QEA Bridge Of Death™

Annie Kroo and Samantha Young

April 2017

1 Exercise 1

In Exercise 1, we qualitatively determined how the Neato would move.

Exercise 1	
Case	Quantitative Prediction
Both wheels move forward same positive velocity	Moving straight forward
Both wheels move backwards same negative velocity	Moving straight backwards
One wheel drives forward and the other backward with equal speed	Turn in place
One wheel drives forward while the other remains stationary	Pivot around one wheel

Table 1: Qualitative Predictions

2 Exercise 2

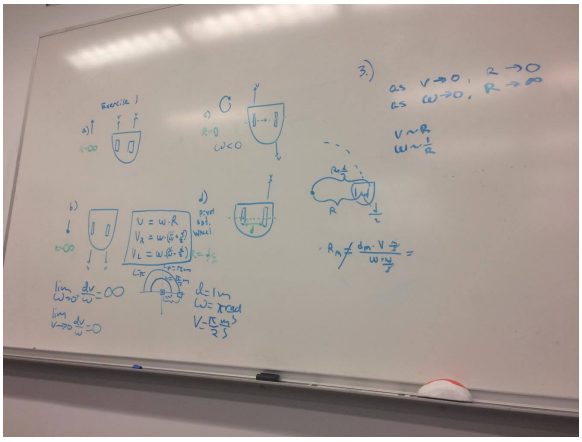

Figure 1: The board captures exercises 1, 2 and 3

3 Exercise 3

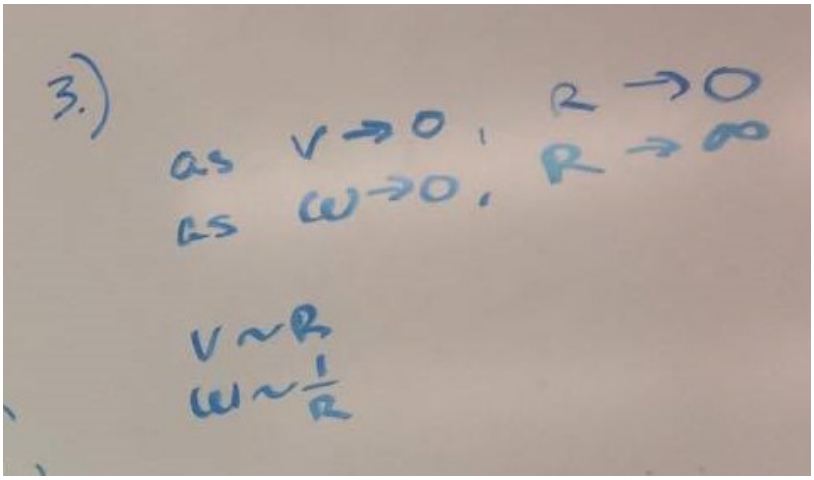

Figure 2: The beginning of our derivations

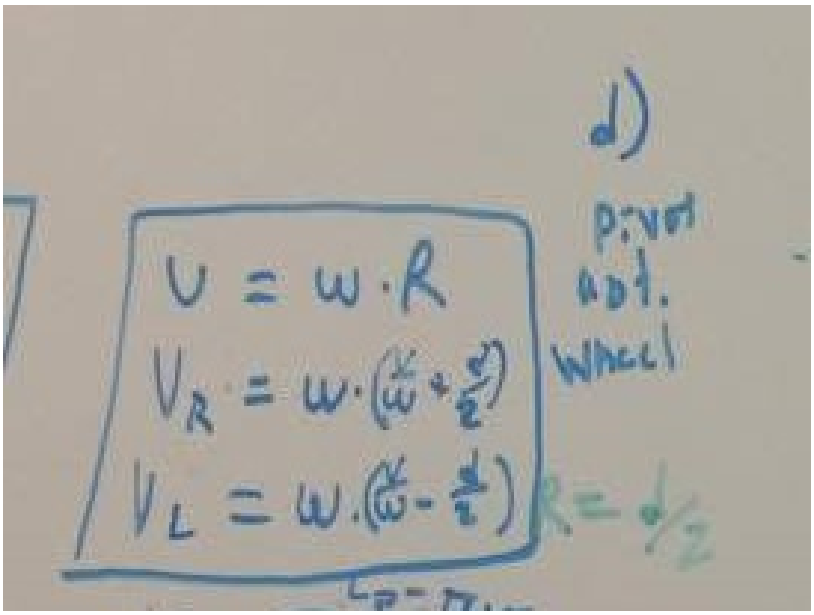

Figure 3: Complete equations

4 Exercise 4

4.1 Experiment 1

4.1.1 Experiment Goal

Qualitatively validate the that our predicted R value is the actual R value.

4.1.2 Procedure

We will have our robot complete a full circle path turning about an R value of 0.247. This will create a large circular path. In order to implement this, we used the equation $vR = \omega * ((v/w) + (d/2))$ and the equation $vL = \omega * ((v/w) + (d/2))$ where v is the linear velocity, ω is the angular velocity, and d is the distance between the wheels of the Neato. To find the actual R value we attached a marker to our robot so that it drew a circle, which we could then use to find the distance to the center of the path, in our case a circle, traced.

4.1.3 Measurements

We will measure how far the point of which the robot turns, R, from the center of the robot. We have the calculated value for R and will directly measure the experimental value. By comparing these measurements we can find the error that occurs when the robot drives and determine if our calculations are correct.

4.1.4 Weaknesses

This experiment assumes there is no wheel slippage and the robot instantaneously accelerates to the desired velocity.

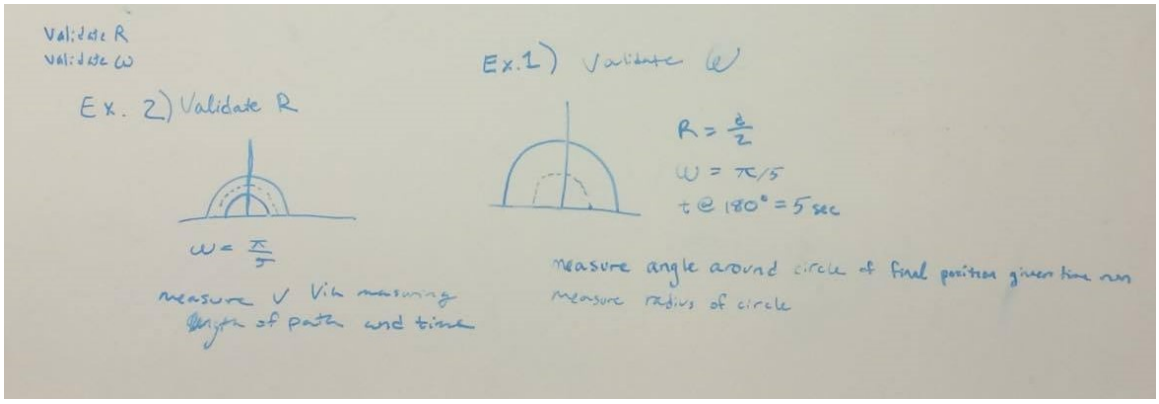

Figure 4: Creating the experiements

5 Exercise 5

5.1 Results

This experiment had a percent error of 3.6% between teh actual and predicted R values.

4/6/17 3:39 AM C:\Users\akroo\Documents\...\excercise5.m 1 of 1

```
%To initialize run the following line to set up ros's connection to matlab
% rosinit('10.0.75.2',11311,'NodeHost','10.0.75.1')
%--------------------------------------------------------------------------%
pub = rospublisher('/raw_vel');          %set up publisher for velocity to be written
to later
msg = rosmessage(pub);                   %establishes message object containing
publisher
d = 0.24765;                             %distance between wheels in meters
w = pi/5;                                %angular velocities
v = d*w;                                 %linear velocity
vR = w*((v/w)+(d/2));                    %right wheel velocity
vL = w*((v/w)-(d/2));                    %left wheel velocity
endTime = 2*pi/w;                        %time it theoretically takes to complete one
circle
msg.Data = [vL, vR];                     %writes velocities to message data object
send(pub, msg);                          %sends velocities to wheels
tStart = tic();                          %establishes start time of while loop
elapsed = 0;                             %establish elapsed
while elapsed <= endTime
    elapsed = toc(tStart);               %check to determine whether to stop neto
end
msg.Data = [0,0];                        %writes zero velocities to data message
object neto
send(pub, msg);                          %sends zero velocities to robot
%rosshutdown
```

Figure 5: Code to make robot go in a circle for the experiment outlined in exercise 4.

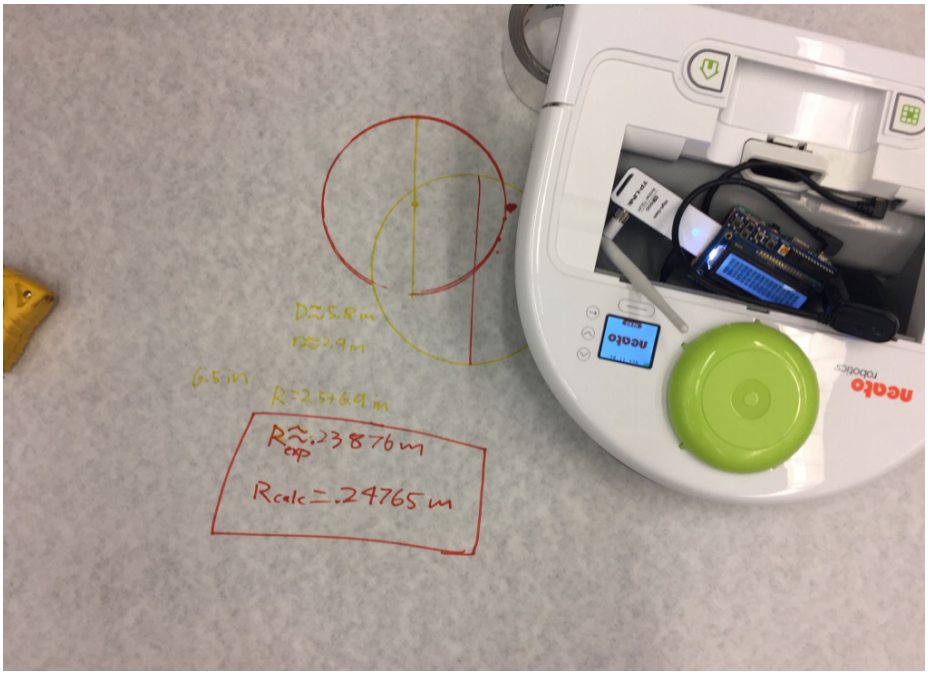

Figure 6: Predicted R vs Actual R

To find the actual R value we attached a marker to our robot so that it drew a circle, which we could then use to find the distance to the center of the path, in our case a circle, traced.

All of our code for this project can be found at the git hub repository: https://github.com/anniekroo/BridgeOfDeath

6 Exercise 6

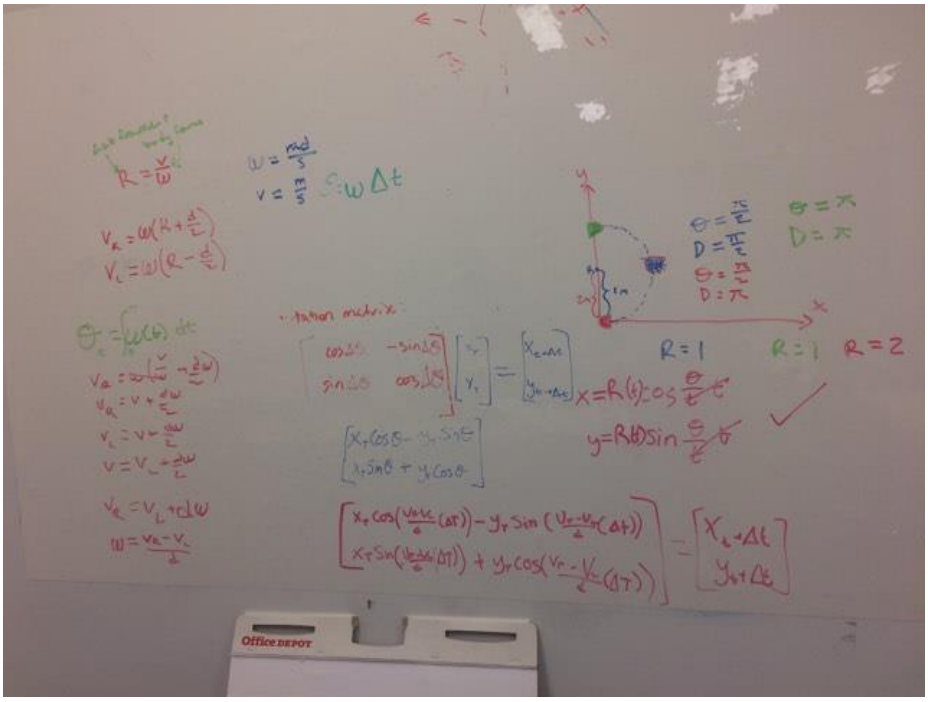

Figure 7: Creating a new position vector of the robot given a change in time.

To get a new x and y position from a starting location, we began by finding ω in terms of the left and right wheel velocities for the neato robot.

$$\omega = \frac{V_R - V_L}{d}$$

We then rotated our frame of reference and made the change in position in that direction equal to velocity multiplied by the change in time with the the following operation:

$$\begin{pmatrix} x_{t+\Delta t} \\ y_{t+\Delta t} \end{pmatrix} = v\Delta t \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix} \begin{pmatrix} x_t \\ y_t \end{pmatrix}$$

Using the equation

$$\theta = \omega\Delta t$$

we substituted our previously derived ω equation into this equation for θ. We then replaced all θ in the rotational matrix with this new substitution.

7 Exercise 7

4/6/17 3:39 AM C:\Users\akroo\Documents\...\excercise7.m 1 of 2

```
%To initialize run the following line to set up ros's connection to matlab
% rosinit('10.0.75.2',11311,'NodeHost','10.0.75.1')
%--------------------------------------------------------------------------%
pub = rospublisher('/raw_vel');          %set up publisher for velocity to be written
to later
msg = rosmessage(pub);                   %establishes message object containing
publisher
d = 0.24765;                             %distance between wheels in meters
w = pi/5;                                %angular velocities
v = d*w;                                 %linear velocity
vR = w*((v/w)+(d/2));                    %right wheel velocity
vL = w*((v/w)-(d/2));                    %left wheel velocity
endTime = 2*pi/w;                        %time it theoretically takes to complete one
circle
msg.Data = [vL, vR];                     %writes velocities to message data object
send(pub, msg);                          %sends velocities to wheels
sub_enc = rossubscriber('/encoders');    %set up subscriber for encoders to be written
to later
tStart = tic();                          %creating start time
elapsed = 0;
endTime = 10;                            %run time
pos = [];
sampleTime = .1;                         %time step to change velocities at
while elapsed <= endTime
    for e = 1:(endTime*(1/sampleTime))
        a = tic();
        pos = [pos, sub_enc.LatestMessage.Data]; %creating matrix of positions
        pause(sampleTime-(toc(a)));
    end
    elapsed = toc(tStart);
end
msg.Data = [0,0];                        %stopping robot after end time
send(pub, msg);
%--------------------------------------------------------------------------%
%POST PROCESSING
vR = pos(1,:);                           %
vL = pos(2,:);
w = (pos(1,:)-pos(2,:))./0.24765;
dD = [];
for i = 2:(size(vR, 2))
    dD = [dD, pos(:,i)-pos(:,i-1)]; %difference of positions
end
v = dD./sampleTime;
x = [];
y = [];
oldTurn = 0;
for i = 1:size(v, 2)
    newTurn = w(i)*sampleTime + oldTurn;    %calculating new theta
    x = [x, v(:,i).*cos(newTurn)];          %calculating new position x
```

4/6/17 3:39 AM C:\Users\akroo\Documents\...\excercise7.m 2 of 2

```
    y = [y, v(:,i).*sin(newTurn)];          %calculating new position y
    oldTurn = newTurn;
end
clf
plot(x(2,:),y(2,:), '.')                    %plotting right wheel
hold on
plot(x(1,:),y(1,:), '.')                    %plotting left wheel
%rosshutdown
```

Figure 8: Code to make robot go in a circle and plot location based on encoder values.

8 Exercise 8

4/6/17 3:39 AM C:\Users\akroo\Documents\...\excercise8.m 1 of 2

```
%To initialize run the following line to set up ros's connection to matlab
% rosinit('10.0.75.2',11311,'NodeHost','10.0.75.1')
%--------------------------------------------------------------------------%
syms u;
% encode the fact that u is a real number (allows simplifications)
assume(u,'real');
% create a symbolic expression for ellipse
R = sym([.5*cos(u), .75*sin(u), 0]);

% compute the tangent vector
T = diff(R);
% compute That.  Simplify will make sure things are in a sane form.
That = simplify(T ./ norm(T));
N = simplify(diff(That));
Bhat = simplify(cross(That, N));
%--------------------------------------------------------------------------%
pub = rospublisher('/raw_vel'); %set up publisher for velocity to be written to later
msg = rosmessage(pub);          %set up message of publisher for velocity
d = 0.24765;                    %m
w = Bhat(3);                    %equating Bhat to angular velocity
v = simplify(norm(T));          %theoretical R of d

vR = w.*((v./w)+(d./2));        % Equations for the wheels of the neto robot
vL = w.*((v./w)-(d./2));        % given angular velocity and linear velocity
endTime = 17;                   % stops robot after set amount of time
timeStep = .1;                  % determines how often we change our velocities

tStart = tic();                 % starts timer counting how long the program has been
running
while elapsed <= endTime        % checks that the program has been running for less
than endTime
    for i = 0:timeStep:endTime  % runs through each time step and writes to neto
        startLoopTime = tic();  % establishes start time for loop to check how long
it takes to run loop
        u = i/4.5;              % creates a u value to be substituted into symbolic
function
        instVR = (double (subs(vR)))/4.5;   % creates instantaneous velocity for
right wheel
        instVL = (double (subs(vL)))/4.5;   % creates instantaneous velocity for
right wheel
        msg.Data = [instVL, instVR];        % writes data to data object in message
        send(pub, msg);                     % sends message to neto
        pause(timeStep - toc(startLoopTime))% delays re-running the loop for timeStep
seconds from the start of the loop
    end
    elapsed = toc(tStart);                  %recreates elapsed to check whether or
not the while loop should end
end
```

4/6/17 3:39 AM C:\Users\akroo\Documents\...\excercise8.m 2 of 2

```
msg.Data = [0,0];               % writes 0 velocities to data object of message
send(pub, msg);                 % sends 0 velocities to neto
%rosshutdown
```

Figure 9: Code to make robot go in a circle and plot location based on encoder values.

9 Exercise 9

To traverse the bridge of death our Neato followed the curve

$$r(u) = -2a((l - cos(u))cos(u) + (1 - l))\hat{\imath} + 2a(l - cos(u))sin(u)\hat{\jmath}(a = 0.4, l = 0.4)$$

as seen in the figure below.

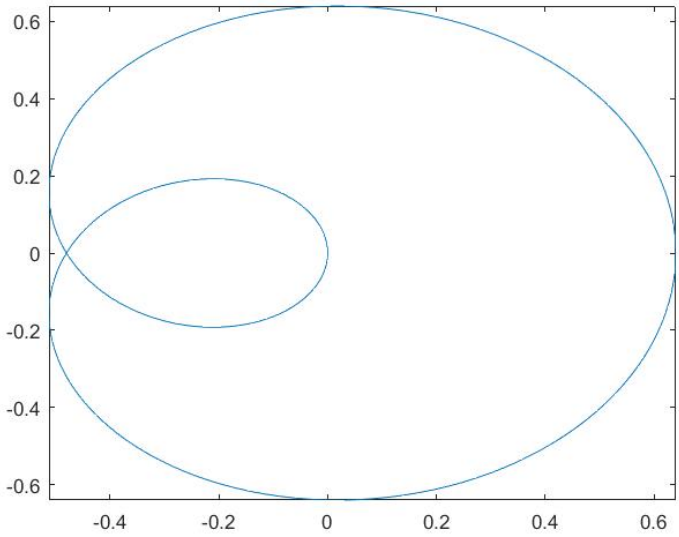

Figure 10: The expected path of the Neato.

```
%To initialize run the following line to set up ros's connection to matlab
% rosinit('10.0.75.2',11311,'NodeHost','10.0.75.1')
%--------------------------------------------------------------------------%
syms u;
% encode the fact that u is a real number (allows simplifications)
assume(u,'real');
a = .4;
l = .4;
% create a symbolic expression for the curve of the bridge of death
R = sym([-2.*(a).*((l - cos(u)).*cos(u) + (1 - l)); 2.*a.*(l - cos(u)).* sin(u); 0]);

% compute the tangent vector
T = diff(R);
% compute That.  Simplify will make sure things are in a sane form.
That = simplify(T ./ norm(T));
N = simplify(diff(That));
Bhat = simplify(cross(That, N));
%--------------------------------------------------------------------------%
pub = rospublisher('/raw_vel'); %set up publisher for velocity to be written to later
msg = rosmessage(pub);          %set up message of publisher for velocity
d = 0.24765;                    %m
w = Bhat(3);                    %equating Bhat to angular velocity
v = simplify(norm(T));          %theoretical R of d

vR = w.*((v./w)+(d./2));        % Equations for the wheels of the neto robot
vL = w.*((v./w)-(d./2));        % given angular velocity and linear velocity
endTime = 17;                   % stops robot after set amount of time
timeStep = .1;                  % determines how often we change our velocities

tStart = tic();                 % starts timer counting how long the program has been
running
while elapsed <= endTime        % checks that the program has been running for less
than endTime
    for i = 0:timeStep:endTime  % runs through each time step and writes to neto
        startLoopTime = tic();  % establishes start time for loop to check how long
it takes to run loop
        u = i/4.5;              % creates a u value to be substituted into symbolic
function
        instVR = (double (subs(vR)))/4.5;   % creates instantaneous velocity for
right wheel
        instVL = (double (subs(vL)))/4.5;   % creates instantaneous velocity for
right wheel
        msg.Data = [instVL, instVR];        % writes data to data object in message
        send(pub, msg);                     % sends message to neto
        pause(timeStep - toc(startLoopTime)% delays re-running the loop for timeStep
seconds from the start of the loop
    end
    elapsed = toc(tStart);                  %recreates elapsed to check whether or
not the while loop should end
```



```
end

msg.Data = [0,0];               % writes 0 velocities to data object of message
send(pub, msg);                 % sends 0 velocities to neto
%rosshutdown
```

Figure 11: Code to make robot go in a circle and plot location based on encoder values.

A video of our robot traversing the bridge of death can be found at: https://www.youtube.com/watch?v=6rNPUTRn2qAfeature=youtu.be

10 General Process

In order to create a program that drives a Neato, we integrated our knowledge of degrees of freedom, changing reference frames, and concepts of tangent and normal velocity vectors. We began by coming up with equations relating the velocities that we would write to our left and right wheels to the overall angular and linear velocities of our robot. To do this we related linear(v) and angular(ω) velocity to the location about which a robot turns. From there we related the velocity components to the wheel velocities with the equation below where d is the distance between the wheels of the neato, ω is the angular velocity, v is the linear velocity, and R is the point about which the robot turns and can be described by $R = \frac{v}{\omega}$.

$$V_R = \omega(R + \frac{d}{2})$$

$$V_L = \omega(R - \frac{d}{2})$$

To verify our equations we teamed up with another group to prove our two key assumptions. In our experiment we proved that R, the point at which the robot turns about, is equal to linear velocity over angular velocity. To do this, we calculated an R value for a path that we programed into our robot and compared the theoretical R value to the actual R value. To find the actual R value we attached a marker to our robot so that it drew a circle, which we could then use to find the distance to the center of the path, in our case a circle, traced. Our error for this experiment was 3.6%.The other group meanwhile validated omega by estimating the time it would take to make a complete circle and determining if the robot did complete a circle in a calculated time. This too yielded a positive result.

Although we considered how to use matrix multiplication to shift our frames of reference and the applicability of this to the use of encoders to more accurately pinpoint the location of our neato, we did not use this method in our final code.

To have our neato cross the bridge of death, we combined the idea of change of reference frame with the equations that we generated to define the left and right wheel velocities. To do this we wrote a program that found the angular velocity as a function of time by taking the cross product of the tangent unit vector and the normal vector. With the angular velocity, we were able to find the left and right wheel velocities as a function of time. Finally, we were able to use a for loop to step through time and write wheel velocities to the neato thus causing it to follow the curve.