MAS416

- https://www.facebook.com/uiamechatronics

MAS416 – Course Policy

- 40% Digital Exam
  - You will have 8 quizzes.
  - Quizzes should be done Tuesdays 2-5.
  - Each quiz includes 1 question.
  - You need to register for the quiz and then you will do it on the campus.
  - You will receive an update when the registration for the quiz is open.
  - You can do the quiz multiple times with multiple attempts.

- 60% Project report and presentation

Lecture 1-1: Overview

- Integrators
- Numerical Integrators
- Forward Euler (First Order)

Ordinary Differential Equations (ODEs): Initial-Value Problems

$$\dot{\underline{q}} = \underline{\Phi}(t, \underline{q})$$

$$\begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{Bmatrix} = \begin{Bmatrix} \Phi_1(t, \underline{q}) \\ \Phi_2(t, \underline{q}) \\ \vdots \\ \Phi_n(t, \underline{q}) \end{Bmatrix}$$

$$\underline{q}(t=0) = \begin{Bmatrix} q_1^{(0)} \\ q_2^{(0)} \\ \vdots \\ q_n^{(0)} \end{Bmatrix}$$

- $\underline{q} = \{q_1 \quad q_2 \quad \dots \quad q_n\}'$: array of state variables
- $\underline{\Phi}$: array of characteristic functions

Ordinary Differential Equations (ODEs): Initial-Value Problems

- ODE is solved during a time interval $t \in \left[t_{start}, t_{final} \right]$
- We are interested in finding the behavior of state variables or any other function of state variables as a function of time

$$\underline{\dot{q}} = \underline{\Phi}(t, \underline{q}) \qquad \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{Bmatrix} = \begin{Bmatrix} \Phi_1(t, \underline{q}) \\ \Phi_2(t, \underline{q}) \\ \vdots \\ \Phi_n(t, \underline{q}) \end{Bmatrix}$$

- We always need an initial condition to find the solution of ODE

$$\underline{q}(t=0) = \begin{Bmatrix} q_1^{(0)} \\ q_2^{(0)} \\ \vdots \\ q_n^{(0)} \end{Bmatrix}$$

Grand Scheme: Continuous Solver

$$\underline{q}, t \;\Rightarrow\; \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{Bmatrix} = \begin{Bmatrix} \Phi_1(t,\underline{q}) \\ \Phi_2(t,\underline{q}) \\ \vdots \\ \Phi_n(t,\underline{q}) \end{Bmatrix} \;\Rightarrow\; \dot{\underline{q}} \;\overset{\int}{\Rightarrow}\; \underline{q}$$

Grand Scheme: Numerical Solver

$$\begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{Bmatrix} = \begin{Bmatrix} \Phi_1(t,\underline{q}) \\ \Phi_2(t,\underline{q}) \\ \vdots \\ \Phi_n(t,\underline{q}) \end{Bmatrix}$$

$$\underset{t=0}{\begin{Bmatrix} q_1^{(0)} \\ q_2^{(0)} \\ \vdots \\ q_n^{(0)} \end{Bmatrix}} \xRightarrow{\begin{Bmatrix} \Phi_1(t,\underline{q}) \\ \Phi_2(t,\underline{q}) \\ \vdots \\ \Phi_n(t,\underline{q}) \end{Bmatrix}} \begin{Bmatrix} \dot{q}_1^{(0)} \\ \dot{q}_2^{(0)} \\ \vdots \\ \dot{q}_n^{(0)} \end{Bmatrix} \xRightarrow{\int} \underset{t=t_1}{\begin{Bmatrix} q_1^{(t_1)} \\ q_2^{(t_1)} \\ \vdots \\ q_n^{(t_1)} \end{Bmatrix}} \xRightarrow{\begin{Bmatrix} \Phi_1(t,\underline{q}) \\ \Phi_2(t,\underline{q}) \\ \vdots \\ \Phi_n(t,\underline{q}) \end{Bmatrix}} \begin{Bmatrix} \dot{q}_1^{(t_1)} \\ \dot{q}_2^{(t_1)} \\ \vdots \\ \dot{q}_n^{(t_1)} \end{Bmatrix} \xRightarrow{\int} \underset{t=t_2}{\begin{Bmatrix} q_1^{(t_2)} \\ q_2^{(t_2)} \\ \vdots \\ q_n^{(t_2)} \end{Bmatrix}} \xRightarrow{\begin{Bmatrix} \Phi_1(t,\underline{q}) \\ \Phi_2(t,\underline{q}) \\ \vdots \\ \Phi_n(t,\underline{q}) \end{Bmatrix}}$$

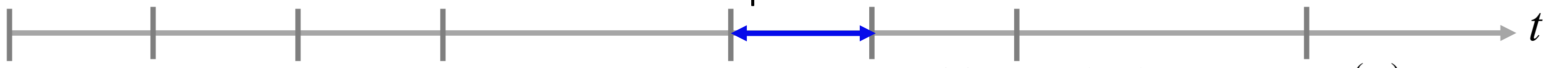
Grand Scheme: Numerical Solver

$$\underline{q}^{(1)} \qquad \underline{q}^{(2)} \qquad \underline{q}^{(j-1)} \qquad \underline{q}^{(j)} \qquad \underline{q}^{(j+1)}$$

$$\begin{Bmatrix} q_1^{(1)} \\ q_2^{(1)} \\ \vdots \\ q_n^{(1)} \end{Bmatrix} \quad \begin{Bmatrix} q_1^{(2)} \\ q_2^{(2)} \\ \vdots \\ q_n^{(2)} \end{Bmatrix} \qquad \begin{Bmatrix} q_1^{(j-1)} \\ q_2^{(j-1)} \\ \vdots \\ q_n^{(j-1)} \end{Bmatrix} \begin{Bmatrix} q_1^{(j)} \\ q_2^{(j)} \\ \vdots \\ q_n^{(j)} \end{Bmatrix} \begin{Bmatrix} q_1^{(j+1)} \\ q_2^{(j+1)} \\ \vdots \\ q_n^{(j+1)} \end{Bmatrix}$$

Time Step $j=1$ $\quad j=2$ $\qquad j-1$ $\quad j$ $\quad j+1$ $\qquad j=m$

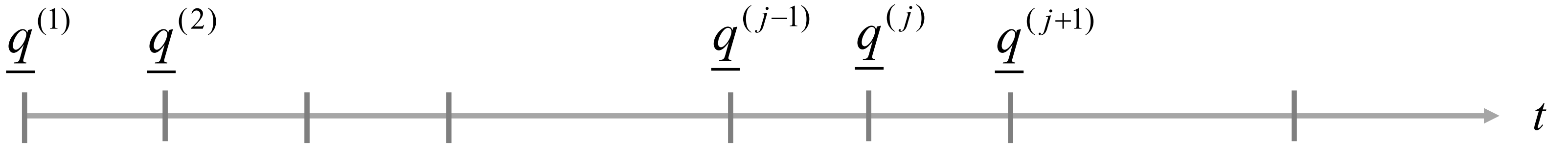
Step size = h



$t=t^{(1)}$ $\quad t=t^{(2)}$ $\qquad t=t^{(j-1)}$ $\; t=t^{(j)}$ $\; t=t^{(j+1)}$ $\qquad t=t^{(m)}$

Numerical Solver for Initial-Value Problems

- Explicit method: calculates $\underline{q}^{(j+1)}$ using $\underline{q}^{(j)}$ and $t^{(j)}$.
- Implicit method: finds a solution by solving an equation involving both $\underline{q}^{(j)}$ and $\underline{q}^{(j+1)}$.
    - The implicit algorithms are iterative.
    - We need an estimate of $\underline{q}^{(j+1)}$ to start the iteration of the formula.
    - We can use an explicit formula to estimate $\underline{q}^{(j+1)}$ (*predictor* step).
    - Then the implicit formula is used to correct the predicted value of $\underline{q}^{(j+1)}$ (*corrector* step). Normally, an algorithm iterates on the $\underline{q}^{(j+1)}$ in the corrector step.



Numerical Solver for Initial-Value Problems

- Numerical algorithms based on *Taylor series expansion*
  - They use Taylor series expansion to find the derivatives of the function with respect to t. A k-th order algorithm needs the derivatives up to k-1. These derivatives, in general, are not readily available.

- Numerical algorithms based on *Polynomial approximation*
  - A polynomial of sufficiently high order can, in principle, be used to calculate $\underline{q}^{(j+1)}$ to any desired accuracy. A k-th-degree polynomial is referred to as an algorithm of order k. In practice, the amount of computation increases with the order of polynomial.

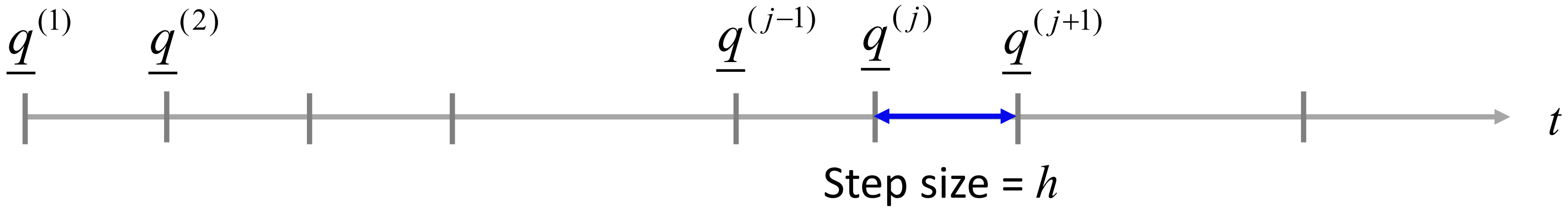
Numerical Solver for Initial-Value Problems

- The accuracy of an algorithm is directly proportional to its *order*.

- In general, higher order algorithms generate more accurate solution.

Integration Based on Taylor Series Expansion

$$\underline{q}^{(j+1)} = \underline{q}^{(j)} + \frac{1}{1!}\dot{\underline{q}}^{(j)}h + \frac{1}{2!}\ddot{\underline{q}}^{(j)}h^2 + \frac{1}{3!}\dddot{\underline{q}}^{(j)}h^3 + \ldots$$



Forward Euler (First Order)

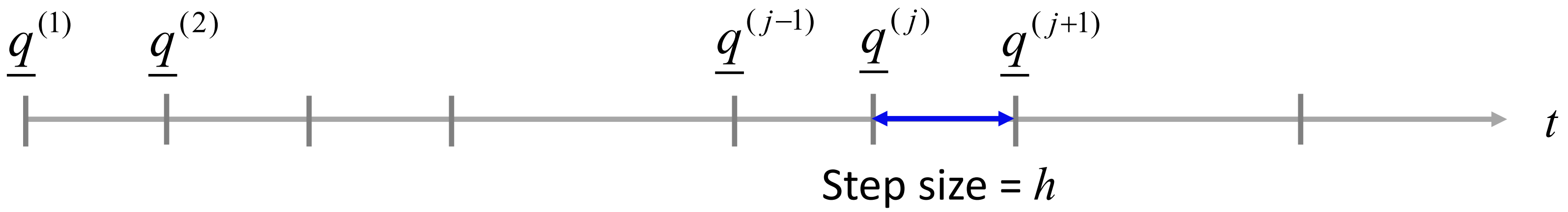
$$\underline{q}^{(j+1)} = \underline{q}^{(j)} + \frac{1}{1!}\underline{\dot{q}}^{(j)}h + \frac{1}{2!}\underline{\ddot{q}}^{(j)}h^2 + \frac{1}{3!}\underline{\dddot{q}}^{(j)}h^3 + \dots$$

$$\underline{q}^{(j+1)} = \underline{q}^{(j)} + \frac{1}{1!}\underline{\dot{q}}^{(j)}h$$

$$\boxed{\underline{q}^{(j+1)} = \underline{q}^{(j)} + \underline{\dot{q}}^{(j)}h}$$

$$\underline{\dot{q}} = \underline{\Phi}(t,\underline{q}) \qquad \underline{\dot{q}}^{(j)} = \underline{\Phi}(t^{(j)},\underline{q}^{(j)})$$

$$\boxed{\underline{q}^{(j+1)} = \underline{q}^{(j)} + \underline{\Phi}(t^{(j)},\underline{q}^{(j)})h}$$



State Variables

$$\underline{\dot{q}} = \underline{\Phi}(t, \underline{q}) \qquad\qquad \underline{q}$$

Physics	Differential Equations	State Variables	Order of the ODE
Mechanics (mass)	$\ddot{x} = \Phi(t, x, \dot{x})$	$\underline{q} = \{x \;\; \dot{x}\}'$	2
Mechanics (inertia)	$\ddot{\theta} = \Phi(t, \theta, \dot{\theta})$	$\underline{q} = \{\theta \;\; \dot{\theta}\}'$	2
Electronics	$\dot{v} = \Phi(t, i, v)$	$q = v$	1
Electronics	$\frac{d}{dt} i = \Phi(t, i, v)$	$q = i$	1
Hydraulics	$\dot{p} = \Phi(t, p)$	$q = p$	1

Lecture 1-2: Overview

- Example of Numerical Integration of a First Order System

Example: Numerical Integration of a First Order System

$$\dot{x} = -2xt$$

$$x(0) = 1$$

$$t \in [0,3]s$$

$$\dot{x} = -2xt$$

$$\frac{dx}{dt} = -2xt$$

$$\frac{dx}{x} = -2tdt$$

$$\ln x = -t^2 + C$$

$$\ln 1 = -0^2 + C \qquad C = 0$$

$$\ln x = -t^2$$

$$x = e^{-t^2}$$

$$\dot{x} = -2xt$$

$$x(0) = 1$$

$$t \in [0,3]s$$

State variable: x

$$\boxed{q^{(j+1)} = q^{(j)} + \dot{q}^{(j)} h}$$

$$x^{(j+1)} = x^{(j)} + \dot{x}^{(j)} h$$

$$\dot{x}^{(j)} = -2x^{(j)} t^{(j)}$$

Example: Numerical Integration of a First Order System

MATLAB Program

```
clc; close all; clear all
%Time data
EndTime=3.0;
StepTime=1e-5; %h
%Initialize time and state variables
Time=0.0;
x=1;
Counter=1;
%Start time integration
while Time<EndTime
    % Computing qDot
    xDot = -2*x*Time;
    % Save data for plotting
    Time_Plot(Counter)=Time;
    x_Plot(Counter)=x;
    xDot_Plot(Counter)=xDot;
    %Time integrate
    x=x+xDot*StepTime;
    Time=Time+StepTime;
    Counter=Counter+1;
end
plot(Time_Plot,x_Plot,'LineWidth',2);

% Exact Analytical Solution
xExact = exp(-Time_Plot.^2);
hold on
plot(Time_Plot,xExact,'LineWidth',2);
grid on
```

$$\dot{x} = -2xt$$
$$x(0) = 1$$
$$t \in [0,3]s$$

$$x = e^{-t^2}$$

$$x^{(j+1)} = x^{(j)} + \dot{x}^{(j)} h$$
$$\dot{x}^{(j)} = -2x^{(j)} t^{(j)}$$

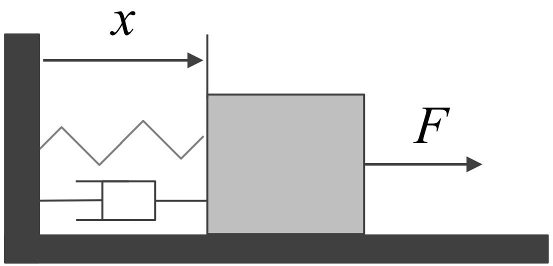
Lecture 1-3: Overview

- Example of Numerical Integration of a Second Order System

- Runge-Kutta Algorithm

Example: Numerical Integration of a Second Order System

The differential equation $2\ddot{x}+6\dot{x}+10x=15$ governs the motion of the following mass-spring-damper system, where x is the position in meters, $\dot{x}$ is the velocity in m/s, and $\ddot{x}$ is the acceleration of the block in m/s^2. Determine the state variables of the system. Write a forward Euler integration scheme to simulate the motion of the system from 0 to 40 seconds with the initial conditions $x(0)=4m$ and $\dot{x}(0)=0m/s$. Use the following values for the integration time step: 1e-5, 1e-2, 1e-1, 0.9 s. What do you observe?



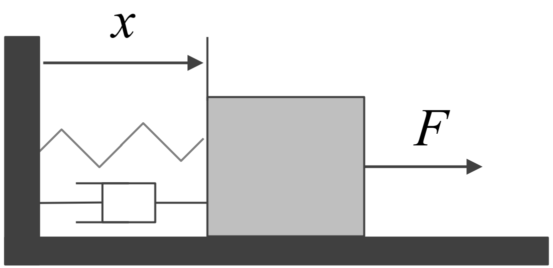
Example: Numerical Integration of a Second Order System

$$2\ddot{x} + 6\dot{x} + 10x = 15$$

State variables: $\underline{q} = \begin{Bmatrix} x \\ \dot{x} \end{Bmatrix}$

$$x(0) = 4$$
$$\dot{x}(0) = 0$$

$$\boxed{q^{(j+1)} = q^{(j)} + \dot{q}^{(j)} h}$$



$$x^{(j+1)} = x^{(j)} + \dot{x}^{(j)} h$$

$$\dot{x}^{(j+1)} = \dot{x}^{(j)} + \ddot{x}^{(j)} h$$

$$\ddot{x}^{(j)} = \frac{1}{2}(15 - 10x^{(j)} - 6\dot{x}^{(j)})$$

Example: Numerical Integration of a Second Order System

MATLAB Program

```
clc; clear;
tic
%Time data
EndTime=50;
StepTime=1e-5;
%Initialize time and state variables
Time=0.0;
x=4;
xDot=0;
Counter=1;
%Start time integration
while Time<EndTime
    xDotDot=1/2*(15-10*x-6*xDot);
    %Save data for plotting
    Time_Plot(Counter)=Time;
    x_Plot(Counter)=x;
    xDot_Plot(Counter)=xDot;
    xDotDot_Plot(Counter)=xDotDot;
    %Time integrate
    x=x+xDot*StepTime;
    xDot=xDot+xDotDot*StepTime;
    Time=Time+StepTime;
    Counter=Counter+1;
end
toc
```

$$x^{(j+1)} = x^{(j)} + \dot{x}^{(j)} h$$

$$\dot{x}^{(j+1)} = \dot{x}^{(j)} + \ddot{x}^{(j)} h$$

$$\ddot{x}^{(j)} = \frac{1}{2}(15 - 10x^{(j)} - 6\dot{x}^{(j)})$$

Runge-Kutta Algorithm

$$q^{(j+1)} = q^{(j)} + \Delta t\, f$$

$$f = \tfrac{1}{6}\left(f_1 + 2f_2 + 2f_3 + f_4\right)$$

$$f_1 = \Phi(t^{(j)}, q^{(j)})$$

$$f_2 = \Phi\left(t^{(j)} + \frac{\Delta t}{2}, q^{(j)} + \frac{\Delta t}{2} f_1 \right)$$

$$f_3 = \Phi\left(t^{(j)} + \frac{\Delta t}{2}, q^{(j)} + \frac{\Delta t}{2} f_2 \right)$$

$$f_4 = \Phi\left(t^{(j)} + \frac{\Delta t}{2}, q^{(j)} + \Delta t\, f_3 \right)$$

- A fourth-order algorithm, therefore, truncation error remains relatively small even for a relatively large step size.
- The function $\Phi(t,q)$ must be evaluated four times at each time step while the values of the function are not used in any subsequent computations.
- It is not as efficient as some of the multi-step algorithms.

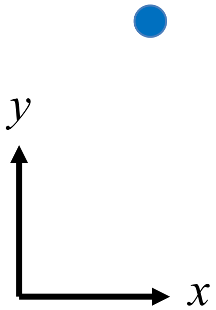
Lecture 1-4: Overview

- Modeling Mechanical Systems
- Reference Frame
- Degrees-of-Freedom

Degrees-of-Freedom

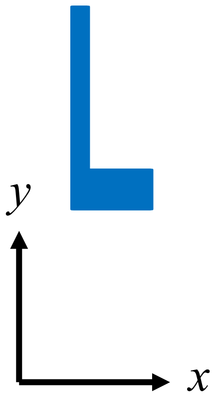
- A mechanical system's DoF is equal to the number of independent entities needed to uniquely define its position in space at any given time.

- A free particle has 2 DoF in plane motion.



Degrees-of-Freedom

- A free body (link) on a plane has 3 DoFs.
- Any general motion (displacement) of a free planar link can be decomposed into three independent motions.



Lecture 1-5: Overview

- One Dimensional Mechanics: Translational Motion
- Linear Spring
- Linear Damper
- Mass-Spring-Damper System

One Dimensional Mechanics: Translational Motion

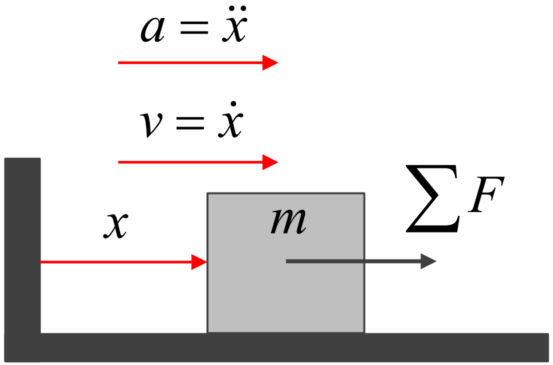
- x: Coordinate of the particle

- Newton's Second Law of Motions

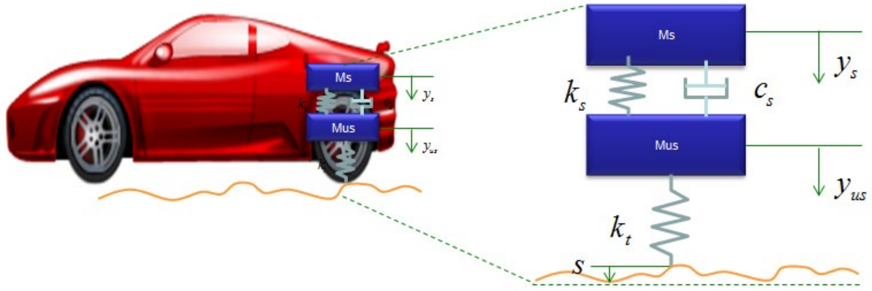
$$\sum F = ma = m\ddot{x}$$

- Here x is the absolute coordinate since it is measured from a fixed reference (e.g., wall)

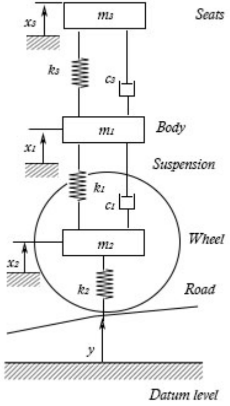
- The acceleration on the right-hand-side must be absolute acceleration.



Why One-Dimensional Mechanics: Translational Motion



http://www.sharetechnote.com/html/DE_Modeling_Example_SpringMass.html

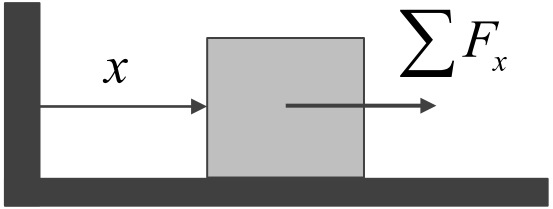


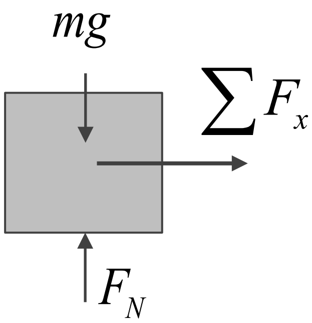
https://study.com/academy/answer/the-figure-shows-a-quarter-car-model-that-includes-the-mass-of-the-seats-including-passengers-the-constants-k-3-and-c-3-represent-the-stiffness-and-damping-in-the-seat-supports-to-derive-the-equa.html

One Dimensional Mechanics: Translational Motion

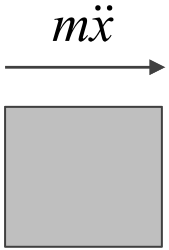
- Newton's Second Law of Motions

$$\sum F_x = ma_x = m\ddot{x}$$





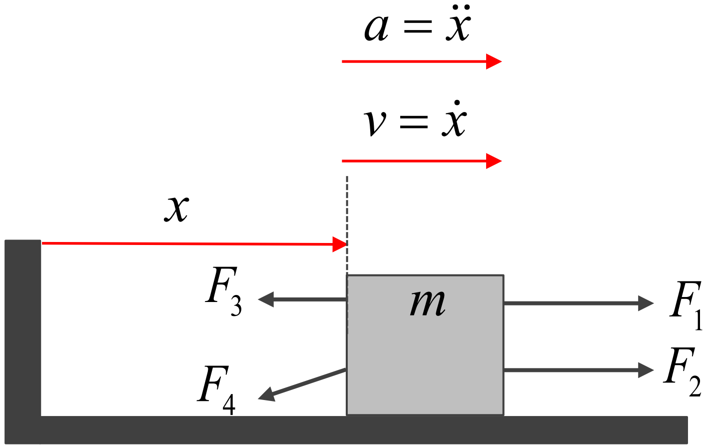
Free-body Diagram



Kinetic Diagram

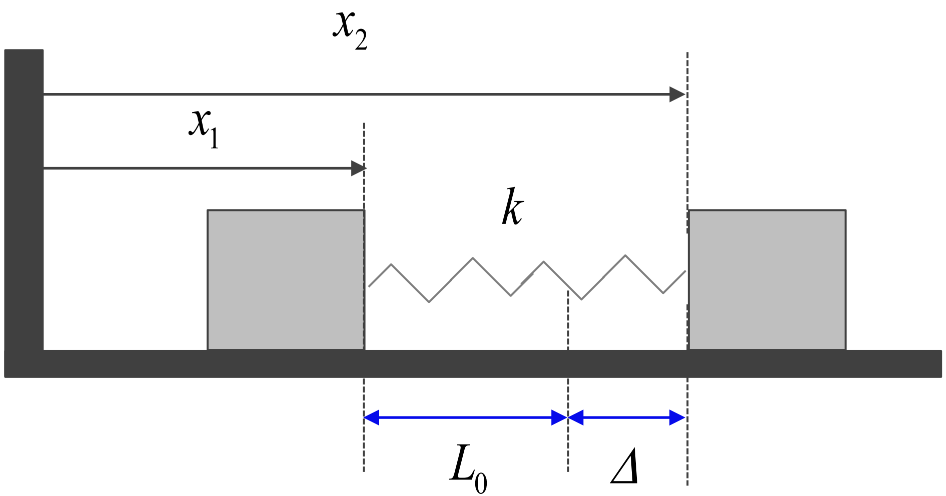
One Dimensional Mechanics: Translational Motion

$$\sum F = ma = m\ddot{x}$$



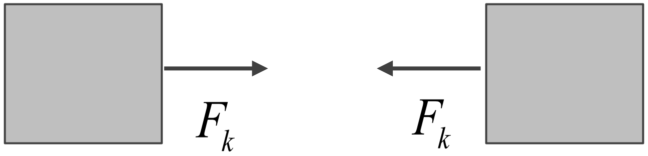
Linear Spring

- L_0 : undeformed/unstretched length of spring
- Δ : the deformation of the spring
- $x_2 - x_1$: current length of the spring
- $\Delta = (x_2 - x_1) - L_0$
- $F_k = k\Delta$

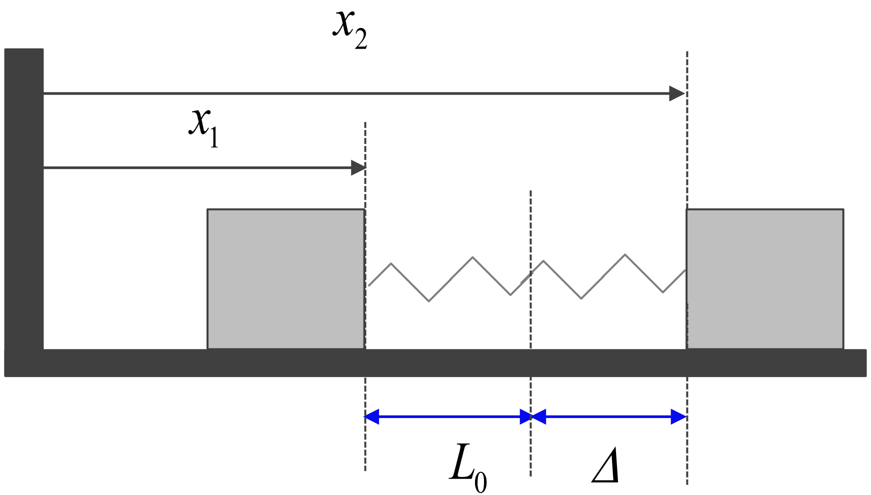


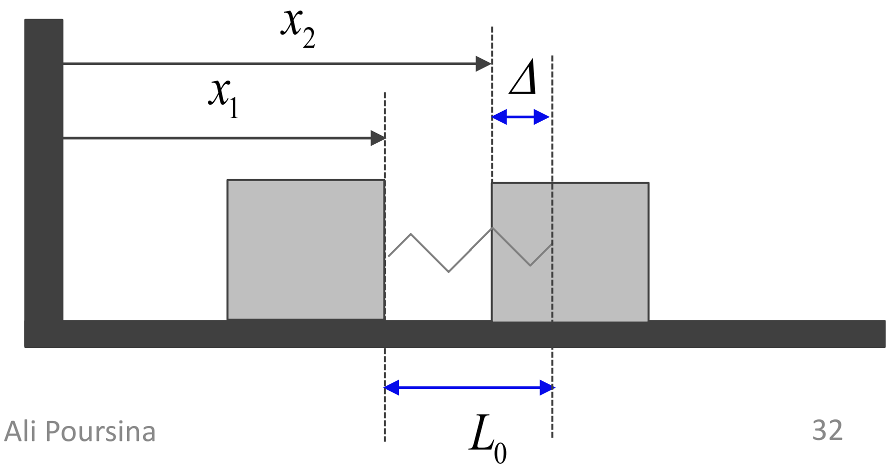
Linear Spring

- $\Delta = (x_2 - x_1) - L_0$
- $F_k = k\Delta$
- $\Delta > 0$: Spring is stretched
  - It pulls the bodies towards each other



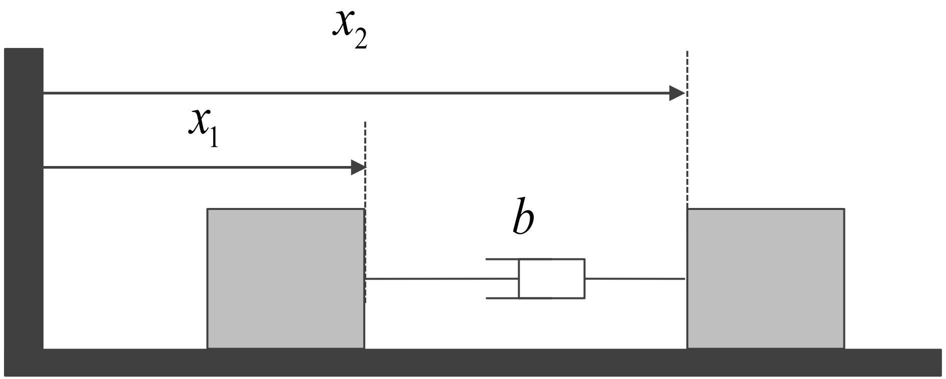
- $\Delta < 0$: Spring is compressed
  - It pushes the bodies apart

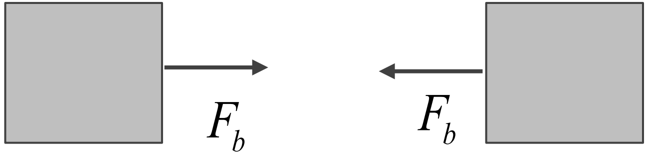




Linear Damper

- $\dot{\Delta} = \dot{x}_2 - \dot{x}_1$
- $F_b = b\dot{\Delta}$
- $\dot{\Delta} = \dot{x}_2 - \dot{x}_1 > 0$: Damper is elongating
    - It pulls the bodies towards each other



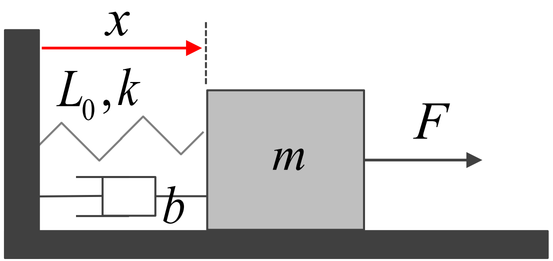


- $\dot{\Delta} = \dot{x}_2 - \dot{x}_1 < 0$: Damper is shortening
    - It pushes the bodies apart

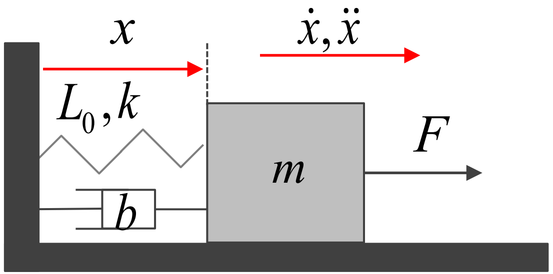
Mass-Spring-Damper System

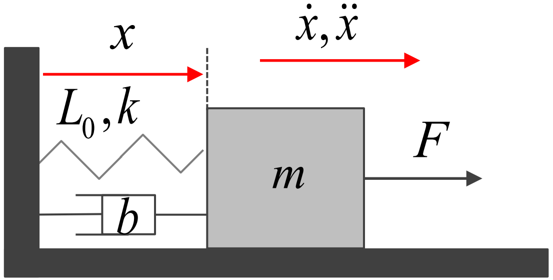
Form the equations of motion of the system.
Determine the state variables.
How many initial conditions are required to simulate the system?
Run your simulation for 10 seconds and plot state variables.

$$m = 2000\,gr$$
$$k = 50\,N/m$$
$$L_0 = 0.1\,m$$
$$b = 20\,Ns/m$$
$$F = 10\sin(\omega t)$$
$$\omega = \pi/2\ rad/s$$
$$x(0) = 0.3\,m$$
$$\dot{x}(0) = 0.0\,m/s$$



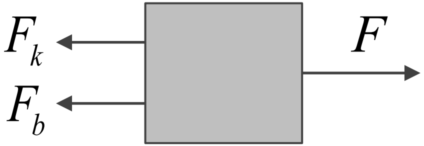
Mass-Spring-Damper System

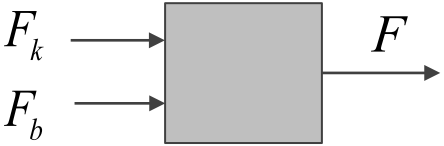




$$\underline{q} = \begin{Bmatrix} q_1 \\ q_2 \end{Bmatrix} = \begin{Bmatrix} x \\ \dot{x} \end{Bmatrix}$$

$$\underline{\dot{q}} = \begin{Bmatrix} \dot{x} \\ \ddot{x} \end{Bmatrix}$$





$$\Delta = x - L_0 \qquad F_k = k\Delta$$

$$\dot{\Delta} = \dot{x} \qquad F_b = b\dot{\Delta}$$

$$\sum F = m\ddot{x} \qquad F - F_k - F_b = m\ddot{x}$$

$$\Delta = L_0 - x \qquad F_k = k\Delta$$

$$\dot{\Delta} = -\dot{x} \qquad F_b = b\dot{\Delta}$$

$$\sum F = m\ddot{x} \qquad F + F_k + F_b = m\ddot{x}$$

Example: Numerical Integration of a Second Order System

MATLAB Program

```
clc;  clear all; close all;
tic

% System Properties
m = 2; % kg (mass of the particle)
k = 50; % N/m (spring stiffness)
L0 = 0.1; % m (undeformed length of the spring)
b = 20; % Ns/m (damping coefficient)
F_amp = 10; % N (Amplitude of the applied
force)
omega = pi/2; % rad/s (frequency of the applied
force)

%Time data
EndTime=10;
StepTime=0.01
```

```
%Initialize time and state variables
Time=0.0;
x=0.3;
xDot=0;
Counter=1;
```

$$m = 2000 gr$$

$$k = 50 N / m$$

$$L_0 = 0.1 m$$

$$b = 20 Ns / m$$

$$F = 10 \sin(\omega t)$$

$$\omega = \pi / 2 \; rad / s$$

$$x(0) = 0.3 m$$

$$\dot{x}(0) = 0.0 m / s$$

Example: Numerical Integration of a Second Order System

MATLAB Program

```
%Start time integration
while Time<EndTime
 % Compute external force
    F = F_amp * sin(omega*Time);

    % Compute the spring force
    delta = x - L0; % deformation of the spring
    Fk = k*delta; % spring force

    % Compute the damping force
    deltaDot = xDot;
    Fb = b *deltaDot; % damper force

    % Compute the acceleration
    xDotDot=1/m*(F-Fk-Fb);

    %Save data for plotting
    Time_Plot(Counter)=Time;
    x_Plot(Counter)=x;
    xDot_Plot(Counter)=xDot;
    xDotDot_Plot(Counter)=xDotDot;

%Time integrate
    x=x+xDot*StepTime;
    xDot=xDot+xDotDot*StepTime;
    Time=Time+StepTime;
    Counter=Counter+1;
end
toc
```

$$\Delta = x - L_0 \qquad F_k = k\Delta$$

$$\dot{\Delta} = \dot{x} \qquad F_b = b\dot{\Delta}$$

$$\ddot{x}^{(j)} = \frac{1}{m}(F - F_k - F_b)$$

$$x^{(j+1)} = x^{(j)} + \dot{x}^{(j)}h$$

$$\dot{x}^{(j+1)} = \dot{x}^{(j)} + \ddot{x}^{(j)}h$$

Example: Numerical Integration of a Second Order System

MATLAB Program

```
subplot(3,1,1)
plot(Time_Plot,x_Plot,'LineWidth',2)% Plotting x
xlabel('$t~(s)$','Interpreter','latex');
ylabel('$x~(m)$','Interpreter','latex');
box on
grid on
hold on
subplot(3,1,2)

plot(Time_Plot,xDot_Plot,'LineWidth',2) % Plotting xDot
xlabel('$t~(s)$','Interpreter','latex');
ylabel('$\dot{x}~(m/s)$','Interpreter','latex')
box on
grid on
```

```
subplot(3,1,3)
plot(Time_Plot,xDotDot_Plot,'LineWidth',2) % Plotting xDot
xlabel('$t~(s)$','Interpreter','latex');
ylabel('$\ddot{x}~(m/s^2)$','Interpreter','latex')
box on
grid on
```