=1:N  
    K(i:i+1, i:i+1) = K(i:i+1, i:i+1) + ke;  
end  
M=eye(N+1)*5;  
  
M(1,1)=10^12; % MAKING MASS-1 (m1) VERY HEAVY!!! TO REPLICATE WALL CONSTRAINT
```

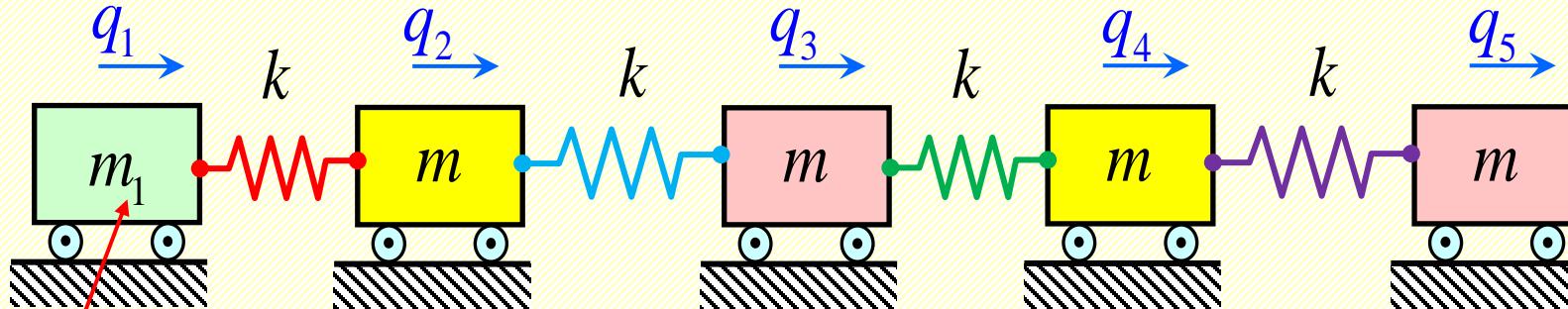
```
[U,D]=eig(K,M);  
disp(sprintf('w_1 = %7.4f [rad/s]',sqrt(D(1,1))))  
disp(sprintf('w_2 = %7.4f [rad/s]',sqrt(D(2,2))))  
commandwindow
```

First frequency is ZERO, because system admits rigid body mode

Command Window
New to MATLAB? See resources for [Getting Started](#).

w_1 = 0.0000 [rad/s]
w_2 = 2.9352 [rad/s]

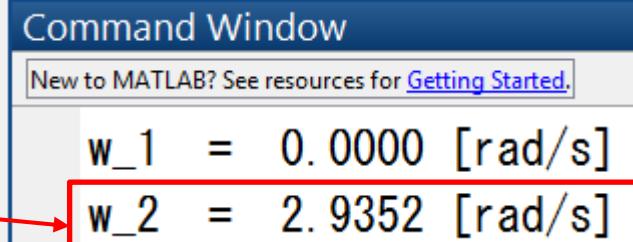
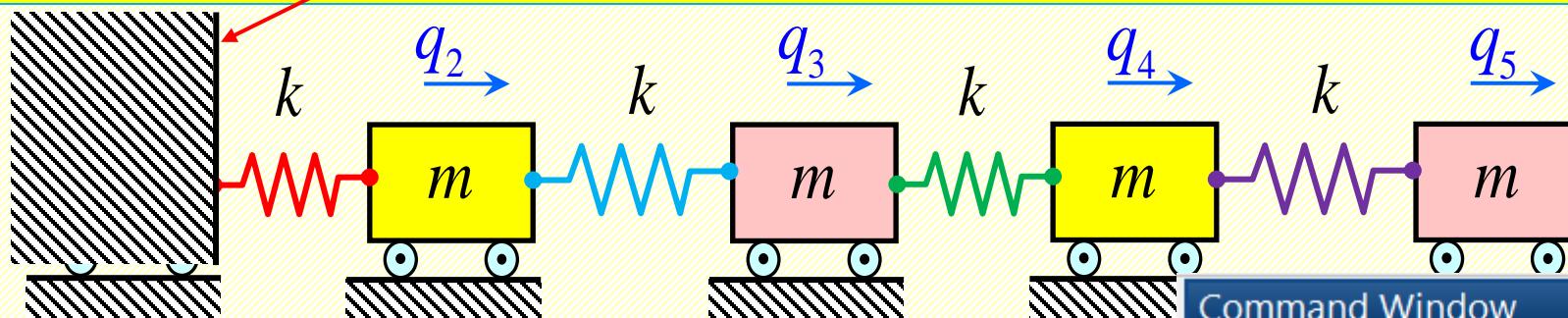
Interpretation of Results



If mass-1 is very heavy, it can replicate a WALL CONSTRAINT. However, with this technique, we have effective reduction by 1 of the DOFs in the unconstrained system.

Hint for A-1:

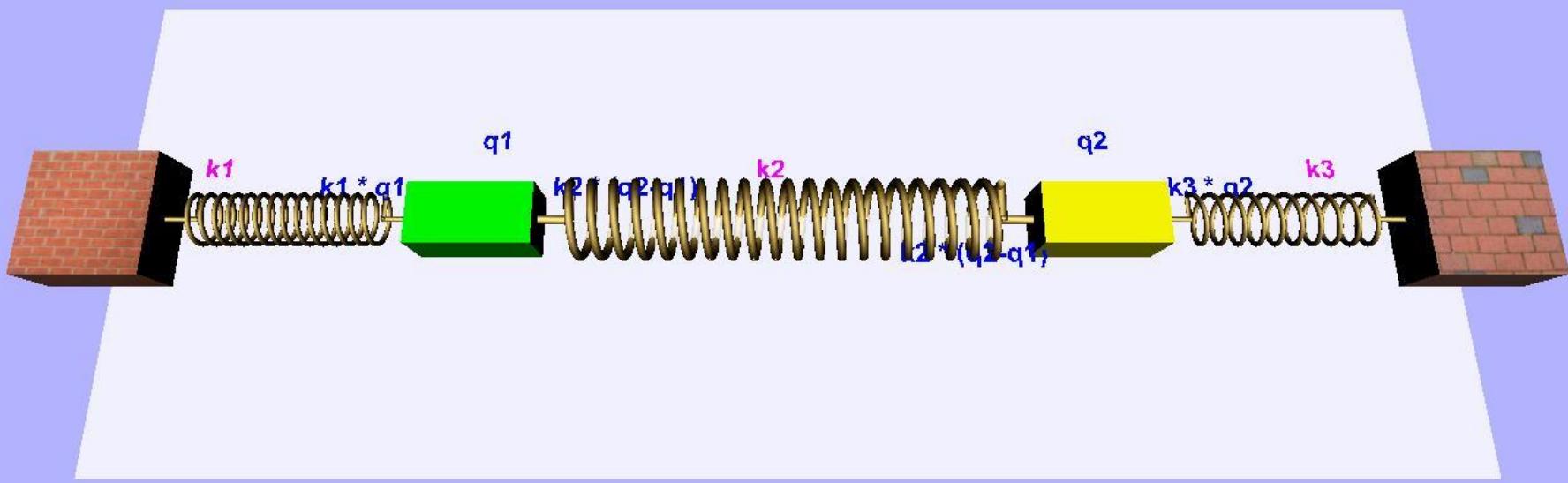
Therefore, in order to get 1st elastic frequency, we need to take second frequency of the FREE system.
In order to get 2nd elastic frequency, we need to take third frequency of the FREE system, etc.



First elastic frequency for the CONSTRAINED system

Multi-DOF mass-spring systems: Derivation of EOM using Newton's Second Law & Free-Body Diagrams (FBDs).

Demo in Virtual Reality



RMIT, School of Aerospace, Mech & Manuf Engng
Copyright P.M.Trivailo July-2008

To use sensors **PLACE POINTER OVER SPRING**

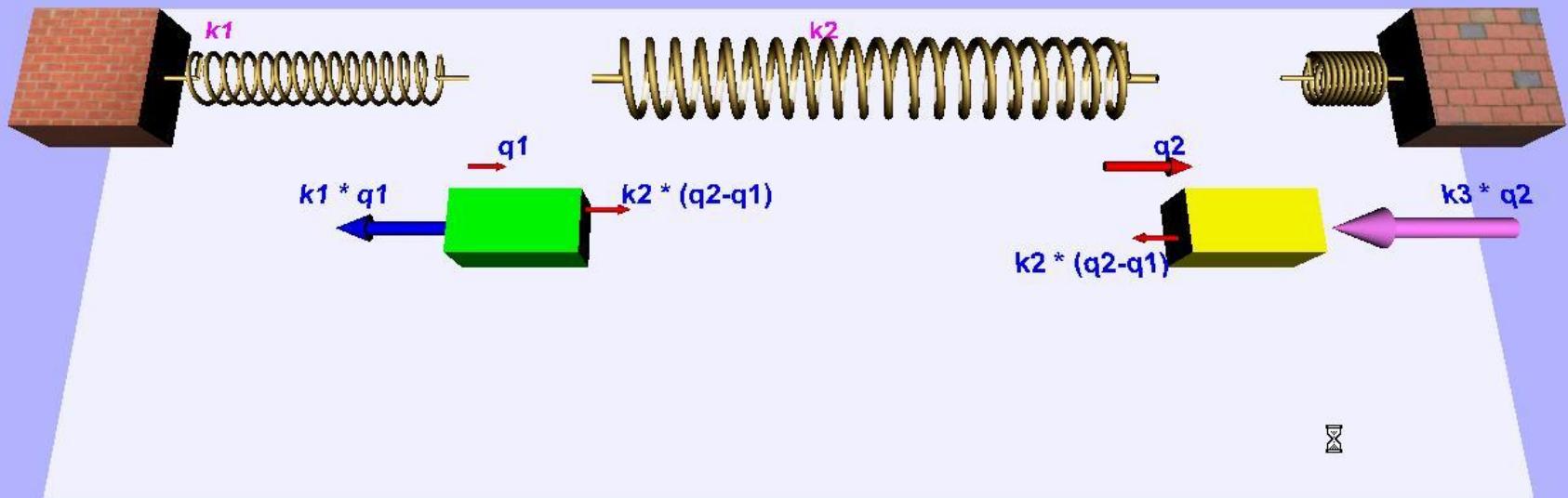
PARALLEL GRAPHICS



RMIT, School of Aerospace, Mech & Manuf Engng
Copyright P.M.Trivailo July-2008

To use sensors PLACE POINTER OVER SPRING





RMIT, School of Aerospace, Mech & Manuf Engng
Copyright P.M.Trivailo July-2008

To use sensors PLACE POINTER OVER SPRING

CORTONA
VRML CLIENT

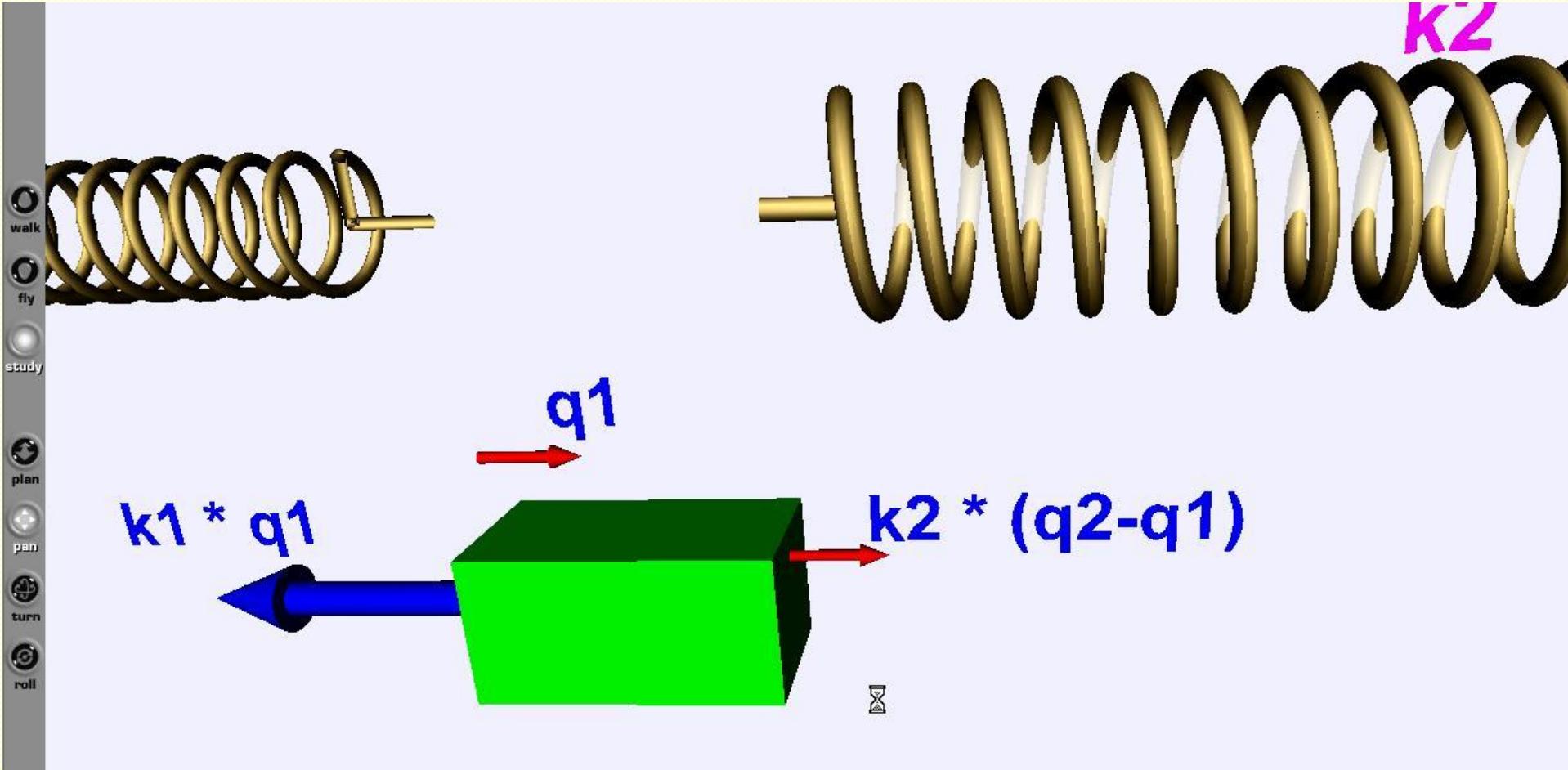
goto

align

view

restore

fit



RMIT, School of Aerospace, Mech & Manuf Engng
Copyright P.M.Trivailo July-2008

To use sensors PLACE POINTER OVER



goto



align



view



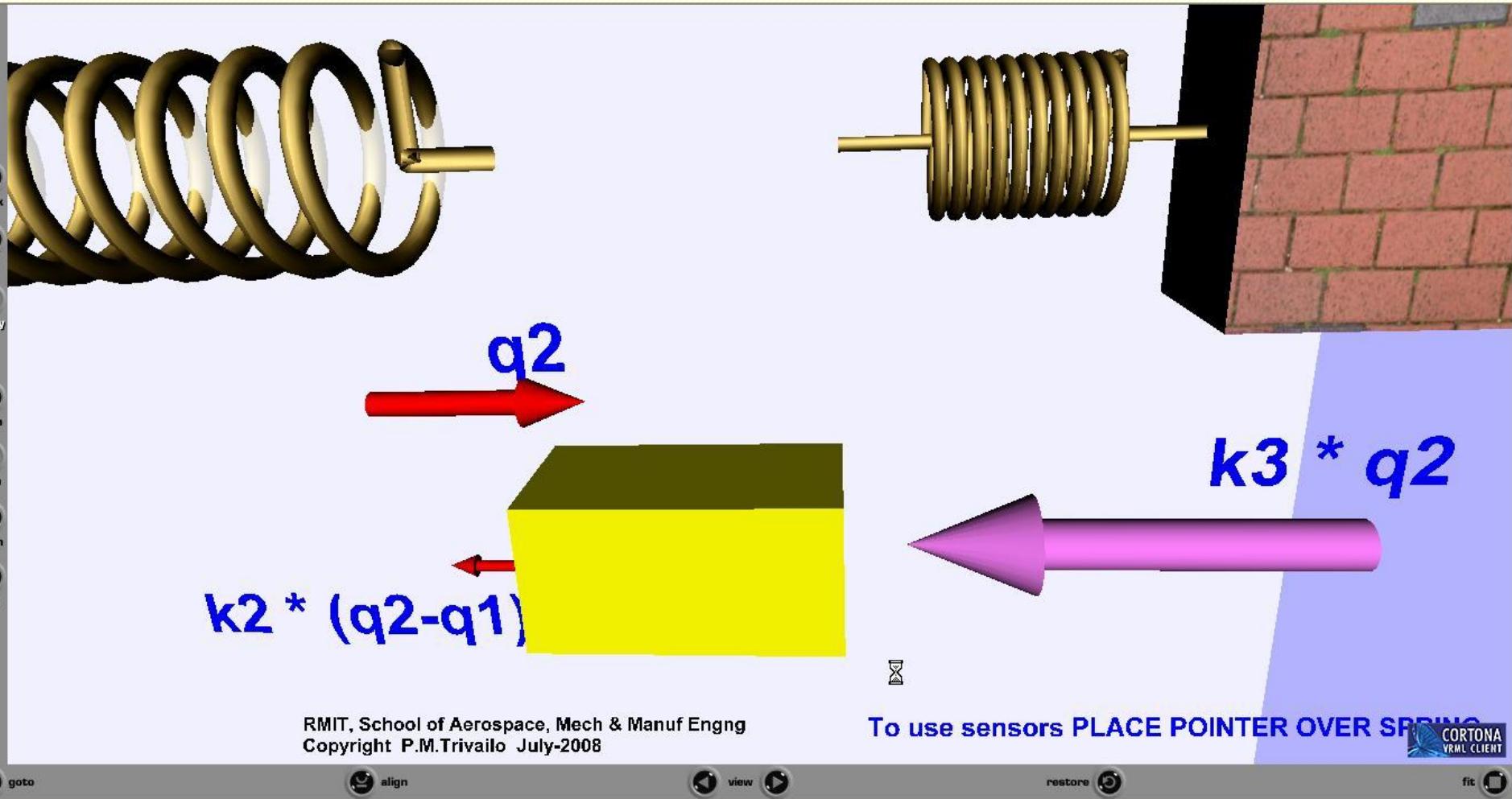
play



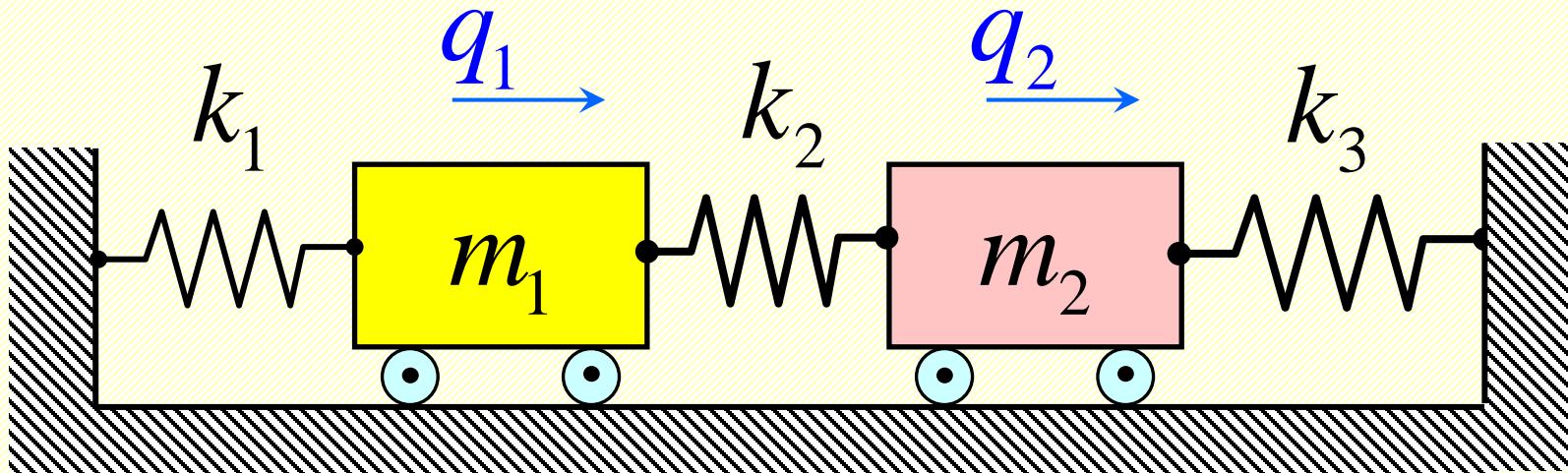
restore



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



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



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

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

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

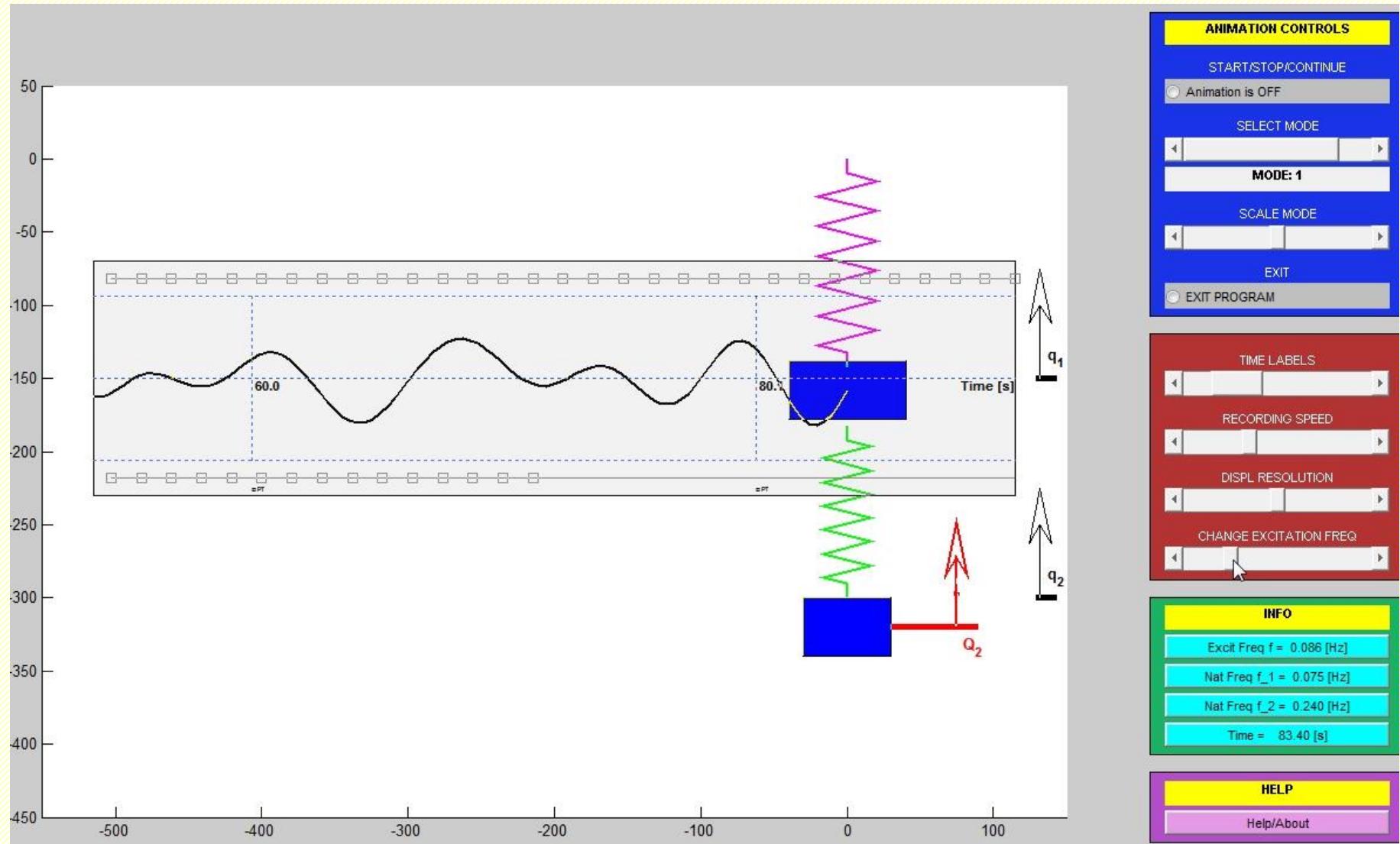
EQUATIONS IN STATE-SPACE FORM

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

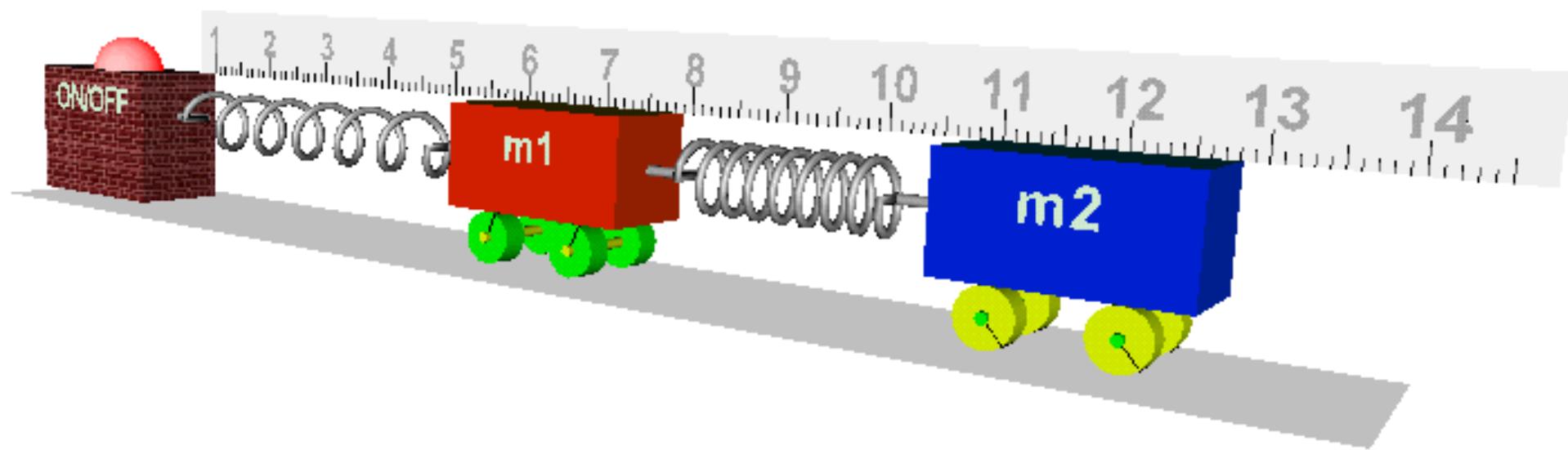
$$y = Cx + Du$$

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

SOLVING RESPONSES FOR MULTI-DOF SYSTEMS: MATLAB and ODExxx



Virtual Laboratory



Virtual Laboratory



EIGENVALUE PROBLEM FOR MULTI-DOF SYSTEMS: SOLVING EP, USING MATLAB

EIGENVALUE PROBLEM

Matrix Equations of Motion:



Equation of Motion of a Damped n-DOF System

$$[m]\{\ddot{q}(t)\} + [c]\{\dot{q}(t)\} + [k]\{q(t)\} = \{Q(t)\}$$

Equation of Free Motion of an Undamped n-DOF System

$$[m]\{\ddot{q}(t)\} + [k]\{q(t)\} = \{0\}$$

EIGENVALUE PROBLEM FORMULATION

$$[m]\{\ddot{q}(t)\} + [k]\{q(t)\} = \{0\}$$

$$\{q(t)\} = e^{i\omega t} \{u\}$$

Eigenvalue Problem for an n -DOF System:

$$[k]\{u\} = \omega^2[m]\{u\}$$

or

$$([k] - \omega^2[m])\{u\} = \{0\}.$$

(7)

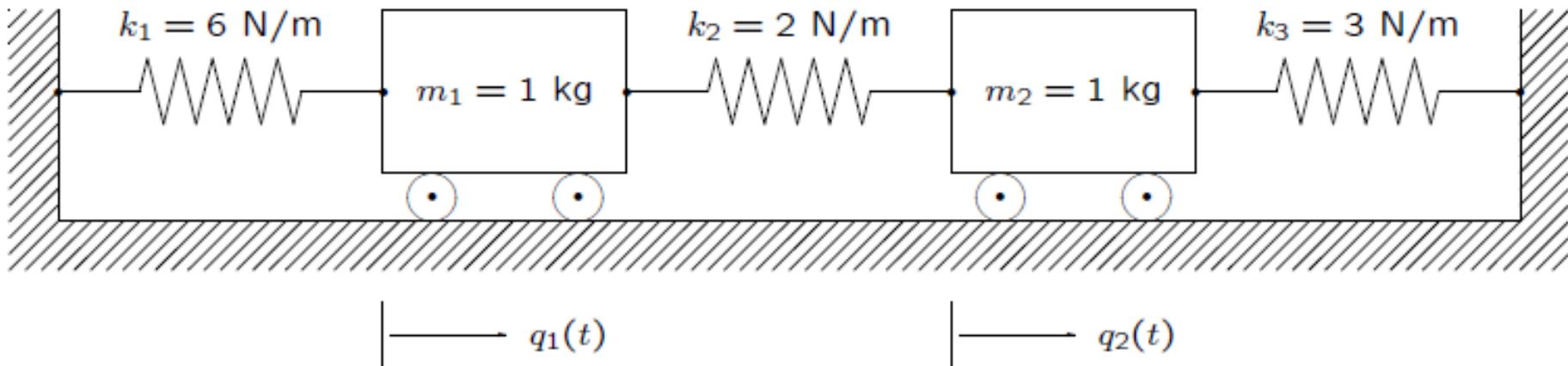
CHARACTERISTIC EQUATION

The problem of determining the values of the parameter ω^2 for which (7) possesses nontrivial solutions is known as eigenvalue problem.

Characteristic Equation for a n-DOF System:

$$\Delta(\omega^2) = |[k] - \omega^2[m]| = 0. \quad (8)$$

Example: 2-DOF MASS-SPRING SYSTEM



$$[m] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad [k] = \begin{bmatrix} -8 & -2 \\ -2 & 5 \end{bmatrix}; \quad \{q(t)\} = e^{i\omega t} \{u\}$$

$$-\omega^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} + \begin{bmatrix} -8 & -2 \\ -2 & 5 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}.$$

$$\begin{bmatrix} 8 - \omega^2 & -2 \\ -2 & 5 - \omega^2 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}.$$

$$\det \begin{pmatrix} 8 - \omega^2 & -2 \\ -2 & 5 - \omega^2 \end{pmatrix} = 0.$$

$$(\omega^2)^2 - 13(\omega^2) + 36 = 0.$$

$$\lambda_1 = [\omega^2]_1 = 4 \left(\frac{\text{rad}}{\text{s}} \right)^2; \quad \lambda_2 = [\omega^2]_2 = 9 \left(\frac{\text{rad}}{\text{s}} \right)^2$$

$$\lambda_1 = \omega_1^2 = 4 \left(\frac{\text{rad}}{\text{s}} \right)^2.$$

$$\begin{bmatrix} 8 - 4 & -2 \\ -2 & 5 - 4 \end{bmatrix} \begin{Bmatrix} u_{11} \\ u_{21} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}.$$
$$\begin{bmatrix} 4 & -2 \\ -2 & 1 \end{bmatrix} \begin{Bmatrix} u_{11} \\ u_{21} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}.$$

$$4u_{11} - 2u_{21} = 0$$

$$\begin{Bmatrix} u_{11} \\ u_{21} \end{Bmatrix} = \begin{Bmatrix} 1 \\ 2 \end{Bmatrix}, \begin{Bmatrix} 3 \\ 6 \end{Bmatrix}, \dots$$

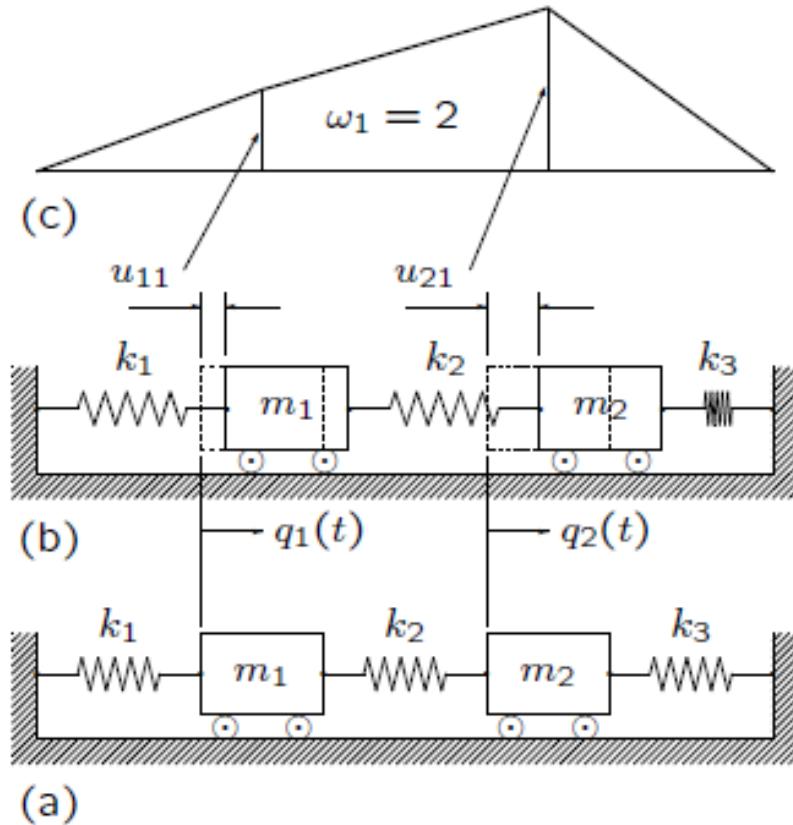
$$\lambda_2 = \omega_2^2 = 9 \left(\frac{\text{rad}}{\text{s}} \right)^2.$$

$$\begin{bmatrix} 8 - 9 & -2 \\ -2 & 5 - 9 \end{bmatrix} \begin{Bmatrix} u_{12} \\ u_{22} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}.$$
$$\begin{bmatrix} -1 & -2 \\ -2 & -4 \end{bmatrix} \begin{Bmatrix} u_{12} \\ u_{22} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}.$$

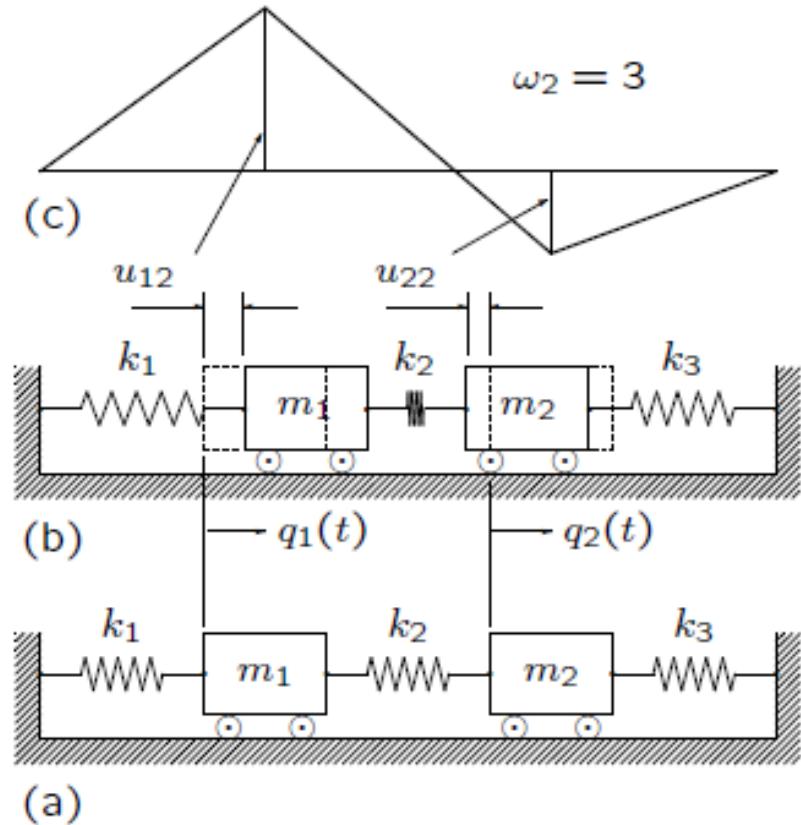
$$-u_{12} - 2u_{22} = 0$$

$$\begin{Bmatrix} u_{12} \\ u_{22} \end{Bmatrix} = \begin{Bmatrix} 1 \\ -\frac{1}{2} \end{Bmatrix}, \begin{Bmatrix} 4 \\ -2 \end{Bmatrix}, \dots$$

$$\begin{Bmatrix} u_{11} \\ u_{21} \end{Bmatrix} = \begin{Bmatrix} 1 \\ 2 \end{Bmatrix}, \begin{Bmatrix} 2 \\ 4 \end{Bmatrix}, \dots$$



$$\begin{Bmatrix} u_{12} \\ u_{22} \end{Bmatrix} = \begin{Bmatrix} 2 \\ -1 \end{Bmatrix}, \begin{Bmatrix} 4 \\ -2 \end{Bmatrix}, \dots$$



Solve Eigenvalue Problem using MATLAB

MATLAB IN VIBRATION ANALYSIS

MATLAB Command to Calculate

Eigenvalues and Eigenvectors of the System:

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

or

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

Normalization Scheme for the Natural Modes:

$$\{u\}_r^T [m] \{u\}_r = 1, \quad (r = 1, 2, \dots, n).$$

Consequence from Normalization of the Natural Modes:

$$\{u\}_r^T [k] \{u\}_r = \omega_r^2, \quad (r = 1, 2, \dots, n).$$

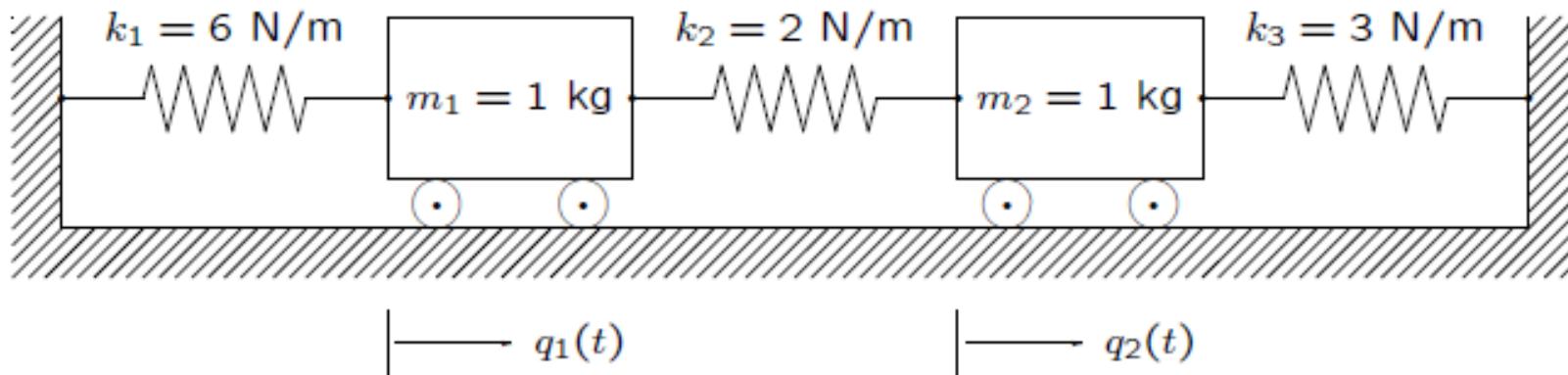
NORMALISATION OF EIGENVECTORS USING MATLAB

To calculate the Normalised Modal Matrix

use the following MATLAB command:

$$Un = U / \text{sqrt}(U' * m * U).$$

ORTHOGONALITY: 2-DOFS EXAMPLE



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

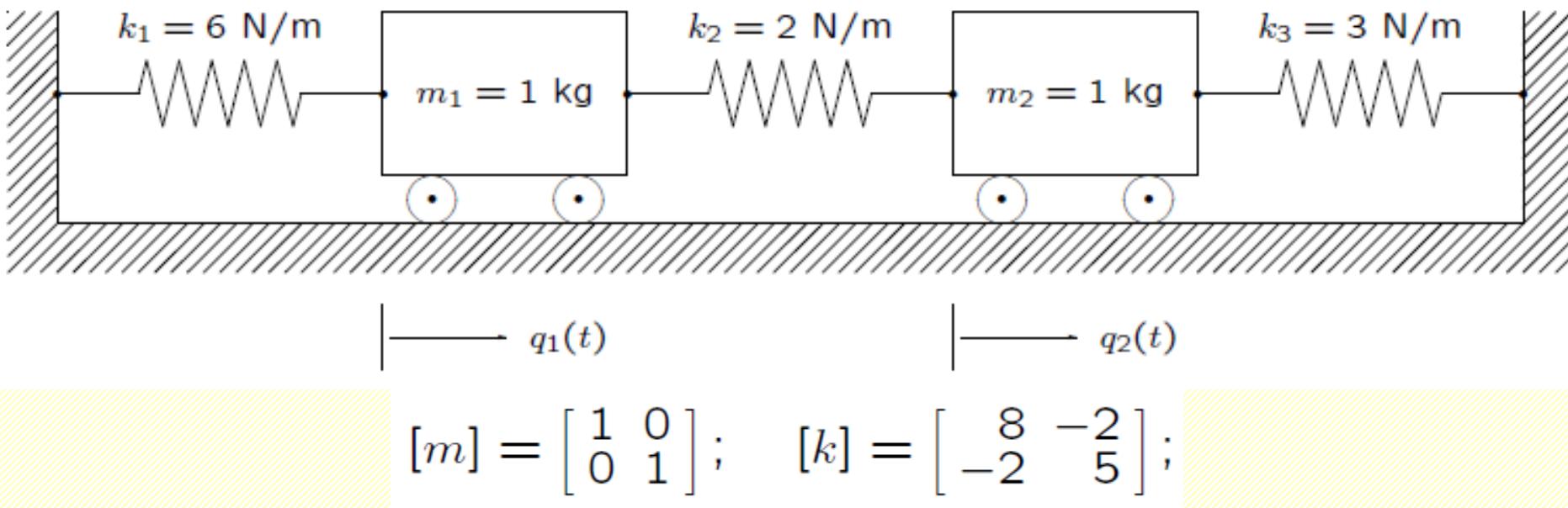
$$\{u\}_1 = \left\{ \begin{array}{c} u_{11} \\ u_{21} \end{array} \right\} = \left\{ \begin{array}{c} 1 \\ 2 \end{array} \right\}, \quad \{u\}_2 = \left\{ \begin{array}{c} u_{12} \\ u_{22} \end{array} \right\} = \left\{ \begin{array}{c} 2 \\ -1 \end{array} \right\},$$

$$\{u\}_1^T [m] \{u\}_2 =$$

$$= \begin{Bmatrix} u_{11} \\ u_{21} \end{Bmatrix}^T [m] \begin{Bmatrix} u_{12} \\ u_{22} \end{Bmatrix} =$$

$$= \begin{Bmatrix} 1 \\ 2 \end{Bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} 2 \\ -1 \end{Bmatrix} = 0$$

Numerical Example



```
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



EIGENVECTORS (or 'MODE SHAPES')

$D =$

4

0

0

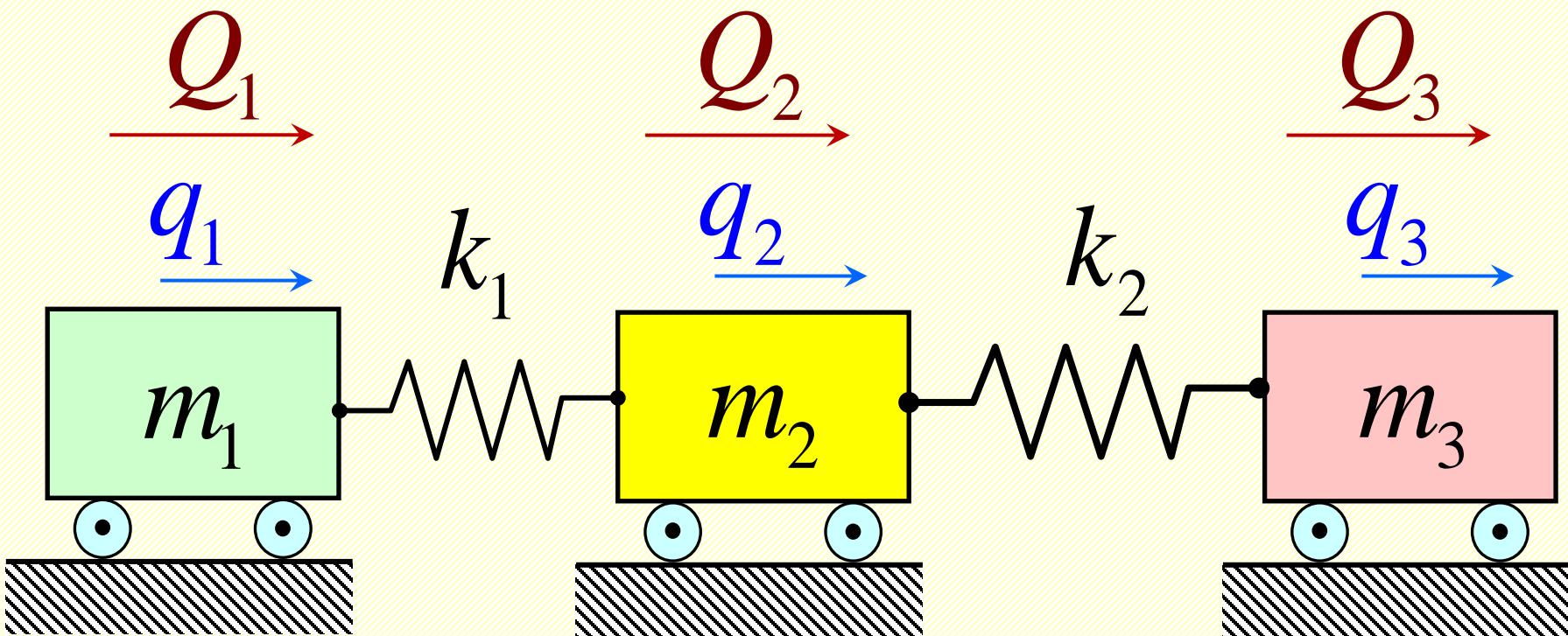
9

EIGENVALUES

MODE-1

MODE-2

Example: 3-DOF System



$$\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \begin{Bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{Bmatrix} + \begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} q_1 \\ q_2 \\ q_3 \end{Bmatrix} = \begin{Bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{Bmatrix}$$

```

1 %% MIET2389 Eigenvalue Problem Example
2 - close('all'); clear; clc;
3 - m1=4; m2=9; m3=1;
4 - k1=1000; k2=500;
5 - m=[m1 0 0; 0 m2 0; 0 0 m3];
6 - k=[k1 -k1 0; -k1 k1+k2 -k2; 0 -k2 k2];
7 - [U,D]=eig(k,m);
8 - fprintf('===== Modal Matrix [U]: ======\n')
9 - fprintf(' %9.6f %9.6f %9.6f \n',U(1,:));
10 - fprintf(' %9.6f %9.6f %9.6f \n',U(2,:));
11 - fprintf(' %9.6f %9.6f %9.6f \n',U(3,:));
12 - fprintf('===== Natural Frequencies [rad/s] : ======\n')
13 - fprintf('w1= %9.6f [rad/s]; w2= %9.6f [rad/s]; w3= %9.6f [rad/s]\n',sqrt(diag(D)));
14 - fprintf('===== Natural Frequencies [Hz] : ======\n')
15 - fprintf('f1= %9.6f [Hz]; f2= %9.6f [Hz]; f3= %9.6f [Hz]\n',sqrt(diag(D))/2/pi);

```

Command Window

```

===== Modal Matrix [U]: ======
-0.267261 0.408248 0.109109
-0.267261 -0.136083 -0.145479
-0.267261 -0.408248 0.872872
===== Natural Frequencies [rad/s] : ======
w1= 0.000000 [rad/s]; w2= 18.257419 [rad/s]; w3= 24.152295 [rad/s]
===== Natural Frequencies [Hz] : ======
f1= 0.000000 [Hz]; f2= 2.905758 [Hz]; f3= 3.843957 [Hz]

```

Note: 'zero' natural frequency corresponds to the 'rigid body mode'

EQUATIONS IN STATE-SPACE FORM

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

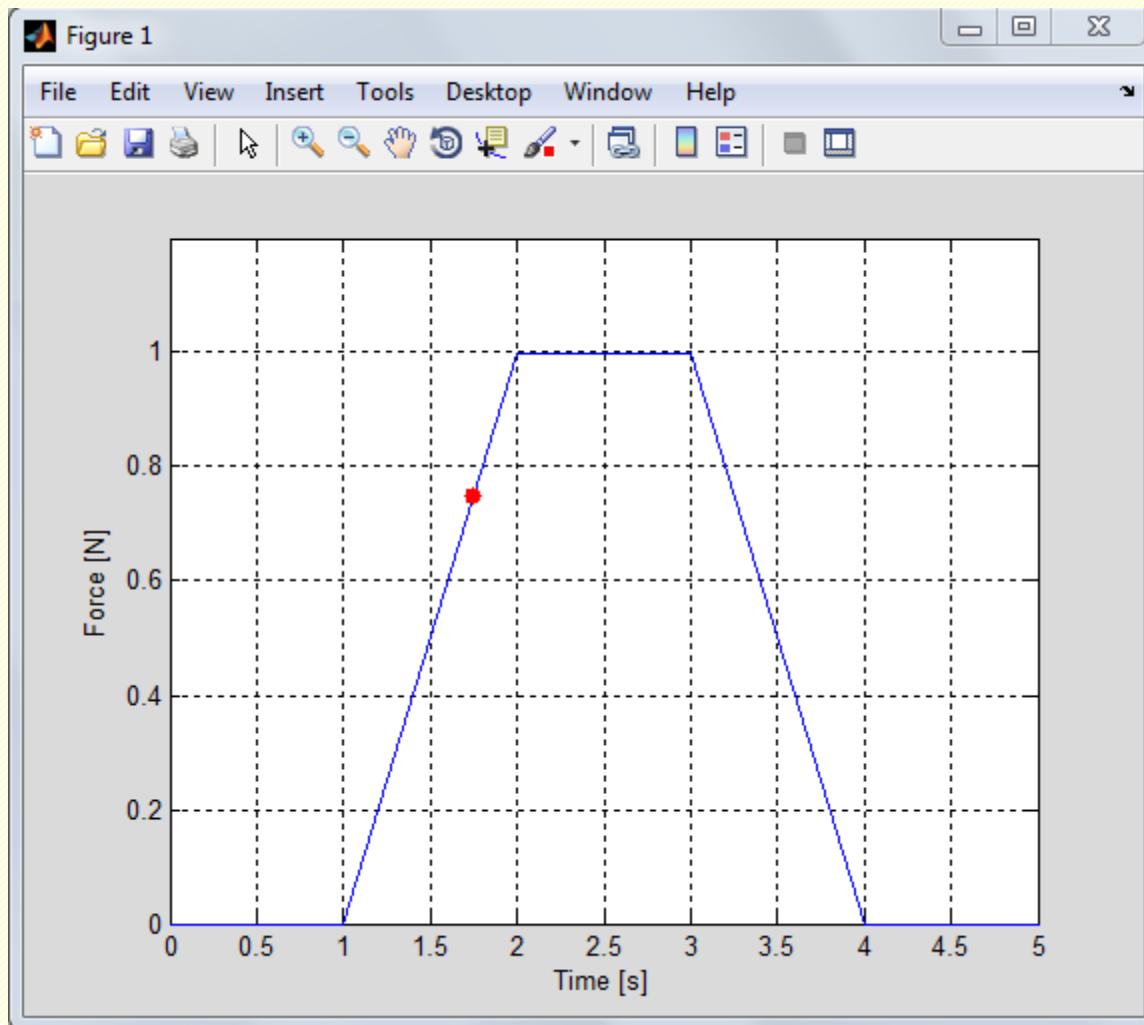
$$y = Cx + Du$$

EXAMPLES (Class Work)

Exploring “interp1” Command

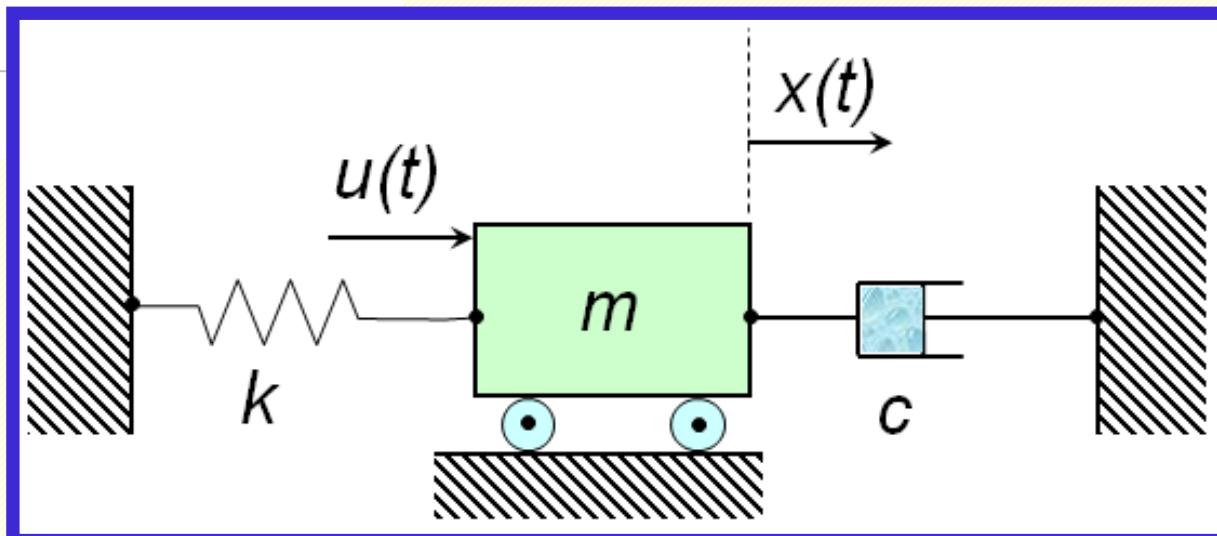
```
1 %% MY FIRST EXAMPLE IN MATLAB
2 % Programmed by Prof P.M.Trivailo
3 %
4 %
5
6 %% Step-1: Enter the data
7 t=[0 1 2 3 4 10];
8 F=[0 0 1 1 0 0];
9
10 %% Step-2: Plot the data
11 plot(t,F);
12 % alternatively use
13 % gg1=line('XData',t, 'YData', F,'Color',[0 0 1], 'LineWidth',3)
14 axis
15 xlabel('Time [s]')
16 ylabel('Force [N]')
17
```

```
18 %% Step-3: Previewing of the Plot and Adjusting the Axes
19 axis([0 max(t) 0 2])
20 grid
21
22
23 %% Step-4: Draw a Dot
24 gg2= line('XData',0,'YData', 0,'Color',[1 0 0], 'MarkerSize',20,'Marker','.')
25
26 %% Step-5 & 6: Illustrating the 'interp1' command;
27 % Illustrating the cells and incremental feature
28 % Enter increment (for example 0.1) in the '-' window
29 % for the cells and click '+'
30 t_interp=1.75;
31 F_interp = interp1(t,F,t_interp)
32 set(gg2, 'XData',t_interp,'YData', F_interp);
33 axis([0 5 0 1.2])
```



FORCE RESPONSE: 1-DOFS

```
1 %% 1-DOF SYSTEM: RESPONSE
2 % Solving Differential Eqs of Motion using MATLAB
3 % Programmed by Prof P.M.Trivailo
4 %
5 %
6 %
7 - global tt FF A B C D
8
9 %% Enter System Characteristics
10 - m = 2; k = 200;
11 - c = 2.3;
12 - %c = 0;
13
14 %% Enter Initial Conditions (ICs)
15 - q0 = [0]; q0_dot = [0];
16
17 %% Establish the IC vector
18 - x0 = [q0 ; q0_dot];
```

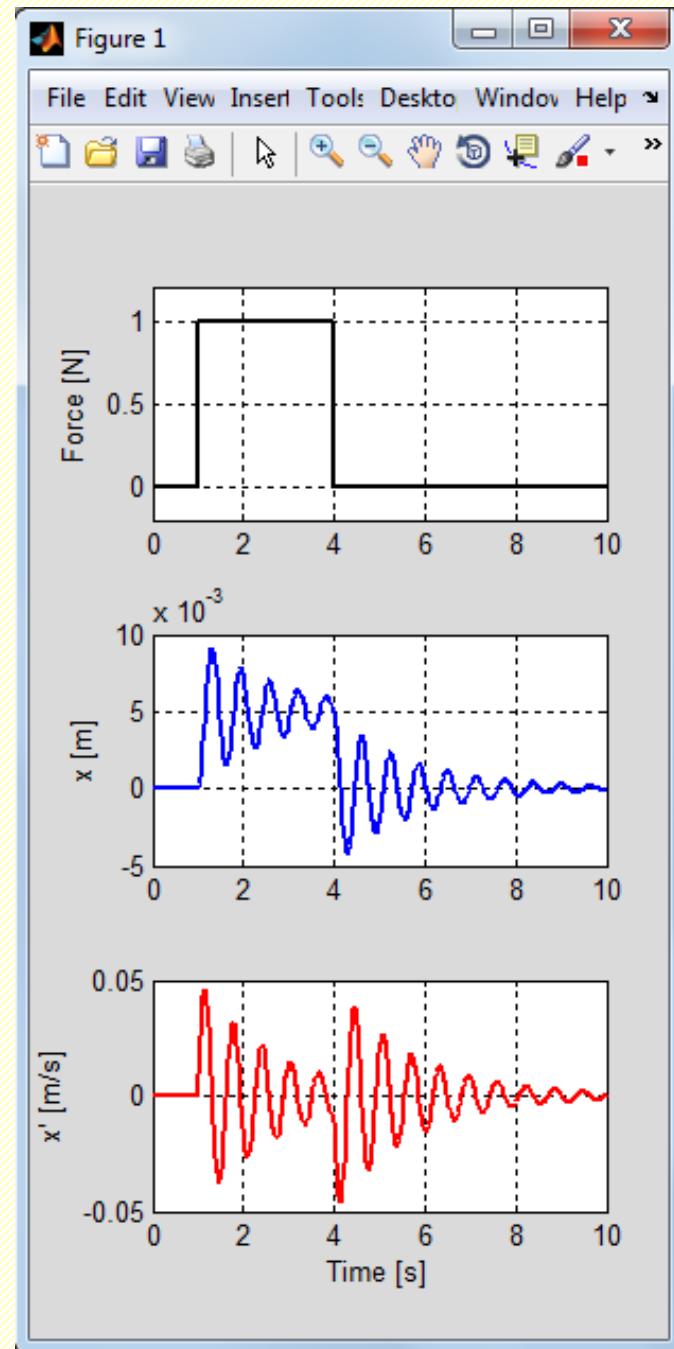


```
19
20 %% Specify the EXTERNAL Excitation Force
21 %tt=[0 1 2 3 4 10];
22 - tt=[0 1 1.001 3.999 4 10];
23 - FF=[0 0 1 1 0 0]*1;
24
25 %% Create matrices for state-space formulation
26 - dof=size(m,1);
27 - A = [zeros(dof, dof) eye(dof, dof) ; -inv(m)*k -inv(m)*c];
28 - B = [zeros(dof, dof) ; inv(m)];
29
30 % These matrices will be needed IF SIMULINK is used
31 - C = eye(2*dof, 2*dof);
32 - D = zeros(2*dof, dof);
33
34 %% CORE!!! SOLVING DIFFERENTIAL EQS USING ODE23 Integration Procedure
35 - [t_out,x_out] = ode23('PMTode',[0 max(tt)],x0);
36
```

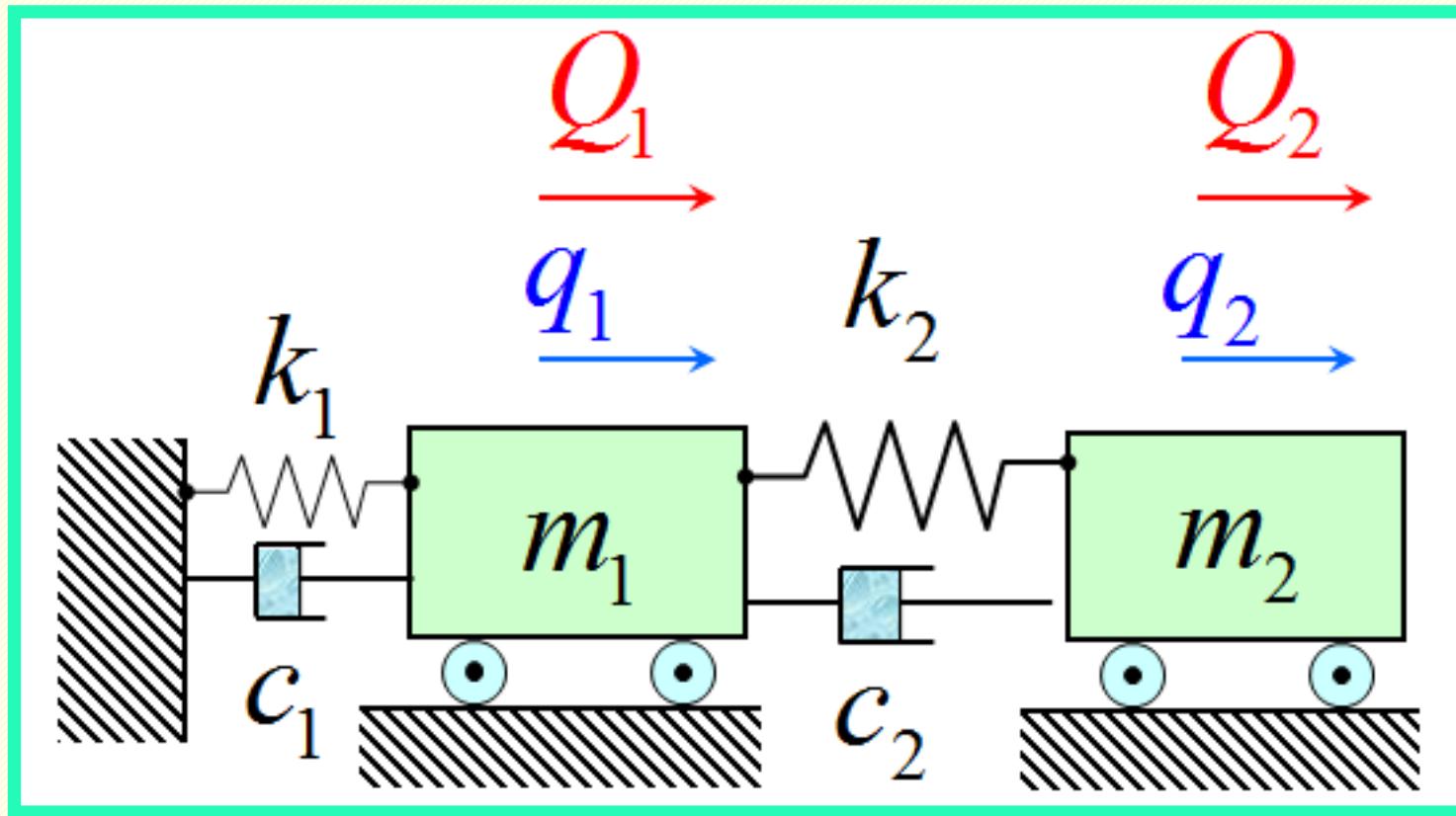
```

37 %% PLOTTING Results
38 % Plotting the excitation function first
39 subplot(3,1,1)
40 gg1=plot(tt,FF,'k');
41 axis([0 max(tt) -0.2 max(FF)+0.2])
42 xlabel('Time [s]')
43 ylabel('Force [N]')
44 grid
45
46 % Plotting Response of the 1-DOF system
47 subplot(3,1,2)
48 gg2=plot(t_out,x_out(:,1), 'b');
49 xlabel('Time [s]')
50 ylabel('x [m]')
51 grid
52
53 subplot(3,1,3)
54 gg3=plot(t_out,x_out(:,2), 'r');
55 set([gg1, gg2, gg3], 'LineWidth', 2)
56 xlabel('Time [s]')
57 ylabel('x' [m/s]')
58 grid

```



FORCE RESPONSE: 2-DOFS



$$[m] = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}; \quad [c] = \begin{bmatrix} c_1 + c_2 & -c_1 \\ -c_2 & c_2 \end{bmatrix}; \quad [k] = \begin{bmatrix} k_1 + k_2 & -k_1 \\ -k_2 & k_2 \end{bmatrix}$$

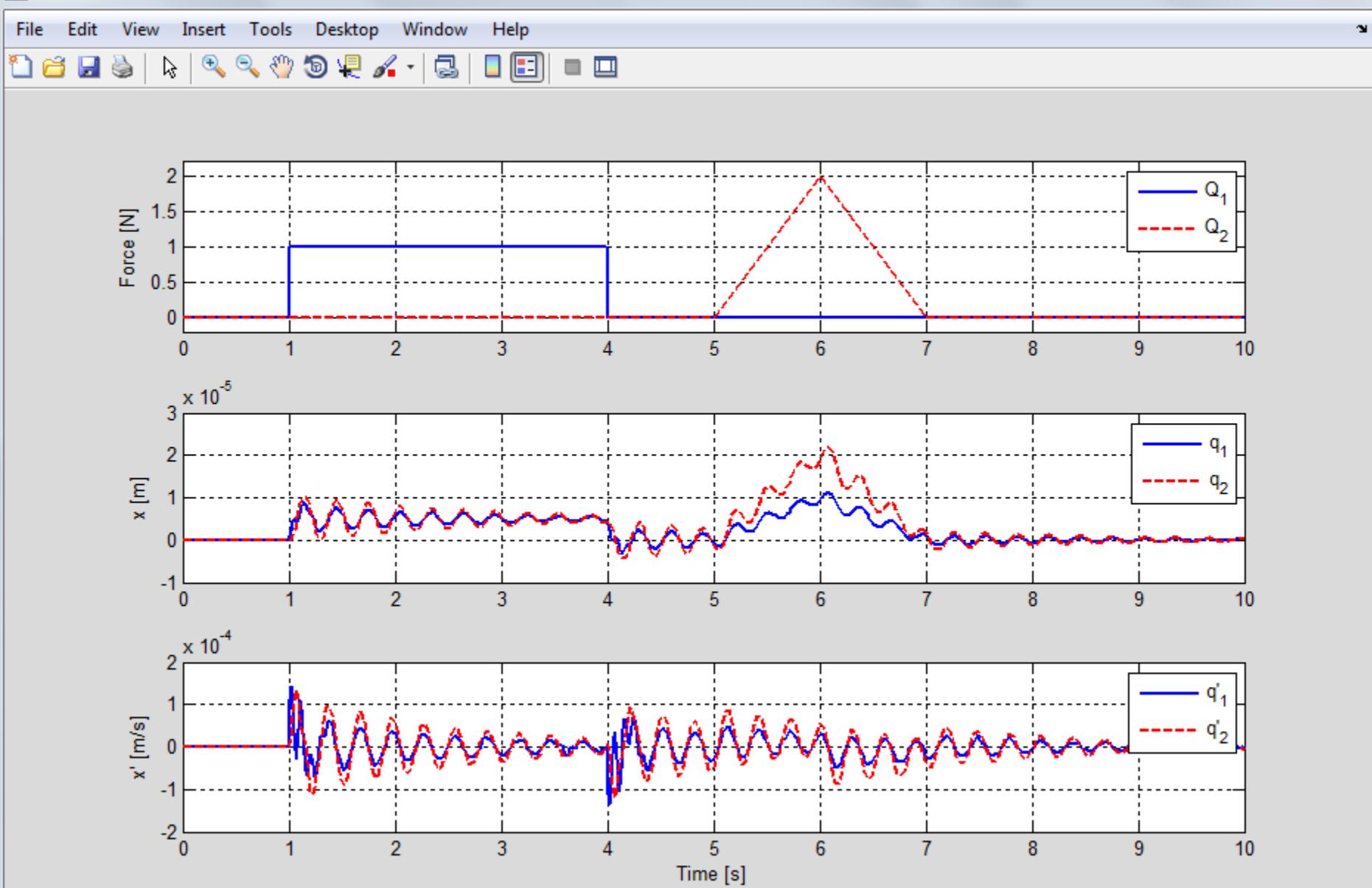
FORCE RESPONSE: 2-DOFS

```
1 %% 2-DOF SYSTEM: RESPONSE
2 % Solving Differential Eqs of Motion for 2DOFS using MATLAB
3 % Programmed by Prof P.M.Trivailo
4 %
5 %
6 %
7 - global tt1 FF1 tt2 FF2 A B C D
8
9 %% Enter System Characteristics
10 - m = 1e2*[1 0 ; 0 2];
11 - k = 1e5*[4 -2 ; -2 2];
12 - c = 1e2*[8 -4 ; -4 4];
13
14 %% Enter Initial Conditions (ICs)
15 - q0 = [0; 0];
16 - q0_dot = [0; 0];
17
18 %% Establish the IC vector
19 - x0 = [q0 ; q0_dot];
```

```
20
21 %% Specify the EXTERNAL Excitation Force
22 %tt=[0 1 2 3 4 10];
23 tt1=[0 1 1.001 3.999 4 10];
24 FF1=[0 0 1 1 0 0]*1;
25 tt2=[0 5 6 7 10];
26 FF2=[0 0 2 0 0];
27
28 %% Create matrices for state-space formulation
29 dof=size(m,1);
30 A = [zeros(dof, dof) eye(dof, dof) ; -inv(m)*k -inv(m)*c];
31 B = [zeros(dof, dof) ; inv(m)];
32
33 % These matrices will be needed IF SIMULINK is used
34 C = eye(2*dof, 2*dof);
35 D = zeros(2*dof, dof);
36
37 %% CORE!!! SOLVING DIFFERENTIAL EQS USING ODE23 Integration Procedure
38 [t_out,x_out] = ode23('PMTode2',[0 max(tt1)],x0);
39
```

```
41 % Plotting the excitation function first
42 subplot(3,1,1)
43 gg1=plot(tt1,FF1,'b',tt2,FF2,'r--');
44 axis([0 max(tt2) -0.2 max(FF2)+0.2])
45 ylabel('Force [N]')
46 legend('Q_1','Q_2')
47 grid
48
49 % Plotting Response of the 2-DOF system
50 subplot(3,1,2)
51 gg2=plot(t_out,x_out(:,1),'b',t_out,x_out(:,2),'r--');
52 ylabel('x [m]')
53 legend('q_1','q_2')
54 grid
55
56 subplot(3,1,3)
57 gg3=plot(t_out,x_out(:,3),'b',t_out,x_out(:,4),'r--');
58 set([gg1, gg2, gg3], 'LineWidth', 2)
59 xlabel('Time [s]')
60 ylabel('x'' [m/s]')
61 legend('q''_1','q''_2')
62 grid
```

Figure 1



Non-Linear Systems? WHY BOTHER ?

Examples

MODELLING: “CHAIN FOUNTAIN”

EXAMPLE OF A GREAT PRESENTATION



Courtesy: Steve Mould, <https://www.youtube.com/watch?v=dQJBBklpQQ>



Self siphoning beads

1,925,211 views

9.6K

157

SHARE

...

EXAMPLE OF A GREAT PRESENTATION



Courtesy: Steve Mould, <https://www.youtube.com/watch?v=wmFi1xhz90Q>



Investigating the "Mould effect" | Steve Mould | TEDxNewcastle

73,236 views

1.8K

19

SHARE

...

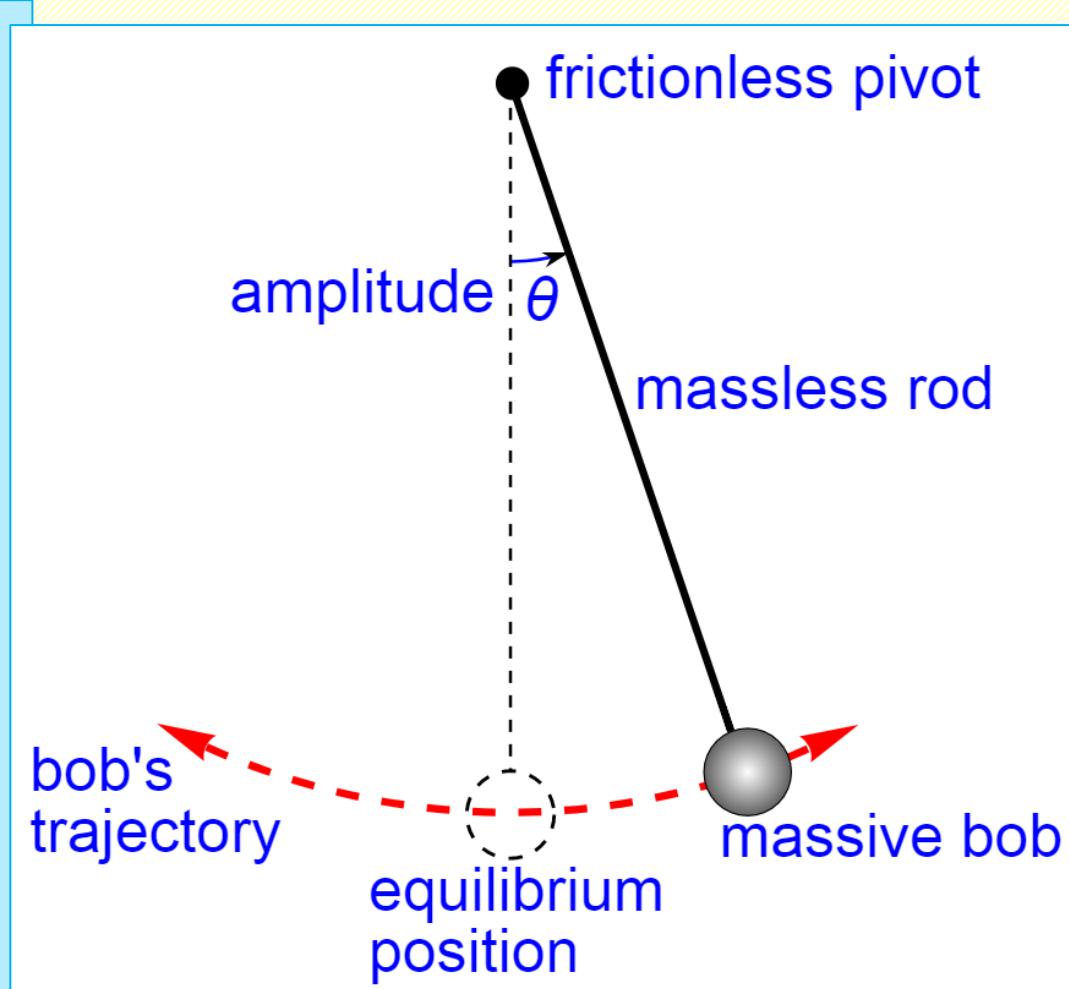
What is Pendulum?

A **pendulum** is a weight suspended from a pivot so that it can swing freely.

When a pendulum is displaced sideways from its resting, equilibrium position, it is subject to a restoring force due to gravity that will accelerate it back toward the equilibrium position.

When released, the restoring force acting on the pendulum's mass causes it to oscillate about the equilibrium position, swinging back and forth.

The time for one complete cycle, a left swing and a right swing, is called the period. The period depends on the length of the pendulum and also to a slight degree on the amplitude, the width of the pendulum's swing.



Courtesy: <https://en.wikipedia.org/wiki/Pendulum>

What is Pendulum?

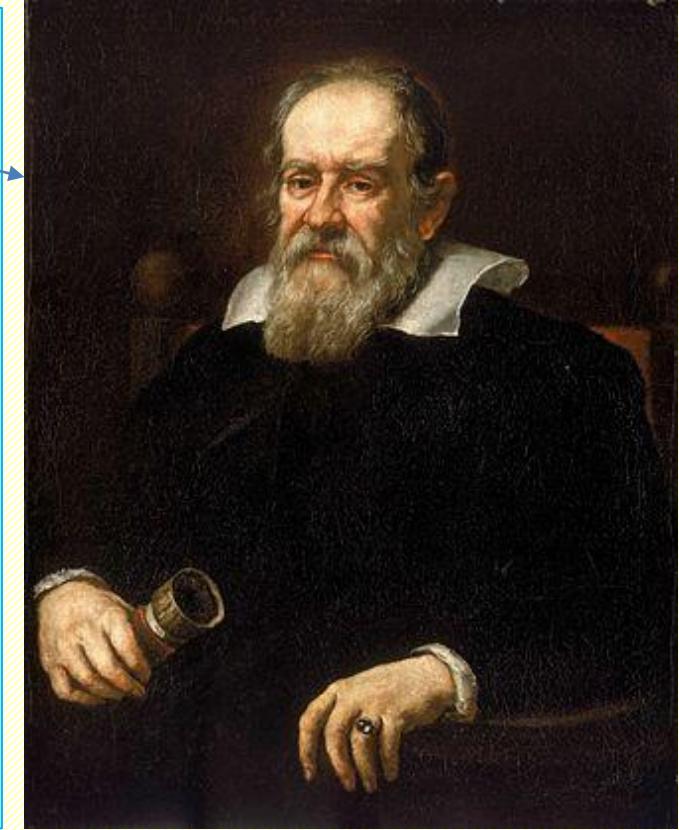
From the first scientific investigations of the pendulum around 1602 by **Galileo Galilei**, the regular motion of pendulums was used for timekeeping, and was the world's most accurate timekeeping technology until the 1930s.

The pendulum clock invented by **Christian Huygens** in 1658 became the world's standard timekeeper, used in homes and offices for 270 years, and achieved accuracy of about one second per year before it was superseded as a time standard by the quartz clock in the 1930s.

Pendulums are also used in scientific instruments such as accelerometers and seismometers. Historically they were used as gravimeters to measure the acceleration of gravity in geophysical surveys, and even as a standard of length.

The word "pendulum" is new Latin, from the Latin ***pendulus***, meaning 'hanging'.

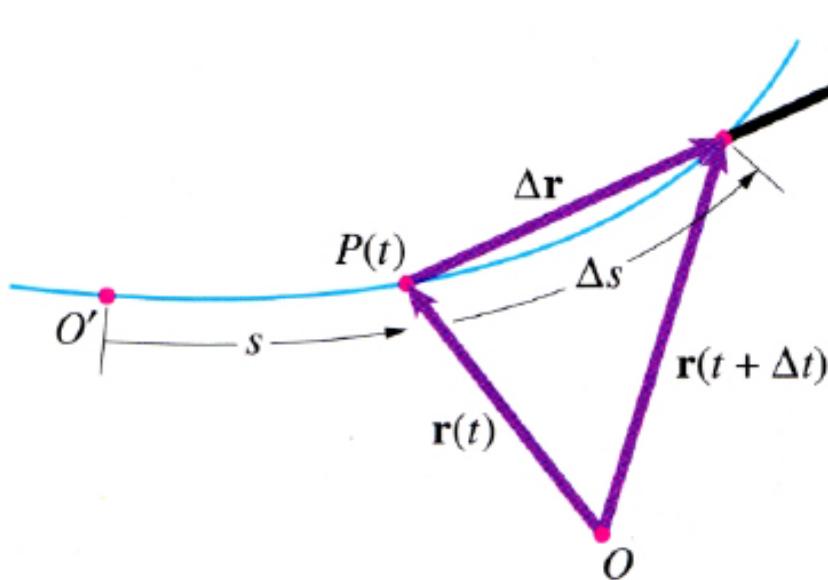
Galileo Galilei (15 Feb 1564[3] – 8 Jan 1642) was an Italian polymath. Galileo is a central figure in the transition from natural philosophy to modern science and in the transformation of the scientific Renaissance into a scientific revolution.



Courtesy: <https://en.wikipedia.org/wiki/Pendulum> https://en.wikipedia.org/wiki/Galileo_Galilei

Normal & Tangential Coordinates (Slide from the previous Lecture)

Curvilinear Motion: Normal & Tangential Coord



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

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

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

Normal acceleration

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

Tangential acceleration

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

Derive EOM for Pendulum: Method-1

Newton's Second Law

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

(for translational motion, $m = \text{const}$)

is equivalent to

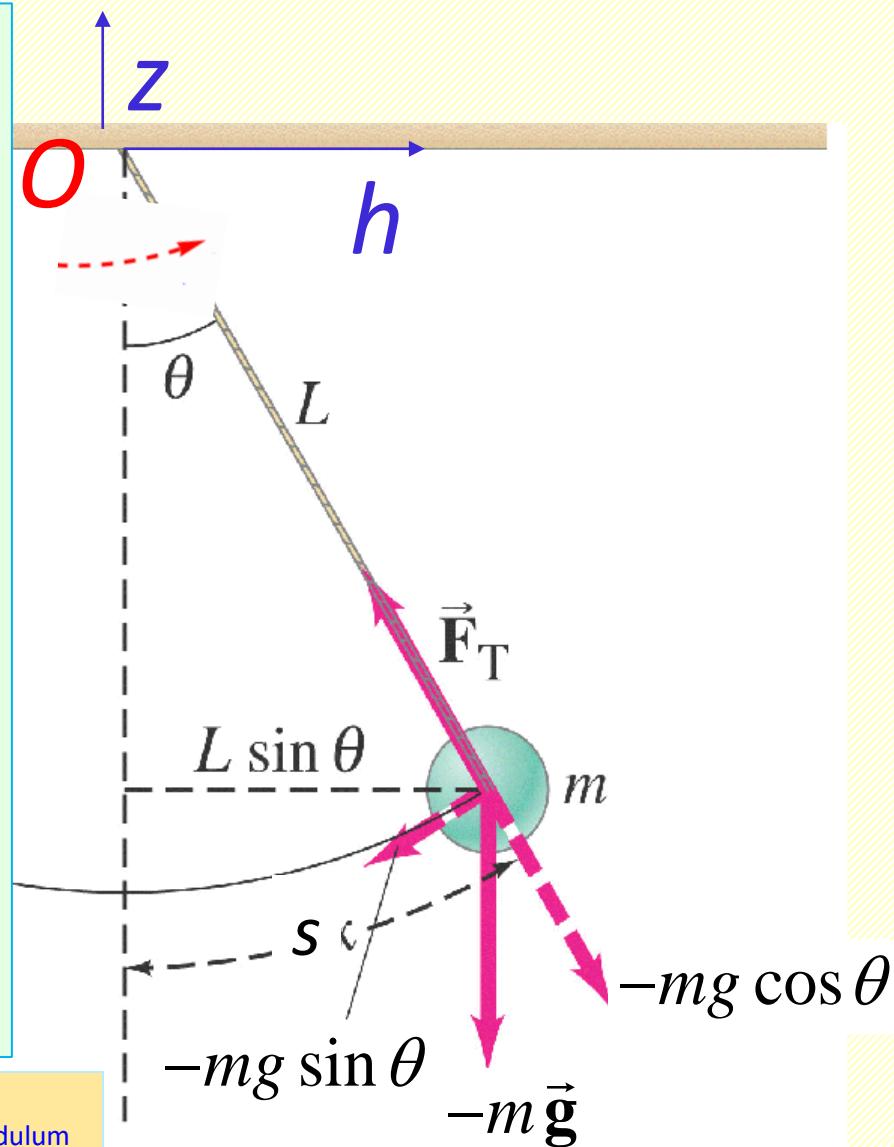
$$\sum F_t = m \frac{dv}{dt} \quad (\text{eq. along tangential direction})$$

and

$$\sum F_n = m \frac{v^2}{\rho} \quad (\text{eq. along normal direction})$$

Image (PMT adapted), Courtesy:

<https://www.quora.com/How-do-you-calculate-work-done-by-tension-on-a-pendulum>



Derive EOM for Pendulum: Method-1

$$\sum F_t = m \frac{dv}{dt}$$

Develop Left Hand Side ($v = \dot{s}$; $\dot{v} = \ddot{s}$):

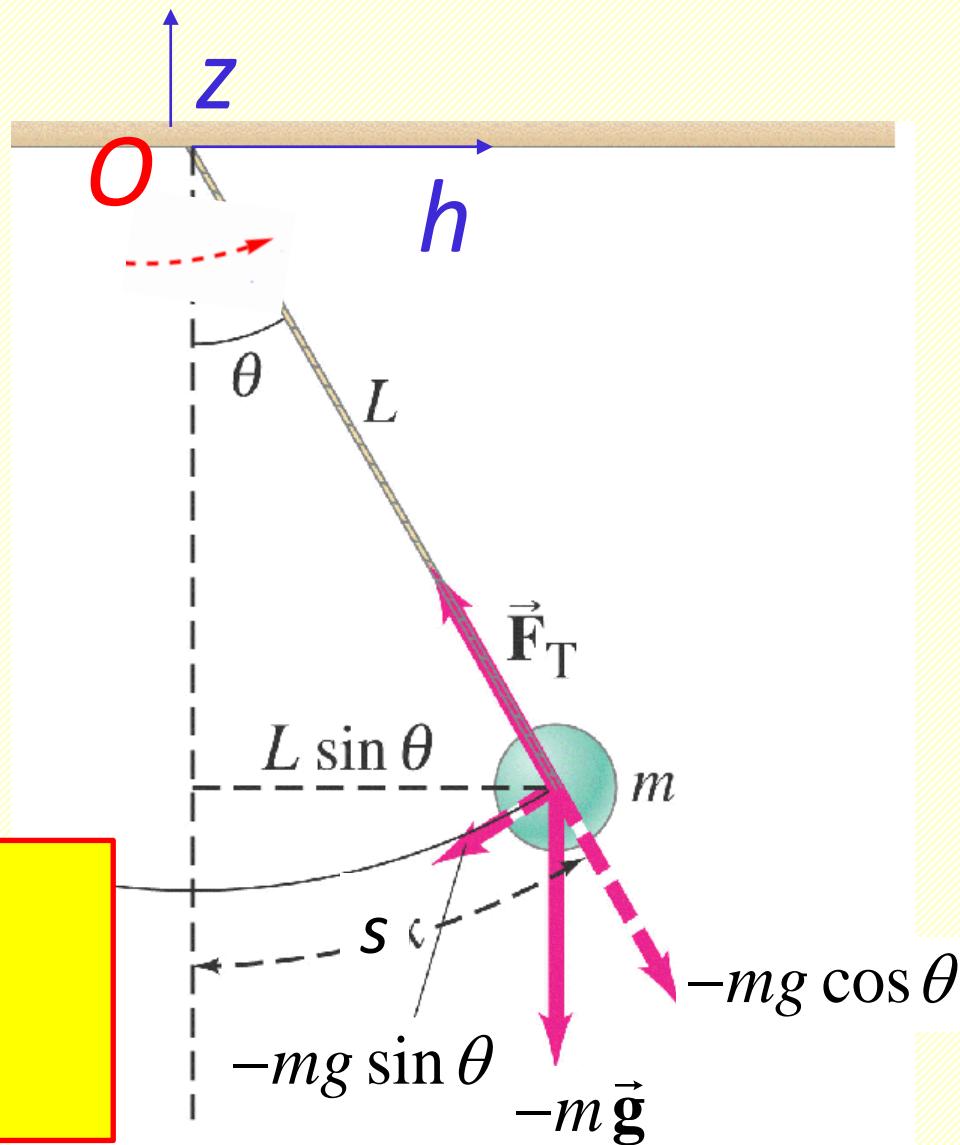
$$\sum F_t = -mg \sin \theta$$

Develop Right Hand Side:

$$m \frac{dv}{dt} = m \frac{d^2 s}{dt^2} = m \frac{d^2(L\theta)}{dt^2} = L \frac{d^2 \theta}{dt^2}$$

$$\text{LHS=RHS} \quad \Rightarrow$$

$$\frac{d^2 \theta}{dt^2} + \frac{g}{L} \sin \theta = 0$$



Derive EOM for Pendulum: Method-2

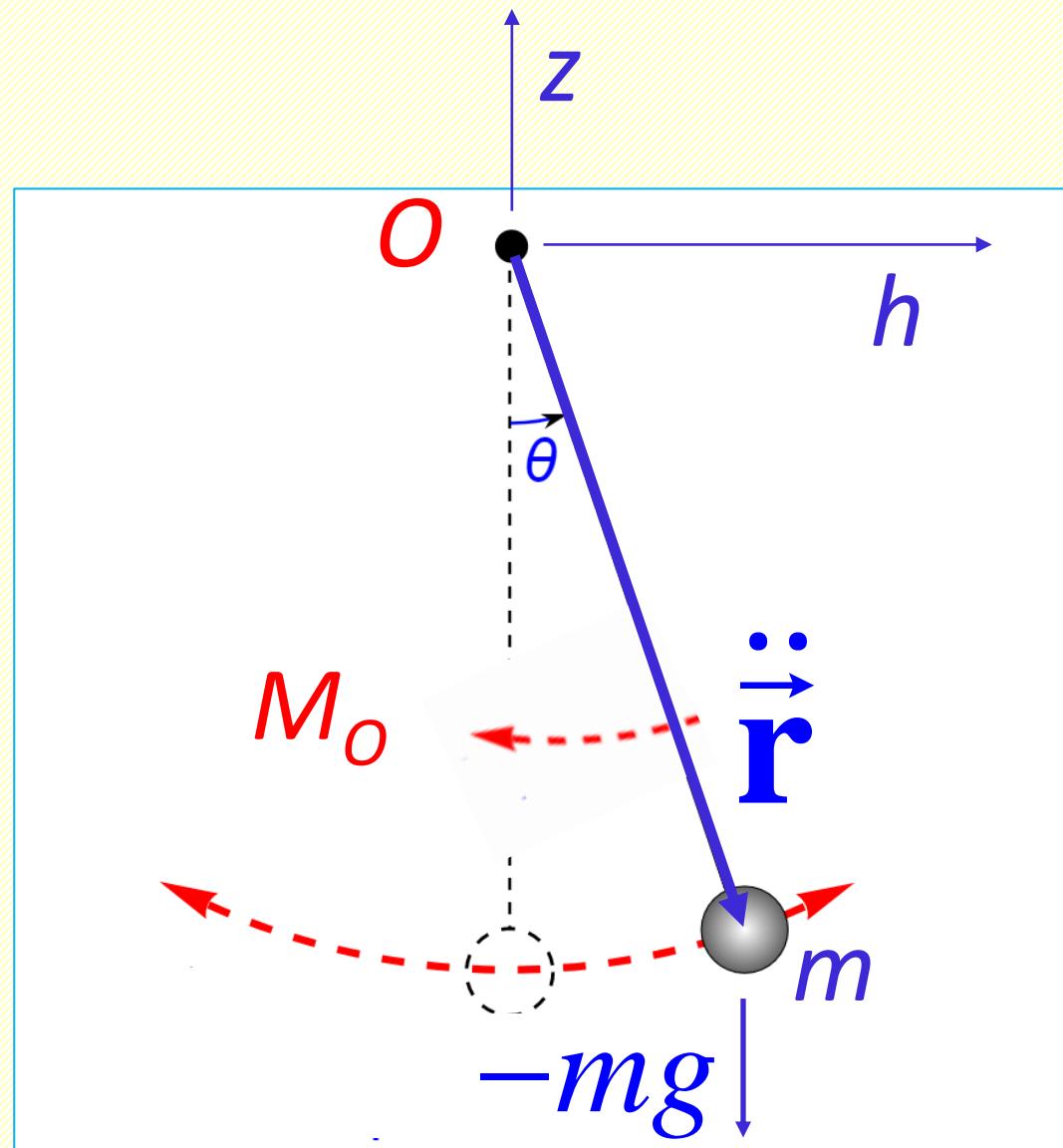
$$\sum \vec{F} = m \ddot{\vec{r}}$$

(for translational motion,
 $m = \text{const}$)

is equivalent to

$$\sum M_O = I_O \ddot{\theta}$$

(for rotational motion,
 $I_O = \text{const}$)



Derive EOM for Pendulum: Method-2

$$\sum M_o = I_o \frac{d^2\theta}{dt^2}$$

Develop Left Hand Side:

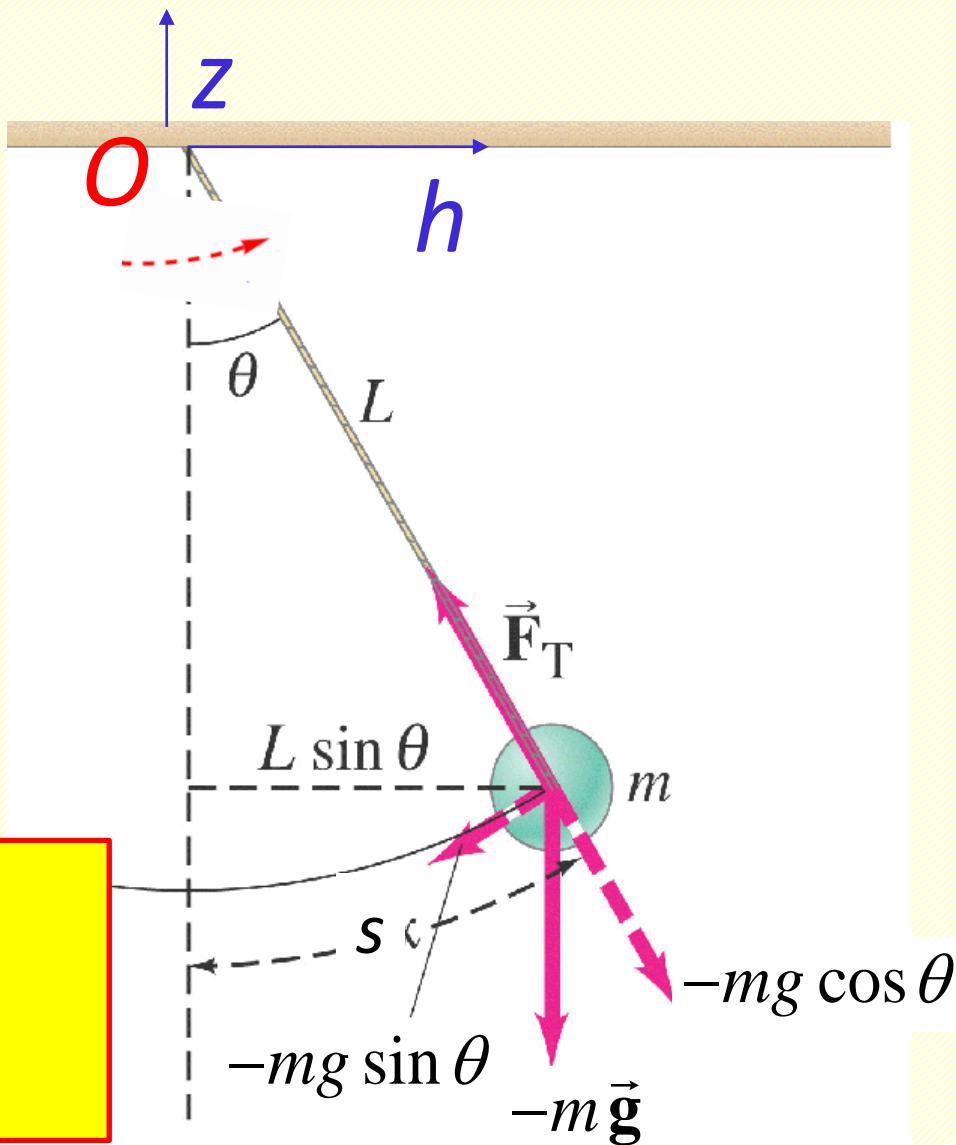
$$\sum M_o = -mg \sin \theta \times L$$

Develop Right Hand Side:

$$I_o \frac{d^2\theta}{dt^2} = mL^2 \frac{d^2\theta}{dt^2}$$

$$\text{LHS=RHS} \quad \Rightarrow$$

$$\frac{d^2\theta}{dt^2} + \frac{g}{L} \sin \theta = 0$$



Why on previous slide $I_o=mL^2$???

On one hand, Kinetic Energy of the System
(due to translational considerations):

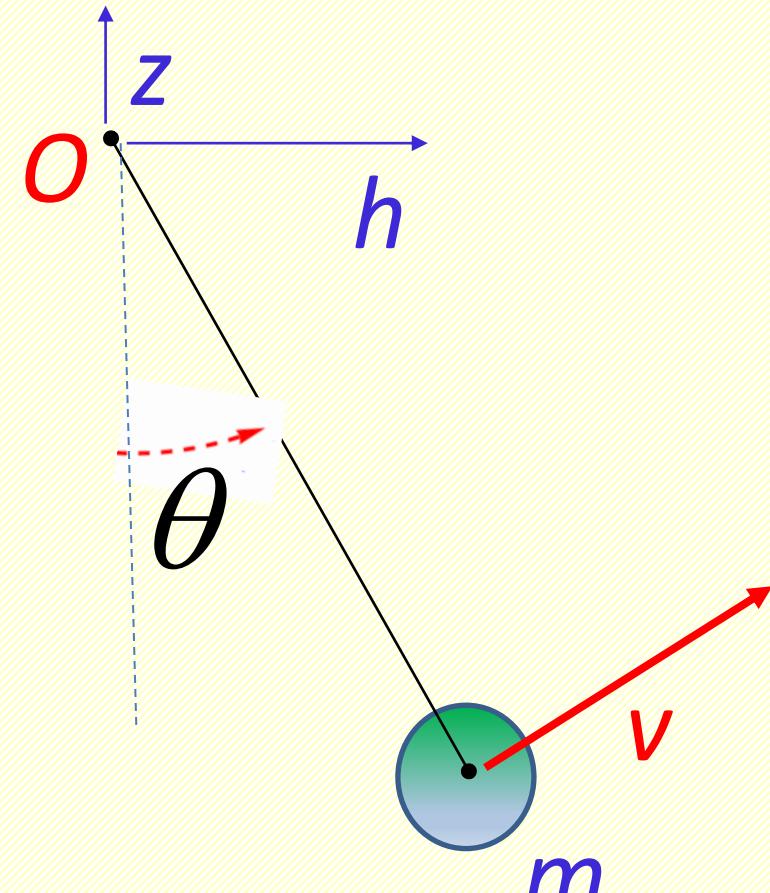
$$T = \frac{1}{2}mv^2 = \frac{1}{2}m(\omega L)^2 = \frac{1}{2} \boxed{mL^2} \dot{\theta}^2$$

On the other hand,
(due to rotational considerations):

$$T = \frac{1}{2}I_o\omega^2 = \frac{1}{2} \boxed{I_o} \dot{\theta}^2$$

LHS=RHS \Rightarrow

$$\boxed{I_o = mL^2}$$



Why on previous slide $I_o=mL^2$???

On one hand, Kinetic Energy of the System
(due to translational considerations):

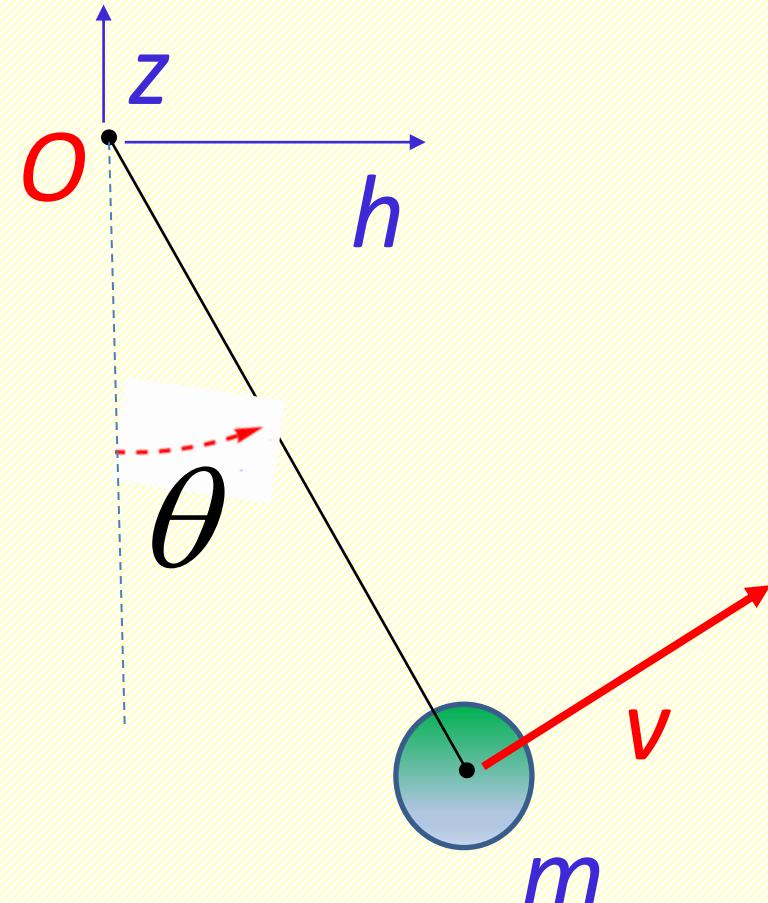
$$T = \frac{1}{2}mv^2 = \frac{1}{2}m(\omega L)^2 = \frac{1}{2} \boxed{mL^2} \dot{\theta}^2$$

On the other hand,
(due to rotational considerations):

$$T = \frac{1}{2}I_o\omega^2 = \frac{1}{2} \boxed{I_o} \dot{\theta}^2$$

LHS=RHS \Rightarrow

$$\boxed{I_o = mL^2}$$



!!!!!!

CORE TECHNIQUE

IN THIS COURSE:

NUMERICAL SOLUTION,

ILLUSTRATED

WITH PENDULUM EXAMPLE

!!!!!!

ONE-FILE SOLUTION:

i.e. using Anonymous Function
to describe EOM

Re-Write EOM using System's States!

$$\frac{d^2\theta}{dt^2} + \frac{g}{L} \sin \theta = 0$$

$$\begin{cases} x_1 = \theta \\ x_2 = \dot{\theta} \end{cases} \quad \text{therefore } \dot{x}_1 = x_2$$

Also, physical EOM
can be re-written as

$$\dot{x}_2 = -\frac{g}{L} \sin \theta$$

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\frac{g}{L} \sin \theta \end{cases}$$

Re-Write EOM using System's States!

$$\frac{d^2\theta}{dt^2} + \frac{g}{L} \sin \theta = 0$$

$$\begin{cases} x_1 = \theta \\ x_2 = \dot{\theta} \end{cases}$$

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\frac{g}{L} \sin \theta \end{cases}$$

SOLUTION: Numerical (!!!) Method

MATLAB SCRIPT (no annotations)

```
%% SIMPLE PENDULUM: NON-LINEAR EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2018
%-----
clear; close all; clc;
g=9.81; L=2; tmax=18; thD0=170; th_dot0=0;
th0=thD0*pi/180; t=[0:0.001:1]*tmax;
pend_xdot = @(t, x) ([x(2); -(g/L)*sin(x(1))]);
[tt,zz]=ode45(pend_xdot,t,[th0; th_dot0]);
plot(tt, zz(:,1)*180/pi,'LineWidth',3,'Color',[0 0.5 1]);
grid on;
xl=xlabel('$t$ [s]'); yl=ylabel('$\theta$ [deg]');
set([xl,yl],'Interpreter','LaTeX')
title('Pendulum (No Air Drag): $\theta$ Time History');
set(gca, 'FontSize',18); hold on;
```

$$\begin{cases} x_1 = \theta \\ x_2 = \dot{\theta} \end{cases}$$

SOLUTION: Numerical (!!!) Method

MATLAB SCRIPT (with annotations)

```
%% SIMPLE NON-LINEAR  
% Designed by Prof P.M.  
  
clear; close all; clc;  
g=9.81; L=2; tmax=18; thD0=170; th_dot0=0;  
th0=thD0*pi/180; t=[0:0.001:1]*tmax;  
pend_xdot = @(t, x) ([x(2); -(g/L)*sin(x(1))]);  
  
[tt,zz]=ode45(pend_xdot,t,[th0; th_dot0]);  
  
plot(tt, zz(:,1)*180/pi, 'LineWidth', 3, 'Color', [0 0.5 1]);  
grid on;  
xl=xlabel('t [s]'); yl=ylabel('theta [rad]');  
set([xl,yl], 'Interpreter', 'LaTeX')  
title('Pendulum Motion');  
set(gca, 'FontSize', 14);
```

1. Input of the data

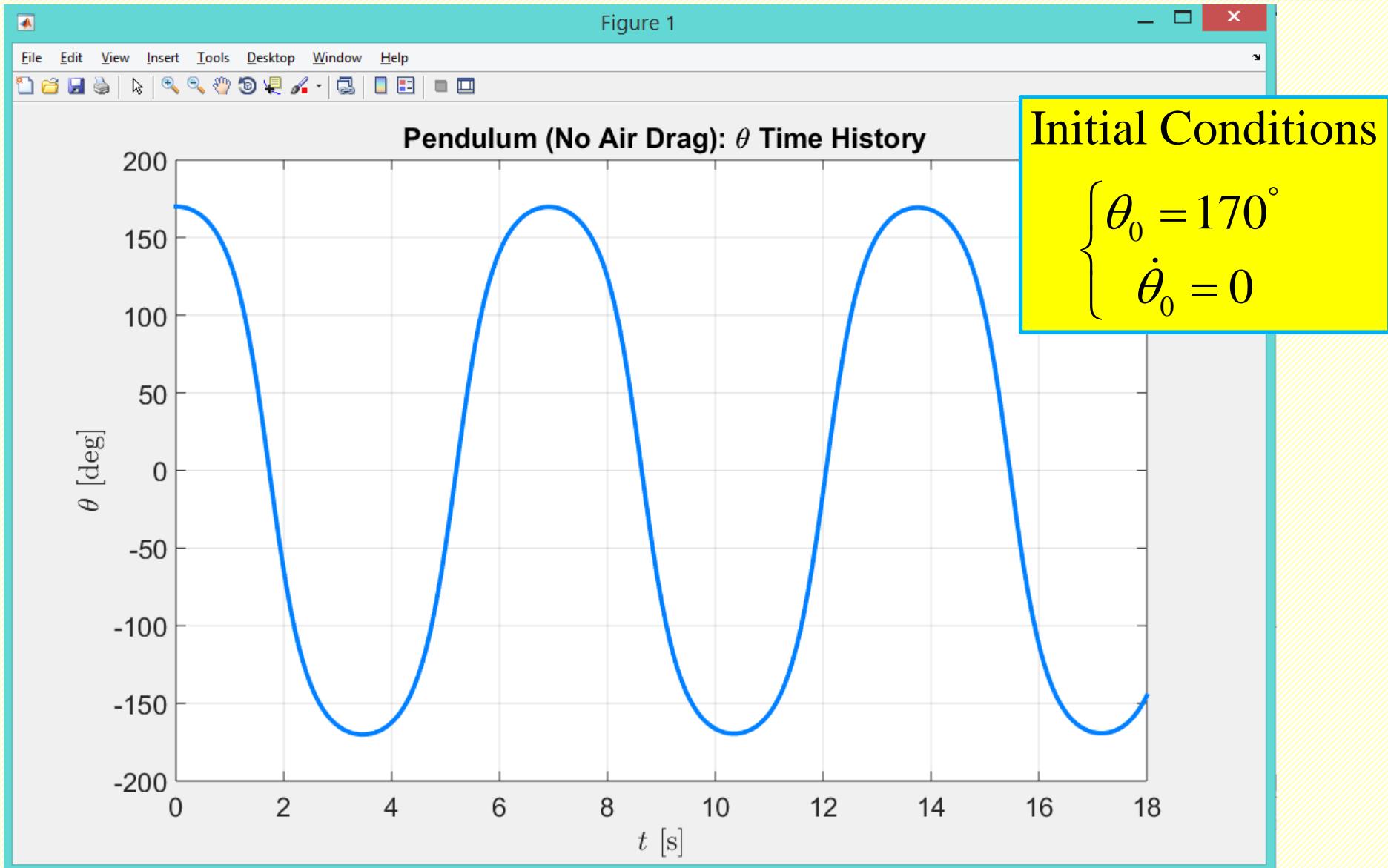
2.

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\frac{g}{L} \sin \theta \end{cases}$$

3. Calling ode45 procedure !

4. Plotting results

Output of the MATLAB script

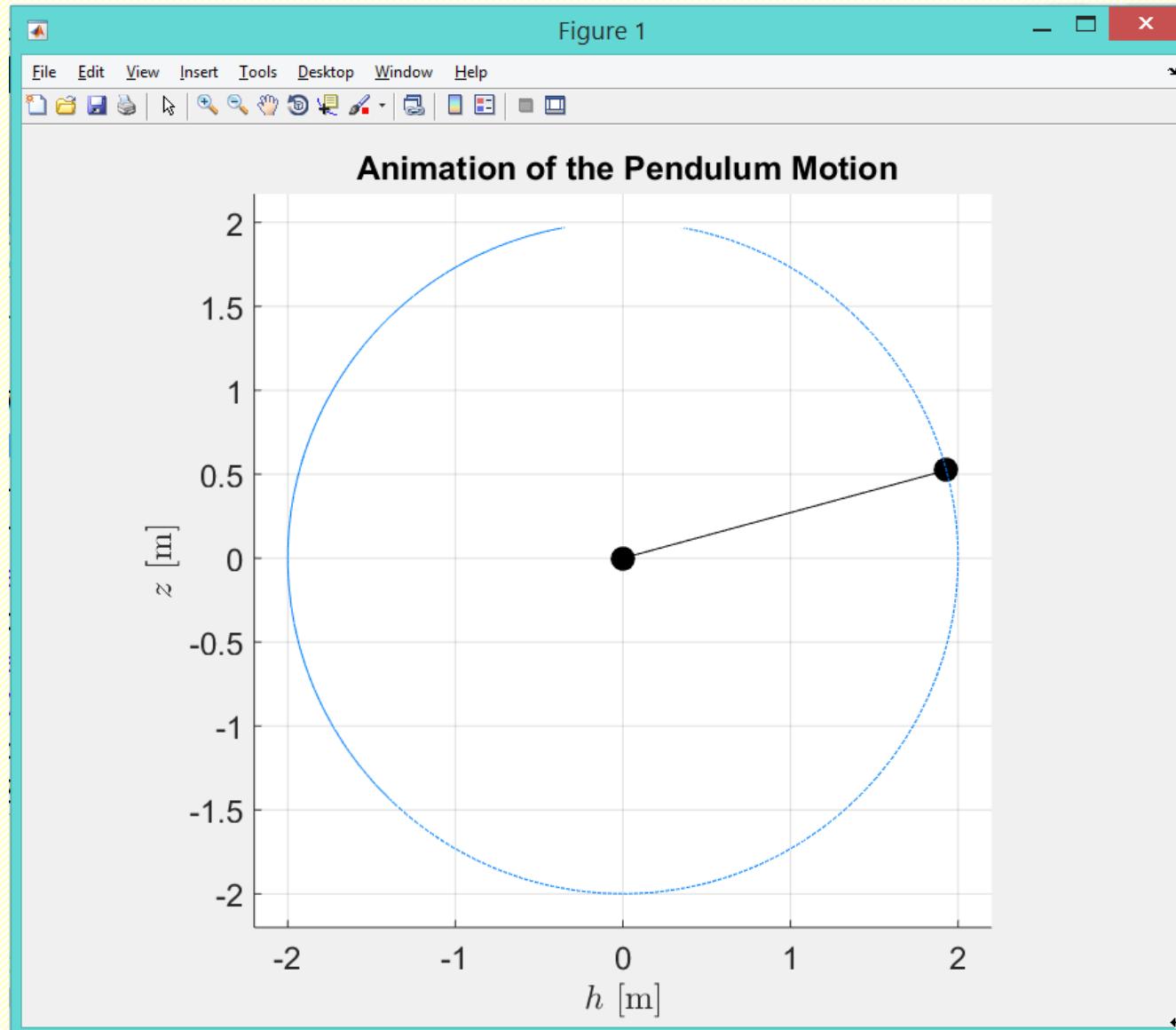


ANIMATION OF THE SYSTEM (!!!)

MATLAB SCRIPT (no annotations)

```
%% ANIMATION OF SIMPLE PENDULUM: NON-LINEAR EXAMPLE
% Designed by Prof P.M.Trivailo (C) 2018
figure;
add=0.1*L; % for better visualisation: increase "camera" embrace
hmax=max(L*sin(zz(:,1)))+add;
zmin=-L-add; zmax=max([-L*cos(zz(:,1)); 0])+add;
axis([-hmax hmax zmin zmax]); axis equal;
set(gca, 'XLimMode', 'manual'); hold on; grid on;
g=line('XData', [0 L*sin(th0)], ...
        'YData', [0 -L*cos(th0)], 'Marker', '.', 'MarkerSize', 48);
xlabel('$h$ [m]', 'Interpreter', 'LaTeX');
ylabel('$z$ [m]', 'Interpreter', 'LaTeX');
title('bf Animation of the Pendulum Motion');
set(gca, 'FontSize', 18); hold on;
plot(L*sin(zz(:,1)), -L*cos(zz(:,1))), 'LineStyle', ':', 'Color', [0
0.5 1]);
for i=1:length(tt),
    set(g, 'XData', [0 L*sin(zz(i,1))], 'YData', [0 -L*cos(zz(i,1))]);
    drawnow; pause(0.01);
end
```

Animation: Screen Snap-shot



LINEARISATION of EOMs: COMPARING LINEAR and NONLINEAR MODELS

Example: 1-DOF Pendulum (Non-Linear)

Example: Response of a 1DOF Pendulum Using Non-Linear AND Linearised EOMs

Let us consider **free vibrations** of a mass m suspended on the inextensible cable of constant length L . Using any of the known methods, and considering a 2D motion of the system, we can derive the non-linear differential equation of motion of the system in terms of the angular displacement θ - the only DOF of the system:

$$\ddot{\theta} + \frac{g}{L} \sin \theta = 0 \quad (\text{non-linear equation}) \quad (8)$$

Note that for this equation would be very similar to the equation for the 1-DOF inverted pendulum (3) except that for the sign of the second term.

The state-space equivalent of the Eq.(8) is:

$$\begin{cases} \dot{x}_1 = x_2; \\ \dot{x}_2 = -\frac{g}{L} \sin \theta \end{cases} \quad \text{non-linear model} \quad (9)$$

We can linearise the non-linear equation (8) as fol-

Example: 1-DOF Pendulum (Linearised)

lows:

$$\ddot{\theta} + \frac{g}{L}\theta = 0 \quad (\text{linearised equation}) \quad (10)$$

The state-space equivalent of Eq.(10) is:

$$\begin{cases} \dot{x}_1 = x_2; \\ \dot{x}_2 = -\frac{g}{L}x_1 \end{cases} \quad \text{linearised model} \quad (11)$$

where $x = [\theta; \dot{\theta}]$. This result can be re-written in the

matrix form:

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} = \begin{bmatrix} 0 & 1 \\ -(g/L) & 0 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} \quad \text{linearised model} \quad (12)$$

Note that these linearised equations in the state-space are for the **free-vibrations case** (i.e. the case, when the system is excited via initial conditions only, and via the application of the external force).

Scripts: 1-DOF Pendulum

(Nonlinear Model versus Linear Model: 2-file solution)

MATLAB script (MAIN PROGRAM):

```
%% Simple 1-DOF Pendulum
%% Designed by Prof P.M.Trivailo
% RMIT University
global g L tt FF A B

%% Enter system's parameters:
g=9.81; % m/s^2
L=2; % m
x0=[60*pi/180; 0]; % 60deg initial deflection
tmax=14;
tt=[0 tmax]; FF=[0 0];
```

This is where the
ARBITRARY
EXTERNAL FORCE
CAN BE ADDED!

Scripts: 1-DOF Pendulum (Nonlinear Model versus Linear Model: 2-file solution)

```
%% Running fully Non-Linear Model first
[tt1,xx1]=ode45('myodeNonLin',[0 tmax],x0);

%% Then running a Linearised Model
A=[0 1; -g/L 0]; B=[0; 0];
[tt2,xx2]=ode45('PMTode',[0 tmax], x0);

%% Plot results on the same plot
plot(tt1, xx1(:,1)*180/pi,'b',tt2,xx2(:,1)*180/pi,'r:');
legend('Non-Linear Model','Linearised Model')
xlabel('Time [s]'); ylabel('\theta [deg]'); grid
```

Scripts: 1-DOF Pendulum (Nonlinear Model versus Linear Model: 2-file solution)

MATLAB script for the **1-DOF SYSTEM**
(SEPARATE file 'myodeNonLin.m' with DIFF EOMs):

```
function [x_dot] = myodeNonLin(t,x)  
global g L  
  
x_dot(1,1)= x(2);  
x_dot(2,1)= -(g/L)*sin(x(1));
```

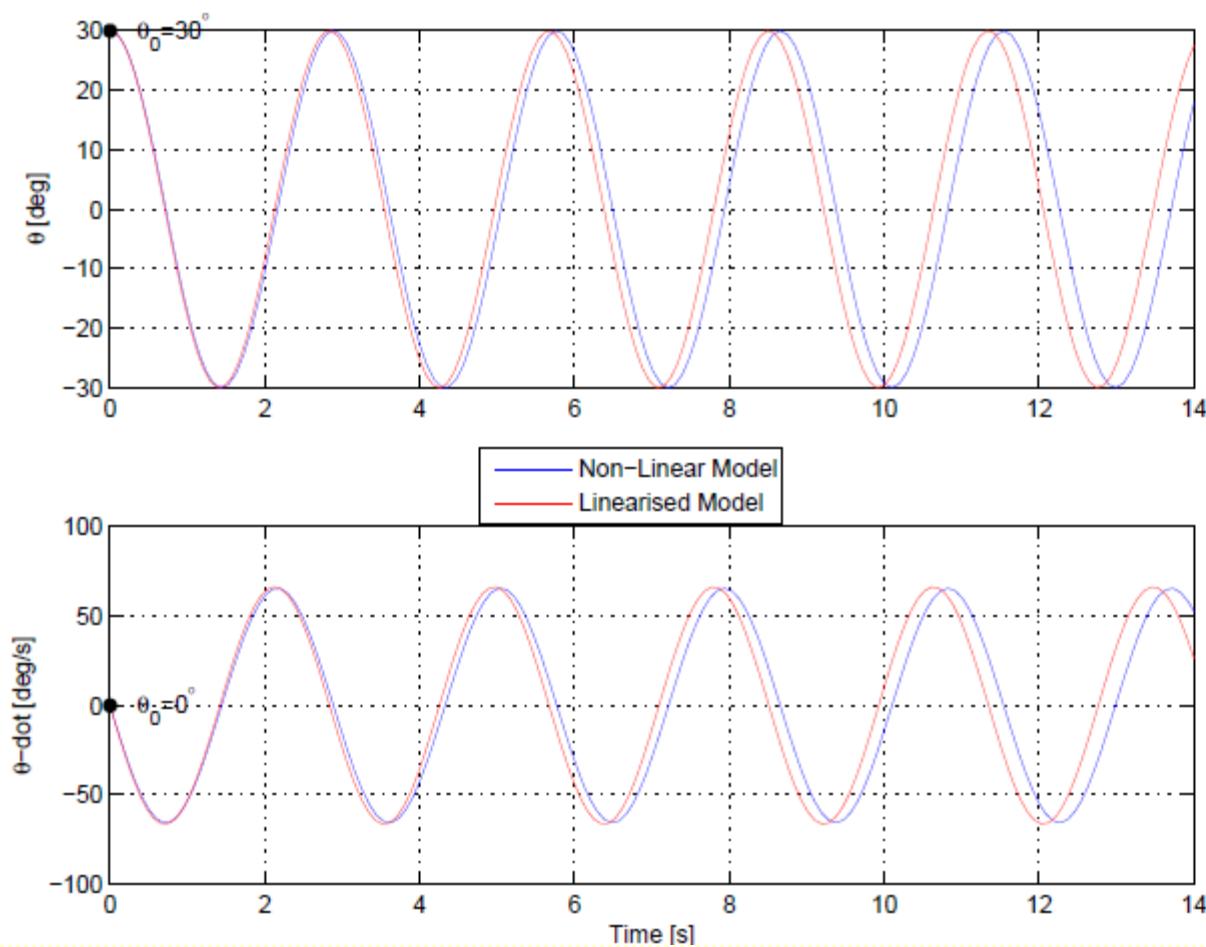
Scripts: 1-DOF Pendulum (Nonlinear Model versus Linear Model: 2-file solution)

MATLAB script for the **1-DOF SYSTEM**
(SEPARATE file ' PMTode.m' with DIFF EOMs)

```
function [x_dot] = PMTode(t,x)
global tt FF A B
Q=interp1(tt,FF,t');
x_dot = A*x+B*Q;;
```

Response to Initial Excitation

Compare Linearised and Non-Linear Model



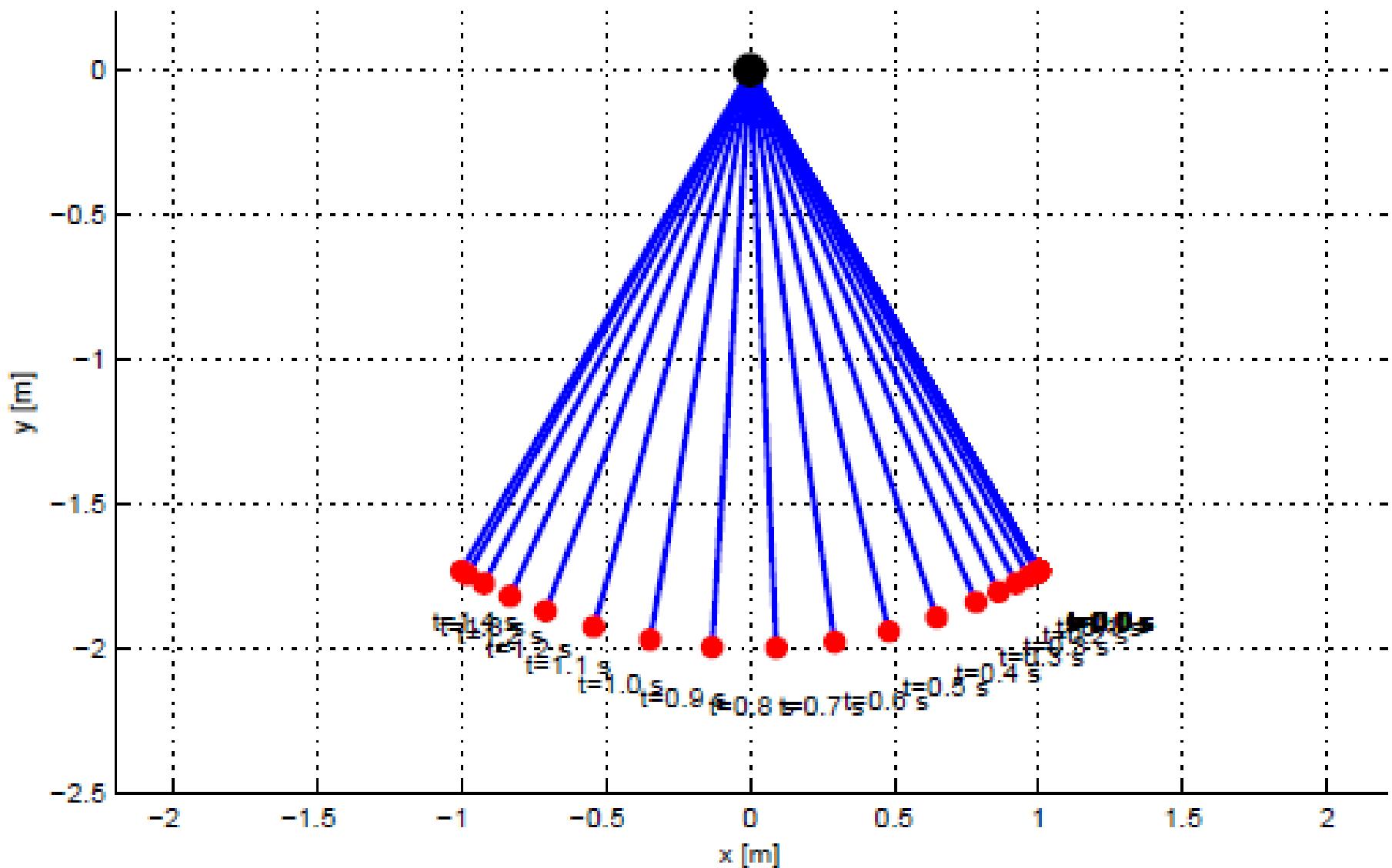
$$\theta_0 = 30^\circ$$

Results: 1-DOF Pendulum (Nonlinear Model versus Linear Model)

By the way, the PERIOD of oscillations can be calculated for small angles, using

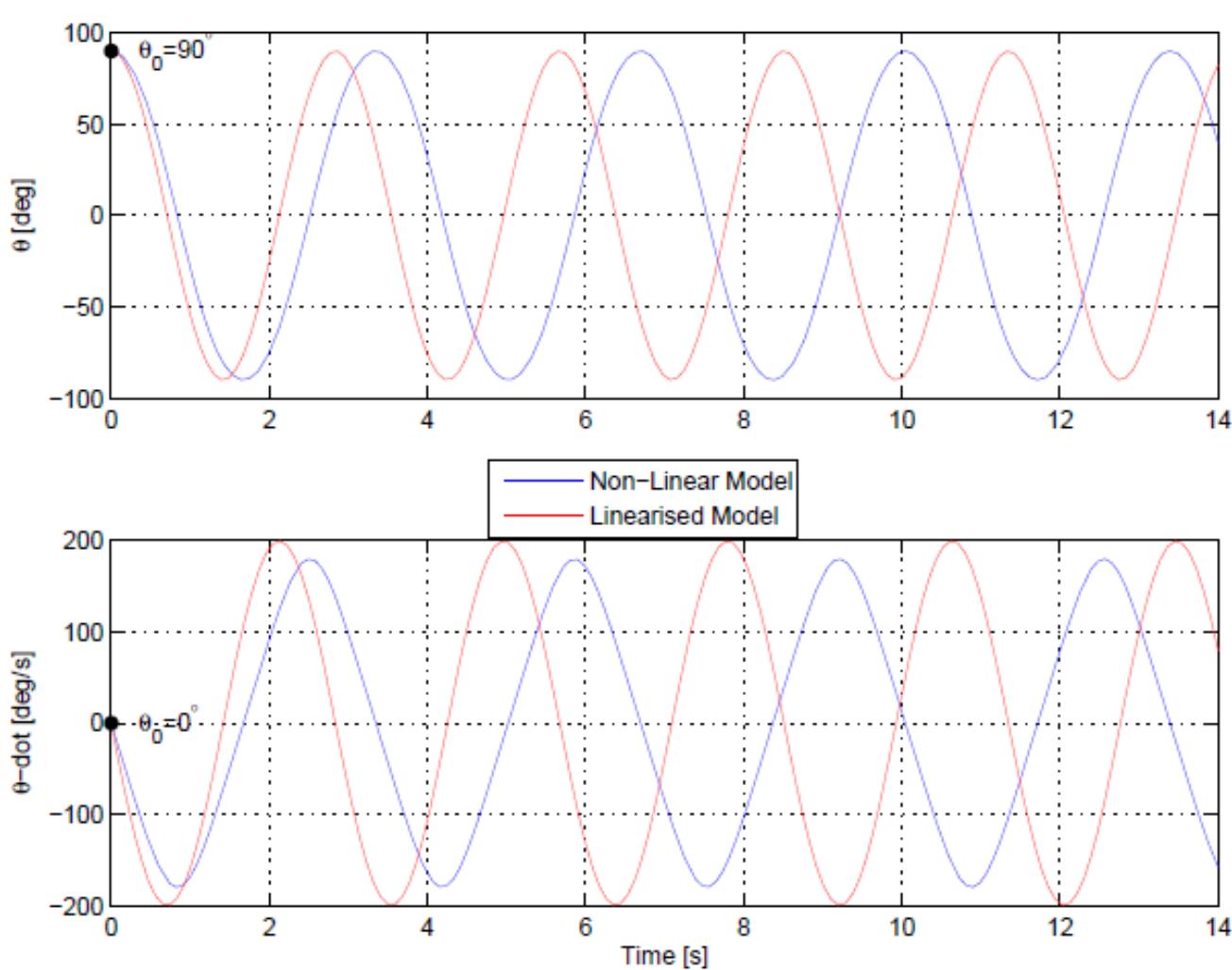
$$T \approx 2\pi \sqrt{\frac{L}{g}}$$

You may wish to check this equation using the simulation data

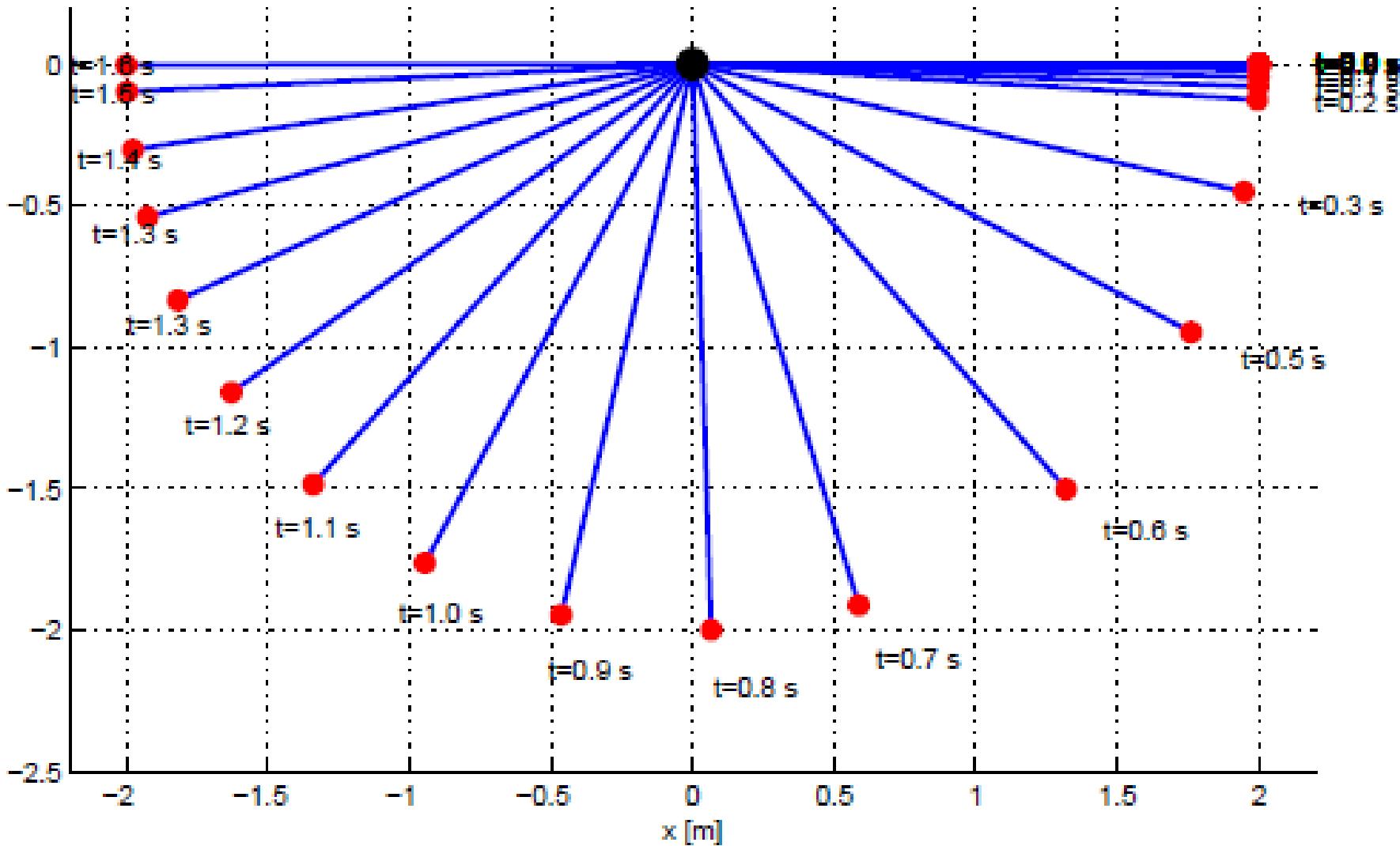


Results: 1-DOF Pendulum (Nonlinear Model versus Linear Model)

Compare Linearised and Non-Linear Model

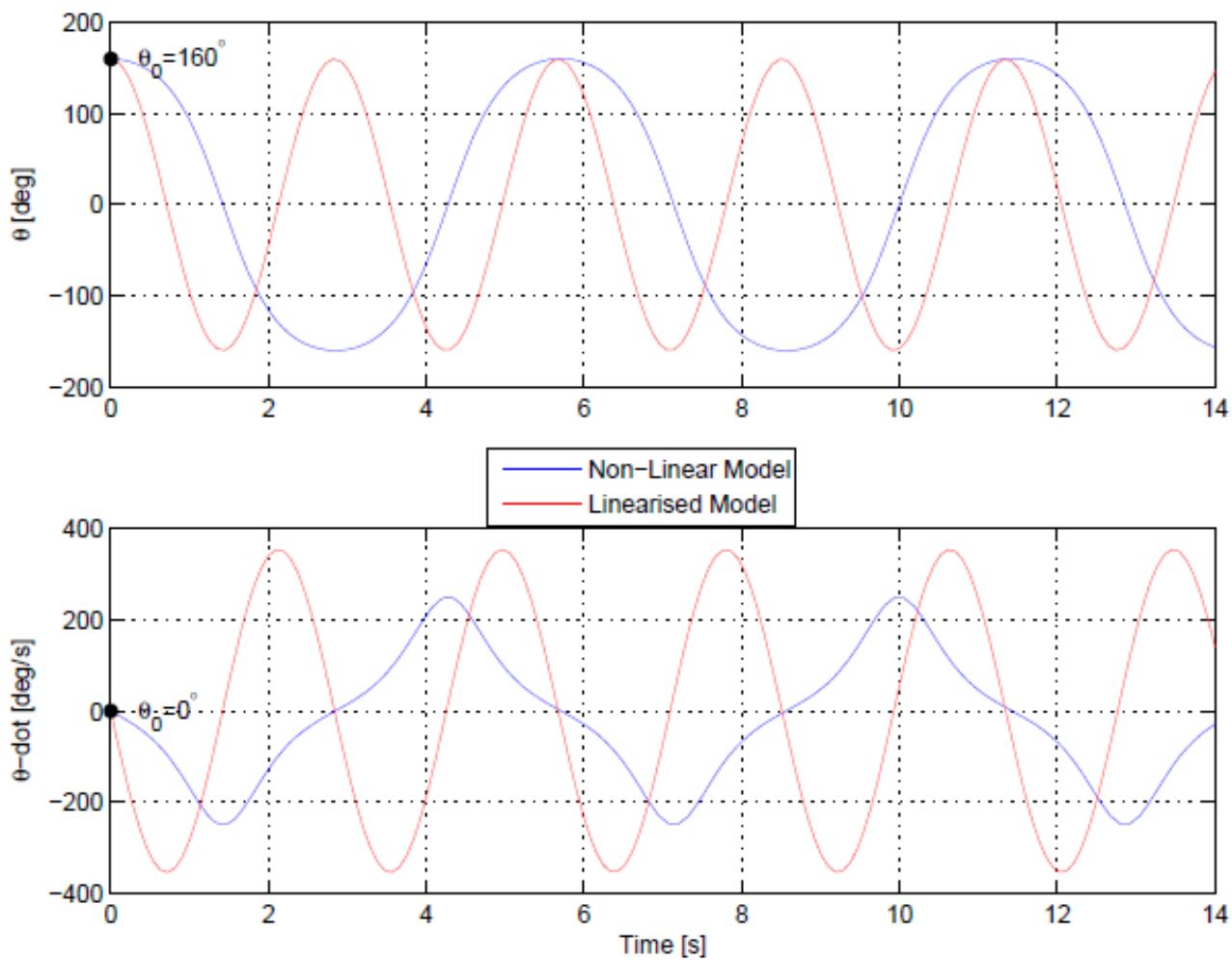


$$\theta_0 = 90^\circ$$

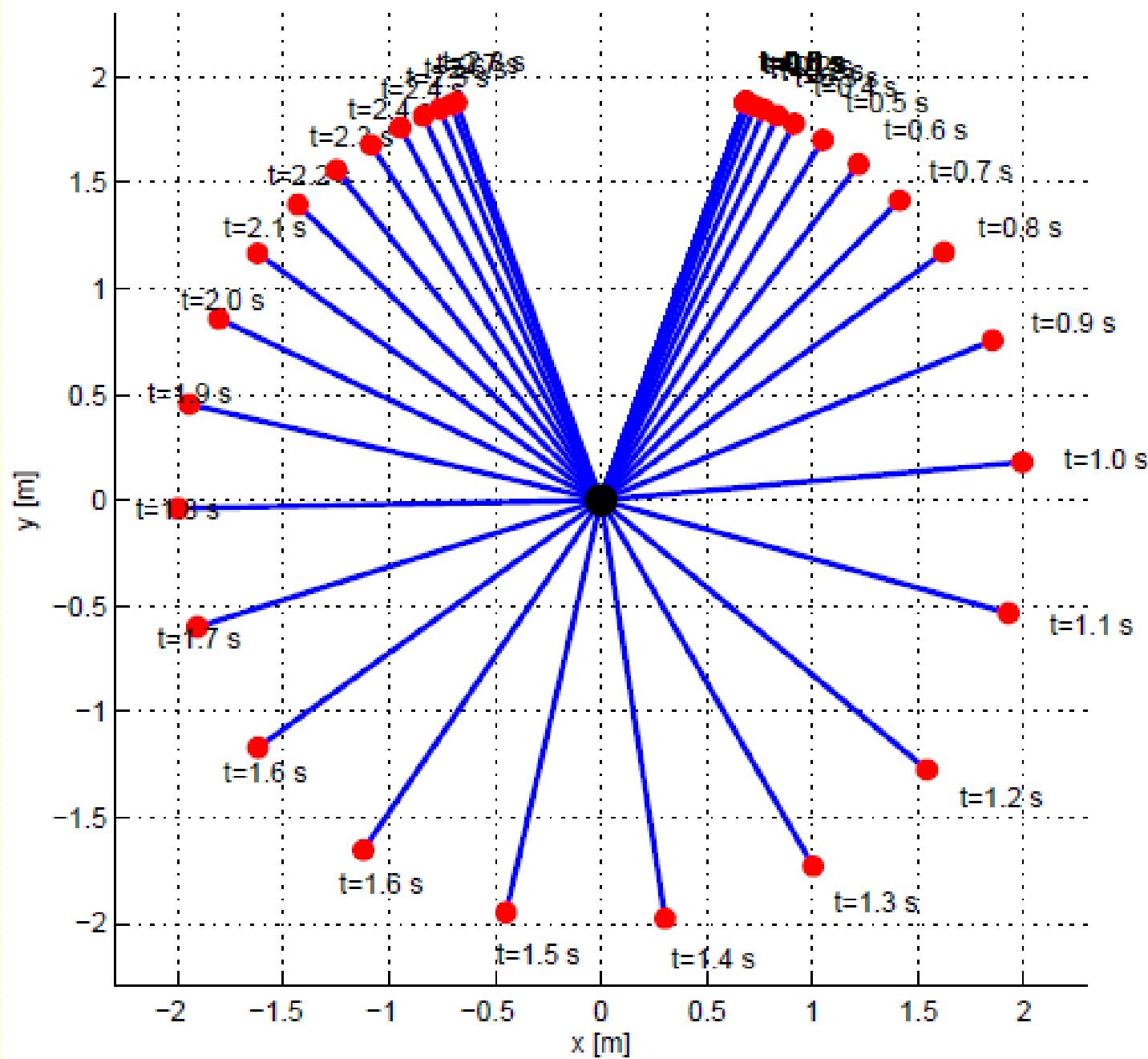


Results: 1-DOF Pendulum (Nonlinear Model versus Linear Model)

Compare Linearised and Non-Linear Model



$$\theta_0 = 160^\circ$$



END OF LECTURE-2b SLIDES

APPENDIX

Other Examples