Contents

Chapter 1

Sinusoidal Turn Profiles

1.1 Principles

As with the other phased turn profiles, the aim is to have the robot transition from purely linear motion through to a constant radius turn and then back to purely linear motion.

In this image you can see how the angular velocity increases and decreases in the first and last phase. The shape of the curves in these phases follows one quarter of a sinusoid.

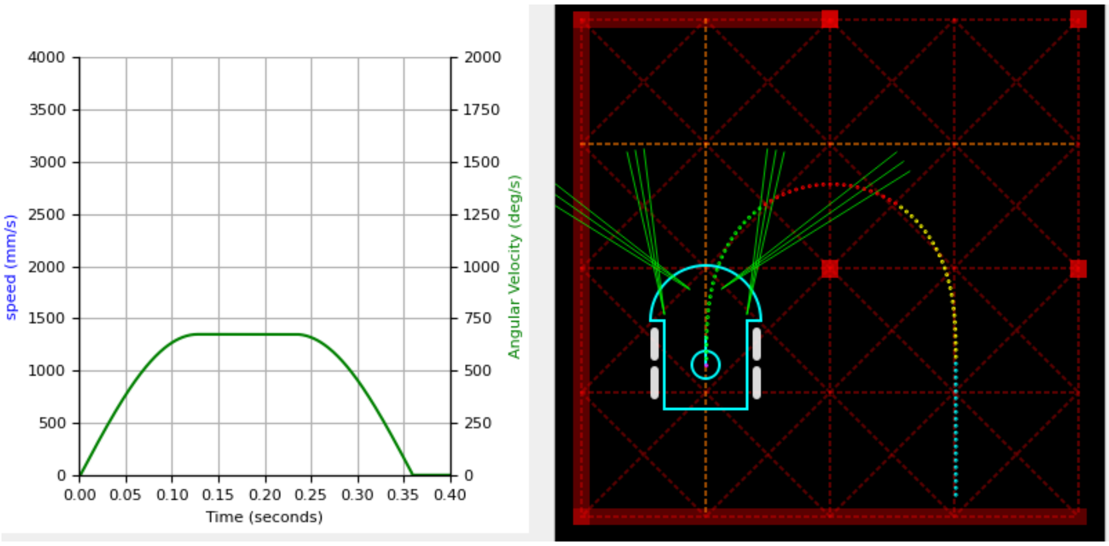

Figure 1.1: A 180 degree sinusoidal profile turn

The original idea of a sinusoidal profile was to try and keep the sum of the forces on the tyre constant throughout the turn. Those forces are the tangential braking/accelerating force and the lateral, centripetal force.

There are specific parameters that depend upon the dimensions of the robot which will ensure the constant force constraint is met. In practice though, better overall performance can be achieved by taking a more general apporach to decidingthe exact length of each phase.

1.2 Turn design

Designing the turn assumes that it will be performed at a constant velocity, v, and starts with three parameters.

- First is the total angle of the turn, θ_0
- Second is the desired centripetal acceleration, A_0
- Third is the desired radius of the central phase of the turn, R_0

Note that the turn radius, R_0, is describing the actual radius of the central phase of the turn. The *effective radius* will be larger.

1.3 Setup

It is convenient to define the turn speed in terms of centripetal acceleration rather than velocity because it is the centripetal acceleration that limits grip and performance directly. Given the desired centripetal acceleration, the required speed can be calculated from:

$$v = \sqrt{A_0 R_0} \tag{1.1}$$

Now we can calculate the angular velocity, ω_0, in radians per second, needed for the robot to perform the central, constant radius phase:

$$\omega_0 = \frac{v}{R_0} \tag{1.2}$$

The robot starts the turn with an angular velocity of zero and must transition to $\omega = \omega_0$ over a fixed distance x_Δ. This distance can be chosen as a turn paramater but is usually the same at all speeds for a given turn type. While the choice of x_Δ is arbitrary it should be noted that small values will cause large braking/acceleration loads on the tyres and larger values will make the overal turn radius larger.

1.4 Transition

During the first, accelerating transition phase of the sinusoidal profile the current angular velocity will vary as a function of the sine of the distance:

$$\omega = \omega_0 sin\left(\frac{\pi}{2}\frac{v}{x_\Delta}t\right) \tag{1.3}$$

During the second, decelerating transition phase, the robot's angular velocity must be reduced to zero as a function of this sinusoid:

$$\omega = \omega_0 sin\left(\frac{\pi}{2}\left(1 - \frac{v}{x_\Delta}t\right)\right) \tag{1.4}$$

1.5 Execution

To perform the full turn in the robot, the three phases must be executed in turn. The first phase is not difficult to implement. The robot must simply increase the angular velocity according to (3) above. The only remaining calculation is to determine when to begin the third, deceleration phase. There is no clear way to calculate the distance over which the middle phase must run though some implementations may just make this another parameter. In that way, the total angle for the turn can be adjusted by changing that distance. For example, a 90° turn could be extended to 135° or 180° by simply changing the length of the constant-radius phase. In the same way, errors in the drive train or control system might mean that the robot does not turn by exactly the calculated amount and so the actual angle could be adjusted by changing the centre section length.

A better solution is to calculate the angle through which the robot must turn during the accelerating and decelerating phases and use that information to determine when the decelerating phase must begin.

1.6 Transition angle

The accelerating and decelerating transition angles,θ_Δ, are the same and can be calculated by integrating (3):

$$\theta_\Delta = \int_0^{x_\Delta/v} \omega_0 \sin\left(\frac{\pi v}{2x_\Delta}t\right) dx$$

$$= \left[-\frac{2x_\Delta}{\pi v}\omega_0 \cos\left(\frac{\pi v}{2x_\Delta}t\right) + C\right]_0^{x_\Delta/v}$$

$$\therefore \theta_\Delta = \frac{2x_\Delta}{\pi v}\omega_0 \tag{1.5}$$

Since the angular velocity, $\omega_0 = v/R_0$

$$\theta_\Delta = \frac{2x_\Delta}{\pi R_0} \tag{1.6}$$

1.7 Continuous Sinusoid

The three-phase turn calculations are not overly difficult to implement and provide for many opportunities to fine-tune the actual path of the robot. Even so, it might be convenient to use a single continuous function to determine the angular velocity throughout the turn.

Consider the case where the turn transition angle, θ_Δ is exactly half the total turn angle, θ_T. That would permit the angular velocity throughout the turn to be calculated from a single sinusoidal function. All that is required is to find the value of x_Δ that satisfies the condition

$$\theta_\Delta = \frac{\theta_0}{2} = \frac{2x_\Delta}{\pi R_0}$$

$$x_\Delta = \frac{\pi\theta_0 R_0}{4} \tag{1.7}$$

1.8 Implementation

With the relationships established between the turn angle, θ_0 (in radians), the desired minimum radius, R_0, and the desired centripetal acceleration, A_0, the continuous sinusoid profile is easy to implement. All that is required is that the robot calculate the transition distance x_Δ from (1.7) and then run at a constant velocity, v, while generating an angular velocity from these functions:

$$\omega_0 = \frac{v}{R_0} \tag{1.8}$$

$$\omega = \omega_0.sin\left(\frac{\pi x}{x_\Delta}\right), 0 \leq x \leq x_\Delta \tag{1.9}$$

For the 180 degree turn illustrated at the start of this paper, the result looks like this:

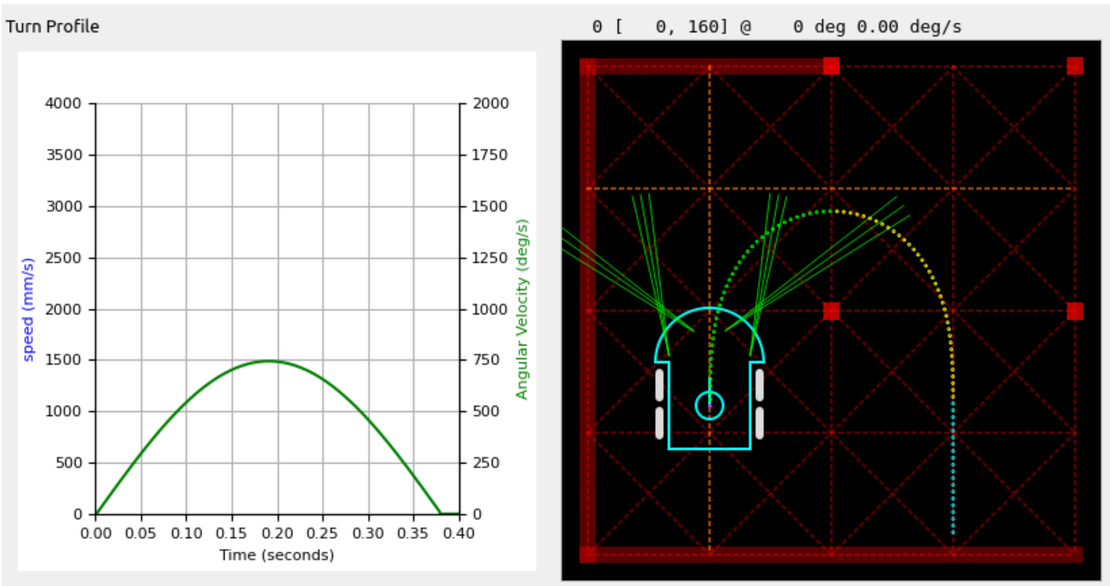

Figure 1.2: A 180 degree continuous sinusoidal profile turn

The initial acceleration required of the tyres is significantly less at the expense of a somewhat higher maximum centripetal acceleration although that is over a shorter period. At the speed shown, the continuous form of the turn will take an extra 23ms - an increase of 6.5%. It should be noted that, on a small microcontroller, the calculation of the sine() function may impose unacceptable overheads and approximations or lookup tables may be preferable.