# RIYA Week 2 Presentation

## Dynamics of 2-Spring Stack

Jacob Thomas Sony
IIT Bombay

## Objective

- To simulate the dynamics of a mass suspended on a 2-spring stack in MATLAB

## Tasks accomplished

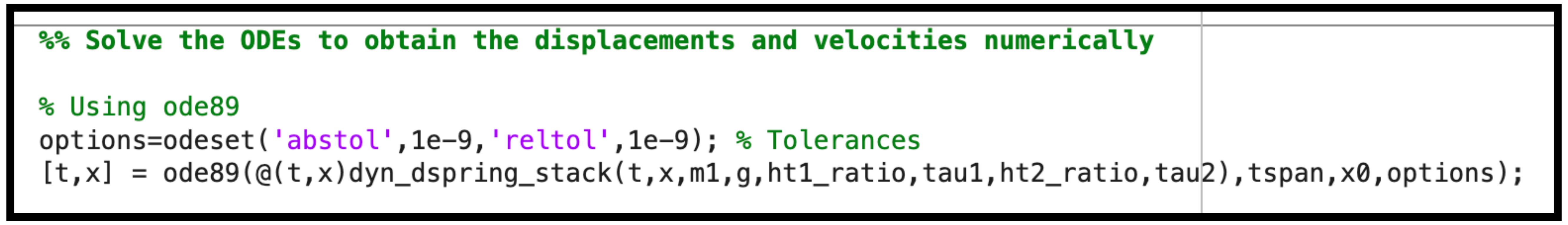
- Created MATLAB scripts for the dynamics of the 2-spring stack, for simulating the dynamics and for animating the motion of the mass
- Implemented two approaches for simulation -
  **1)** Virtual Mass method and **2)** Force equality method

# Code

## Parameters of the System

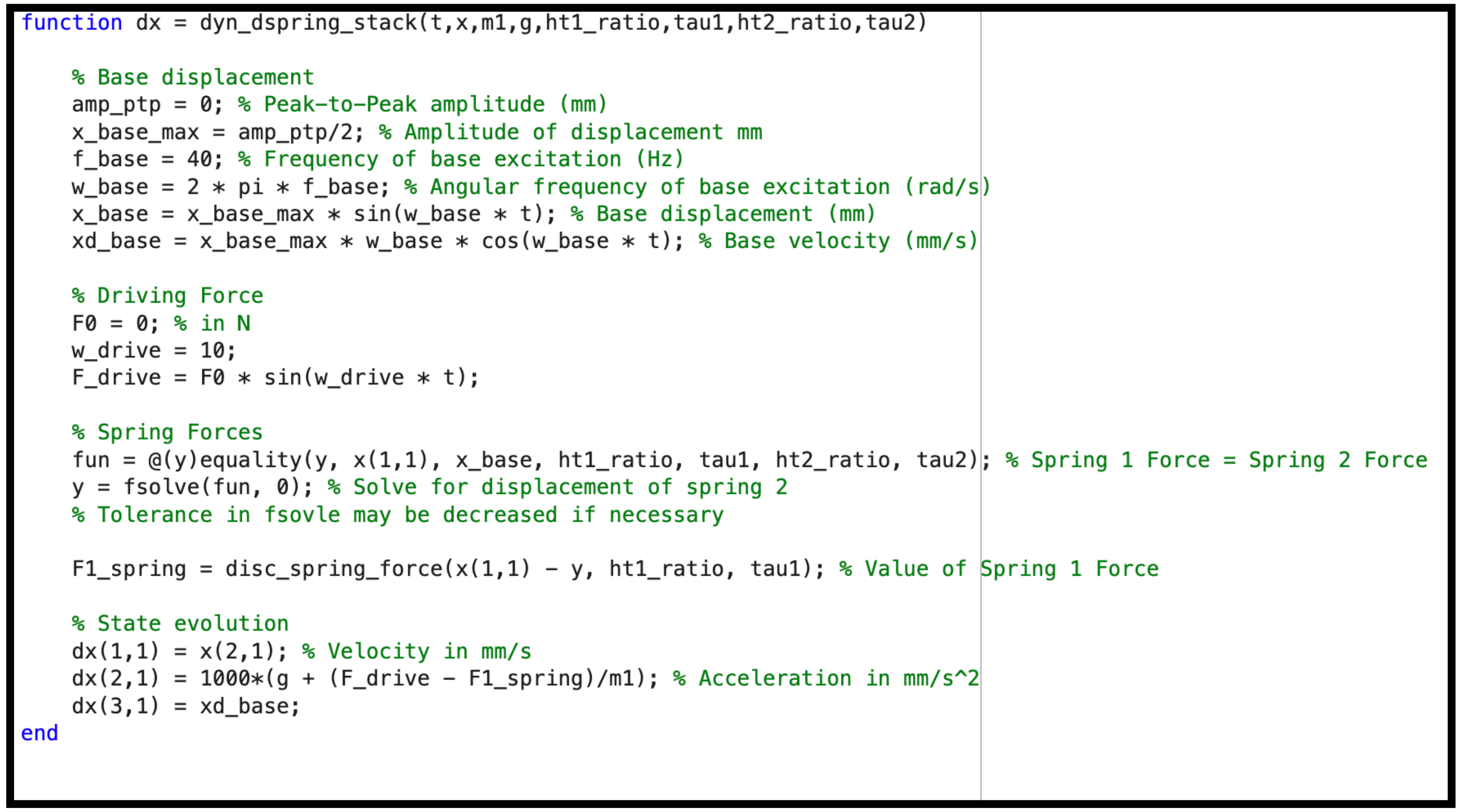
```matlab
%% Simulation of 2-spring stack - Mass system in MATLAB

% The code below is used to simulate the dynamics of a 2-spring stack
% consisting of disc-springs with a mass attached to it.
% The displacement and velocity profiles of the mass are plotted
% and an animation of the spring-mass motion is also shown

% Convention - Positive values of displacement and velocity are in the
% downward direction

%% System Parameters
m1 = 10.7; % Mass (kg)
g = 9.81; % Acceleration due to gravity (m/s^2)
ht1_ratio = sqrt(2); % Ratio of height to thickness of spring 1 (the top spring)
tau1 = 0.5; % Thickness of spring 1 (mm)
ht2_ratio = sqrt(2); % Ratio of height to thickness of spring 2 (the bottom spring)
tau2 = 0.5; % Thickness of spring 2 (mm)
h01 = ht1_ratio * tau1; % Height of spring 1 (mm)
h02 = ht2_ratio * tau2; % Height of spring 2 (mm)
l0 = 4*(h01 + h02); % Natural length of the spring + spacers (mm)
l1 = 4*h02 + h01; % Distance between the base and the top of spring 2 (mm)
```

# Code



```
%% Solve the ODEs to obtain the displacements and velocities numerically

% Using ode89
options=odeset('abstol',1e-9,'reltol',1e-9); % Tolerances
[t,x] = ode89(@(t,x)dyn_dspring_stack(t,x,m1,g,ht1_ratio,tau1,ht2_ratio,tau2),tspan,x0,options);
```

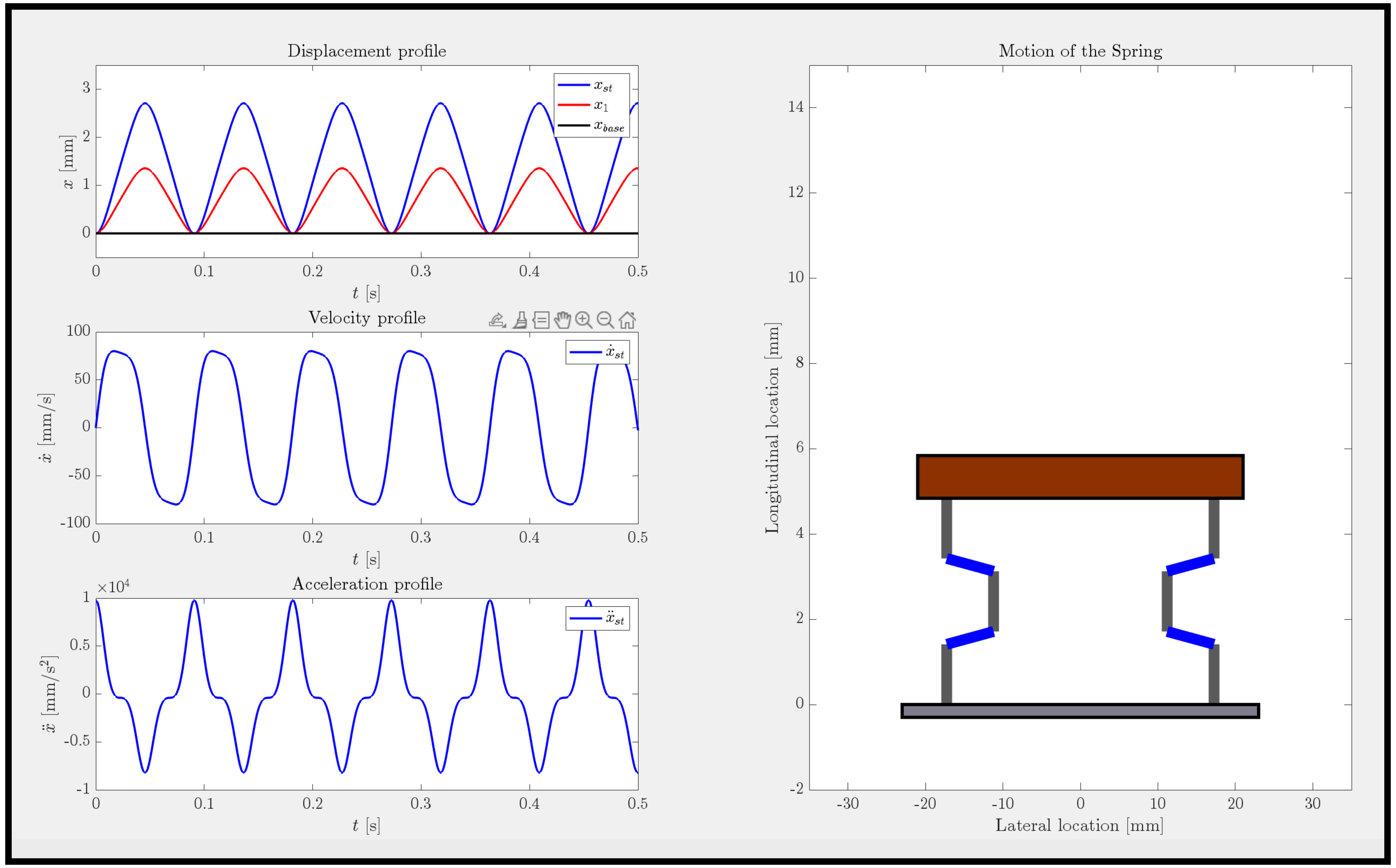
**Numerical ODE solving using ode45 / ode89**



```
function dx = dyn_dspring_stack(t,x,m1,g,ht1_ratio,tau1,ht2_ratio,tau2)

    % Base displacement
    amp_ptp = 0; % Peak-to-Peak amplitude (mm)
    x_base_max = amp_ptp/2; % Amplitude of displacement mm
    f_base = 40; % Frequency of base excitation (Hz)
    w_base = 2 * pi * f_base; % Angular frequency of base excitation (rad/s)
    x_base = x_base_max * sin(w_base * t); % Base displacement (mm)
    xd_base = x_base_max * w_base * cos(w_base * t); % Base velocity (mm/s)

    % Driving Force
    F0 = 0; % in N
    w_drive = 10;
    F_drive = F0 * sin(w_drive * t);

    % Spring Forces
    fun = @(y)equality(y, x(1,1), x_base, ht1_ratio, tau1, ht2_ratio, tau2); % Spring 1 Force = Spring 2 Force
    y = fsolve(fun, 0); % Solve for displacement of spring 2
    % Tolerance in fsovle may be decreased if necessary

    F1_spring = disc_spring_force(x(1,1) - y, ht1_ratio, tau1); % Value of Spring 1 Force

    % State evolution
    dx(1,1) = x(2,1); % Velocity in mm/s
    dx(2,1) = 1000*(g + (F_drive - F1_spring)/m1); % Acceleration in mm/s^2
    dx(3,1) = xd_base;
end
```

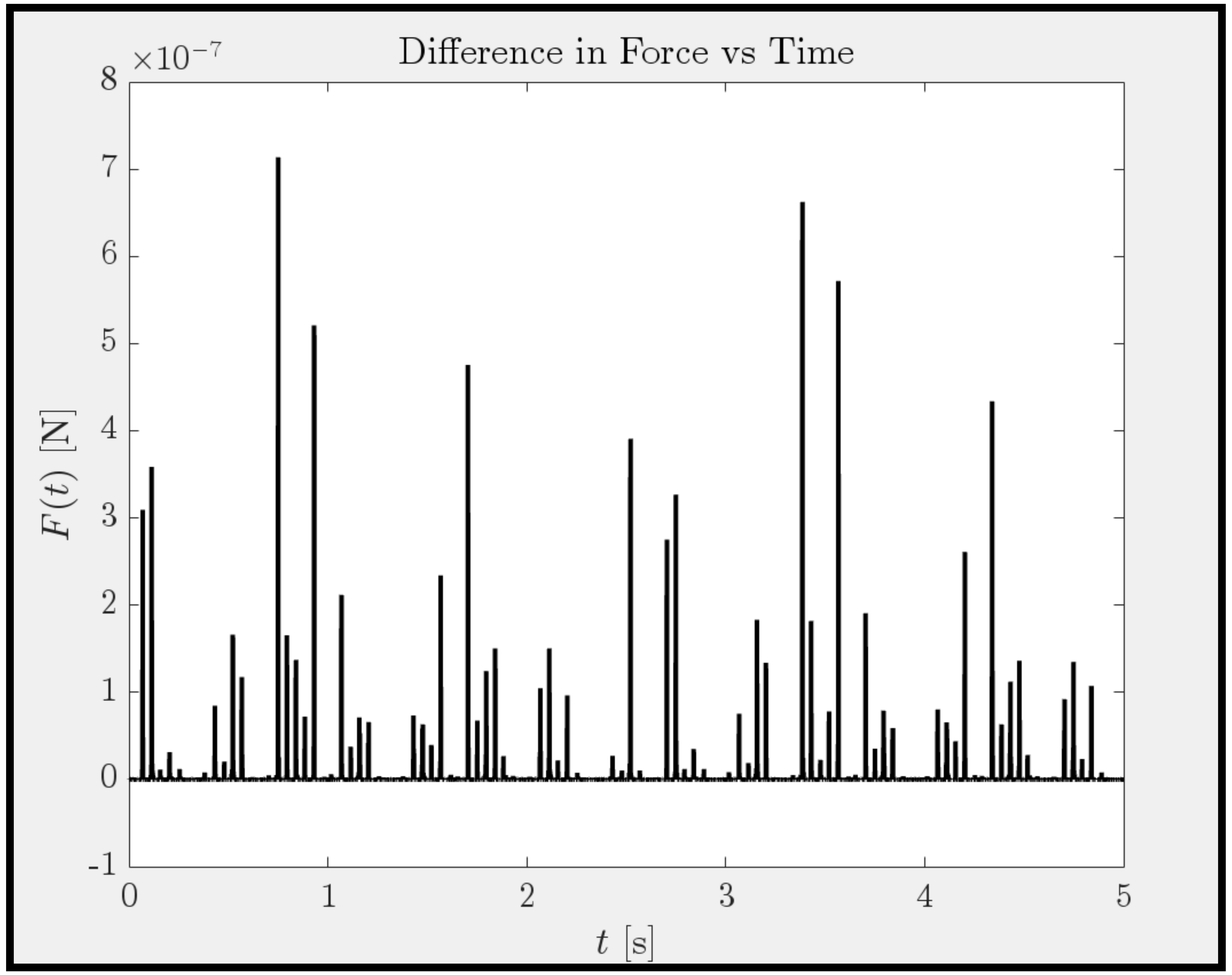
**Function with the ODEs representing the dynamics using NLM**

# Results



**Case 1 :** $\frac{h_1}{t} = \frac{h_2}{t} = 1.41$, **Initial Conditions -** $x(0) = 0, \dot{x}(0) = 0$

# Results



In the **Force equality method**, forces in the two springs are equal to within 1e-7 which is accurate enough

In the **Virtual mass method**, the accuracy increases the smaller the mass become but there is numerical instability and it is difficult to reach an accuracy of 1e-7

**Case 1 :** $\frac{h_1}{t} = \frac{h_2}{t} = 1.41$, **Initial Conditions -** $x(0) = 0, \dot{x}(0) = 0$

## Scope for Future work

- Solve for **static equilibrium state** of the 2-spring stack and then proceed
- Incorporate **base displacement** into the simulation as well