Term Project: Evolving Soft Robots (Phase C)

Submitted by: Yaoyao Xu (UNI: yx2725)
Ruoyao Zhao (UNI: rz2535)

Course number: MECSE4510
Instructor: Hod Lipson

Submitted time: 11/16/2021
Grace Hour Used: 1 h
Grace Hour Left: 98 h

Results

The Fastest Robot

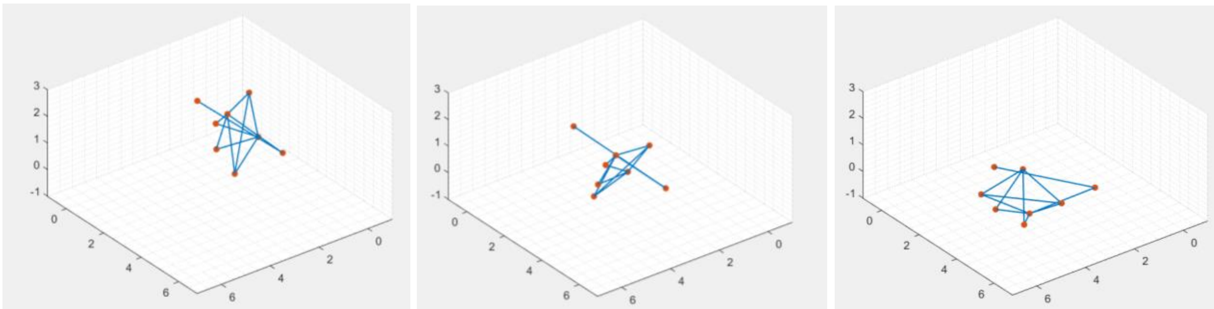


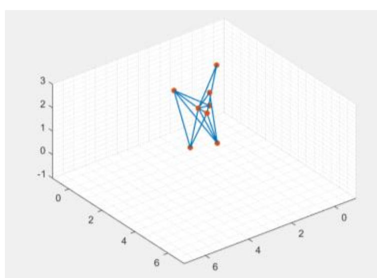
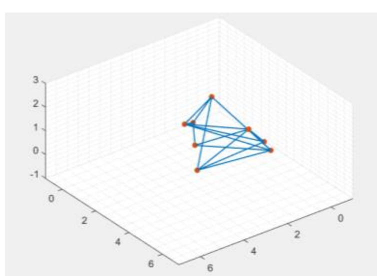
Figure 1 The Fastest Robot in Three Frames of its Motion

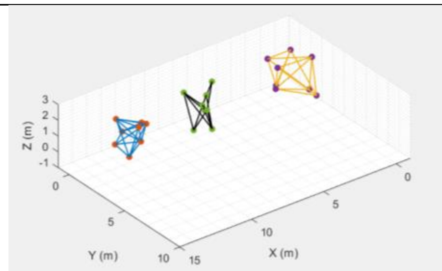
Video Link: https://youtu.be/D6y6CjSrBHU

Table 1 Speed of the Fastest Robot

Maximum Speed	1.49313 m/s
Evaluations	2500
Time Step	1e-03

Other Robots

Other Robots	Figure	Video Link
Walking Robot		https://youtu.be/spOXDt5zhMo
Crawling Robot		https://youtu.be/cxCp2TEV4bA
Multi-Robot		https://youtu.be/r3qTwtTjKOE

Robot Zoo

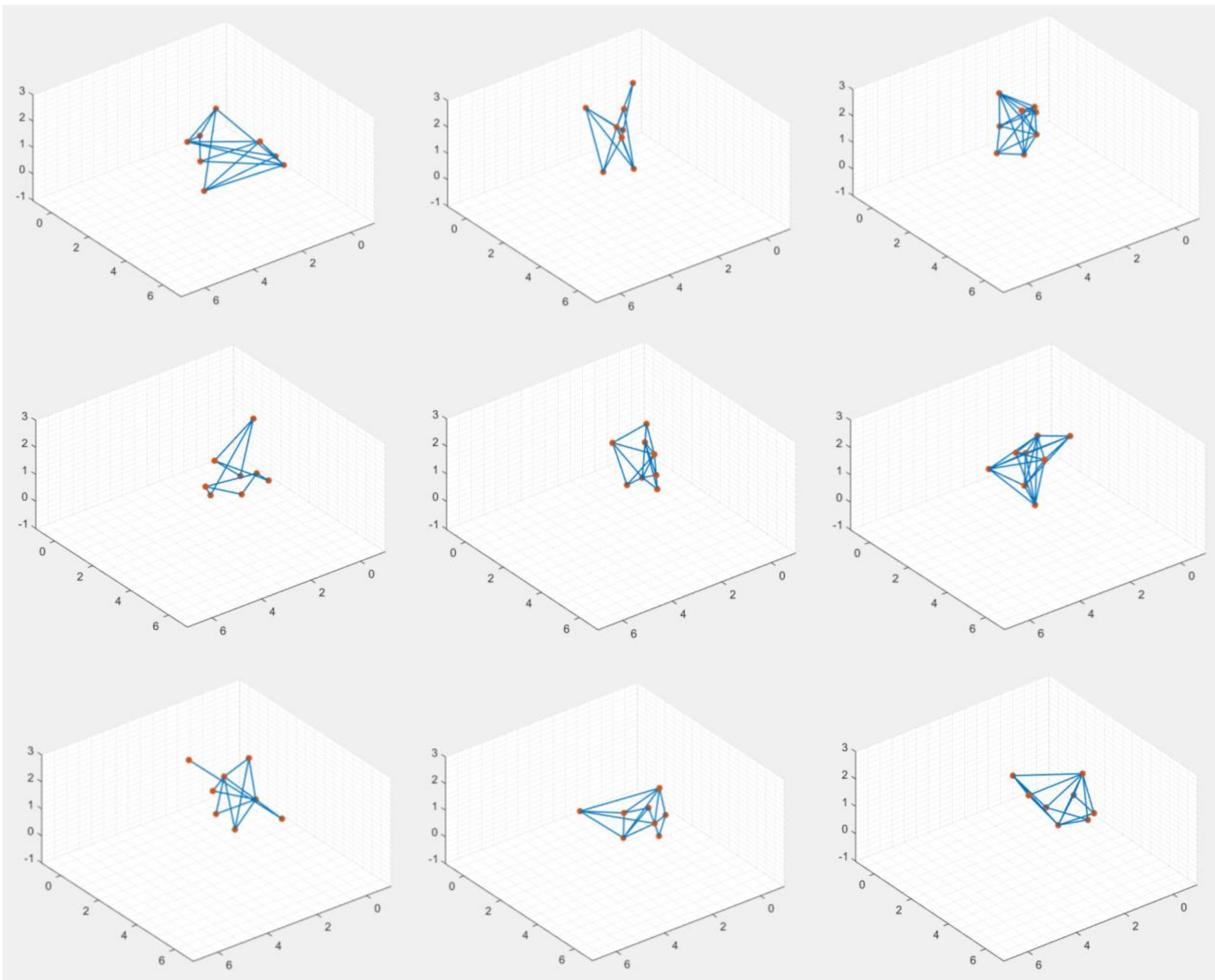


Figure 2 Robot Zoo with 9 Robots

The link to our final presentation:
https://docs.google.com/presentation/d/101UdNmg2QQK7p05Dgr-TPGEJRatbVaUS/edit?usp=sharing&ouid=114120210428284577005&rtpof=true&sd=true

Methods

The purpose of this project is to evolve robots to travel longer distance in a physical simulator, which contains moving and bouncing motion on a flat ground with fraction. We used GA to evolve the robot, the environmental conditions are sent to be acceleration of gravity (g) = [0, 0, -9.81] m/s^2, time step (dt) = 1e-3 with a spring evaluation of 1000 per second, mass of total mass point = 0.8 kg, stiffness of ground = 50 N/m, the friction coefficient of ground = 0.5, and the damping coefficient = 0.29. We programmed the GA in Matlab.

Parameters

The original robot will have 8 masses with its total mass to be 0.8kg. The original positions of these 8 masses are chosen randomly from a 6 x 6 x 6 cube coordinates. The springs link these masses randomly. A sine function is applied to each spring to make it be compressed or extended. The expression of this sine function is $L = a + b(\omega t + c)$, where a is the original rest length of spring, and L is the length after changing. Direct encoding is chosen for programming. There are 6 variables set up at the beginning, connectivity and number of springs, b (from -0.5 to 0.5), c (from 0 to 2 pi), the spring constant k (from 20 to 50), the masses distribution, and the positions of the masses. A matrix that contains different connectivity, b, c, k, mass distribution, and masses position values for every spring will be initialized.

The motion of each mass is calculated separately, and the forces and motion of the spring from the sine function is also divided into X, Y, and Z axes and recombined around the 8 masses. When the mass touched the ground, all the masses are considered separately, and the ground is regarded as a spring with its k to be 50 N/m, and friction coefficient to be 0.5. the damping force is set to $F_c = damp \times V^2$, where V is the velocity.

When evolving the motion of cube, the connectivity of springs, the b, c from sine function, the spring coefficient, the mass distribution, and the masses positions are treated as the chromosome in GA. Since there are 8 masses, the maximum number of springs is set to be 28. For the connectivity and positions of springs, if a spring is going to be eliminated, its b, c, and k are multiplied by 0; if a spring is going to kept, its b, c, and k remain unchanged. The mass distribution is randomly separated into 8 masses, and the total mass is always 0.8 kg. For the positions of masses, masses are randomly chosen from a 6 x 6 x 6 cube coordinate contains 216 points. The population size is set to be 50, the number of evaluations is assigned to be 2500, we run the GA three times. The selection method here is the tournament, in which 7 members are selected from the population, and the best two of them are used to be the parents.

The fitness function will be the distance the robot traveled. A fixed number of chromosomes would be selected and put into crossover and mutation. The crossover will randomly exchange points in b, c, k, mass distribution, and mass positions of two parents to create a child. Because the b, c, and k of unconnected spring are 0, connectivity and the number of springs is crossed when crossing b, c, and k.

The mutation will randomly choose two points in the gene and change them. The data of the fastest robot will be stored. The program will finally find 3 sets of "best values for spring connectivity, b, c, and k mass distributions and mass positions", We take the average of them to get the best-representing data.

Conclusion

As figure 1 and 2 show, the averaged longest distance the robot can travel in 3 seconds is 4.48m. As figure 3 shows, the curve does not converge because the number of populations, iterations, and total runs is too small. However, due to the limit of MATLAB performance and the personal computer hardware environment, it took more than 4 and a half hours for the program to finish all the calculations. Besides, when testing the motion of the cube, time step 'dt' and range of the spring coefficient 'k' will make a big difference to the speed. A large spring coefficient and a long-time step will cause a large error in the calculation. In order to reduce the error, we set dt to be 0.001s, and k of ground and spring are set to be around 50 N/m, which makes the 100 times smaller. In addition, damping is used to reduce the energy generated by the error, which makes the simulation system more stable.

Performance Plots

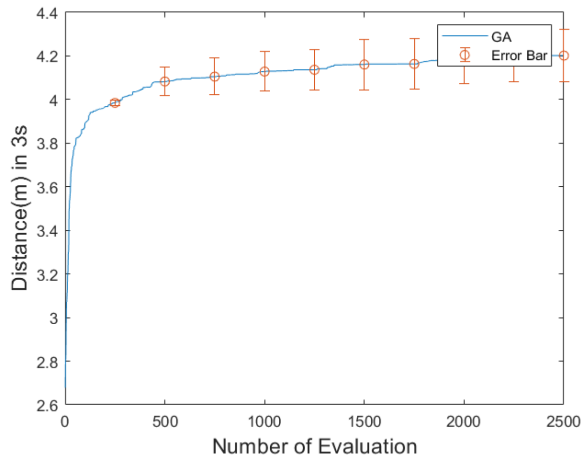


Figure 3 Learning Curve of the Fastest Robot

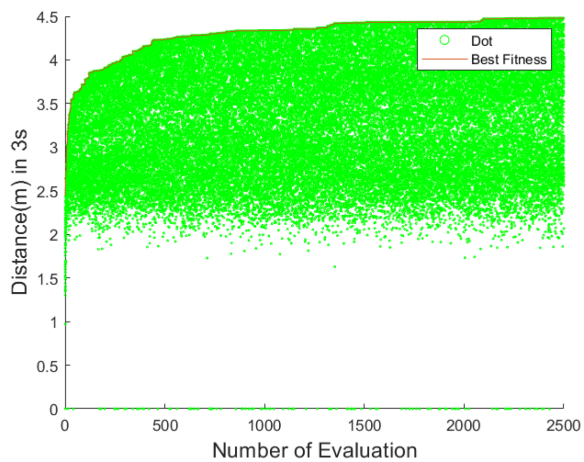


Figure 4 Dot Plot of the Fastest Robot

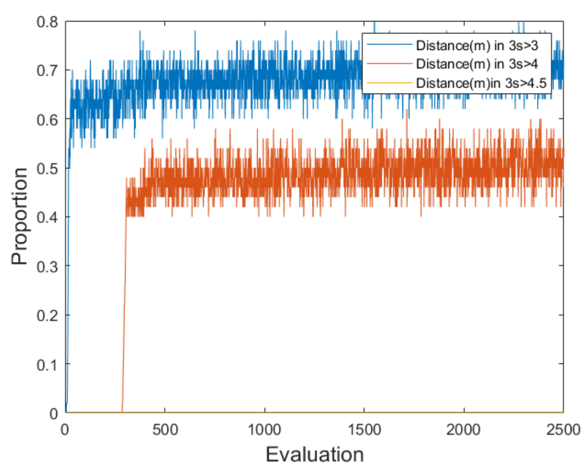


Figure 5 Convergence Plot of the Fastest Robot

Appendix

GA

```
%% Initialization:
% Start iterating time
clearvars
close all
clc
n = 28;
w = pi*10;
%% Genetic Algorithm
run = 3; % running times
iter = 2500; % evaluate times
popnum = 50; % number of population
path_new = NaN(iter, n+1);
kept = NaN(iter, 1);
tdistance = NaN(popnum, 1);
Tdistance = NaN(popnum,iter);
cood=6;
alldata = NaN(iter,run);
bckmv=zeros(popnum,(28*3+8+24));
for j = 1:run
    % Generate initial populations randomly:
pop = NaN(popnum, n);
for i=1:popnum
 springnum=randi([0, 1], [28, 1]);% decide number and positon of springs
 b = springnum .*(rand(28,1)*(0.5)-0.25); % range of b
 c = springnum .*rand(28,1)*2*pi; %range of c
 k =springnum .*( rand(28,1)*(20)+30);% range of k

 mm=rand(8,1);% decide mass distribution
 m=mm*0.8/sum(mm);
 vn=randperm(cood^3,8);
 v=locationsix(vn,cood)';

 bckmv(i,:) = [b;c;k;m;v]; % create gene
end
for N = 1:iter
 tic
 for i = 1:popnum
 tdistance(i) = Motion(w,bckmv(i,:));
 end
 tdistance(isnan(tdistance))=0;
 Tdistance(:,N) = tdistance; % Store all the distances
 if N == 800 || N == 1200 || N == 2500
 filename = strcat('data_', num2str(j), '_', num2str(N), '.mat');
 save(filename);
 end
 [path_best,idx_best] = max(tdistance); % Find the longest path and its position in matrix
 fit = tdistance; % value of fitness

 % store the best value:
 kept(N) = path_best;
 bckmv2=zeros(15,(28*3+8+24));
 bckmv3=zeros(15,(28*3+8+24));


 for i = 1:15
 bckmv1 = Rank(bckmv,fit); % Tournament Selection
 bckmv2(i,:) = Cross(bckmv1); % Crossover

 nm= randperm(popnum,1);
 bckmv3(i,:) = Mutate(bckmv(nm,:)); % Mutation

 end

 [~,idx] = sort(tdistance,'descend');

 bckmv = [bckmv(idx(1:20),:);bckmv2;bckmv3]; % Get good gene from parents
```

```
toc
end
alldata(:,j) = kept;
end
```

Motion function

```
function [tdistance] = Motion(w,bckmv)
%% debug
% clc
% clear
% w=100;
%
% m=ones(1,8)*0.1;
% b = zeros(1,28);
% c = zeros(1,28);
% k =1000*ones(1,28);
% L0=0.1;
% v=[0 0 L0 L0 0 0 L0 L0 0 L0 L0 0 0 L0 L0 0 0 0 0 0 L0 L0 L0 L0 ];
% bckmv=[b c k m v];
%% put gene in to data
b = bckmv(1:28);
c = bckmv(29:56);
k = bckmv(57:84);
m = bckmv(85:92)';
v1= bckmv(93:100)';
v2= bckmv(101:108)';
v3= bckmv(109:116)';
%% Define global variables:
global g
 g = -9.81; % in m/s^2 gravity
global dt
 dt = .001;% step time
global T; % Time
 T = 0;

kg = 50 ; % gorud coefficient
u = 0.5;% groung fridction coeficient
damp=0.29;% damping coeficient
Vo = [v1,v2,v3]; % initial positon of robot
V = [Vo(:,1),Vo(:,2),Vo(:,3)];
[l]=lengthorigin(Vo); % origin length of all spring
% set start condition
posx = V(:,1);
posy = V(:,2);
posz = V(:,3);
vx = zeros(length(posx),1);%
vy = zeros(length(posx),1);
vz = zeros(length(posx),1);
agx = zeros(length(posx),1);%
agy = zeros(length(posx),1);
agz = zeros(length(posx),1);
%
n = 0;
while n<= 3/dt
 n = n+1;
 l0 = -b.*sin(w*T+c) + b.*sin(w*(T+dt)+c);%% calulate motion from sine wave
 V = [posx,posy,posz];
 [Fspring,mo] = spring(l,V,k,l0); % divide spring force and sine motion in to masses
 mox = mo(:,1);
 moy = mo(:,2);
 moz = mo(:,3);
 % calcualte motion
 mtx=vx*dt + mox;
 mty=vy*dt + moy;
 mtz=vz*dt + moz;
 % calculate total motion
 posx = posx + mtx; %%
 posy = posy + mty;
 posz = posz + mtz;
 for i=1: length(posx) %when masses touch ground
 if posz(i) <= 0
```

```
 agz(i)=-kg*posz(i)/m(i);
 if mtx(i)^2+mty(i)^2 ==0% if masses have friciton
 agx(i)=0;
 agy(i)=0;
 else
 agx(i)=kg*posz(i)/m(i)*u*(mtx(i)/sqrt(mtx(i)^2+mty(i)^2));
 agy(i)=kg*posz(i)/m(i)*u*(mty(i)/sqrt(mtx(i)^2+mty(i)^2));
 end
 else
 agz(i)=0;
 agx(i)=0;
 agy(i)=0;
 end
 end
 Fspringx = Fspring(:,1);
 Fspringy = Fspring(:,2);
 Fspringz = Fspring(:,3);


 aspringx = Fspringx./m;
 aspringy = Fspringy./m;
 aspringz = Fspringz./m;


 % convert damping force to velocity change
 vdx=vx*damp.*abs(vx)./m*dt;
 vdy=vy*damp.*abs(vy)./m*dt;
 vdz=vz*damp.*abs(vz)./m*dt;

 % get the new velocity
 vx = vx + aspringx*dt+agx*dt-vdx;
 vy = vy + aspringy*dt+agy*dt-vdy;
 vz = vz + aspringz*dt+g*dt+agz*dt-vdz;%%
%
% Distancex(:,n) = posx;
% Distancey(:,n) = posy;
% Distancez(:,n) = posz;
%
 T = T+dt;
end
tdistance = sqrt((mean(posx))^2 + (mean(posy))^2); % calculae motion
end
```

Spring Function

```
function [Fspring,moveL0] = spring(l,v,k,L0)
% l is orginal length of cube, v is current cube point position, k is spring constant
% to increase speed, loop is not used
% the code is changed from phase b
%% debug
% clc
% clear
% l1 = ones(12,1);
% l2 = ones(12,1)* 2^0.5;
% l3 = ones(12,1)* 3^0.5;
% l=[l1;l2;l3];
%
%
% ll= 0.0009;
% k=ones(1,28);
% v=[ 0 0 0;
% 0 ll 0;
% ll ll 0;
% ll 0 0;
% 0 0 ll*1.9898;
% 0 ll ll*1.9898;
% ll ll ll*1.9898;
% ll 0 ll*1.9898;];
% load('data.mat')
% v=[posx, posy posz];
%
```

```
%
% L0=ones(1,28)*0.1;
%%


%Calculate the force from spring and the motion from L0
%Calulate the length
 lx1=(sum((v(1,:)-v(4,:)).^2))^0.5;
 x14=(v(1,1)-v(4,1));
 y14=(v(1,2)-v(4,2));
 z14=(v(1,3)-v(4,3));
 lx2=(sum((v(2,:)-v(3,:)).^2))^0.5;
 x23=(v(2,1)-v(3,1));
 y23=(v(2,2)-v(3,2));
 z23=(v(2,3)-v(3,3));
 lx3=(sum((v(5,:)-v(8,:)).^2))^0.5;
 x58=(v(5,1)-v(8,1));
 y58=(v(5,2)-v(8,2));
 z58=(v(5,3)-v(8,3));
 lx4=(sum((v(6,:)-v(7,:)).^2))^0.5;
 x67=(v(6,1)-v(7,1));
 y67=(v(6,2)-v(7,2));
 z67=(v(6,3)-v(7,3));
 ly1=(sum((v(1,:)-v(2,:)).^2))^0.5;
 x12=(v(1,1)-v(2,1));
 y12=(v(1,2)-v(2,2));
 z12=(v(1,3)-v(2,3));
 ly2=(sum((v(3,:)-v(4,:)).^2))^0.5;
 x34=(v(3,1)-v(4,1));
 y34=(v(3,2)-v(4,2));
 z34=(v(3,3)-v(4,3));
 ly3=(sum((v(5,:)-v(6,:)).^2))^0.5;
 x56=(v(5,1)-v(6,1));
 y56=(v(5,2)-v(6,2));
 z56=(v(5,3)-v(6,3));
 ly4=(sum((v(7,:)-v(8,:)).^2))^0.5;
 x78=(v(7,1)-v(8,1));
 y78=(v(7,2)-v(8,2));
 z78=(v(7,3)-v(8,3));
 lz1=(sum((v(1,:)-v(5,:)).^2))^0.5;
 x15=(v(1,1)-v(5,1));
 y15=(v(1,2)-v(5,2));
 z15=(v(1,3)-v(5,3));
 lz2=(sum((v(2,:)-v(6,:)).^2))^0.5;
 x26=(v(2,1)-v(6,1));
 y26=(v(2,2)-v(6,2));
 z26=(v(2,3)-v(6,3));
 lz3=(sum((v(3,:)-v(7,:)).^2))^0.5;
 x37=(v(3,1)-v(7,1));
 y37=(v(3,2)-v(7,2));
 z37=(v(3,3)-v(7,3));
 lz4=(sum((v(4,:)-v(8,:)).^2))^0.5;
 x48=(v(4,1)-v(8,1));
 y48=(v(4,2)-v(8,2));
 z48=(v(4,3)-v(8,3));
 %calculate force and motion in xyz direction
 fx14=(l(1)-lx1)*k(1);
 fx14x=fx14/lx1*x14;
 fx14y=fx14/lx1*y14;
 fx14z=fx14/lx1*z14;
 mx14x=L0(1)/lx1*x14;
 mx14y=L0(1)/lx1*y14;
 mx14z=L0(1)/lx1*z14;

 fx23=(l(2)-lx2)*k(2);
 fx23x=fx23/lx2*x23;
 fx23y=fx23/lx2*y23;
 fx23z=fx23/lx2*z23;
 mx23x=L0(2)/lx2*x23;
 mx23y=L0(2)/lx2*y23;
 mx23z=L0(2)/lx2*z23;
```

```
fx58=(l(3)-lx3)*k(3);
fx58x=fx58/lx3*x58;
fx58y=fx58/lx3*y58;
fx58z=fx58/lx3*z58;
mx58x=L0(3)/lx3*x58;
mx58y=L0(3)/lx3*y58;
mx58z=L0(3)/lx3*z58;


fx67=(l(4)-lx4)*k(4);
fx67x=fx67/lx4*x67;
fx67y=fx67/lx4*y67;
fx67z=fx67/lx4*z67;
mx67x=L0(4)/lx4*x67;
mx67y=L0(4)/lx4*y67;
mx67z=L0(4)/lx4*z67;
fy12=(l(5)-ly1)*k(5);
fy12x=fy12/ly1*x12;
fy12y=fy12/ly1*y12;
fy12z=fy12/ly1*z12;
my12x=L0(5)/ly1*x12;
my12y=L0(5)/ly1*y12;
my12z=L0(5)/ly1*z12;

fy34=(l(6)-ly2)*k(6);
fy34x=fy34/ly2*x34;
fy34y=fy34/ly2*y34;
fy34z=fy34/ly2*z34;
my34x=L0(6)/ly2*x34;
my34y=L0(6)/ly2*y34;
my34z=L0(6)/ly2*z34;

fy56=(l(7)-ly3)*k(7);
fy56x=fy56/ly3*x56;
fy56y=fy56/ly3*y56;
fy56z=fy56/ly3*z56;
my56x=L0(7)/ly3*x56;
my56y=L0(7)/ly3*y56;
my56z=L0(7)/ly3*z56;

fy78=(l(8)-ly4)*k(8);
fy78x=fy78/ly4*x78;
fy78y=fy78/ly4*y78;
fy78z=fy78/ly4*z78;
my78x=L0(8)/ly4*x78;
my78y=L0(8)/ly4*y78;
my78z=L0(8)/ly4*z78;

fz15=(l(9)-lz1)*k(9);
fz15x=fz15/lz1*x15;
fz15y=fz15/lz1*y15;
fz15z=fz15/lz1*z15;
mz15x=L0(9)/lz1*x15;
mz15y=L0(9)/lz1*y15;
mz15z=L0(9)/lz1*z15;

fz26=(l(10)-lz2)*k(10);
fz26x=fz26/lz2*x26;
fz26y=fz26/lz2*y26;
fz26z=fz26/lz2*z26;
mz26x=L0(10)/lz2*x26;
mz26y=L0(10)/lz2*y26;
mz26z=L0(10)/lz2*z26;

fz37=(l(11)-lz3)*k(11);
fz37x=fz37/lz3*x37;
fz37y=fz37/lz3*y37;
fz37z=fz37/lz3*z37;
mz37x=L0(11)/lz3*x37;
mz37y=L0(11)/lz3*y37;
```

```
mz37z=L0(11)/lz3*z37;

fz48=(l(12)-lz4)*k(12);
fz48x=fz48/lz4*x48;
fz48y=fz48/lz4*y48;
fz48z=fz48/lz4*z48;
mz48x=L0(12)/lz4*x48;
mz48y=L0(12)/lz4*y48;
mz48z=L0(12)/lz4*z48;

%calculate length
lxy1=(sum((v(1,:)-v(3,:)).^2))^0.5;
x13=(v(1,1)-v(3,1));
y13=(v(1,2)-v(3,2));
z13=(v(1,3)-v(3,3));
lxy2=(sum((v(2,:)-v(4,:)).^2))^0.5;
x24=(v(2,1)-v(4,1));
y24=(v(2,2)-v(4,2));
z24=(v(2,3)-v(4,3));
lxy3=(sum((v(5,:)-v(7,:)).^2))^0.5;
x57=(v(5,1)-v(7,1));
y57=(v(5,2)-v(7,2));
z57=(v(5,3)-v(7,3));
lxy4=(sum((v(6,:)-v(8,:)).^2))^0.5;
x68=(v(6,1)-v(8,1));
y68=(v(6,2)-v(8,2));
z68=(v(6,3)-v(8,3));
lxz1=(sum((v(1,:)-v(8,:)).^2))^0.5;
x18=(v(1,1)-v(8,1));
y18=(v(1,2)-v(8,2));
z18=(v(1,3)-v(8,3));

lxz2=(sum((v(4,:)-v(5,:)).^2))^0.5;
x45=(v(4,1)-v(5,1));
y45=(v(4,2)-v(5,2));
z45=(v(4,3)-v(5,3));

lxz3=(sum((v(2,:)-v(7,:)).^2))^0.5;
x27=(v(2,1)-v(7,1));
y27=(v(2,2)-v(7,2));
z27=(v(2,3)-v(7,3));

lxz4=(sum((v(3,:)-v(6,:)).^2))^0.5;
x36=(v(3,1)-v(6,1));
y36=(v(3,2)-v(6,2));
z36=(v(3,3)-v(6,3));

lyz1=(sum((v(1,:)-v(6,:)).^2))^0.5;
x16=(v(1,1)-v(6,1));
y16=(v(1,2)-v(6,2));
z16=(v(1,3)-v(6,3));

lyz2=(sum((v(2,:)-v(5,:)).^2))^0.5;
x25=(v(2,1)-v(5,1));
y25=(v(2,2)-v(5,2));
z25=(v(2,3)-v(5,3));

lyz3=(sum((v(3,:)-v(8,:)).^2))^0.5;
x38=(v(3,1)-v(8,1));
y38=(v(3,2)-v(8,2));
z38=(v(3,3)-v(8,3));

lyz4=(sum((v(4,:)-v(7,:)).^2))^0.5;
x47=(v(4,1)-v(7,1));
y47=(v(4,2)-v(7,2));
z47=(v(4,3)-v(7,3));
%caculate force and motion in xyz direction
fxy13=(l(13)-lxy1)*k(13);
fxy13x=fxy13/lxy1*x13;
fxy13y=fxy13/lxy1*y13;
fxy13z=fxy13/lxy1*z13;
```

```
mxy13x=L0(13)/lxy1*x13;
mxy13y=L0(13)/lxy1*y13;
mxy13z=L0(13)/lxy1*z13;

fxy24=(l(14)-lxy2)*k(14);
fxy24x=fxy24/lxy2*x24;
fxy24y=fxy24/lxy2*y24;
fxy24z=fxy24/lxy2*z24;
mxy24x=L0(14)/lxy2*x24;
mxy24y=L0(14)/lxy2*y24;
mxy24z=L0(14)/lxy2*z24;

fxy57=(l(15)-lxy3)*k(15);
fxy57x=fxy57/lxy3*x57;
fxy57y=fxy57/lxy3*y57;
fxy57z=fxy57/lxy3*z57;
mxy57x=L0(15)/lxy3*x57;
mxy57y=L0(15)/lxy3*y57;
mxy57z=L0(15)/lxy3*z57;

fxy68=(l(16)-lxy4)*k(16);
fxy68x=fxy68/lxy4*x68;
fxy68y=fxy68/lxy4*y68;
fxy68z=fxy68/lxy4*z68;
mxy68x=L0(16)/lxy4*x68;
mxy68y=L0(16)/lxy4*y68;
mxy68z=L0(16)/lxy4*z68;

fxz18=(l(17)-lxz1)*k(17);
fxz18x=fxz18/lxz1*x18;
fxz18y=fxz18/lxz1*y18;
fxz18z=fxz18/lxz1*z18;
mxz18x=L0(17)/lxz1*x18;
mxz18y=L0(17)/lxz1*y18;
mxz18z=L0(17)/lxz1*z18;

fxz45=(l(18)-lxz2)*k(18);
fxz45x=fxz45/lxz2*x45;
fxz45y=fxz45/lxz2*y45;
fxz45z=fxz45/lxz2*z45;
mxz45x=L0(18)/lxz2*x45;
mxz45y=L0(18)/lxz2*y45;
mxz45z=L0(18)/lxz2*z45;

fxz27=(l(19)-lxz3)*k(19);
fxz27x=fxz27/lxz3*x27;
fxz27y=fxz27/lxz3*y27;
fxz27z=fxz27/lxz3*z27;
mxz27x=L0(19)/lxz3*x27;
mxz27y=L0(19)/lxz3*y27;
mxz27z=L0(19)/lxz3*z27;

fxz36=(l(20)-lxz4)*k(20);
fxz36x=fxz36/lxz4*x36;
fxz36y=fxz36/lxz4*y36;
fxz36z=fxz36/lxz4*z36;
mxz36x=L0(20)/lxz4*x36;
mxz36y=L0(20)/lxz4*y36;
mxz36z=L0(20)/lxz4*z36;

fyz16=(l(21)-lyz1)*k(21);
fyz16x=fyz16/lyz1*x16;
fyz16y=fyz16/lyz1*y16;
fyz16z=fyz16/lyz1*z16;
myz16x=L0(21)/lyz1*x16;
myz16y=L0(21)/lyz1*y16;
myz16z=L0(21)/lyz1*z16;

fyz25=(l(22)-lyz2)*k(22);
fyz25x=fyz25/lyz2*x25;
fyz25y=fyz25/lyz2*y25;
```

```
 fyz25z=fyz25/lyz2*z25;
 myz25x=L0(22)/lyz2*x25;
 myz25y=L0(22)/lyz2*y25;
 myz25z=L0(22)/lyz2*z25;

 fyz38=(l(23)-lyz3)*k(23);
 fyz38x=fyz38/lyz3*x38;
 fyz38y=fyz38/lyz3*y38;
 fyz38z=fyz38/lyz3*z38;
 myz38x=L0(23)/lyz3*x38;
 myz38y=L0(23)/lyz3*y38;
 myz38z=L0(23)/lyz3*z38;

 fyz47=(l(24)-lyz4)*k(24);
 fyz47x=fyz47/lyz4*x47;
 fyz47y=fyz47/lyz4*y47;
 fyz47z=fyz47/lyz4*z47;
 myz47x=L0(24)/lyz4*x47;
 myz47y=L0(24)/lyz4*y47;
 myz47z=L0(24)/lyz4*z47;
 %calculte length
 lxyz1=(sum((v(1,:)-v(7,:)).^2))^0.5;
 x17=(v(1,1)-v(7,1));
 y17=(v(1,2)-v(7,2));
 z17=(v(1,3)-v(7,3));
 lxyz2=(sum((v(3,:)-v(5,:)).^2))^0.5;
 x35=(v(3,1)-v(5,1));
 y35=(v(3,2)-v(5,2));
 z35=(v(3,3)-v(5,3));
 lxyz3=(sum((v(2,:)-v(8,:)).^2))^0.5;
 x28=(v(2,1)-v(8,1));
 y28=(v(2,2)-v(8,2));
 z28=(v(2,3)-v(8,3));
 lxyz4=(sum((v(4,:)-v(6,:)).^2))^0.5;
 x46=(v(4,1)-v(6,1));
 y46=(v(4,2)-v(6,2));
 z46=(v(4,3)-v(6,3));
%calculate force and motion in xyz direction
 fxyz17=(l(25)-lxyz1)*k(25);
 fxyz17x=fxyz17/lxyz1*x17;
 fxyz17y=fxyz17/lxyz1*y17;
 fxyz17z=fxyz17/lxyz1*z17;
 mxyz17x=L0(25)/lxyz1*x17;
 mxyz17y=L0(25)/lxyz1*y17;
 mxyz17z=L0(25)/lxyz1*z17;

 fxyz35=(l(26)-lxyz2)*k(26);
 fxyz35x=fxyz35/lxyz2*x35;
 fxyz35y=fxyz35/lxyz2*y35;
 fxyz35z=fxyz35/lxyz2*z35;
 mxyz35x=L0(26)/lxyz2*x35;
 mxyz35y=L0(26)/lxyz2*y35;
 mxyz35z=L0(26)/lxyz2*z35;

 fxyz28=(l(27)-lxyz3)*k(27);
 fxyz28x=fxyz28/lxyz3*x28;
 fxyz28y=fxyz28/lxyz3*y28;
 fxyz28z=fxyz28/lxyz3*z28;
 mxyz28x=L0(27)/lxyz3*x28;
 mxyz28y=L0(27)/lxyz3*y28;
 mxyz28z=L0(27)/lxyz3*z28;

 fxyz46=(l(28)-lxyz4)*k(28);
 fxyz46x=fxyz46/lxyz4*x46;
 fxyz46y=fxyz46/lxyz4*y46;
 fxyz46z=fxyz46/lxyz4*z46;
 mxyz46x=L0(28)/lxyz4*x46;
 mxyz46y=L0(28)/lxyz4*y46;
 mxyz46z=L0(28)/lxyz4*z46;

 % combine force on masses
```

```
 Fspring=zeros(8,3);
 Fspring(1,1)=(fx14x+fy12x+fz15x+fxy13x+fxz18x+fyz16x+fxyz17x);
 Fspring(1,2)=(fx14y+fy12y+fz15y+fxy13y+fxz18y+fyz16y+fxyz17y);
 Fspring(1,3)=(fx14z+fy12z+fz15z+fxy13z+fxz18z+fyz16z+fxyz17z);

 Fspring(2,1)=(fx23x-fy12x+fz26x+fxy24x+fxz27x+fyz25x+fxyz28x);
 Fspring(2,2)=(fx23y-fy12y+fz26y+fxy24y+fxz27y+fyz25y+fxyz28y);
 Fspring(2,3)=(fx23z-fy12z+fz26z+fxy24z+fxz27z+fyz25z+fxyz28z);

 Fspring(3,1)=(-fx23x+fy34x+fz37x-fxy13x+fxz36x+fyz38x+fxyz35x);
 Fspring(3,2)=(-fx23y+fy34y+fz37y-fxy13y+fxz36y+fyz38y+fxyz35y);
 Fspring(3,3)=(-fx23z+fy34z+fz37z-fxy13z+fxz36z+fyz38z+fxyz35z);

 Fspring(4,1)=(-fx14x-fy34x+fz48x-fxy24x+fxz45x+fyz47x+fxyz46x);
 Fspring(4,2)=(-fx14y-fy34y+fz48y-fxy24y+fxz45y+fyz47y+fxyz46y);
 Fspring(4,3)=(-fx14z-fy34z+fz48z-fxy24z+fxz45z+fyz47z+fxyz46z);

 Fspring(5,1)=(fx58x+fy56x-fz15x+fxy57x-fxz45x-fyz25x-fxyz35x);
 Fspring(5,2)=(fx58y+fy56y-fz15y+fxy57y-fxz45y-fyz25y-fxyz35y);
 Fspring(5,3)=(fx58z+fy56z-fz15z+fxy57z-fxz45z-fyz25z-fxyz35z);

 Fspring(6,1)=(fx67x-fy56x-fz26x+fxy68x-fxz36x-fyz16x-fxyz46x);
 Fspring(6,2)=(fx67y-fy56y-fz26y+fxy68y-fxz36y-fyz16y-fxyz46y);
 Fspring(6,3)=(fx67z-fy56z-fz26z+fxy68z-fxz36z-fyz16z-fxyz46z);

 Fspring(7,1)=(-fx67x+fy78x-fz37x-fxy57x-fxz27x-fyz47x-fxyz17x);
 Fspring(7,2)=(-fx67y+fy78y-fz37y-fxy57y-fxz27y-fyz47y-fxyz17y);
 Fspring(7,3)=(-fx67z+fy78z-fz37z-fxy57z-fxz27z-fyz47z-fxyz17z);

 Fspring(8,1)=(-fx58x-fy78x-fz48x-fxy68x-fxz18x-fyz38x-fxyz28x);
 Fspring(8,2)=(-fx58y-fy78y-fz48y-fxy68y-fxz18y-fyz38y-fxyz28y);
 Fspring(8,3)=(-fx58z-fy78z-fz48z-fxy68z-fxz18z-fyz38z-fxyz28z);
 % combine Motion on masses
 moveL0=zeros(8,3);
 moveL0(1,1)=(mx14x+my12x+mz15x+mxy13x+mxz18x+myz16x+mxyz17x)/7;
 moveL0(1,2)=(mx14y+my12y+mz15y+mxy13y+mxz18y+myz16y+mxyz17y)/7;
 moveL0(1,3)=(mx14z+my12z+mz15z+mxy13z+mxz18z+myz16z+mxyz17z)/7;

 moveL0(2,1)=(mx23x-my12x+mz26x+mxy24x+mxz27x+myz25x+mxyz28x)/7;
 moveL0(2,2)=(mx23y-my12y+mz26y+mxy24y+mxz27y+myz25y+mxyz28y)/7;
 moveL0(2,3)=(mx23z-my12z+mz26z+mxy24z+mxz27z+myz25z+mxyz28z)/7;

 moveL0(3,1)=(-mx23x+my34x+mz37x-mxy13x+mxz36x+myz38x+mxyz35x)/7;
 moveL0(3,2)=(-mx23y+my34y+mz37y-mxy13y+mxz36y+myz38y+mxyz35y)/7;
 moveL0(3,3)=(-mx23z+my34z+mz37z-mxy13z+mxz36z+myz38z+mxyz35z)/7;

 moveL0(4,1)=(-mx14x-my34x+mz48x-mxy24x+mxz45x+myz47x+mxyz46x)/7;
 moveL0(4,2)=(-mx14y-my34y+mz48y-mxy24y+mxz45y+myz47y+mxyz46y)/7;
 moveL0(4,3)=(-mx14z-my34z+mz48z-mxy24z+mxz45z+myz47z+mxyz46z)/7;

 moveL0(5,1)=(mx58x+my56x-mz15x+mxy57x-mxz45x-myz25x-mxyz35x)/7;
 moveL0(5,2)=(mx58y+my56y-mz15y+mxy57y-mxz45y-myz25y-mxyz35y)/7;
 moveL0(5,3)=(mx58z+my56z-mz15z+mxy57z-mxz45z-myz25z-mxyz35z)/7;

 moveL0(6,1)=(mx67x-my56x-mz26x+mxy68x-mxz36x-myz16x-mxyz46x)/7;
 moveL0(6,2)=(mx67y-my56y-mz26y+mxy68y-mxz36y-myz16y-mxyz46y)/7;
 moveL0(6,3)=(mx67z-my56z-mz26z+mxy68z-mxz36z-myz16z-mxyz46z)/7;

 moveL0(7,1)=(-mx67x+my78x-mz37x-mxy57x-mxz27x-myz47x-mxyz17x)/7;
 moveL0(7,2)=(-mx67y+my78y-mz37y-mxy57y-mxz27y-myz47y-mxyz17y)/7;
 moveL0(7,3)=(-mx67z+my78z-mz37z-mxy57z-mxz27z-myz47z-mxyz17z)/7;

 moveL0(8,1)=(-mx58x-my78x-mz48x-mxy68x-mxz18x-myz38x-mxyz28x)/7;
 moveL0(8,2)=(-mx58y-my78y-mz48y-mxy68y-mxz18y-myz38y-mxyz28y)/7;
 moveL0(8,3)=(-mx58z-my78z-mz48z-mxy68z-mxz18z-myz38z-mxyz28z)/7;
end
```

Calculate the original length

```
function [l]=lengthorigin(v)
%% calculate the origin length for the spring
l=zeros(28,1);
```

```
l(1)=(sum((v(1,:)-v(4,:)).^2))^0.5;
l(2)=(sum((v(2,:)-v(3,:)).^2))^0.5;
l(3)=(sum((v(5,:)-v(8,:)).^2))^0.5;
l(4)=(sum((v(6,:)-v(7,:)).^2))^0.5;
l(5)=(sum((v(1,:)-v(2,:)).^2))^0.5;
l(6)=(sum((v(3,:)-v(4,:)).^2))^0.5;
l(7)=(sum((v(5,:)-v(6,:)).^2))^0.5;
l(8)=(sum((v(7,:)-v(8,:)).^2))^0.5;
l(9)=(sum((v(1,:)-v(5,:)).^2))^0.5;
l(10)=(sum((v(2,:)-v(6,:)).^2))^0.5;
l(11)=(sum((v(3,:)-v(7,:)).^2))^0.5;
l(12)=(sum((v(4,:)-v(8,:)).^2))^0.5;
l(13)=(sum((v(1,:)-v(3,:)).^2))^0.5;
l(14)=(sum((v(2,:)-v(4,:)).^2))^0.5;
l(15)=(sum((v(5,:)-v(7,:)).^2))^0.5;
l(16)=(sum((v(6,:)-v(8,:)).^2))^0.5;
l(17)=(sum((v(1,:)-v(8,:)).^2))^0.5;
l(18)=(sum((v(4,:)-v(5,:)).^2))^0.5;
l(19)=(sum((v(2,:)-v(7,:)).^2))^0.5;
l(20)=(sum((v(3,:)-v(6,:)).^2))^0.5;
l(21)=(sum((v(1,:)-v(6,:)).^2))^0.5;
l(22)=(sum((v(2,:)-v(5,:)).^2))^0.5;
l(23)=(sum((v(3,:)-v(8,:)).^2))^0.5;
l(24)=(sum((v(4,:)-v(7,:)).^2))^0.5;
l(25)=(sum((v(1,:)-v(7,:)).^2))^0.5;
l(26)=(sum((v(3,:)-v(5,:)).^2))^0.5;
l(27)=(sum((v(2,:)-v(8,:)).^2))^0.5;
l(28)=(sum((v(4,:)-v(6,:)).^2))^0.5;
end
```

Decide the masses positions

```
function [v] = locationsix(vn,cood)
% find the start position of masses
v=zeros(1,length(vn)*3);
% ensure the coordinate
p=zeros(cood^3,3);
n=1;
for a =1:cood
 for b=1:cood
 for c=1:cood
 p(n,:)=[(a-1), (b-1), (c-1)];
 n=n+1;
 end
 end
end
% find the masses position
p=p*0.5;
for i= 1: length(vn)
v(i)=p(vn(i),1);
v(i+length(vn))=p(vn(i),2);
v(i+length(vn)*2)=p(vn(i),3);
end
```

tournament selection

```
function chrom_selected = Rank(chrom,fit)
%% tournament selection
num=length (fit);
choose=randperm(num,7); % choose 7 group member
score=fit(choose); % rank them
name=chrom(choose,:);
[~,idx] = sort(score,'descend'); % choose best 2 of them
chrom_selected=name(idx(1:2),:);
end
```