2D simulation of rotating clock hands
Project 1 for Modeling and Simulation in Science, Engineering and Economics

Ruishan Lin

Courant Institute of Mathematical Sciences - New York University

Abstract. Using the idea from `Dynamics of Structures: Networks of Springs and Dashpots with Point Masses at Nodes` [1], I built a model of two clock hands spinning around the clock center. By implementing different values of mass, initial velocity, dashpot constant, and spring constant, as well as the magnitude of the time step, I aim to unravel the effects of the physical variables on the spinning motion.

Keywords: spring · motion · mechanics.

1 Introduction

We know the basic structure of a clock consists of a fixed center and 2 or 3 spinning hands with fixed angular velocities and fixed lengths. Clocks in real life have very complicated energy structures and time-keeping devices to keep measures accurate. For this simulation, however, we focus only on the movements on the dial, which are the behaviors of the two hands when given initial velocity.

In this project, I treat the hour hand and the minute hand both as springs. Their initial lengths have a fixed ratio of 2:1 (2 being the minute hand and 1 being the hour hand). Then it's crucial to implement the correct initial velocities to the minute hand and the hour hand so that they could move like the real clock hands. Because the clock hands are springs, their length changed as they rotate. Trails are plotted to better understand their movements on the dial.

The central idea of this project lies upon the foundation of the law of motion for springs. The designs of this program is to function as such: when one implements an initial velocity onto the structure, the program goes on to figure out the motion in the rest of the cycles by itself.

2 Governing formulas and equations

2.1 Angular and tangential velocity

Since the model focuses on rotation, it's essential to emphasize the formula for angular velocity,

$$\omega = d\theta/dt \tag{1}$$

and tangential velocity,

$$V_t = \omega \cdot r \tag{2}$$

where r means radius. In the tests, we give the clock hands an initial velocity from the left to the right, if we start from 12 o'clock. The angular velocity of the hour hand is 1/60 of that of a minute hand. Meanwhile, the length of the hour hand is half of that of the minute hand. Usually, angular velocities are set to be positive in the counterclockwise direction. In this case, since both hands are traveling clockwise, I treat their angular velocities as positive, in calculating the tangential velocity as positive numbers.

2.2 Equations of motion

Newton's second law,

$$F = m \cdot a \tag{3}$$

$$a = dU/dt \tag{4}$$

is another crucial foundation of the simulation. Mass m and velocity U are represented by matrix in a system of nodes and links.

2.3 Hooke's law

The force needed to extend or compress a spring by some distance is linearly proportional to that distance. If k is the spring constant, s is the change in distance,we have the linear relation,

$$F_s = ks \tag{5}$$

2.4 Dashpot constant

We also simulate the ideal dashpot force onto each link. The force produced is linearly proportional to the force applied. μ is the dashpot constant or the dashpot impedance. The relationship between the applied force f and the velocity v is

$$f = \mu \cdot v \tag{6}$$

3 Mathematical modeling

3.1 Basic structure

To build a basic structure, I start with a 3 by 2 matrix X:

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 2 \end{pmatrix}$$

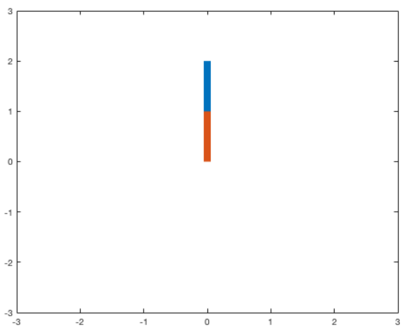


Fig. 1. basic structure 1

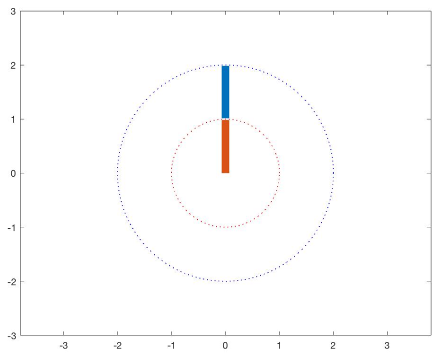


Fig. 2. basic structure 2

which represents the positional information of the 3 nodes, each row representing the coordinates of a node.

Now, we need to give orders to connect the nodes. Introduce the structure-building variables, jj: $\begin{pmatrix}1\\2\end{pmatrix}$ and kk: $\begin{pmatrix}2\\3\end{pmatrix}$. jj builds the link of the hour hand; kk builds the link of the minute hand. We also initiate variable FreeNodes: $\begin{pmatrix}1\\3\end{pmatrix}$ which indicates the two nodes that rotates have initial coordinates (0,1) and (0,2).

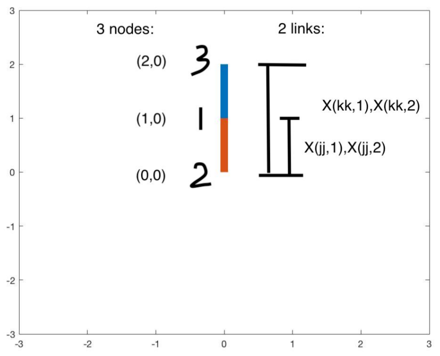


Fig. 3. nodes and links

3.2 initial values

On the structure

Matrix name	Size	Description
M	3 by 1	mass of each node
S	2 by 1	spring constant of each link
D	2 by 1	dashpot constant of each link
U	3 by 2	velocity of each node

Table 3.2.1

1. Mass of this model is added onto each of the 3 node;
2. Velocity matrix records the coordinates of each node. Similar to the positional matrix X;
3. All values for the for parameters are initialized and subject to changes to study their effects on the model.

Rotation

Variable	Unit	Description	Value
dt	minute	time step	1
Tclock	minutes	period	60
omega_hh	RPM	angular velocity of the hour hand	0.0087
omega_mh	RPM	angular velocity of the minute hand	0.1047
Len_hh	meters	length of the hour hand	1
Len_mh	meters	length of the minute hand	2

Table 3.2.2

1. RPM stands for "radians per minute";
2. The basic unit is minute, since the second hand is not present in our clock;
3. $omega\_hh = 2 \cdot pi/(12 \cdot Tclock)$; $omega\_mh = 2 \cdot pi/Tclock$.

initial velocity Our simulation starts at sharply 12 o'clock. Therefore, the clock hands are given horizontal initial velocities on node 1 and node 3. As (2) suggests, we calculate the x-component of the initial velocities

$$omega\ hh \cdot Len\ hh$$

for the hour hand and

$$omega\ mh \cdot Len\ mh$$

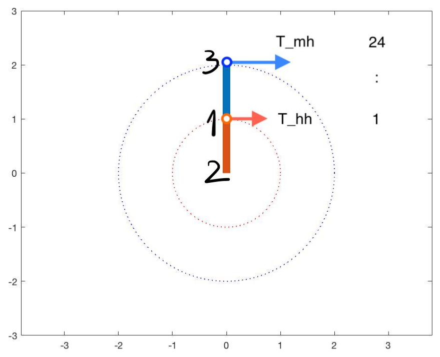


Fig. 4. initial velocity

for the minute hand. As shown in Table 3.2.2

$$omega\ mh = 12 \cdot omega\ hh$$

and

$$Len\ mh = 2 \cdot Len\ hh$$

Hence, we obtain

$$\frac{omega\ hh \cdot Len\ hh}{omega\ mh \cdot Len\ mh} = \frac{1}{12 \cdot 2} = \frac{1}{24}$$

3.3 Physical parameters

The motion on the springs [2] Governing by (3),(4),(5), we can deduce the formula by

$$M_k \frac{dU_k}{dt} = \sum_{j \in\ N(k)} T_{jk} \frac{X_j - X_k}{\|X_j - X_k\|} + F_k \tag{7}$$

$$\frac{dX_k}{dt} = U_k \tag{8}$$

$$T_{jk} = S_{jk}(\|X_j - X_k\| - R^0_{jk}) + D_{jk} \frac{X_j - X_k}{\|X_j - X_k\|} \cdot (U_j - U_k) \tag{9}$$

Note:

1. j,k are indexes of nodes; Since masses are added onto nodes, we can also treat j,k as masses.
2. (6) is a statement of balance; F_k is the general external force. the external force of the entire spring. T_{jk} is a scalar that represents the magnitude of the spring force. $\frac{X_j - X_k}{\|X_j - X_k\|}$ is a unit vector that determines the direction of the force.

3. (8) unravels the force on the string due to the motion, which consists of two components: (1) the force needed to create the change of length; (2) the dashpot force or the resisting force;
4. Lines 116-119 in **Section 5 Matlab Code** are the translation of (6),(7), and (8) into code.

We apply the idea of forward Euler method, which approximates the time derivative as such

$$\frac{dU}{dt}(\Delta t + t) \approx \frac{U(t + \Delta t - U(t)}{\Delta t} \tag{10}$$

Forward Euler's approximation enable us to loop in Matlab to calculate and update related variables at each time step.
Lines 140 to 143 in **Section 5. Matlab Code** use the idea of forward Euler method.

4 Results and discussions

Test 1 Our model successfully let the program figure out the rotation by itself. Given an initial velocity of the horizontal direction, the two clock hands made of spring have different motions when their initial physical parameters vary. The most basic case is when all values of the parameters are set to 1. Namely, we have M = $\begin{pmatrix}1\\1\\1\end{pmatrix}$, S = $\begin{pmatrix}1\\1\end{pmatrix}$, D = $\begin{pmatrix}1\\1\end{pmatrix}$, and dt = 1.

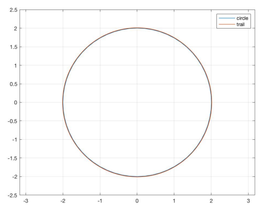


Fig. 5. minute hand

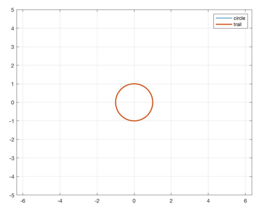


Fig. 6. clock hand

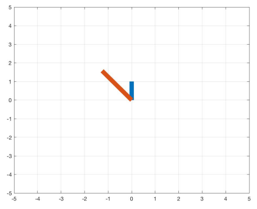


Fig. 7. ending position

Fig.5. and Fig.6.are the trails of the two hands are circular after running for

12 rounds. However, there's a delay in the minute hand, meaning that after 12 cycles, it did not return to its original position. I computed the angle between the two hands to be 33.18 °**(Fig.7)**.

4.1 Trails

To study how the physical parameters influence the motion, I conducted a series of tests on the trails of the rotations. Here are some interesting results:

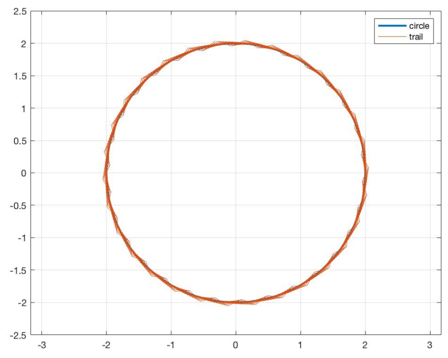


Fig. 8. Test 2.1

Test 2.1 Parameters: M = $\begin{pmatrix}1\\1\\0.8\end{pmatrix}$, S = $\begin{pmatrix}1.3\\1.2\end{pmatrix}$, D = $\begin{pmatrix}1\\1\end{pmatrix}$, and dt = 1.

Here, we are able to see the wobbles as the spring compresses and extends from its original length **(Fig.8.)**.

Test 2.2 Parameters: M = $\begin{pmatrix}1\\1\\1\end{pmatrix}$, S = $\begin{pmatrix}1\\1\end{pmatrix}$, D = $\begin{pmatrix}1.8\\1.8\end{pmatrix}$, and dt = 1.

I only modified dashpot constants, compared to Test1. **(Fig.9)**, the trail of the minute hand obtained from this test, shows that there were lots of vibration going on. The spreading trails from the clock center do not cover all the direction, suggesting that the course is not so continuous or circular. This does align with our expectations: when a spring extends and contracts while rotating, its trail should be in the radiant shape. After consulting with Prof. Peskin, who suggested that dt (time step) could be too large, I changed dt to 0.5 to plot movement one more time, and get drastically different result: we are again getting a circle!**(Fig.10)**

How do different parameters affect the trail In my simulation, dt plays a crutial role in the shape of the trails. For each set of M,D, and S, there's a certain

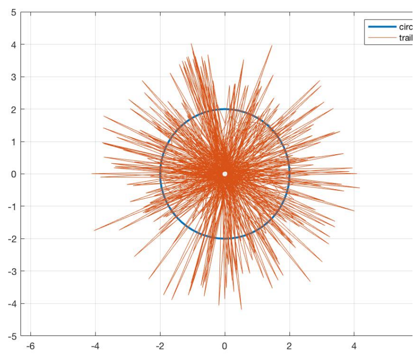


Fig. 9. Test 2.2

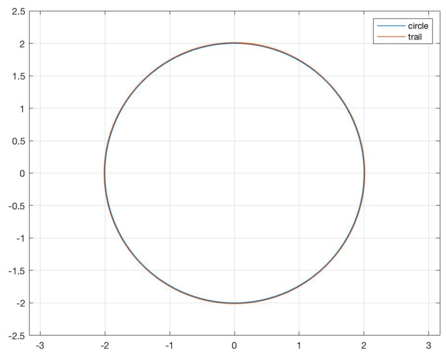


Fig. 10. Test 2.2(1)

threshold of dt which divides the shape into "circular looking" and "radiant". Due to the limited time and resources, I have not yet figured out whether there's a numerical relationship between dt and each of the parameter. The following are a comple qualitative observations and reflections:

1. When dt exceeds certain number, the coordinates stored in the matrix X would get too big that the springs blows out; We should always pick a small value for dt, ideally no more than 1;
2. When each node is assigned with a greater mass, their trail suggests that they stay extended during the entire rotation, as shown in **figure 11**;
3. Spring constant affects on the trail in such a way that it makes the trail "thicker", as shown in **figure 12**;
4. When spring constants are eliminated, dashpot constants within certain range would keep elongating the length of the spring, as shown in **figure 13**;
5. Dashpot constant's effects can be almost eliminated when dt is small enough.

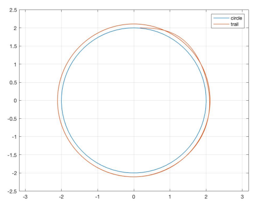


Fig. 11. observation 2

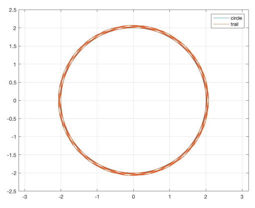


Fig. 12. observation 3

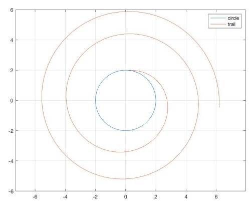
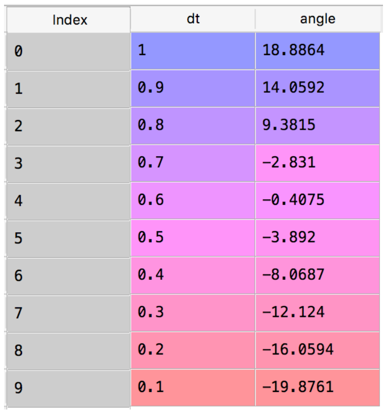


Fig. 13. observation 4

4.2 Loss of synchronization

What I mean by desynchronization is, as **Fig.7** shows, the angle between the two clock hands after they finish the 12 cycles of rotations. Why is the minute hand not able to catch up with the hour hand, which points towards their initial direction? I conducted a series of tests for a pair of clock hands with the following initial values: M $= \begin{pmatrix} 3 \\ 3 \\ 3 \end{pmatrix}$, S $= \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, D $= \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. I wonder what positions would the two clock hands end up with if I only adjusted dt from 0.1 to 1.



Index	dt	angle
0	1	18.8864
1	0.9	14.0592
2	0.8	9.3815
3	0.7	-2.831
4	0.6	-0.4075
5	0.5	-3.892
6	0.4	-8.0687
7	0.3	-12.124
8	0.2	-16.0594
9	0.1	-19.8761

Fig. 14. Table 4.2.2

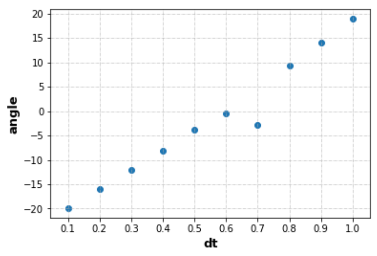
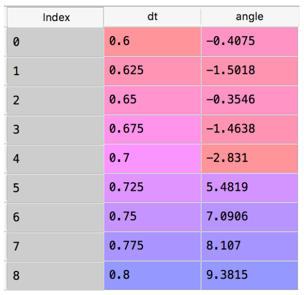


Fig. 15. Scatter plot 1



Index	dt	angle
0	0.6	-0.4075
1	0.625	-1.5018
2	0.65	-0.3546
3	0.675	-1.4638
4	0.7	-2.831
5	0.725	5.4819
6	0.75	7.0906
7	0.775	8.107
8	0.8	9.3815

Fig. 16. Table 4.2.3

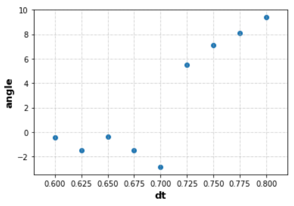


Fig. 17. Local scatter plot

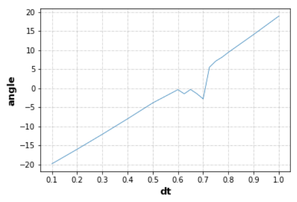


Fig. 18. The whole picture

When dt is between 0.1 and 0.6 and between 0.8 and 1.0, the angle and dt seem to be linearly correlated. However, when dt is between 0.6 and 0.8, the pattern breaks. So I conducted additional tests between the dt range of 0.6 and

0.8. There are some fluctuations there, especially between 0.65 and 0.725, as shown in **Fig. 17**

Summary From **Fig.18**, we see that every different value of dt can result in a different ending position of the clock hands. Due to the numerical and computing nature of Matlab, we are able to see the impacts more directly from the trails or the animation. It is still unclear What causes the fluctuation between 0.6 to 0.8, but it might be the effects of dashpot impedance. Further physical knowledge is needed to unravel such myth.

5 Matlab Code

```
close all; clear all; clc;
% building a string of springs
NDIM = 2;%dimention 2D
%set up the basic structure
num_nodes = 3;
num_links = 2;

%The arrays that link the nodes
jj = zeros(num_links,1);
kk = zeros(num_links,1);

% set numerical parameters
T_clock = 60;% meaning 60 minutes
dt = 0.9;

% set physical parameters
omega = 2*pi/T_clock;
M = zeros(num_nodes,1); % mass of nodes
S = zeros(num_links,1); % link spring constant
D = zeros(num_links,1); % link dashpot constant
%Masses and the spring constant
M(1) = 1;
M(2) = 1;
M(3) = 1;
S(1) = 0.3;
S(2) = 0.5;
D(1) = 1;
D(2) = 1;

% set positions of nodes
X = zeros(num_nodes, NDIM);
X(1,1) = 0; % node 1
```

```
X(1,2) = 1;
X(2,1) = 0; % node 2
X(2,2) = 0;
X(3,1) = 0; % node 3
X(3,2) = 2;

% build structure by creating links
jj(1) = 1; % link 1
kk(1) = 2;
jj(2) = 2; % link 2
kk(2) = 3;

%plot the initial structure
figure(1);
x = X(jj,1);
y = X(jj,2);
m = X(kk,1);
n = X(kk,2);
plot(m',n','linewidth',8)
xlim([-3 3])
ylim([-3 3])
axis equal
hold on
plot(x',y','linewidth',8)
hold on
viscircles([0,0],1,'color','red','LineStyle',':','
    LineWidth',0.8)
hold on
viscircles([0,0],2,'color','blue','LineStyle',':','
    LineWidth',0.8)

% Build a mechanic clock
center = [0 0]; % center of the clock
% specify free nodes with prescribed displacement
free_nodes = [1,3];
% initialize rest length
DX = X(jj,:) - X(kk,:);
Rzero = sqrt(sum(DX.^2,2));

Len_mh=Rzero(2);%length of the minute hand
Len_hh=Rzero(1);%length of the hour hand

for t=0:dt:T_clock

```

```
    theta = pi/2-t*omega; %the angle by which the minute hand spins
    gamma=pi/2-t*omega/12; %the angle by which the hour hand spins
    X(free_nodes(2),:) = center+Len_mh*[cos(theta),sin(theta)];
    X(free_nodes(1),:) = center+Len_hh*[cos(gamma),sin(gamma)];
    figure(2);
    m1 = [X(kk,1),X(jj,1)];
    n1 = [X(kk,2),X(jj,2)];
    plot(m1',n1','linewidth',8)
    grid on
    xlim([-5 5])
    ylim([-5 5])
    pause(0.01)
    %frame = getframe(gcf); %get frame
    %writeVideo(myVideo, frame);

end

% initialize velocities to zero
U = zeros(num_nodes, NDIM);


%reset the positions
X(1,1) = 0; % node 1
X(1,2) = 1;
X(2,1) = 0; % node 2
X(2,2) = 0;
X(3,1) = 0; % node 3
X(3,2) = 2;
X_copy=X;
omega_hh=2*pi/(12*T_clock);%angular velocity of the hour hand
omega_mh=2*pi/T_clock;%angular velocity of the minute hand

U(free_nodes(1),:)=[omega_hh*Len_hh 0];%initial velocity for the hour hand
U(free_nodes(2),:)=[omega_mh*Len_mh 0];%initial velocity for the minute hand
x_hh=[]; %store hour hand position updated by the program
x_mh=[]; %store minute hand position updated by the program
```

```
x_mech_hh=[]; %store hour hand position controlled by
    human
x_mech_mh=[]; %store minute hand position controlled by
    human

for t=0:dt:12*T_clock

    DX = X(jj,:) - X(kk,:);
    DU = U(jj,:) - U(kk,:);
    R = sqrt(sum(DX.^2,2));
    T = S.*(R - Rzero) + (D./R).*sum(DX.*DU,2);
    TR = T./R;
    FF = [TR TR].*DX;

    F = zeros(num_nodes, NDIM);
    % compute forces on the nodes
    for l = 1:num_links
        F(kk(l),:) = F(kk(l),:) + FF(l,:);
        F(jj(l),:) = F(jj(l),:) - FF(l,:);
    end

    figure(3);
    x = [X(jj,1) X(kk,1)];
    y = [X(jj,2) X(kk,2)];
    plot(x',y','linewidth',8)
    grid on
    xlim([-5 5])
    ylim([-5 5])
    pause(0.01)

    % update the free nodes
    U(free_nodes(1),:) = U(free_nodes(1),:) + dt*F(
        free_nodes(1),:)./[M(free_nodes(1)) M(free_nodes
        (1))];
    X(free_nodes(1),:) = X(free_nodes(1),:) + dt*U(
        free_nodes(1),:);
    U(free_nodes(2),:) = U(free_nodes(2),:) + dt*F(
        free_nodes(2),:)./[M(free_nodes(2)) M(free_nodes
        (2))];
    X(free_nodes(2),:) = X(free_nodes(2),:) + dt*U(
        free_nodes(2),:);

    x_hh = [x_hh;X(1,:)]; %list to store all the
        positional values of the hour hand
    x_mh = [x_mh;X(3,:)]; %list to store all the
        positional values of the minute hand

    theta1 = pi/2-t*omega;
    gamma1=pi/2-t*omega/12;
    X_copy(free_nodes(2),:) = center+Len_mh*[cos(theta1),
        sin(theta1)];
    X_copy(free_nodes(1),:) = center+Len_hh*[cos(gamma1),
        sin(gamma1)];
    x_mech_hh=[x_mech_hh;X_copy(1,:)];
    x_mech_mh=[x_mech_mh;X_copy(3,:)];
end

figure(4)
plot(x_mech_hh(:,1),x_mech_hh(:,2),'linewidth',0.8);
hold on
plot(x_hh(:,1),x_hh(:,2),'linewidth',0.6);
grid on
xlim([-5 5])
ylim([-5 5])
legend('circle','trail')
axis equal

figure(5)
plot(x_mech_mh(:,1),x_mech_mh(:,2),'linewidth',0.4);
hold on
plot(x_mh(:,1),x_mh(:,2),'linewidth',0.4);
grid on
xlim([-6 6])
ylim([-6 6])
legend('circle','trail')
axis equal
% calculate the theta between two hands
%current length
Rnow = sqrt(sum(DX.^2,2))
%a dot b/|a||b| = cos theta
% acosd returns the angle in degrees
angle = acosd (sum(X(1,:).*X(3,:))/(Rnow(1)*Rnow(2)))
```

6 Ackowledgement

This is my first project with Matlab and I'm grateful for the help and guidance from Prof. Charlie Peskin and Guanhua Sun.
My inspiration for this simulation came from the code spring_string_2d.m contributed by Charles Puelz [3].

The modeling workshop on `Latex` held by Prof. Mutiara Sondjaja and Ziye Ye has also helped me in writing this paper.

References

[1] Charlie Peskin. *Dynamics of Structures: Networks of Springs and Dashpots with Point Masses at Nodes*. URL: `https://www.math.nyu.edu/faculty/peskin/modsim_lecture_notes/structural_mechanics.pdf`.

[2] Charlie Peskin and Charles Puelz. *Mathematical modeling in science, engineering, and economics*. URL: `https://modelingsimulation.github.io/undergraduateClass/notes_MATH395.pdf`.

[3] Charles Puelz. *spring_string_2D.m/undergraduateClass/ModelingSimulation*. URL: `https://github.com/ModelingSimulation/undergraduateClass/blob/master/spring_string_2d.m`.