Mathematics in Autonomous Robotics Control - VEX V5 Robotics Competition

Lin, Jiahong Ericsson
Senior Programmer, 936A
HKIS VEX Robotics Club
`251010@hkis.edu.hk`

You, Qi Aaron
Senior Programmer, 936A
HKIS VEX Robotics Club
`250710@hkis.edu.hk`

Liu, Qinan Andy
Senior Programmer, 936A
HKIS VEX Robotics Club
`250886@hkis.edu.hk`

November 9, 2024

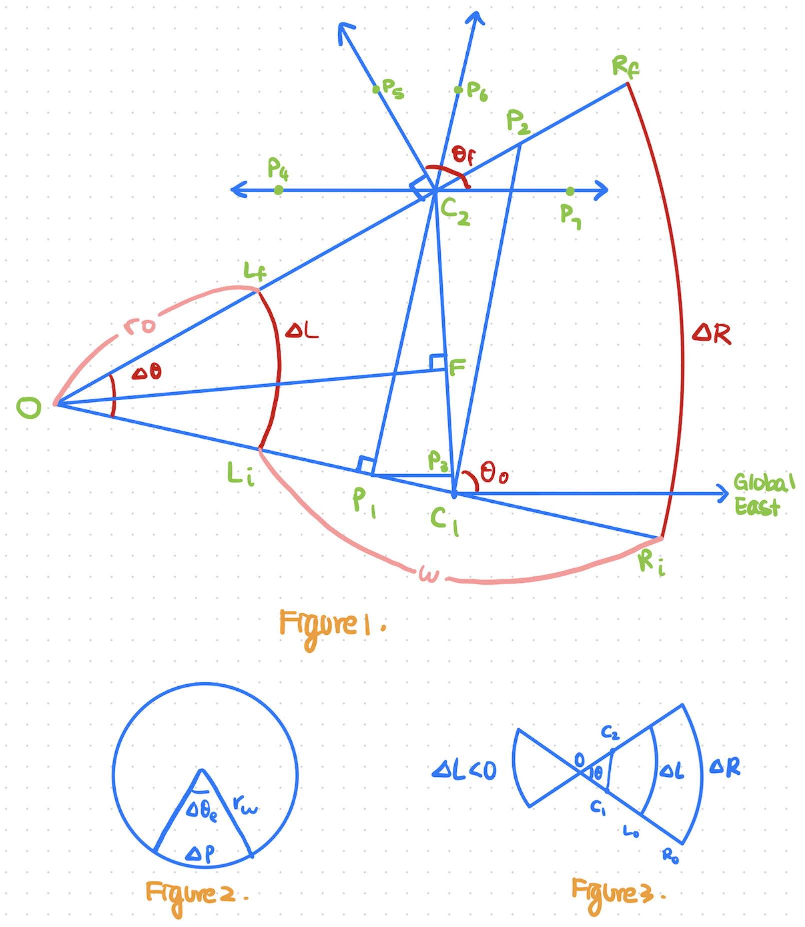
Abstract

The precise control of motor output values to follow a smooth path predetermined from set waypoints is crucial to the success of the autonomous period. Therefore, the mathematics behind these algorithms have been laid out in the document below. Mathematics enabling odometry for accurate coordinate tracking using motor encoder readings, cubic splining for smooth (see section Cubic Splines for definition) path interpolation (based on a custom-implemented matrix library), pure pursuit for smooth path following, and PID for output power limiting is described.

Contents

1 Odometry

Odometry is employed to track the live coordinates of the robot, given only the information outputted by the motor encodings on either side. In other words, given a set of encoder readings over time, the task is to estimate the position of the robot at any point of time in the period over which the readings were taken. For the calculations, the following diagrams shall be used.



(1)

1.1 Constructions

In (1), refer to Figure 1.

$$\overline{P_1C_2P_6} \perp \overline{OR_1} \tag{2}$$

$$\overline{P_1P_3}\text{is global east.} \tag{3}$$

$$\overline{P_4C_2P_7} \parallel \overline{P_1P_3} \tag{4}$$

$$\overline{C_2P_5} \perp \overline{OC_2} \tag{5}$$

1.2 Calculating Positional Change

In (1), refer to Figure 2.

Let the wheel radius be r_w, and $\Delta\theta_e$ be the change in encoder position in radians.

$$\Delta p = \Delta\theta_e \cdot r \tag{6}$$

1.3 Calculating heading change

1.3.1 Case 1: Wheels have Same Sign

In (1), refer to Figure 1.

$$\begin{cases} \Delta\theta r_0 = \Delta L \\ \Delta\theta\left(r_0 + w\right) = \Delta\theta r_0 + \Delta\theta w = \Delta\theta R \end{cases}$$

$$\Rightarrow \Delta\theta w = \Delta R - \Delta L$$

$$\Rightarrow \Delta\theta = \frac{\Delta R - \Delta L}{w}$$

1.3.2 Case 2: Wheels have Opposite Sign

In (1), refer to Figure 3.

$$w = L_0R_0 = OR_0 - OL_0 = \frac{\Delta R}{\theta} - \frac{\Delta L}{\theta} = \frac{\Delta R - \Delta L}{\theta}$$

$$\Rightarrow \Delta\theta = \frac{\Delta R - \Delta L}{w}$$

In both cases,

$$\Delta\theta = \frac{\Delta R - \Delta L}{w} \tag{7}$$

1.4 Calculating Final Heading

In (1), refer to Figure 1.

$$\mathrm{m}\angle OC_2P_1 = \frac{\pi}{2} - \Delta\theta$$

$$\overline{C_2P_4} \parallel \overline{C_1P_1} \Rightarrow \mathrm{m}\angle P_4C_2P_1 = \theta_0$$

$$\Rightarrow \mathrm{m}\angle OC_2P_4 = \theta_0 - \left(\frac{\pi}{2} - \Delta\theta\right) = \theta_0 + \Delta\theta - \frac{\pi}{2}$$

$$\Rightarrow \mathrm{m}\angle P_5C_2P_4 = \frac{\pi}{2} - \left(\theta_0 + \Delta\theta - \frac{\pi}{2}\right) = \pi - \theta_0 - \Delta\theta$$

$$\Rightarrow \mathrm{m}\angle P_5C_2P_7 = \pi - (\pi - \theta_0 - \Delta\theta) = \theta_f = \theta_0 + \Delta\theta \tag{8}$$

1.5 Calculating Straight-Line Distance

1.5.1 Case 1: Nonzero Theta

In (1), refer to Figure 1.

$$r = r_0 + \frac{w}{2} = \frac{\Delta L}{\Delta\theta} + \frac{w}{2} = OC_1 = OC_2$$

Construct the perpendicular bisector of $\overline{C_1C_2}$, $\overline{OF}$:

$$C_1F = C_2F = r\sin\frac{\Delta\theta}{2}$$

$$\Rightarrow C_1C_2 = 2r\sin\frac{\Delta\theta}{2} \tag{9}$$

1.5.2 Case 2: Zero Theta

Since both sides have had the same track change:

$$C_1C_2 = \Delta L = \Delta R \tag{10}$$

1.6 Change of Basis

In (1), refer to Figure 1.

$$\triangle OC_1C_2 \text{ isosceles}$$

$$\Rightarrow \mathrm{m}\angle OC_1C_2 = \frac{\pi - \Delta\theta}{2}$$

$$\Rightarrow \mathrm{m}\angle C_2P_3P_2 = \frac{\pi}{2} - \frac{\pi - \Delta\theta}{2} = \frac{\Delta\theta}{2}$$

Define offset $k = \theta_0 + \frac{\Delta\theta}{2}$. Set up a change of basis:

$$A = \begin{bmatrix} \cos -k & \cos\frac{\pi}{2} - k \\ \sin -k & \sin\frac{\pi}{2} - k \end{bmatrix} = \begin{bmatrix} \cos k & \sin k \\ -\sin k & \cos k \end{bmatrix}$$

In this system, C_1C_2 lies on the horizontal axis and the change in coordinate is merely the displacement calculated in the previous step. To undo the change of basis to get the translation vector in the global coordinate system, multiply by the inverse matrix:

$$A^{-1} = \begin{bmatrix} \cos k & -\sin k \\ \sin k & \cos k \end{bmatrix} \tag{11}$$

1.7 Summary

$$\Delta\theta = \frac{\Delta R - \Delta L}{w} \tag{12}$$

$$d = \begin{cases} 2r\sin\frac{\Delta\theta}{2} & \Delta\theta \neq 0 \\ \Delta L & \Delta\theta = 0 \end{cases} \tag{13}$$

$$\vec{t} = d \cdot \begin{bmatrix} \cos k & -\sin k \\ \sin k & \cos k \end{bmatrix} \tag{14}$$

1.8 A Note on Gear Ratios

The math in this section assumed that the motor cartridge was 1-to-1 in the calculation of the distance change. In practice, the change in track should be calculated taking gear ratios into account. Therefore, the code in Section 5 also includes a scalar which will account for gear ratio effects.

2 Cubic Splines

Cubic splines are used to interpolate a smooth path which passes through certain waypoints. Given a set of waypoints $\mathbf{P}$, the task is to output a new set of points $\mathbf{P}_{\text{interp}}$ which passes through all of the points in $\mathbf{P}$ while also satisfying certain criteria for smoothness and continuity.

2.1 Definitions

We define a cubic spline interpolation function as a piecewise function S of n cubic equations:

$$\forall 1 \le k \le n : S\left(x; \vec{c}\right) = f_k\left(x\right), x_{k-1} \le x < x_k \tag{15}$$

Here, we define each f and its derivatives as follows:

$$\begin{cases} f_k\left(x\right) = a_k x^3 + b_k x^2 + c_k x + d_k \\ \frac{d}{dx} f_k\left(x\right) = 3a_k x^2 + 2b_k x + c_k \\ \frac{d^2}{dx^2} f_k\left(x\right) = 6a_k x + 2b_k \end{cases} \tag{16}$$

We also define a function $\mathbb{S} : X \times Y \to \mathbb{R}^{4n}$ which returns the spline interpolation function coefficients from input points $\mathbf{X}$ and $\mathbf{Y}$:

$$\mathbb{S}\left(\mathbf{X}, \mathbf{Y}\right) = \vec{c} = \begin{bmatrix} a_1 \\ b_1 \\ \vdots \\ c_n \\ d_n \end{bmatrix} \tag{17}$$

We assume $\{x_k\}$ is strictly increasing. There are $4n$ unknown variables requiring $4n$ equations.

2.2 Restrictions

We restrict S, requiring it to pass through each of the $n+1$ waypoints and belong to differentiability class C^2. That is, $f^{(2)}$ is continuous. Thus, $\forall 1 \le k < n$:

$$\begin{cases} f_k\left(x_{k-1}\right) = y_{k-1} \\ f_k\left(x_k\right) = y_k \\ \frac{d}{dx} f_k\left(x_k\right) - \frac{d}{dx} f_{k+1}\left(x_k\right) = 0 \\ \frac{d^2}{dx^2} f_k\left(x_k\right) - \frac{d^2}{dx^2} f_{k+1}\left(x_k\right) = 0 \end{cases} \tag{18}$$

Additionally,

$$\begin{cases} \frac{d^2}{dx^2} f_1\left(x_0\right) = 0 \\ \frac{d^2}{dx^2} f_n\left(x_n\right) = 0 \\ f_n\left(x_{n-1}\right) = y_{n-1} \\ f_n\left(x_n\right) = y_n \end{cases} \tag{19}$$

The $n-1$ systems of 4 equations in (19) and the additional 4 equations in (18) give us $4\left(n-1\right)+4 = 4n$ equations to work with, as desired.

2.3 Matrix Form

We write the resultant system of equations as a matrix. Aside from the boundary conditions, the following structure is repeated for the section of the matrix beginning at index $(4k-2, 4k-3)$, valid $\forall 1 \leq k < n$:

$$\mathbf{S} = \begin{bmatrix} x_{k-1}^3 & x_{k-1}^2 & x_{k-1} & 1 & 0 & 0 & 0 & 0 \\ x_k^3 & x_k^2 & x_k & 1 & 0 & 0 & 0 & 0 \\ 3x_k^2 & 2x_k & 1 & 0 & -3x_k^2 & -2x_k & -1 & 0 \\ 6x_k & 2 & 0 & 0 & -6x_k & -2 & 0 & 0 \end{bmatrix} \tag{20}$$

Specifically:

$$\forall 1 \leq k < n \left(\forall 1 \leq i \leq 4 \left(\forall 1 \leq j \leq 8 \left(\mathbf{M}_{4k-2+i, 4k-3+j} = \mathbf{S}_{i,j}\right)\right)\right) \tag{21}$$

Where $\mathbf{S} \in \mathbb{R}^{4 \times 8}$ is defined in (20) and the rest of the matrix $\mathbf{M}$ is filled with 0. The boundary conditions are filled in rows $1, 4n-2, 4n-1, 4n$, respectively. The full matrix $\mathbf{M} \in \mathbb{R}^{4n \times (4n+1)}$ is given below:

$$\mathbf{M} = \left[\begin{array}{ccccccccccccccccc|c} 6x_0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_0^3 & x_0^2 & x_0 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_0 \\ x_1^3 & x_1^2 & x_0 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_1 \\ 3x_1^2 & 2x_1 & 1 & 0 & -3x_1^2 & -2x_1 & -1 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6x_1 & 2 & 0 & 0 & -6x_1 & -2 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & x_{n-2}^3 & x_{n-2}^2 & x_{n-2} & 1 & 0 & 0 & 0 & 0 & y_{n-2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & x_{n-1}^3 & x_{n-1}^2 & x_{n-1} & 1 & 0 & 0 & 0 & 0 & y_{n-1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 3x_{n-1}^2 & 2x_{n-1} & 1 & 0 & -3x_{n-1}^2 & -2x_{n-1} & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 6x_{n-1} & 2 & 0 & 0 & -6x_{n-1} & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & x_{n-1}^3 & x_{n-1}^2 & x_{n-1} & 1 & y_{n-1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & x_n^3 & x_n^2 & x_n & 1 & y_n \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 6x_n & 2 & 0 & 0 & 0 \end{array}\right] \tag{22}$$

Letting $\mathbf{R} = \operatorname{rref}(\mathbf{M})$, the result is then the last column of the matrix $\mathbf{R}$. More formally, letting $\vec{v} \in \mathbb{R}^{4n+1}$ so that $\vec{v} = [0, 0, \cdots, 0, 1]^\intercal$, the vector of coefficients $\vec{c}$ is given by:

$$\vec{c} = \begin{bmatrix} a_1 \\ b_1 \\ \vdots \\ c_n \\ d_n \end{bmatrix} = \mathbf{R}\vec{v} \tag{23}$$

2.4 Parametrization

We have defined a spline function S to pass through way-points and satisfy certain constraints and solved for the coefficients of the cubic pieces. However, this has not accounted for the fact that most paths are not, in fact, functions. Thus, we employ parametrization. Let the set of points be $\mathbf{P} = \{(x_k, y_k)\}$. We define $\mathbf{X} = \{x_k\}$ and $\mathbf{Y} = \{y_k\}$ and perform the cubic spline interpolation on $\mathbf{X}$ and $\mathbf{Y}$ independently. We use a separate parameter $\mathbf{p} = \{x\}$ of the same size ($|\mathbf{X}| = |\mathbf{Y}| = |\mathbf{P}|$) to obtain the vectors $\vec{c}_x = \mathbb{S}(\mathbf{P}, \mathbf{X})$ and $\vec{c}_y = \mathbb{S}(\mathbf{P}, \mathbf{Y})$, respectively. It is then possible to run interpolation on the axes separately, using new parameters p_{interp} so that $\mathbf{X}_{\text{interp}} = S(p_{\text{interp}}; \vec{c}_x)$ and $\mathbf{Y}_{\text{interp}} = S(p_{\text{interp}}; \vec{c}_y)$. The result $\mathbf{P}_{\text{interp}}$ is then $\left\{\left(\mathbf{X}_{\text{interp}_k}, \mathbf{Y}_{\text{interp}_k}\right)\right\}$, $\forall 0 \leq k < |\mathbf{X}_{\text{interp}}|$.

3 Pure Pursuit

We now have defined a method to obtain the coordinates for a smooth path. The next step is to follow this path. Pure Pursuit is defined by always looking for a point on the path some look-ahead distance l in front of the robot, and adjusting the steering power accordingly. Therefore, we need to make calculations to determine the target point and the relative steering power.

3.1 Intersection of a Circle and Line

We have the robot centered at (h, k) with a look-ahead distance of l, and a set of points on the path $\{(x_k, y_k)\}$. We iterate through solving in reverse so that the point found is always closest to the end and that no point that has already been traced is visited again. In the case that $x_k \neq x_{k+1}$, we have:

$$\begin{cases} y = mx + b \\ (x-h)^2 + (y-k)^2 = l^2 \end{cases} \tag{24}$$

We can write $y = mx + b$ after solving for x:

$$\left(m^2+1\right)x^2 + 2\left(mb - mk^2\right)x + \left(h^2 + k^2 + b^2 - l^2 - 2bk\right) = 0 \tag{25}$$

$$\Rightarrow x = \frac{-2\left(mb - mk^2\right) \pm \sqrt{\left(2\left(mb - mk^2\right)\right)^2 - 4\left(m^2+1\right)\left(h^2+k^2+b^2-l^2-2bk\right)}}{2\left(m^2+1\right)} \tag{26}$$

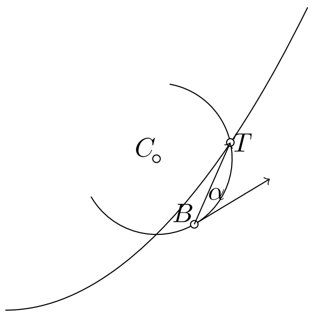
In the other case that $x_k = x_{k+1}$, we can similarly find:

$$y = k \pm \sqrt{l^2 - (h - x_k)} \tag{27}$$

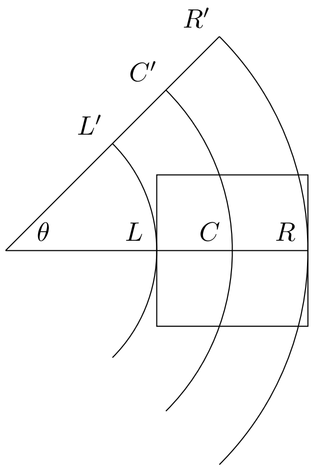
Importantly, note that for each solved (x, y) it must hold that $x \in [x_k, x_{k+1})$ and $y \in [y_k, y_{k+1})$. We say that such a point lies on segment k of the path (starting from $k = 0$).

3.2 Relative Steering Power

In the coordinate plane, we define point B as the robot's coordinate and T as the target point. α is calculated from the robot's initial heading: construct midpoint M of $\overline{BT}$; due to tangency, $m\angle BCM = \alpha$, and, by definition, $r_c = \frac{l}{2}\csc\alpha$. The steering for the left and right sides is calculated as $s_l = r_c - \frac{w}{2}$ and $s_r = r_c + \frac{w}{2}$, where w is the width of the robot. Refer to the diagrams below: (28) is the diagram to solve for the radius r_c, and (29) is the diagram to solve for the steering values s_l and s_r.



(28)



(29)

After finding the steering values, we perform a normalization step so that the values add to 100; this normalization is performed so that PID values can be consistently adjusted. The normalization is as follows:

$$\begin{cases} s_l \leftarrow s_l \cdot \frac{100}{s_l+s_r} \cdot \operatorname{sgn}(\cos\alpha) \\ s_r \leftarrow s_r \cdot \frac{100}{s_l+s_r} \cdot \operatorname{sng}(\cos\alpha) \end{cases} \tag{30}$$

Where $\operatorname{sgn}(\cos\alpha)$ equaling 1 when $\cos\alpha > 0$, 0 when $\cos\alpha = 0$, and -1 otherwise is multiplied by the scale factor to allow the robot to move backward if it is more efficient.

3.3 Distance to Path End

We find the distance from a solved point (x_t, y_t) lying on segment k to the end of the path by summing up the respective distances (the look-ahead distance, the distance from the target to the next point, and the sum of distances of all points from the next point to the end point):

$$d(x_t, y_t) = l + \sqrt{(x_t - x_{k+1})^2 + (y_t - y_{k+1})^2} + \sum_{i=k+1}^{n} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$$

The value k is the unique value such that $x_t \in [x_k, x_{k+1})$ and $y_t \in [y_k, y_{k+1})$, as described above by the definition that "the point lies on segment k of the path".

4 Output Control

The PID controller is defined by the output function $p : \mathbb{R} \to \mathbb{R}$:

$$p(t) = k_p \epsilon(t) + k_i \int_0^t \epsilon(\tau)\, \mathrm{d}\tau + k_d \frac{\mathrm{d}\epsilon(t)}{\mathrm{d}t} \tag{31}$$

In the above equation, $\epsilon(t)$ represents the error value actual$-$target evaluated at time t. Real numbers $k_p, k_i, k_d \geq 0$ represent gain constants, which must be fine-tuned. Then, the first term $k_p\epsilon(t)$ represents the proportional portion, which will increase the output power the farther the greater the error term is. The second term $k_i \int_0^t \epsilon(\tau)\, \mathrm{d}\tau$ represents the integral portion, which will increase output power the longer the error accumulates. The last term $k_d \frac{\mathrm{d}\epsilon(t)}{\mathrm{d}t}$ represents the derivative portion, which restricts output the faster the error rate is changing (known as **anticipatory control**) to prevent overshooting from the proportional and integral terms. In practice, the integral term is calculated by directly summing up the errors, and the derivative is estimated by the signed difference of the last two errors.

5 C++ Implementation

We are in the process of writing and debugging a library that will be used for autonomous and control period robot programming. The quality of the code is much higher than previous years, employing functions and abstraction when necessary so that the code is clean and maintainable, simple yet versatile. Our codebase can be found on this live Github page (private repo) along with all commit history.