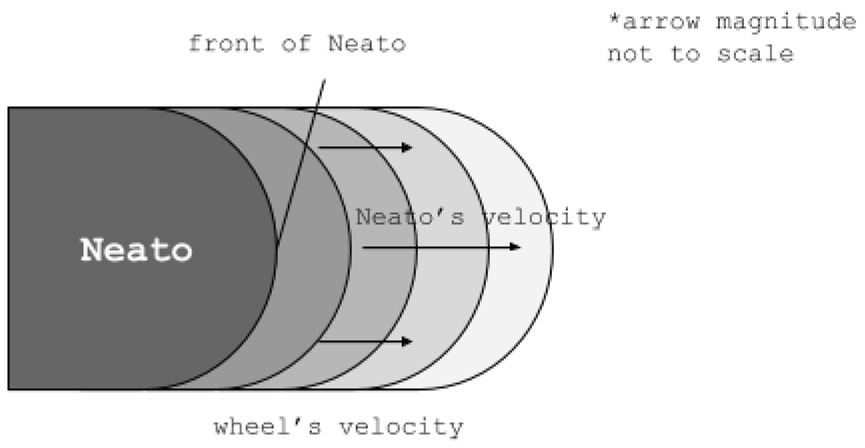
Exercise I

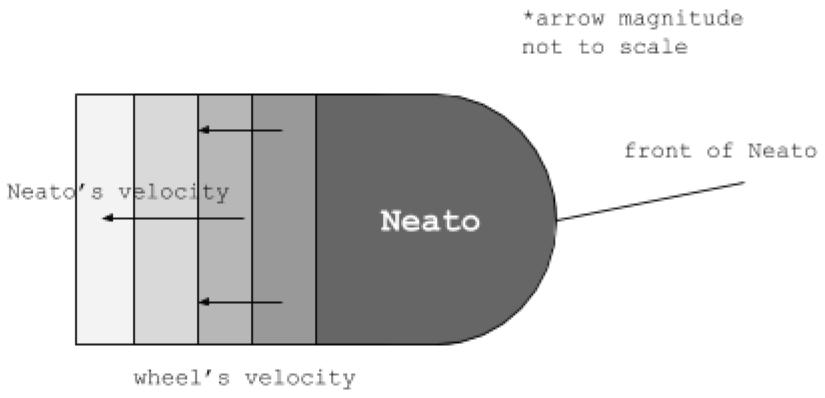
What if both wheels move forward (positive velocity) with equal speed?

The Neato would move forward (in respect with its position).



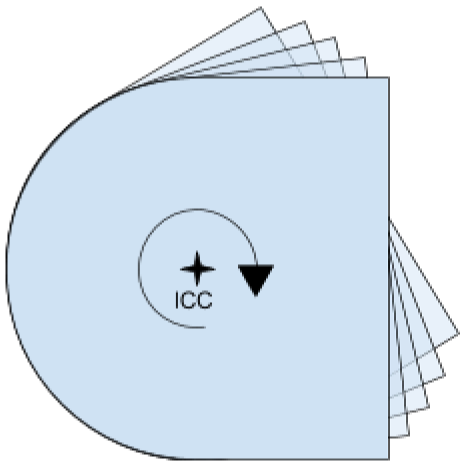
What if both wheels move backward (negative velocity) with equal speed?

The Neato would move backwards (in respect with its position).



What if one wheel drives forward and the other moves backward with equal speed?

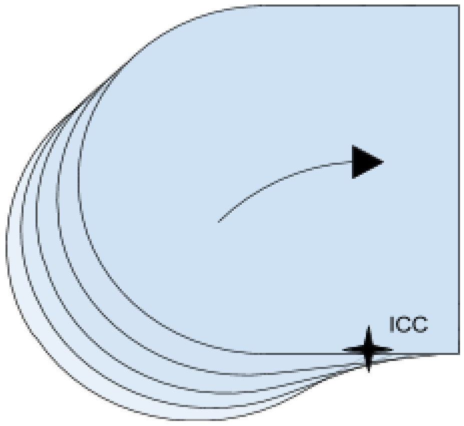
The Neato will rotate "in place," around the point halfway between the two wheels.



R = 0

What if one wheel drives forward while the other remains stationary?

The Neato will rotate forward around the stationary wheel.



R = -d/2

Exercise II

If $V_L = - V_R$ the robot will rotate directly about its center (what is the value of R now?).

Zero. The robot is moving around its center (making it the center of curvature).

What about when $V_L = V_R$?

There isn't one. The curve has no center. (Or, perhaps, $R = \infty$ or $R = -\infty$, for a curve of infinite radius.

For the diagrams you drew in Exercise 1, draw the ICC and describe the value of R qualitatively.

When the Neato moves forward and backwards (the first two questions in Exercise 1) there is no center of curvature.
The second two pictures/cases have the ICC labelled on them.

Exercise III

Assuming no wheel slippage, derive expressions for V_L , V_R , and R in terms of V (the linear velocity of the robot measured at its center), *d* (the robot's wheel base), and *ω* (the angular velocity of the robot about its ICC). As a first sanity check, make sure the units in your solutions make sense. As a second sanity check, make sure that the values of R predicted from your equations make sense for the special cases you worked out in the previous exercise.

$$\frac{V_L+V_R}{2}=V$$
$$\frac{V_L-V_R}{d}=\omega$$
$$V_L=2V-V_R$$
$$2V-V_R-V_R=2(V-V_R)=\omega d$$
$$V_R=V-\frac{1}{2}\omega d$$
$$V_L=2V-(V-\frac{1}{2}\omega d)=V+\frac{1}{2}\omega d$$
$$R=\frac{V}{\omega}$$

Exercise IV

Experiment 1: linear speed

What is the goal of your experiment?

The goal of the experiment is to determine the difference between the programmed/expected velocity behavior and the actual demonstrated velocity.

Describe the experimental procedure. What motor commands will you send to the robot?

The motor commands would tell the robot to move in a circle at a speed that we calculated from the equation/possible even some random arbitrary speed that the Neato could do (though the former would be the better option).

What will you measure?

We will measure the positions and speeds which the Neato's internal wheel encoders report.

How will the measurements help you achieve the goals enumerated in the first question?

Assuming that in Matlab, we have an accurate portrayal of the expected instantaneous and average velocity value, the comparison between the real measurements and the expected measurements would give us an idea of how the robot's behavior differs from the programmed behavior.

Are there any weaknesses in your proposed experiment? Another way to think about it is, are there any major assumptions you have to make in order to carry out the experiment?

This is assuming that the Matlab implementation is accurate in determining instantaneous and average velocity and also that we could get fairly accurate instantaneous and average velocity values from the Neato's encoders.

Experiment 2: rotation

What is the goal of your experiment?

To measure the accuracy of the Neato's motors, as well as the effective distance between the wheels (i.e. the distance which we should put in our equations to model the way it actually moves).

Describe the experimental procedure. What motor commands will you send to the robot?

We will tell the robot to turn on a dime, by driving the left wheel forward and the right backward, both at 50% speed.

What will you measure?

We will measure the Neato's speed of rotation using a digital gyroscope attached firmly (read: not duct tape) to the top of the robot.

How will the measurements help you achieve the goals enumerated in the first question?

If the measured speed of rotation is significantly different from the predicted one, we can adjust our numbers to compensate. Combined with the first experiment, this will give us both the motors' real speeds and the effective distance between them.

Exercise V

Two experiments were run. In the first, the Neato was told to drive forward at a set speed (0.1 m/s, then 0.2 m/s) until the wheel encoders read a certain distance (2m). The time which this took was recorded. This way, we can measure the difference between the speed which we give the Neato and the speed at which it moves.

Here, the speed discrepancy varied with speed. The wheels appeared about 1.6% too slow at 0.1 m/s, and about 3.0% too slow at 0.2 m/s. However, the discrepancy actually makes more sense as a constant time -- about 0.3s -- which is probably a latency of some sort. This lines up with the fact that the Neato's wheels push harder when they are given more resistance, indicating that the encoders are used to adjust speed and so the speed is likely as accurate as the encoders.

The second had the Neato drive in a circle, and then after a number of revolutions (3 or 5) we stopped it and measured the wheel encoder distances. This, combined with the first experiment, gives us the information we need to determine the distance between the Neato's wheels. The distance it returned was 0.253m, which is very close to the 0.24m and 0.25m which other teams have used.

Exercise VI

To find the new theta and the new coordinate points:

$V_L = V - \omega\frac{d}{2}$

$V_R = V + \omega\frac{d}{2}$

$R = \frac{V}{\omega}$

$\theta = \omega\Delta t + \theta_0$

[x'; y'] = [cos θ , - sin θ ; sin θ , cos θ] [x; y]

(subtract the coordinates of the ICC from x and y, and then add it back to x' and y')

To find the coordinates for the instantaneous center of curvature:

$x_{t+\Delta t} = x' + x_{ICC}$

$y_{t+\Delta t} = y' + y_{ICC}$

$\theta = \omega\Delta t$

$x_{ICC} = x_t - R sin\theta$

$y_{ICC} = y_t + R cos\theta$

Exercise VII

To find x and y coordinates:

$\Delta P = \frac{\Delta P_L + \Delta P_R}{2}$

$\Delta x = \Delta P\, cos\theta$

$\Delta y = \Delta P\, sin\theta$

To find velocities:

$\Delta P = V\,\Delta t$

$\Delta P_L = V_L \Delta t$

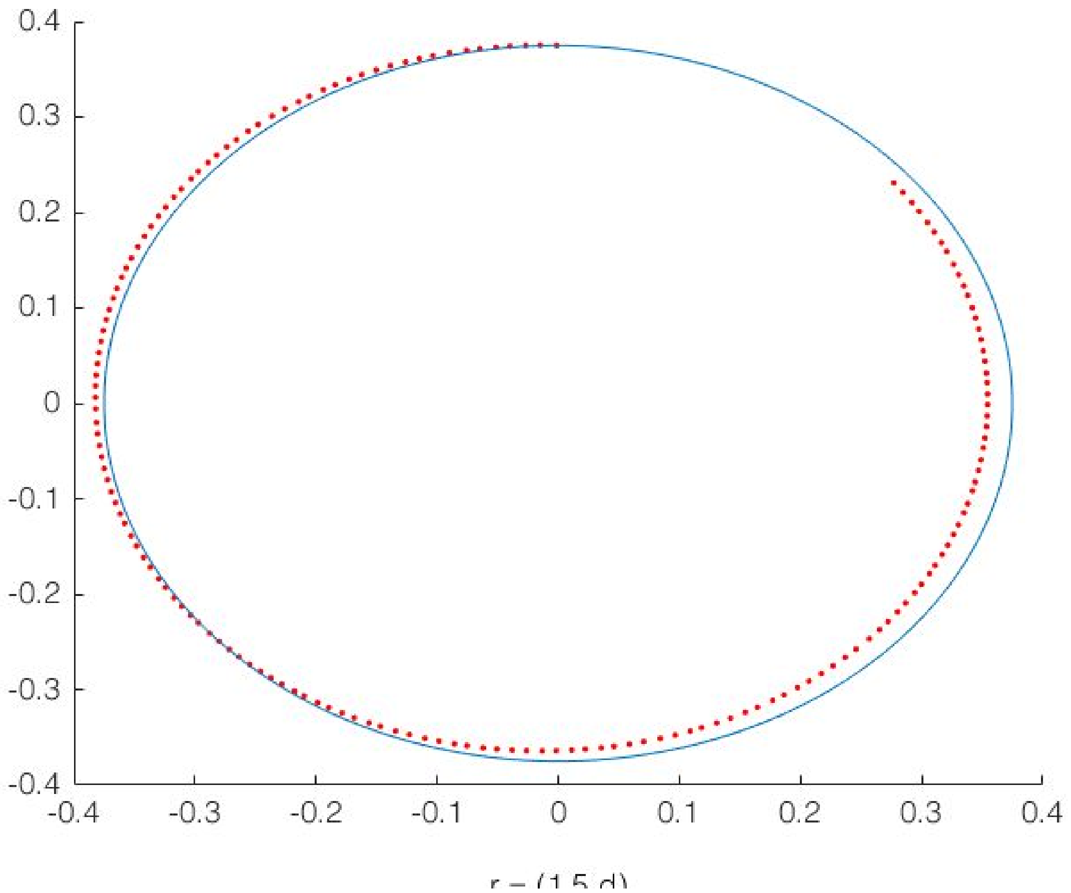
$\Delta P_R = V_R \Delta t$

$V = \frac{V_L + V_R}{2}$

To find theta:

$\theta = \omega \Delta t = \frac{\Delta P_R - \Delta P_L}{d} \Delta t$

Here is a plot of predicted versus actual position:



Exercise VIII

Code is attached.

Exercise IX

Code is attached.
Video link:
https://www.youtube.com/watch?v=zdBVneWNoCk

Prava Dhulipalla and Adam Selker
7 April 2017
QEA I
Module 3: Robots

Bridge of Death: Write-Up

A description of your general process. What strategies did you try? What worked? What didn't? We are looking for something relatively comprehensive. A good length would be about a page.

The objective of the "Bridge of Death" assignment was to program a Neato robot to follow a parametric curve and to hopefully cross a cardioid "bridge of death." Although the bridge had some allowance -- a few centimeters on each side -- precision was still required. Either the code would successfully send the NEATO across the bridge of death or it would not, sending the poor NEATO to it's death (aka hopefully into the hands of an awaiting QEA student but in the more unfortunate cases, the floor).

The centerline of the Bridge follows a cardioid of the equation:
[x,y] = 2a * (l-cos(u)) * [cos(u), sin(u)]
Where t is the equation's parameter, a = 0.4, and l = 0.4.

The only thing that the QEA students were given was the equation of the cardioid that the bridge followed. The only signal that we could send to the NEATO was the velocities for the left and the right wheel. In order to complete the objective, we would have to use the equations to find values for the velocity of the left and right wheel, that changes by a factor of time.

The equations for the left and right wheel were:
$V_L = V - \omega\frac{d}{2}$
$V_R = V + \omega\frac{d}{2}$
Where V_L is the velocity of the left wheel of the NEATO, V_R is the velocity of the right wheel of the NEATO, V is the linear velocity of the NEATO, ω is the angular velocity of the NEATO, and d is the distance between the wheel base.

Now that the equations for the velocities of the left and right wheel were determined, we had to determine the relationship between the cardioid equation and the respective velocities. We also had to determine the value of d for the NEATO.

To determine the value of *d*, we told the robot to drive with one wheel set to twice the speed of the other. This caused it to drive in a circle where the inside wheel tracked τd meters and the outside wheel tracked $2\tau d$ meters. After driving in a total of 14 circles and dividing both wheel odometers by 14, these simple formulae were used to calculate *d.* The number was accurate enough to require no adjustment to succeed on the Bridge of Death.

Then, to determine the speed and angular velocity, we found the normalized tangential equation and the unnormalized normal equation for the cardioid equation. From these, we could find equations for the linear speed and the angular velocity that was dependant on time.

Plugging in these values into the equation and varying it by a small time step, we were able to get the robot to follow the parametric path. However, this way wasn't the only way we considered. The NEATOs have a variety of sensors that we could have used if we didn't want to base it solely on the math. For instance, we could have used edge detectors in order to get the NEATO to avoid the side edges of the bridge, making it so it would follow the path. However, we did not implement this due to learning goals: the goal of the assignment was learning how to use a parametric equation to send commands for the speeds of the wheels, and using feedback from the wheels would have transformed the problem from that of mathematics into something approaching controls engineering. As interesting as this would have been, it was not our goal, and would have been more appropriate for the next unit. In addition, the motors already use the encoders to regulate their speeds, so the gains would have been minimal.

Exercise 8

```
syms t;

% encode the fact that u is a real number (allows simplifications)
assume(t,'real');

a = 0.4;
l = 0.4;
d = 0.254;

speedmult = 0.1;

% create a symbolic expression for an ellipse
R = sym([1/2 * cos(t*speedmult); 3/4 * sin(t*speedmult); 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));
Nhat = simplify(diff(That)./norm(diff(That)));
Bhat = simplify(cross(That, Nhat));

V = simplify(norm(T));
N = simplify(diff(That));
angularV = simplify(cross(That, N));

% make a plot
%hold on;
%fplot3(R(1), R(2), R(3), [0, 2*pi]);
%hold off;

pub = rospublisher('/raw_vel'); %Sender
msgOut = rosmessage(pub);
sub_bump = rossubscriber('/bump');

start = tic;
while true

        t = toc(start);

        omega = double(subs(angularV));
        omega = omega(3);
        speed = double(subs(V));
        Dp = speed * t;

        Vl = speed - omega * d/2;
        Vr = speed + omega * d/2;

        msgOut.Data = [double(Vl), double(Vr)];
        [double(Vl), double(Vr)]
        send(pub, msgOut);


        bumpMessage = receive(sub_bump);
        if any(bumpMessage.Data)
        msgOut.Data = [0,0];
        send(pub, msgOut);
        break
        end
        pause(0.1);

end
```

Exercise 9

```
syms t;

% encode the fact that u is a real number (allows simplifications)
assume(t,'real');

a = 0.4;
l = 0.4;
d = 0.254;

speedmult = 0.1;

% create a symbolic expression for a cardioid
R = sym([-2*a*((l - cos(t*speedmult))*cos(t*speedmult) + (1 - l)); 2*a*(l -
cos(t*speedmult))*sin(t*speedmult); 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));
Nhat = simplify(diff(That)./norm(diff(That)));
Bhat = simplify(cross(That, Nhat));
```

```
V = simplify(norm(T));
N = simplify(diff(That));
angularV = simplify(cross(That, N));

% make a plot
%hold on;
%fplot3(R(1), R(2), R(3), [0, 2*pi]);
%hold off;

pub = rospublisher('/raw_vel'); %Sender
msgOut = rosmessage(pub);
sub_bump = rossubscriber('/bump');

start = tic;
while true

        t = toc(start);

        omega = double(subs(angularV));
        omega = omega(3);
        speed = double(subs(V));

        Dp = speed * t;

        Vl = speed - omega * d/2;
        Vr = speed + omega * d/2;

        msgOut.Data = [double(Vl), double(Vr)];
        [double(Vl), double(Vr)]
        send(pub, msgOut);


        bumpMessage = receive(sub_bump);
        if any(bumpMessage.Data)
        msgOut.Data = [0,0];
        send(pub, msgOut);
        break
        end
        pause(0.1);

end
```