```
%Exercise 36.1
syms u

ri=4 * 0.3960 * cos(2.65 * (u + 1.4));
rj=4*(- 0.99 * sin(u + 1.4));
rk=0*u;
r=[ri,rj,rk];

dr=diff(r,u);
assume(u,{'real','positive'})


T_hat_ugly=dr./norm(dr)
```

```
T_hat_ugly =
```

$$\left(-\frac{5247\sin\left(\frac{53\,u}{20}+\frac{371}{100}\right)}{1250\,\sigma_1}\quad -\frac{99\cos\left(u+\frac{7}{5}\right)}{25\,\sigma_1}\quad 0\right)$$

where

$$\sigma_1=\sqrt{\frac{27531009\left|\sin\left(\frac{53\,u}{20}+\frac{371}{100}\right)\right|^2}{1562500}+\frac{9801\left|\cos\left(u+\frac{7}{5}\right)\right|^2}{625}}$$

```
T_hat=simplify(T_hat_ugly)
```

```
T_hat =
```

$$\left(-\frac{53\sin\left(\frac{53\,u}{20}+\frac{371}{100}\right)}{\sigma_1}\quad -\frac{50\cos\left(u+\frac{7}{5}\right)}{\sigma_1}\quad 0\right)$$

where

$$\sigma_1=\sqrt{2500\cos\left(u+\frac{7}{5}\right)^2+2809\sin\left(\frac{53\,u}{20}+\frac{371}{100}\right)^2}$$

```
dT_hat=diff(T_hat,u);
N_hat=dT_hat/norm(dT_hat);
N_hat=simplify(N_hat)
```

```
N_hat =
```

$$\left(-\frac{2\ (483625\,\sigma_6 + 702250\,\sigma_5 + 218625\,\sigma_4)}{\sigma_1}\quad \frac{561800\sin\left(u+\frac{7}{5}\right) + 1025285\,\sigma_3 + 463485\,\sigma_2}{\sigma_1}\quad 0\right)$$

where

$$\sigma_1 = 265\ \sqrt{2809\ \left(40\sin\left(u+\frac{7}{5}\right) + 73\,\sigma_3 + 33\,\sigma_2\right)^2 + 2500\ (73\,\sigma_6 + 106\,\sigma_5 + 33\,\sigma_4)^2}$$

$$\sigma_2 = \sin\left(\frac{63\,u}{10} + \frac{441}{50}\right)$$

$$\sigma_3 = \sin\left(\frac{43\,u}{10} + \frac{301}{50}\right)$$

$$\sigma_4 = \cos\left(\frac{93\,u}{20} + \frac{651}{100}\right)$$

$$\sigma_5 = \cos\left(\frac{53\,u}{20} + \frac{371}{100}\right)$$

$$\sigma_6 = \cos\left(\frac{13\,u}{20} + \frac{91}{100}\right)$$

```
B_hat=cross(T_hat,N_hat);
B_hat=simplify(B_hat)
```

```
B_hat =
```

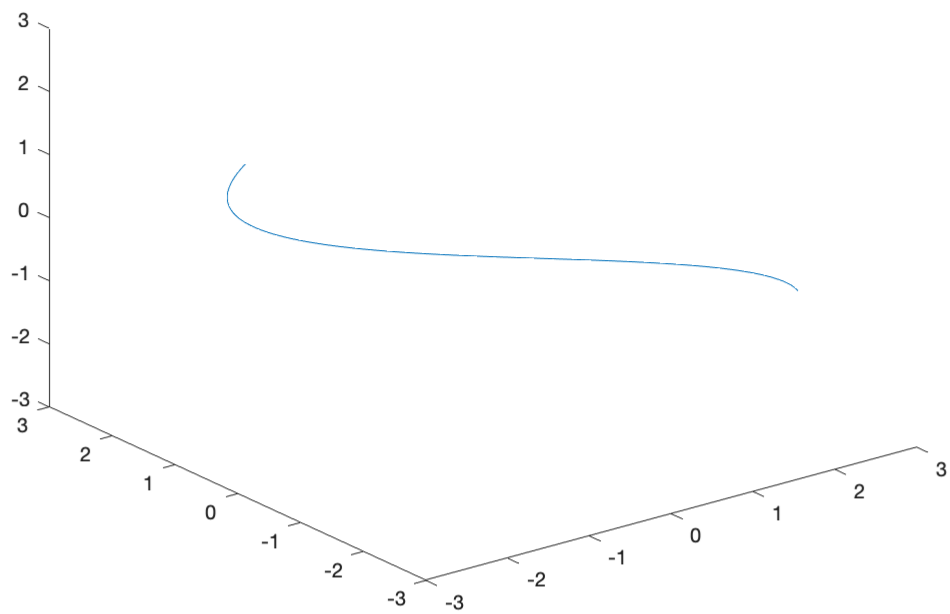
$$\left(0\quad 0\quad -\frac{91250\cos\left(\frac{7\,u}{20}+\frac{49}{100}\right) + \frac{764917\cos\left(\frac{33\,u}{20}+\frac{231}{100}\right)}{2} + \frac{327837\cos\left(\frac{73\,u}{20}+\frac{511}{100}\right)}{2} + 41250}{\sqrt{2809\ \left(40\sin\left(u+\frac{7}{5}\right) + 73\sin\left(\frac{43\,u}{10}+\frac{301}{50}\right) + 33\sin\left(\frac{63\,u}{10}+\frac{441}{50}\right)\right)^2 + 2500\ \left(73\cos\left(\frac{13\,u}{20} +\right.\right.}}\right.$$

where

$$\sigma_1 = \frac{53\,u}{20} + \frac{371}{100}$$

```
R_num = 4;
u_num = linspace(0,3.2);

for n=1:length(u_num)
    r_num(n,:)=double(subs(r,[u],[u_num(n)]));
    T_hat_num(n,:)=double(subs(T_hat,[u],[u_num(n)]));
    N_hat_num(n,:)=double(subs(N_hat,[u],[u_num(n)]));
```

```
    B_hat_num(n,:)=double(subs(B_hat,[u],[u_num(n)]));

    plot3(r_num(:,1),r_num(:,2),r_num(:,3)), axis([-3 3 -3 3 -3 3]),hold on % plot the
    quiver3(r_num(n,1),r_num(n,2),r_num(n,3),T_hat_num(n,1),T_hat_num(n,2),T_hat_num(n,
    quiver3(r_num(n,1),r_num(n,2),r_num(n,3),N_hat_num(n,1),N_hat_num(n,2),N_hat_num(n,
    quiver3(r_num(n,1),r_num(n,2),r_num(n,3),B_hat_num(n,1),B_hat_num(n,2),B_hat_num(n,
    drawnow
end
```




```
%Exercise 36.2 & 36.3
syms t d B
%Getting Linear Velocity Vector
ri=4 * (0.3960 * cos(2.65 * (B*t+ 1.4)));
rj=4*(- 0.99 * sin(B*t + 1.4));
rk = 0*t
```

rk = 0

```
r=[ri,rj,rk];
diffrt=diff(r,t);
assume(t,{'real','positive'})
assume(d,{'real','positive'})
assume(B,{'real','positive'})



%Magnitude of Linear Velocity Vector (Linear Speed)
```

```
maglist = [diffrt]
```

maglist =

$$\left(-\frac{5247\,B\sin\left(\frac{53\,B\,t}{20}+\frac{371}{100}\right)}{1250}\quad -\frac{99\,B\cos\left(B\,t+\frac{7}{5}\right)}{25}\quad 0\right)$$

```
mag = norm(maglist)
```

mag =

$$\sqrt{\frac{9801\,B^2\left|\cos\left(B\,t+\frac{7}{5}\right)\right|^2}{625}+\frac{27531009\,B^2\left|\sin\left(\frac{53\,B\,t}{20}+\frac{371}{100}\right)\right|^2}{1562500}}$$

```
V = mag
```

V =

$$\sqrt{\frac{9801\,B^2\left|\cos\left(B\,t+\frac{7}{5}\right)\right|^2}{625}+\frac{27531009\,B^2\left|\sin\left(\frac{53\,B\,t}{20}+\frac{371}{100}\right)\right|^2}{1562500}}$$

```
%Unit Tangent and Unit Normal Vectors
Unit_Tangent=simplify(diffrt./norm(diffrt))
```

Unit_Tangent =

$$\left(-\frac{53\sin\left(\frac{53\,B\,t}{20}+\frac{371}{100}\right)}{\sigma_1}\quad -\frac{50\cos\left(B\,t+\frac{7}{5}\right)}{\sigma_1}\quad 0\right)$$

where

$$\sigma_1=\sqrt{2500\cos\left(B\,t+\frac{7}{5}\right)^2+2809\sin\left(\frac{53\,B\,t}{20}+\frac{371}{100}\right)^2}$$

```
Dir_Tangent=diff(Unit_Tangent,t);

Unit_Normal = simplify(diff(Unit_Tangent,t)/norm(diff(Unit_Tangent,t)))
```

Unit_Normal =

$$\begin{pmatrix} -\frac{50\,\sigma_3}{\sigma_1} & \frac{53\,\sigma_2}{\sigma_1} & 0 \end{pmatrix}$$

where

$$\sigma_1 = \sqrt{2500\,{\sigma_3}^2 + 2809\,{\sigma_2}^2}$$

$$\sigma_2 = 40\sin\left(B\,t + \frac{7}{5}\right) + 73\sin\left(\frac{43\,B\,t}{10} + \frac{301}{50}\right) + 33\sin\left(\frac{63\,B\,t}{10} + \frac{441}{50}\right)$$

$$\sigma_3 = 73\cos\left(\frac{13\,B\,t}{20} + \frac{91}{100}\right) + 106\cos\left(\frac{53\,B\,t}{20} + \frac{371}{100}\right) + 33\cos\left(\frac{93\,B\,t}{20} + \frac{651}{100}\right)$$

```
%Angular Velocity
omega=simplify(cross(Unit_Tangent,Dir_Tangent))
```

omega =

$$\begin{pmatrix} 0 & 0 & -\frac{265\,B\,\left(73\cos\left(\frac{33\,B\,t}{20} + \frac{231}{100}\right) + 33\cos\left(\frac{73\,B\,t}{20} + \frac{511}{100}\right)\right)}{5000\cos\left(2\,B\,t + \frac{14}{5}\right) - 5618\cos\left(\frac{53\,B\,t}{10} + \frac{371}{50}\right) + 10618} \end{pmatrix}$$

```
load dataset.mat
timeframe=dataset(:,1)
```

```
timeframe = 321×1
    0.0930
    0.1930
    0.2930
    0.3900
    0.4930
    0.5930
    0.6920
    0.7920
    0.8920
    0.9930
      ⋮
```

```
leftpos=dataset(:,2)
```

```
leftpos = 321×1
    0.0402
    0.0605
    0.0809
    0.1013
    0.1217
    0.1420
    0.1718
    0.2015
    0.2312
    0.2609
```

⋮

```
rightpos=dataset(:,3)
```

```
rightpos = 321×1
    0.0528
    0.0796
    0.1063
    0.1331
    0.1599
    0.1867
    0.2198
    0.2529
    0.2860
    0.3192
      ⋮
```

```
leftvelo=diff(leftpos)./diff(timeframe);
rightvelo=diff(rightpos)./diff(timeframe);

velo_exp=(leftvelo+rightvelo)./2
```

```
velo_exp = 320×1
    0.2358
    0.2357
    0.2430
    0.2289
    0.2357
    0.3174
    0.3142
    0.3142
    0.3111
    0.3051
      ⋮
```

```
angular_exp=(rightvelo-leftvelo)./0.235
```

```
angular_exp = 320×1
    0.2721
    0.2721
    0.2805
    0.2642
    0.2721
    0.1469
    0.1453
    0.1453
    0.1439
    0.1411
      ⋮
```

```
avgleft=(mean(reshape(leftvelo,10,[])))';
avgright=(mean(reshape(rightvelo,10,[])))';
avgvelo = (mean(reshape(velo_exp,10,[])))';
angularvelo = (mean(reshape(angular_exp,10,[])))';
```

```
vl = simplify (V - d/2*omega(3));
vr = simplify (V + d/2*omega(3));

t_num = linspace(0,32,32)
```

```
t_num = 1×32
         0    1.0323    2.0645    3.0968    4.1290    5.1613    6.1935    7.2258 ...
```

```
B_num = 0.1
```

```
B_num = 0.1000
```

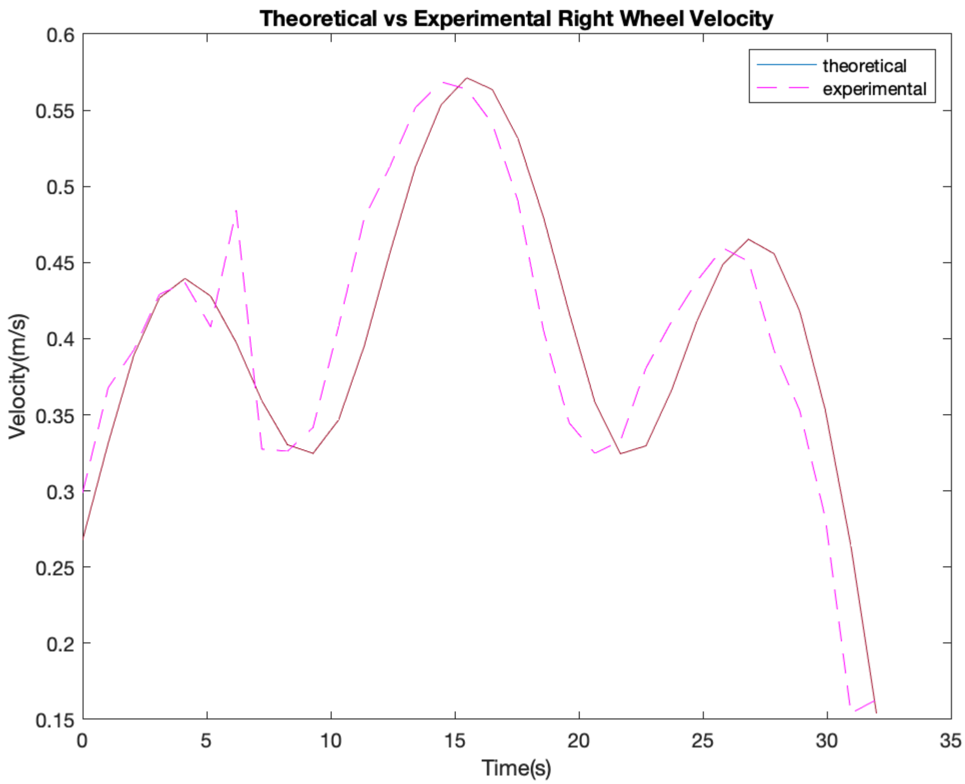
```
d_num = 0.235
```

```
d_num = 0.2350
```

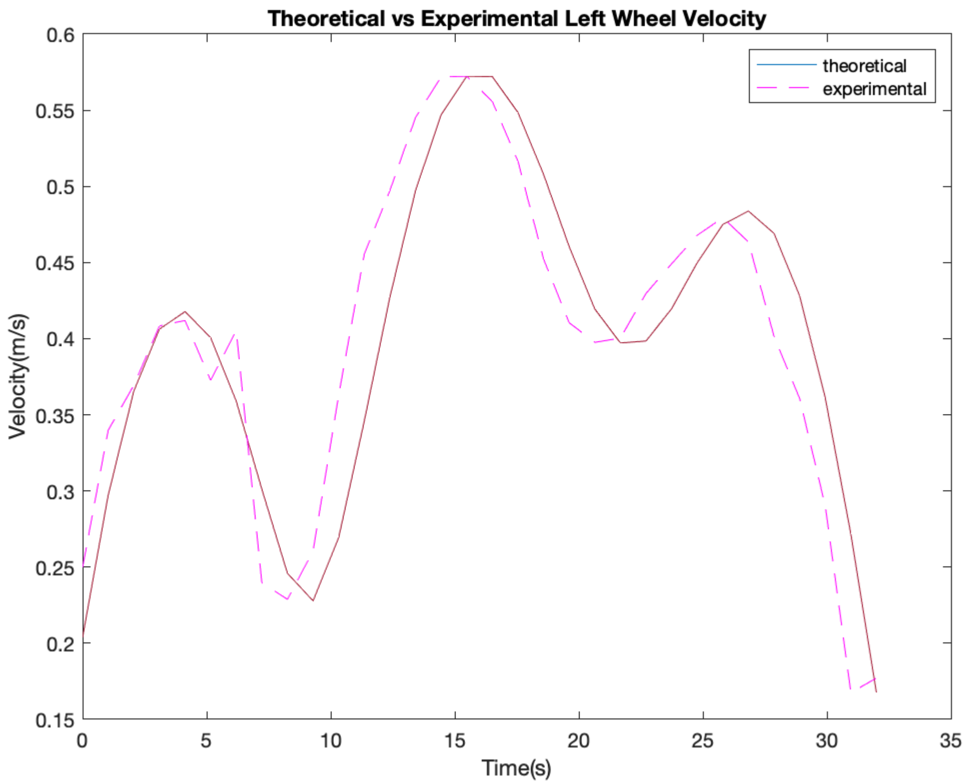
```
tx = t_num'
```

```
tx = 32×1
         0
    1.0323
    2.0645
    3.0968
    4.1290
    5.1613
    6.1935
    7.2258
    8.2581
    9.2903
      ⋮
```

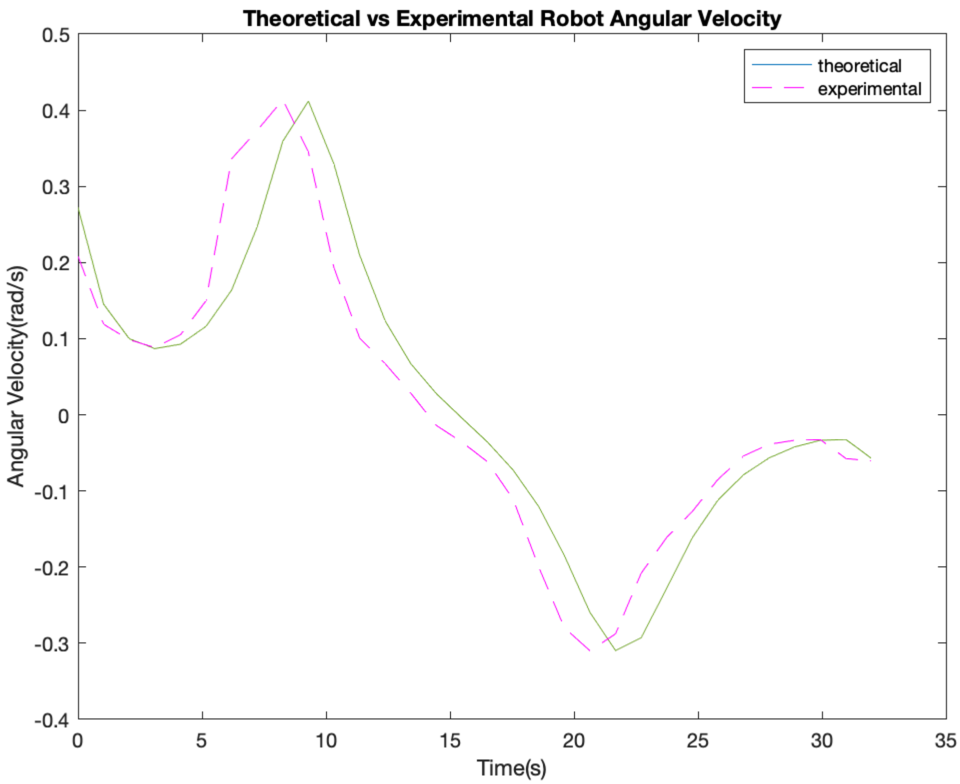
```
for n=1:length(t_num)
    vl_num(n,:)=double(subs(vl,[t,d,B],[t_num(n),d_num,B_num]));
    vr_num(n,:)=double(subs(vr,[t,d,B],[t_num(n),d_num,B_num]));
    omega_num(n,:)=double(subs(omega,[t,d,B],[t_num(n),d_num, B_num]));
    lin_vel(n,:)=double(subs(V,[t,d,B],[t_num(n),d_num, B_num]));
end

figure(1)
title('Theoretical vs Experimental Right Wheel Velocity')
plot(tx(:,1),vr_num(:,1))
hold on
plot(tx(:,1),avgright(:,1),'--m')
xlabel('Time(s)')
ylabel('Velocity(m/s)')
legend('theoretical','experimental')
```



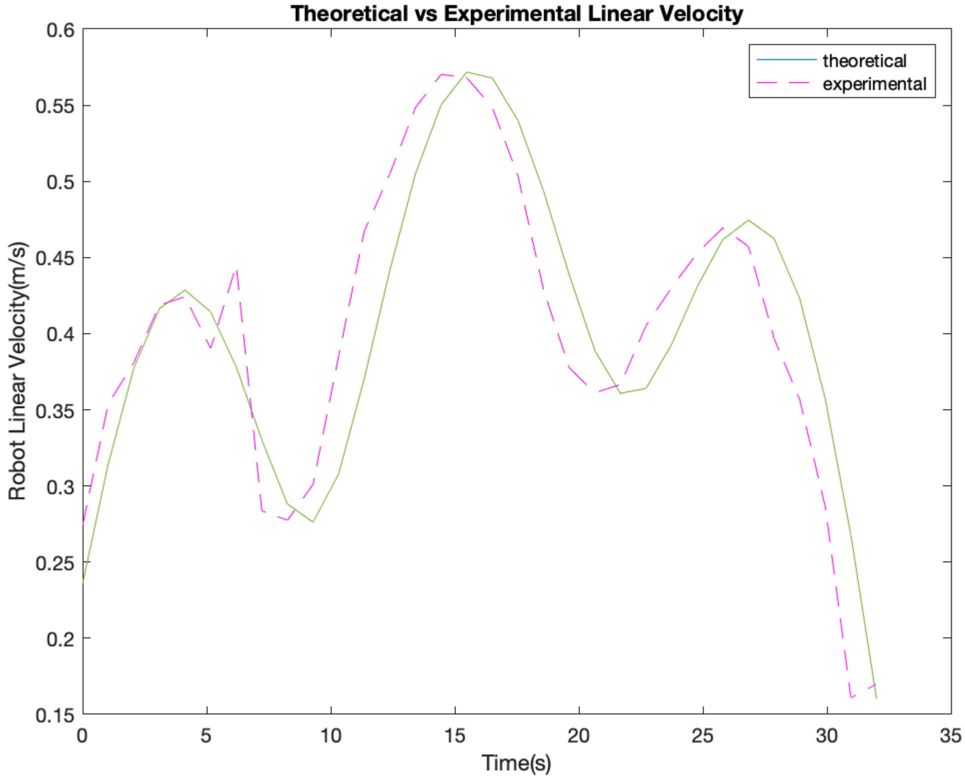

```
figure(2)
title('Theoretical vs Experimental Left Wheel Velocity')
plot(tx(:,1),vl_num(:,1))
hold on
plot(tx(:,1),avgleft(:,1),'--m')
xlabel('Time(s)')
ylabel('Velocity(m/s)')
legend('theoretical','experimental')
```




```
figure(3)
title('Theoretical vs Experimental Robot Angular Velocity')
plot(tx(:,1),omega_num(:,3))
hold on
plot(tx(:,1),angularvelo(:,1),'--m')
xlabel('Time(s)')
ylabel('Angular Velocity(rad/s)')
legend('theoretical','experimental')
```




```
figure(4)
title('Theoretical vs Experimental Linear Velocity')
plot(tx(:,1),lin_vel(:,1))
hold on
plot(tx(:,1),avgvelo(:,1),'--m')
xlabel('Time(s)')
ylabel('Robot Linear Velocity(m/s)')
legend('theoretical','experimental')
```




```
%Exercise 36.4
QEARobo(vl_num,vr_num)
```

```
%Exercise 36.5
load dataset.mat
timeframe=dataset(:,1)
leftpos=dataset(:,2)
rightpos=dataset(:,3)
tstep=0.1

leftvelo=diff(leftpos)./diff(timeframe)
rightvelo=diff(rightpos)./diff(timeframe)

velo_exp=(leftvelo+rightvelo)./2
angular_exp=(rightvelo-leftvelo)./0.235

u = [0,3.2];
% u will be our parameter
syms u;

% this is the equation of the bridge
R = 4*[0.396*cos(2.65*(u+1.4));...
       -0.99*sin(u+1.4);...
       0];
```

```
% tangent vector
T = diff(R);

% normalized tangent vector
That = T/norm(T);

bridgeStart = double(subs(R,u,0))';
startingThat = double(subs(That,u,0))';


heading= startingThat/norm(startingThat)
position=bridgeStart
new_position=position
angle=0

n=1
while n < 321
    new_angle=angle+angular_exp(n,:)*tstep;
    new_heading=[heading(:,1)*cos(new_angle)-heading(:,2)*sin(new_angle),heading(:,1)*s
    new_position=new_position+velo_exp(n,:)*new_heading*tstep;
    position=cat(1,position,new_position);
    %heading=new_heading;
    angle=new_angle;
    n=n+1;
end

p=position
plot3(p(:,1),p(:,2),p(:,3))
```

```
Lwheel_vel = 321×1
    0.4317
    0.3136
    0.2761
    0.2597
    0.2468
    0.2395
    0.2482
    0.2544
    0.2592
    0.2627
      ⋮

RWheel_vel = 321×1
    0.5677
    0.4123
    0.3629
    0.3413
    0.3243
    0.3148
    0.3176
    0.3193
    0.3207
    0.3214
      ⋮

V_exp = 321×1
```

```
    0.4997
    0.3630
    0.3195
    0.3005
    0.2855
    0.2771
    0.2829
    0.2869
    0.2899
    0.2921
      ⋮

omega_exp = 321×1
    0.5787
    0.4198
    0.3694
    0.3473
    0.3299
    0.3202
    0.2954
    0.2764
    0.2617
    0.2497
      ⋮

theta = 321×1
    0.0538
    0.0810
    0.1082
    0.1354
    0.1627
    0.1899
    0.2044
    0.2189
    0.2335
    0.2480
      ⋮

startingPosX = -1.3349
startingPosY = -3.9024
x_position = 2×1
   -1.3349
    1.3857
y_position = 2×1
   -3.9024
    0.0746
Index exceeds the number of array elements (2).
```

```
function get_value(n)
A + x_position(n)+mvecR(n)
end




function collectDataset_sim(datasetname)
% This script provides a method for collecting a dataset from the Neato
% sensors suitable for plotting out a 3d trajectory.  To launch the
% application run:
%
```

```
%    collectDataset_sim('nameofdataset.mat')
%
% Where you should specify where you'd like to the program to save the
% the dataset you collect.
%
% The collected data will be stored in a variable called dataset.  Dataset
% will be a nx6 matrix where each row contains a timestamp, the encoder
% values, and the accelerometer values.  Specifically, here is the row
% format.
%
% [timestamp, positionLeft, positionRight, AccelX, AccelY, AccelZ];
%
% To stop execution of the program, simply close the figure window.

    function myCloseRequest(src,callbackdata)
        % Close request function
        % to display a question dialog box
        % get rid of subscriptions to avoid race conditions
        clear sub_encoders;
        clear sub_accel;
        delete(gcf)
    end

    function processAccel(sub, msg)
        % Process the encoders values by storing by storing them into
        % the matrix of data.
        lastAccel = msg.Data;
    end

    function processEncoders(sub, msg)
        % Process the encoders values by storing by storing them into
        % the matrix of data.
        if ~collectingData
            return;
        end
        currTime = rostime('now');
        currTime = double(currTime.Sec)+double(currTime.Nsec)*10^-9;
        elapsedTime = currTime - start;
        dataset(encoderCount + 1,:) = [elapsedTime msg.Data' lastAccel'];
        encoderCount = encoderCount + 1;
    end

    function keyPressedFunction(fig_obj, eventDat)
        % Convert a key pressed event into a twist message and publish it
        ck = get(fig_obj, 'CurrentKey');
        switch ck
            case 'space'
                if collectingData
                    collectingData = false;
                    dataset = dataset(1:encoderCount, :);
                    save(datasetname, 'dataset');
                    disp('Stopping dataset collection');
                else
                    start = rostime('now');
```

```
                    start = double(start.Sec)+double(start.Nsec)*10^-9;
                    encoderCount = 0;
                    dataset = zeros(100000, 6);
                    collectingData = true;
                    disp('Starting dataset collection');
                end
            end
        end
    global dataset start encoderCount lastAccel;
    lastAccel = [0; 0; 1];      % set this to avoid a very unlikely to occur race condi
    collectingData = false;
    sub_encoders = rossubscriber('/encoders', @processEncoders);
    sub_accel = rossubscriber('/accel', @processAccel);

 f = figure('CloseRequestFcn',@myCloseRequest);
    title('Dataset Collection Window');
    set(f,'WindowKeyPressFcn', @keyPressedFunction);
end



function drive(vl_num,vr_num)

% these are our target wheel velocities
% this publisher lets us command the wheel velocities
pub = rospublisher('/raw_vel');
sub_states = rossubscriber('/gazebo/model_states', 'gazebo_msgs/ModelStates');

msg = rosmessage(pub);

% make sure robot starts out with 0 velocity
msg.Data = [0, 0];
send(pub, msg);
pause(2);

vlp = vl_num'
vrp = vr_num'

start = rostime('now')
pause(1.3)
while 1
    tic
    currTime = rostime('now')
    elapsedTime = currTime - start

    msg.Data = [vlp(round(elapsedTime.seconds)), vrp(round(elapsedTime.seconds))];
    send(pub, msg)
    if elapsedTime > 32
        break;
    end
    pause(1-toc)
end
```

```
% n = 1
% while true
%     tic
%     msg.Data = [vlp(round(n)), vrp(round(n))];
%     send(pub, msg)
%     n = n+1;
%     pause(0.6-toc)
%     if n > 200
%         break;
%     end
% end

% stop the Neato by setting both wheel velocities to 0
msg.Data = [0, 0];
send(pub, msg);
end


function QEARobo(vl_num,vr_num)
% Insert any setup code you want to run here

% define u explicitly to avoid error when using sub functions
% see: https://www.mathworks.com/matlabcentral/answers/268580-error-attempt-to-add-vari
u = [0,3.2];
% u will be our parameter
syms u;

% this is the equation of the bridge
R = 4*[0.396*cos(2.65*(u+1.4));...
       -0.99*sin(u+1.4);...
       0];

% tangent vector
T = diff(R);

% normalized tangent vector
That = T/norm(T);

pub = rospublisher('raw_vel');

% stop the robot if it's going right now
stopMsg = rosmessage(pub);
stopMsg.Data = [0 0];
send(pub, stopMsg);

bridgeStart = double(subs(R,u,0));
startingThat = double(subs(That,u,0));
placeNeato(bridgeStart(1),  bridgeStart(2), startingThat(1), startingThat(2));

% wait a bit for robot to fall onto the bridge
pause(2);

% time to drive!!
```

```
drive(vl_num,vr_num)



% For simulated Neatos only:
% Place the Neato in the specified x, y position and specified heading vector.
function placeNeato(posX, posY, headingX, headingY)
    svc = rossvcclient('gazebo/set_model_state');
    msg = rosmessage(svc);

    msg.ModelState.ModelName = 'neato_standalone';
    startYaw = atan2(headingY, headingX);
    quat = eul2quat([startYaw 0 0]);

    msg.ModelState.Pose.Position.X = posX;
    msg.ModelState.Pose.Position.Y = posY;
    msg.ModelState.Pose.Position.Z = 1.0;
    msg.ModelState.Pose.Orientation.W = quat(1);
    msg.ModelState.Pose.Orientation.X = quat(2);
    msg.ModelState.Pose.Orientation.Y = quat(3);
    msg.ModelState.Pose.Orientation.Z = quat(4);

    % put the robot in the appropriate place
    ret = call(svc, msg);
end
end
```