Night 2: Angular Velocity, NEATOs, and Partial Derivatives

Quantitative Engineering Analysis

Spring 2019

1 *Learning Goals*

By the end of this assignment, you should feel confident with the following:

- Finding angular velocity from a parametric curve.
- Interpreting and writing basic programs for the NEATO.
- Partial Derivatives, and the Chain Rule.

2 *Angular Velocity Revisited [3 hours]*

Suppose we have a 2D parametric curve $\mathbf{r}(t) = f(t)\hat{\mathbf{i}} + g(t)\hat{\mathbf{j}}$. We saw in the Night 1 assignment that the linear velocity vector is given by $\mathbf{r}'(t) = f'(t)\hat{\mathbf{i}} + g'(t)\hat{\mathbf{j}}$.

Determining the expression for the angular velocity $\omega(t)$ of our robot is more involved. Before we derive the correct expression, we will introduce the notion of angular velocity vectors. While expressing angular velocity as a vector may at first seem overly complex (since in this case we know that the robot will rotate about the z-axis, and it seems like we should just be able to compute the scalar magnitude along with whether to turn clockwise or counterclockwise), thinking about the angular velocity as a vector will enable our derivation to be done in a much more straightforward and generalizable manner.

Angular velocity vectors point in the direction about which the body rotates (which for our robot will be along either the positive or negative z-axis). For right-handed coordinate systems (such as the one we are using here), by convention, a positive rotation happens counterclockwise about the direction of the rotation axis. Further, the magnitude of the angular velocity vector indicates the angular speed.

In the problem at the beginning of class on Monday we discussed the coordinate system attached to the robot: the body fixed frame. Because the heading of the robot is locked to the tangent vector $\hat{\mathbf{T}}$ of the curve, we can think of the vector $\hat{\mathbf{T}}$ as being a constant in the body fixed frame of the robot. The body fixed frame is rotating with some unknown angular velocity vector $\boldsymbol{\omega}$ with respect to the room coordinate system. If we wish to know the time derivative of the

tangent vector $\hat{\mathbf{T}} =$ in the room coordinate system, we can use the generalized relationship between the time derivatives of vectors in two coordinate systems which are rotating with an angular velocity vector $\boldsymbol{\omega}$ with respect to each other. This expression is

$$\frac{d\hat{\mathbf{T}}}{dt}|_{room} = \frac{d\hat{\mathbf{T}}}{dt}|_{body} + \boldsymbol{\omega} \times \hat{\mathbf{T}} \tag{1}$$

(Full mathematical derivation here; nice heuristic explanation here.) In the body frame of the robot, $\hat{\mathbf{T}}$ is unchanging, since it is always aligned with the forward direction, so the term $\frac{d\hat{\mathbf{T}}}{dt}|_{body}$ is zero leaving us with

$$\frac{d\hat{\mathbf{T}}}{dt}|_{room} = \boldsymbol{\omega} \times \hat{\mathbf{T}} \tag{2}$$

Then we make use of the scalar triple product, which states that $\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = \mathbf{b}(\mathbf{a} \cdot \mathbf{c}) - \mathbf{c}(\mathbf{a} \cdot \mathbf{b})$. Using this we can derive our angular velocity vector as follows

$$\begin{aligned}
\frac{d\hat{\mathbf{T}}}{dt}|_{room} &= \boldsymbol{\omega} \times \hat{\mathbf{T}} \\
\hat{\mathbf{T}} \times \frac{d\hat{\mathbf{T}}}{dt}|_{room} &= \hat{\mathbf{T}} \times \boldsymbol{\omega} \times \hat{\mathbf{T}} \\
&= \boldsymbol{\omega}(\hat{\mathbf{T}} \cdot \hat{\mathbf{T}}) - \hat{\mathbf{T}}(\hat{\mathbf{T}} \cdot \boldsymbol{\omega}) \\
&= \boldsymbol{\omega}(1) - \hat{\mathbf{T}}(0) \\
&= \boldsymbol{\omega} \\
\Rightarrow \boldsymbol{\omega} &= \hat{\mathbf{T}} \times \frac{d\hat{\mathbf{T}}}{dt}|_{room}
\end{aligned} \tag{3}$$

The x and y components of the angular velocity vector will always be zero because $\hat{\mathbf{T}}$ and $\frac{d\hat{\mathbf{T}}}{dt}|_{room}$ are in the x-y plane and orthogonal. The magnitude of the z-component is the angular speed. If the z-component is positive, we turn counterclockwise at that speed. When it is negative, we turn clockwise at that speed.

Exercise (1) In the Night 1 assignment you found the unit tangent and normal vectors for various paramaterized curves. We will use that information to find linear and angular velocities, then translate those to left and right wheel velocities for the NEATO.

The vector for a circle is given by:

$$\mathbf{r}(t) = R\cos\alpha t\ \hat{\mathbf{i}} + R\sin\alpha t\ \hat{\mathbf{j}},\ \alpha t \in [0, 2\pi]$$

Mathematica hints:

- You can use the apostrophe to take a derivative
- Specify that t and α are real

- Include assumptions about coefficients being positive (however, **note that α does not have to be positive**)
- Use the square root of the dot product rather than `Norm`, especially if you're getting a bunch of absolute values in your answer
- use `Simplify`

(a) What are the linear velocity vector and linear speed?

(b) What is the unit tangent vector for the circle?

(c) What is the unit normal vector?

(d) What is the angular velocity vector?

(e) For the uniform circular motion we have been investigating so far, what does the parameter we have labeled α represent? How is it related to the time it takes to complete one traverse of the circular trajectory?

(f) How would you modify this equation for a circle of radius 1 m?

(g) What value would you choose for α if you want your robot to complete the circle in 30 seconds?

(h) What are the equations for the left and right wheel velocities for the uniform circle? (d=wheel displacement)

(i) What are the left and right wheel velocities needed for a 1 m radius counterclockwise circle to be completed in 30 seconds?

Exercise (2) The vector for an ellipse is given by:

$$\mathbf{r}(t) = a\cos\alpha t\ \hat{\mathbf{i}} + b\sin\alpha t\ \hat{\mathbf{j}},\ \alpha t \in [0, 2\pi]$$

(a) What is the unit tangent vector for the ellipse?

(b) What is the linear velocity vector? How does it differ from the example of the circle?

(c) What is the unit normal vector?

(d) What is the angular velocity vector? How does it differ from the circle?

(e) What are the left and right wheel velocities?

(f) Plot the linear velocity vector as a function of time for various combinations of the paramters a, b, and α.

(g) Plot the angular velocity vector as a function of time for various combinations of the paramters a, b, and α.

(h) Plot the left and right wheel velocities as a function of time for various combinations of the paramters a, b, and α.

3 *Fun with NEATOs [3 hours]*

In the previous section we found the left and right wheel velocities needed to drive a particular trajectory. In this section of the assignment, we will be thinking about how to translate the velocity vectors to a Matlab program that will control your NEATO.

The control of your NEATO is built on top of the Robotic Operating System (ROS), so you will be using ROS commands to control the velocity values for your robot. We will start by playing with some very basic commands, similar to what you did in class.

Consider the program below (You can download the code here):

```
1  function output = driveforward(distance, speed)
2  % driveforward is a simple function that controls the NEATO to drive straight forward
3  % at a designated speed for a set distance
4
5  %This line says we are going to publish to the topic `/raw_vel'
6  %which are the left and right wheel velocities
7  pubvel = rospublisher('/raw_vel')
8
9  %Here we are creating a ROS message
10 message = rosmessage(pubvel);
11
12 %in Matlab tic and toc start and stop a timer. In this program we are making sure we
13 %drive the desired distance by finding the necessary time based on speed
14 tic
15
16 %Set the right and left wheel velocities
17 message.Data = [speed, speed];
18
19 % Send the velocity commands to the NEATO
20 send(pubvel, message);
21 while 1
22     if toc > distance/speed % Here we are saying the if the elapsed time is greater than
23         %distance/speed, we have reached our desired distance and we should stop
24
25         message.Data = [0,0]; % set wheel velocities to zero if we have reached the desire distance
26         send(pubvel, message); % send new wheel velocities
27         break %leave this loop once we have reached the stopping time
28     end
29 end
30
```

This code snippet defines the function "driveforward" which will cause your NEATO to.... you guessed it, drive forward. The function definition is in line 1. You will notice that this function does not have a meaningful output, its sole purpose is to move your robot forward.

3.1 *The structure of a Simple Robot Program*

Let's break down what is happening in this program:

- The inputs to the function are the distance to drive, and the speed.
- Line 7 of the program specifies that you will be publishing to the ROS topic 'raw_vel'.
- Line 10 defines a ROS message which will be sent to the 'raw_vel' topic.
- In line 14, a timer is started using the "tic" command.
- In line 17, a one by two vector with the left and right wheel velocities is assigned to the Matlab structure "message.Data". In this program the velocities are defined at the input to the function, but they could also come from a velocity vector, pre-computed list, etc.
- In line 20, we use the "send" command to send the data in "message" to the ROS topic specified by "pubvel" (in this case, 'raw_vel')

  As soon as we use the "send" command, your robot will start moving according to the wheel velocities in 'message.Data', and will not stop until we tell it to. So, what is happening in the rest of this program? At this point, we are not using a distance sensor on the robot. We will introduce those in class on Thursday, but for now we are basing distance off of time and velocity.
- Line 21 starts a loop that basically monitors how long the robot has been moving forward.
- In line 22 we use the "toc" command to check how much time has elapsed since "tic" was sent. This is very close to the amount of time the robot has been moving. We use the simple fact that $distance = velocity \times time$ to find the elapsed time needed to travel the distance we specified in the function call. We say that if we have reached that maximum time, we send a zero velocity command in lines 25 and 26.
- If we have not reached the maximum elapsed time for our distance, we stay inside the loop and keep checking the time.

Exercise (3) Download the above program and open the m-file in Matlab. It is a function, so it can be called from the command window using the form:

```
output = driveforward(distance, speed)
```

Using this new function, try driving the NEATO for several combinations of distances and speeds. Mark the anticipated finish position on the floor with tape or check. Time your robot as it runs. Do the final distance and time match your expectations?

3.2 *Receiving Sensor Data*

In the previous example program we published to a ROS topic to set the NEATO wheel velocities. In ROS you can also subscribe to a topic to do things like receive sensor data. Download and open the program driveUntilBump in Matlab.

Exercise (4) In line 2 the 'rossubscriber' command is introduced. From the code, what sensor output are we monitoring?

Exercise (5) The variable 'bumpmessage' is a structure. What is the size of 'bumpmessage.data'? What do the values contained in that variable mean?

Exercise (6) What is the 'driveUntilBump' code commanding the robot to do?

(a) Test the 'driveUntilBump' code on a NEATO and verify that your interpretation is correct.

Exercise (7) Modify the 'driveUntilBump' code to make it a function where the robot velocity is an input.

Exercise (8) Using what you have learned from the examples above, write a program that meets the following requirements:

(a) The program commands the robot to drive a designated distance at a chosen speed, and stops when that distance is reached.

(b) If the bump sensor is triggered, the robot reverses direction and backs up for 5 seconds then stops.

Take a short video of your succesful robot, and include the code in your assignment.

Try developing this code on your own first, then if you get stuck, take a look at the program driveUntilBumpThenRunAway for inspiration.

4 *Partial Derivatives and the Chain Rule [2 hours]*

Now work your way through the following sections and problems from the book `Multivariable Calculus by Stewart`, and the book, `A First Course in Mathematical Modeling by Giordano, Fox, and Horton`. You can find pdf copies of the relevant sections on Canvas, subject to the fair use policy. The key material is contained in Section 14.6, although we include a couple of prior sections for completeness.

Exercise (9) Please read Section 14.3 from Stewart on Partial Derivatives. Take notes on important concepts and definitions.

Exercise (10) Please read Section 14.5 from Stewart on The Chain Rule. Take notes on important concepts and definitions.

This is new material on extending the notion of the chain rule from functions of one variable to functions of many variables. The main results are captured in the pink boxes labeled 1 through 4 - these are various cases of the chain rule. Again, this text is written for a student who doesn't have linear algebra. As you read these rules, think about how you might use matrix notation to make this cleaner and more compact. At this stage you should ignore the section on Implicit Derivatives - it will be too confusing and take too long.

Do the following exercises by hand and check your work in Mathematica.

Exercise (11) Complete question 1 from 14.5 Exercises.

Exercise (12) Complete question 5 from 14.5 Exercises.

Exercise (13) Complete question 11 from 14.5 Exercises.