FTC 12566 Equations

Tyler M Silva

January 2019

Abstract

This is a document outlining the equations and mathematical details of the robot for the Aztechs Robotics 12566 FTC team. It will go over many complicated mathematical details and assumes a base knowledge of concepts including partial derivatives, functions in multiple dimensions, and piecewise functions. This robot has many subsystems, but only the holonomic X drive and the lift system have/had complicated equations to it, so that is what this packet will go over.

Contents

1 Driving

$$f(0,0) = [0,0,0,0] \tag{1}$$
$$f(1,1) = [0,-1,0,1] \tag{2}$$
$$f(1,0) = [-1,-1,1,1] \tag{3}$$
$$f(1,-1) = [-1,0,1,0] \tag{4}$$
$$f(0,-1) = [-1,1,1,-1] \tag{5}$$
$$f(-1,-1) = [0,1,0,-1] \tag{6}$$
$$f(-1,0) = [1,1,-1,-1] \tag{7}$$
$$f(-1,1) = [1,0,-1,0] \tag{8}$$
$$f(0,1) = [1,-1,-1,0] \tag{9}$$

This was the first challenge we wanted to overcome. We started by finding out the ideal motor outputs in each of the cardinal directions[1-9]. The first input value is the speed to move right, and the input second value is the speed to move forward, both in percents. After that, the next idea was to make an equation that aligns those points with all partial derivatives equal to 1, -1, or 0. we had a much easier time splitting it up into an equation for each motor (each index of the array)[10-13], using the format f_n, where n is the index of the motor that is being turned.

$$f_1(x,y) = \begin{cases} y - x & x \geq 0 \land y \geq 0 \\ \min(-x, y) & x \geq 0 \land y < 0 \\ \max(-x, y) & x < 0 \land y \geq 0 \\ -x + y & x < 0 \land y < 0 \end{cases} \tag{10}$$

$$f_2(x,y) = \begin{cases} \min(-x, -y) & x \geq 0 \land y \geq 0 \\ -y - x & x \geq 0 \land y < 0 \\ -x - y & x < 0 \land y \geq 0 \\ \max(-x, -y) & x < 0 \land y < 0 \end{cases} \tag{11}$$

$$f_3(x,y) = \begin{cases} x - y & x \geq 0 \land y \geq 0 \\ \max(x, -y) & x \geq 0 \land y < 0 \\ \min(x, -y) & x < 0 \land y \geq 0 \\ -y + x & x < 0 \land y < 0 \end{cases} \tag{12}$$

$$f_4(x,y) = \begin{cases} \max(x,y) & x \geq 0 \land y \geq 0 \\ x+y & x \geq 0 \land y < 0 \\ y+x & x < 0 \land y \geq 0 \\ \min(x,y) & x < 0 \land y < 0 \end{cases} \tag{13}$$

We mostly found these equations by just guessing and checking, though there was a little bit of intuition to it as well. We also wanted to ensure that all partial derivatives would equal 1, -1, or 0 to prove that there is no acceleration [14-21]; otherwise, the robot will not go the direction we want it to.

$$m_{1x}(x,y) = \begin{cases} -1 & x \geq 0 \land y \geq 0 \\ -1 & x \geq 0 \land y < 0 \land -x \leq y \\ 0 & x \geq 0 \land y < 0 \land -x > y \\ -1 & x < 0 \land y \geq 0 \land -x \geq y \\ 0 & x < 0 \land y \geq 0 \land -x < y \\ -1 & x < 0 \land y < 0 \end{cases} \tag{14}$$

$$m_{1y}(x,y) = \begin{cases} 1 & x \geq 0 \land y \geq 0 \\ 0 & x \geq 0 \land y < 0 \land -x \leq y \\ 1 & x \geq 0 \land y < 0 \land -x > y \\ 0 & x < 0 \land y \geq 0 - x \geq y \\ 1 & x < 0 \land y \geq 0 - x < y \\ 1 & x < 0 \land y < 0 \end{cases} \tag{15}$$

$$m_{2x}(x,y) = \begin{cases} -1 & x \geq 0 \land y \geq 0 \land -x \leq -y \\ 0 & x \geq 0 \land y \geq 0 \land -x > -y \\ -1 & x \geq 0 \land y < 0 \\ -1 & x < 0 \land y \geq 0 \\ -1 & x < 0 \land y < 0 \land -x \geq -y \\ 0 & x < 0 \land y < 0 \land -x < -y \end{cases} \tag{16}$$

$$m_{2y}(x,y) = \begin{cases} 0 & x \geq 0 \wedge y \geq 0 \wedge -x \leq -y \\ -1 & x \geq 0 \wedge y \geq 0 \wedge -x > -y \\ -1 & x \geq 0 \wedge y < 0 \\ -1 & x < 0 \wedge y \geq 0 \\ 0 & x < 0 \wedge y < 0 \wedge -x \geq -y \\ -1 & x < 0 \wedge y < 0 \wedge -x < -y \end{cases} \tag{17}$$

$$m_{3x}(x,y) = \begin{cases} 1 & x \geq 0 \wedge y \geq 0 \\ 1 & x \geq 0 \wedge y < 0 \wedge x \geq -y \\ 0 & x \geq 0 \wedge y < 0 \wedge x < -y \\ 1 & x < 0 \wedge y \geq 0 \wedge x \leq -y \\ 0 & x < 0 \wedge y \geq 0 \wedge x > -y \\ 1 & x < 0 \wedge y < 0 \end{cases} \tag{18}$$

$$m_{3y}(x,y) = \begin{cases} -1 & x \geq 0 \wedge y \geq 0 \\ 0 & x \geq 0 \wedge y < 0 \wedge x \geq -y \\ -1 & x \geq 0 \wedge y < 0 \wedge x < -y \\ 0 & x < 0 \wedge y \geq 0 \wedge x \leq -y \\ -1 & x < 0 \wedge y \geq 0 \wedge x > -y \\ -1 & x < 0 \wedge y < 0 \end{cases} \tag{19}$$

$$m_{4x}(x,y) = \begin{cases} 1 & x \geq 0 \wedge y \geq 0 \wedge x \geq y \\ 0 & x \geq 0 \wedge y \geq 0 \wedge x < y \\ 1 & x \geq 0 \wedge y < 0 \\ 1 & x < 0 \wedge y \geq 0 \\ 1 & x < 0 \wedge y < 0 \wedge x \leq y \\ 0 & x < 0 \wedge y < 0 \wedge x > y \end{cases} \tag{20}$$

$$m_{4y}(x,y) = \begin{cases} 0 & x \geq 0 \wedge y \geq 0 \wedge x \geq y \\ 1 & x \geq 0 \wedge y \geq 0 \wedge x < y \\ 1 & x \geq 0 \wedge y < 0 \\ 1 & x < 0 \wedge y \geq 0 \\ 0 & x < 0 \wedge y < 0 \wedge x \leq y \\ 1 & x < 0 \wedge y < 0 \wedge x > y \end{cases} \tag{21}$$

As the equations illustrate, the partial derivative is always either 1, -1, or 0. Hence, the derivative is constant and the function will smoothly transition from any two points on the graph, which means that the correct direction should be applied.

2 Turning

We decided to add the result of the previous function to the result of the turn function produce a final value. We found that turning could be accomplished in a slightly different fashion, as only two touching half-quadrants would be affected. Points outside of that region need only be altered in order to maintain continuity. This was difficult to achieve, but our approach was to determine the values that needed to be changed, and then deduce an equation that smoothed the path between the necessary values. This resulted in us creating two functions per motor: one each for turning right and left. The functions are prototyped similarly to the previous functions, where x is the speed to move right, y is the speed to move forward, and T is the speed to turn right. We denote the functions as f_{ab}, where a is the motor to turn, and b is either r or l, for right and left respectively:

$$t_{1r}(x,y,T) = \begin{cases} -2(-|y|-x)T + (1+y)T & -x \geq |y| \\ (1-y)T & y > |x| \\ (1-x)T & x \geq |y| \\ (1+y)T & -y > |x| \end{cases} \tag{22}$$

$$t_{1l}(x,y,T) = \begin{cases} 2(y+|x|)T + (1+y)T & -y \geq |x| \\ (1-y)T & y > |x| \\ (1-x)T & x \geq |y| \\ (1+y)T & -x > |y| \end{cases} \tag{23}$$

$$t_{2r}(x,y,T) = \begin{cases} 2(y+|x|)T + (1+y)T & -y \geq |x| \\ (1+x)T & -x > |y| \\ (1-y)T & y \geq |x| \\ (1-x)T & -x > |y| \end{cases} \tag{24}$$

$$t_{2l}(x,y,T) = \begin{cases} 2(-|y|+x)T + (1-x)T & x \geq |y| \\ (1+y)T & -y > |x| \\ (1+x)T & -x \geq |y| \\ (1-y)T & y > |x| \end{cases} \tag{25}$$

$$t_{3r}(x,y,T) = \begin{cases} 2(|y|-x)T + (1-x)T & x \geq |y| \\ (1+y)T & -y > |x| \\ (1+x)T & -x \geq |y| \\ (1-y)T & y > |x| \end{cases} \tag{26}$$

$$t_{3l}(x,y,T) = \begin{cases} 2(y-|x|)T + (1-y)T & |x| \leq y \\ (1-x)T & x > |y| \\ (y+1)T & -y \geq |x| \\ (x+1)T & y > |x| \end{cases} \tag{27}$$

$$t_{4r}(x,y,T) = \begin{cases} 2(y-|x|)T + (1-y)T & y \geq |x| \\ (1-x)T & x > |y| \\ (1+y)T & -y \geq |x| \\ (1+x)T & -x > |y| \end{cases} \tag{28}$$

$$t_{4l}(x,y,T) = \begin{cases} 2(-|y|-x)T + (1+x)T & -x \geq |y| \\ (1-y)T & y > |x| \\ (1-x)T & x \geq |y| \\ (1+y)T & -y > |x| \end{cases} \tag{29}$$

3 Proportional Integral Derivative (PID)

The proportional integral derivative (PID) has three components, each of which are calculated separately, multiplied by some scaling value, and then summed together. The proportional component is the easiest to compute, as it is directly proportional to the distance between the robot's current position p_c and the target position p_t. Letting P be the scaling value, this component is simply

$$P(p_t - p_c) \tag{30}$$

The next component, integral, measures the accumulation of error present in the distance travelled, and uses that to overcome friction, as the proportional component will not give the robot enough movement to reach its destination. If there is a very small distance to the endpoint, there will be very little power used, which will likely not be enough to overcome friction. If I is the scale for the integral component, t is a time variable, then the component is given by

$$I\left(\int_0^t (p_t - p_c)dt\right) \tag{31}$$

The last part of PID is derivative. This component ensures that the robot decelerates at a manageable rate; a higher D value corresponds to less deceleration. This can be useful for when the robot is going downhill and needs to stop in a relatively short distance, even if the robot has not been given a large amount of power. Letting D be the scale for this component, we set the component equal to

$$D\left(\frac{d(p_t - p_c)}{dt}\right) \tag{32}$$

We used these components to try and pinpoint the position of the actuator, but that did not work very well, as the physical body of the robot does not provide an easy way for getting the Lexan plates to not hit the walls of the lander without manual operator control. Ultimately, we removed it and opted for full operator control instead.

4 Gyro

We attempted to get an external gyro working on the robot. The only external gyro we have runs on 5V, but the REV expansion hub only has 3.3V analog ports, so we used the REV Logic Level Converter, which converted the 3.3V inputs to 5V. This we eventually found to be the main problem, since this component did not produce a reliably consistent analog signal value. During our prototyping, we tried many ways to write our own gyro code. The first attempt used Riemann summation to numerically integrate the angular rate into an angle. If we let x_i be the angular velocity at a known time t_i, for $i \in \{1, \ldots, n\}$, then we compute the sum

$$\sum_{i=1}^{n} x_i \Delta t_i \tag{33}$$

Each time a data point is recorded, n is increased by 1 and the sum is recomputed. This approach had issues obtaining reliable data; error propagation quickly grew out of hand and was determined to be up to ten orders of magnitude off in some instances. We then went to look for a solution to the problem, and found that the REV Logic Level Converter was not reliably converting analog values. This inspired us to switch to the gyro built into the REV Expansion Hub, which has worked well for us since.

4.1 Enhanced Holonomic Control

The primary motivation for having a working gyro is to have the controls for the robot oriented with respect to the field as opposed to the robot itself; i.e. forward on the joystick moves the robot away from the driver, as opposed to forward from the robot's perspective. Although this feature is still a work in progress, the equation to be implemented is:

$$f(x, y, a) = (\cos(\text{atan2}(x, y) + a)(\sqrt{x^2 + y^2}), \sin(\text{atan2}(x, y) + a)(\sqrt{x^2 + y^2})) \tag{34}$$

where x is the joystick x axis value, y is the joystick y axis value, and a is the amount the robot has turned since the program has started. The above function attempts to rotate the vector of the input of controller by the amount that the robot has rotated

5 Control Goals

- Climb from any point on the field in ¡15 seconds, if possible ¡10
- Clear the depot quickly, without problems
- Navigate the field with obstacles on the field, such as large squares representing robots
- Navigate through a custom obstacle course
- Be able to recover quickly from being pushed
- Control the robot effectively with little visibility.

6 Autonomous

We have a fairly simple strategy in the autonomous period: we want to be of the most help, so we try to cater to whatever the other robot can do. We always land, and have the capability of claiming and parking. We usually ask what the other robot needs to do, and see how our bots can interact the most efficiently and effectively. Our robot is fairly quick, so we aim to interfere with our allied robot as minimally as possible. Our goals for autonomous are as listed.

6.1 Crater Side Start To End In Crater

This autonomous route is the simplest as it requires no turns. It simply lands, unhooks, and then drives forward into the crater:

- ☐ Lands
- ☐ Unhooks (drives slightly to the left)
- ☐ Drives forward into the crater

6.2 Crater Side Start, Claim, Then End In Crater

This route is the most complicated out of the three. It lands, drives over to the depot, claims, then drives back to the crater to park:

- ☐ Lands
- ☐ Unhooks (drives slightly to the left)
- ☐ Drives slightly forward, to approx midway between the lander and the crater
- ☐ Drives diagonally towards the wall in between the depot and the alliance
- ☐ At the wall, it turns and drives the rest of the way to the depot
- ☐ Delivers marker to the depot
- ☐ Drives back to the crater, to park

6.3 Marker Side Start, Claim, Then End In Crater

This is the only route we have that starts on the side with the marker. It lies between the first two routes in terms of difficulty.

- ☐ Lands
- ☐ Unhooks (drives slightly to the left)
- ☐ Drives forward to the depot
- ☐ Claims in the depot
- ☐ Drives to the crater, and parks.