Neato Path-Following Project
QEA1b Spring 2020

Jules Brettle

April 26, 2020

1 Introduction

This goal of this project is to create a MATLAB program that converts a parametric curve in two-dimensional space to a pair of left and right wheel velocities for a two-wheeled robotic Neato. My program can take in any two-dimensional parametric curve for path-following, but this paper will highlight the following path

$$\boldsymbol{r}(u) = 4 * [0.3960cos(5.65(bu + 1.4))\boldsymbol{i} - 0.99sin(bu + 1.4)\boldsymbol{j} + 0\boldsymbol{k}] \text{ where } bu \in [0, 3.2].$$

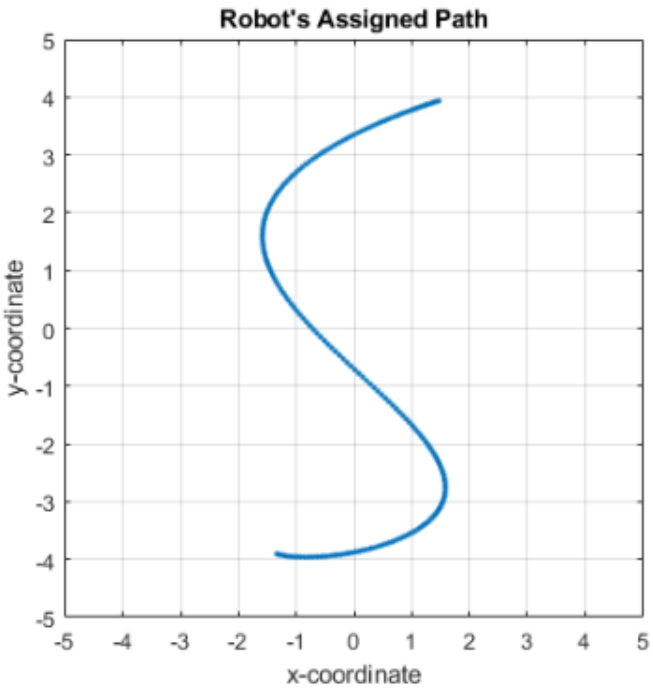

Figure 1: The path to be followed by the robot in this case, defined by the above parametric equation for $\boldsymbol{r}(t)$.

2 Method

The first step is to use the Symbolic Toolbox in MATLAB to define the equation for $\boldsymbol{r}(t)$. We can use the `diff()` function to find the derivative with respect to u and the equation $\hat{T} = \frac{\boldsymbol{r}'}{|\boldsymbol{r}|}$

```
r =
```

$$\left(\frac{198\cos\left(\frac{53\,b\,u}{20}+\frac{371}{100}\right)}{125} \quad -\frac{99\sin\left(b\,u+\frac{7}{5}\right)}{25} \quad 0\right)$$

```
dr=diff(r,u)
dr =
```

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

```
T_hat=dr./norm(dr);
T_hat=simplify(T_hat)
T_hat =
```

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

where

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

Figure 2: The MATLAB vector calculations for $\boldsymbol{r}$ and $\hat{T}$. The equations seem complicated (note the σ substitutions) but in the end we will use the subs() function to convert the vector into 1x200 double arrays representing motor output, so we don't have to deal with them directly.

We then use the magnitude of $\boldsymbol{r}'$ as the linear speed V along the path and $\hat{T} \times \hat{T}'$ as the angular velocity ω. We use these values to find the speeds of the left and right wheels by

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

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

```
linspeed = simplify(norm(dr))
linspeed =
```

$$\frac{99\,|b|\sqrt{2809\sin\left(\frac{53\,b\,u}{20}+\frac{371}{100}\right)^2+2500\cos\left(b\,u+\frac{7}{5}\right)^2}}{1250}$$

```
d = 0.235; % wheel base width, meters
ang_vel = simplify(cross(T_hat,diff(T_hat,u)))
ang_vel =
```

$$\left(0 \quad 0 \quad -\frac{265\,b\left(73\cos\left(\frac{33\,b\,u}{20}+\frac{231}{100}\right)+33\cos\left(\frac{73\,b\,u}{20}+\frac{511}{100}\right)\right)}{2\left(2500\cos\left(2\,b\,u+\frac{14}{5}\right)-2809\cos\left(\frac{53\,b\,u}{10}+\frac{371}{50}\right)+5309\right)}\right)$$

```
VL = simplify(linspeed - ang_vel(3)*d/2);
VR = simplify(linspeed + ang_vel(3)*d/2);
VL, VR
VL =
```

$$\frac{99\,|b|\sqrt{2809\sin\left(\frac{53\,b\,u}{20}+\frac{371}{100}\right)^2+2500\cos\left(b\,u+\frac{7}{5}\right)^2}}{1250}+\frac{2491\,b\left(73\cos\left(\frac{33\,b\,u}{20}+\frac{231}{100}\right)+33\cos\left(\frac{73\,b\,u}{20}+\frac{511}{100}\right)\right)}{160\left(2500\cos\left(2\,b\,u+\frac{14}{5}\right)-2809\cos\left(\frac{53\,b\,u}{10}+\frac{371}{50}\right)+5309\right)}$$

```
VR =
```

$$\frac{99\,|b|\sqrt{2809\sin\left(\frac{53\,b\,u}{20}+\frac{371}{100}\right)^2+2500\cos\left(b\,u+\frac{7}{5}\right)^2}}{1250}-\frac{2491\,b\left(73\cos\left(\frac{33\,b\,u}{20}+\frac{231}{100}\right)+33\cos\left(\frac{73\,b\,u}{20}+\frac{511}{100}\right)\right)}{160\left(2500\cos\left(2\,b\,u+\frac{14}{5}\right)-2809\cos\left(\frac{53\,b\,u}{10}+\frac{371}{50}\right)+5309\right)}$$

Figure 3: The MATLAB calculations for calculating linear speed V, angular velocity ω and the wheel velocities V_L and V_R.

Substituting for $d = 0.235$m (width of wheel base), $b = 0.1$ (to slow to attainable speeds for the Neato motors), and 200 evenly spaced values of u between 0 and 32 seconds gave 200 element long lists for the ideal left and right wheel velocities at the given timestamps (plotted in Figure 4).

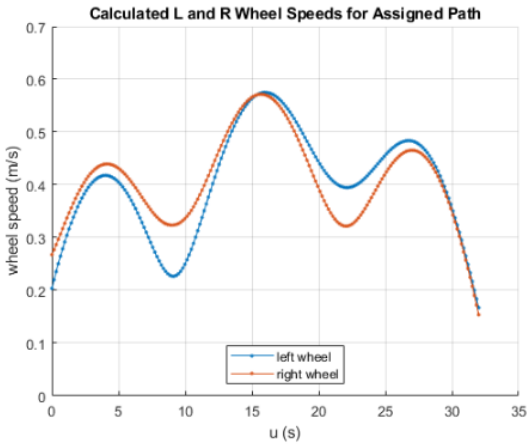

Figure 4: The 200 wheel velocity commands for each of the wheels.

The following code loops through the 200 wheel/motor speed command pairs and records the timestamp the command was actually given. The reasoning behind the `delay` variable (on lines 4 and 18) is given later in the Results section.

```
1  load vlvr.mat % loads u_num, VL_num, and VR_num 1x200 vectors
2  load commandsdata_delay.mat % loads non-adjusted command timestamps
3
4  delay = (commandsdata_delay(length(u_num),1) - u_num(length(u_num)))/length(u_num)
5      % = (non-adjusted time - ideal time)/200 steps. Gives a delay of about 0.0141s
6
7  step_len = u_num(2) - u_num(1);
8      % how long each motor command should run before sending the next set of
       commands
9  num_steps = length(u_num);
10
11 % Neato interface set-up
12 clear commandsdata;
13 pub = rospublisher('/raw_vel');
14 currTime = rostime('now');
15 start = double(currTime.Sec)+double(currTime.Nsec)*10^-9;
16
17 for i = 1:num_steps
18     driveStepLR(pub, step_len-delay,VL_num(i),VR_num(i));
19         % custom fuction that sends motor values for given period of time
20
21     % records timestamp and motor values sent for later plotting
22     currTime = rostime('now');
23     currTime = double(currTime.Sec)+double(currTime.Nsec)*10^-9;
24     elapsedTime = double(currTime - start);
25     commandsdata(i,:) = [elapsedTime VL_num(i) VR_num(i)];
26     commandsdata(i,:)
27 end
28 save('commandsdata.mat', 'commandsdata') % saves for later plotting
29 driveStepLR(pub, step_len,0,0); % stops robot
```

3 Results

When the aforementioned code was run on using the actual simulation with `delay` set to 0, the Neato would drift off course because each step would last slightly longer than it should due to computer processing/communication speed.

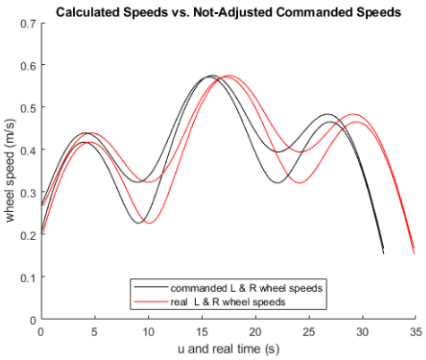

Figure 5: The idea wheel velocities (black) plotted against the actual timestamps recorded during a test. The inherent delays in communication with the Neato stretched the running time from the ideal 32 seconds to about 35 seconds. As the delay accumulates, the robot slowly drifts off-course.

To compensate for the communication delay, we command the motor to run each step for a slightly shorter period of time (adjusted by the variable `delay`) - calculated by determining the difference between the theoretical and actual endpoints and distributing that evenly across the 200 steps. This gives us a `delay` of about 0.0141 seconds.

With that modification, the Neato completes the course perfectly. The encoder data supports this, as the data taken from the encoders (minus some noise) lines up almost perfectly on the wheel speed values calculated in the Method section.

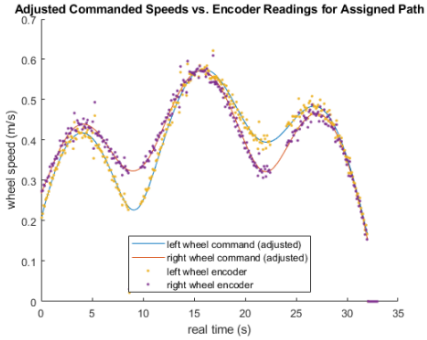

Figure 6: The encoder data from a completed run of the parametric-curve-following code.

4 External Links

A video of the simulated Neato following the path discussed in this paper can be found unlisted on YouTube at https://youtu.be/zKqgU-sNdm4

The full MATLAB code can be found on GitHub at https://github.com/julesbrettle/Neato-Path-Following