Introduction to Motion Profile for VEX Robotics

Benefits, Analytical Derivation, and Implementation using Feedback

Marcos R. Pesante Colón
AON Robotics, University of Puerto Rico
Mayagüez, Puerto Rico
marcos.pesante@upr.edu

Abstract – Motion profiling is a relatively simple and intuitive method to plan a mechanism's motion characteristics starting at a fixed initial point and moving to a fixed destination. It is most frequently focused to convert a simple path to a useful trajectory by specifying time and speed information at each point in the path. Therefore, it might not be appropriate for every task, such actively holding a mechanism's position, yet it is effective at helping control robot drive movements. Furthermore, if this time scaler is given a feedback loop, it has the potential to become an accurate position controller for kinds of movements. Although there exist many variations of motion profiles, we analyze the Trapezoidal Motion Profile in this article. The analytical derivation of this specific type of motion profile is shown in the appendix by using the same kinematic formulas taught at an introductory physics course.

Key Words - Digital Control, Feedback Control Systems, Trajectory Planning, Robot Motion Control

I. Introduction.

Throughout my personal experience in the VEX Robotics Competition, the PID controller [1] has been the dominant control algorithm for competitive teams. Apart from PID, alternative control algorithms are time-based, Bang Bang Controller [2], and Fuzzy Logic [3]. However, a minority of these competitive teams also use motion profiles. In very summarized terms, motion profiling consists of planning how the robot's kinematics are going to look like throughout a particular movement. This is used in conjunction with path planners to determine what voltages to send to the motor to achieve the desired movement.

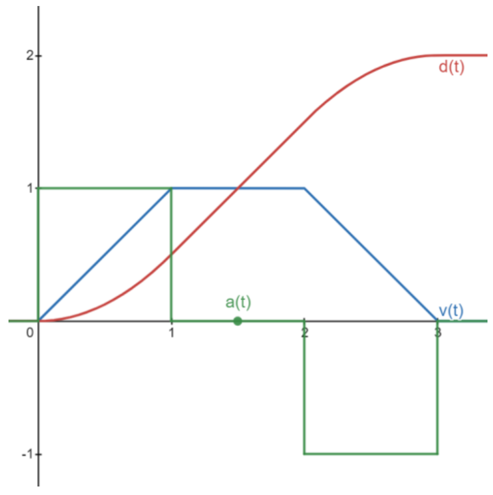

Fig. 1 Example of Trapezoidal Motion Profile

Depending on the application's particular needs there are many variations on how motion profiles are implemented. The three variations that I have seen are predominantly used are: Triangular Motion Profile, Trapezoidal Motion Profile, and S-Curve Motion Profile. They all stem from the basic kinematic motion equations taught at introductory physics courses and have sections, or stages, with different position, speed, and acceleration curves.

$$v_f = v_0 + at \tag{1}$$

$$\Delta x = \frac{v_f + v_0}{2} t \tag{2}$$

$$\Delta x = v_0 t + \frac{1}{2} a t^2 \tag{3}$$

$$v_f^2 = v_0^2 + 2a\Delta x \tag{4}$$

All in all, I have seen motion profiles applied to robot's drive control system (even by the YouTuber Mark Rober [6]) and working with multiple sensors and systems (like Encoder Odometry [4]) to achieve exact and consistent position control. There are libraries like Okapilib [5] that have motion profile capabilities, making such a powerful tool available to a wider audience. For the purpose of this article, we will discuss motion profiles from the bottom up, building them up analytically stage by stage.

II. MOTIVATION.

A. *The PID controller*

Inside the VEX Robotics Competition platform, the PID controller is used like a universal closed-loop controller. Systems like the robot drive, lift, and actuators most frequently use the PID controller. The advantage of being able to tune just three gains to achieve the desired motion for multiple systems while reusing the same code makes this a very attractive control algorithm. It saves time while implementing the algorithm and helps programmers adapt quickly to the code base as there is only one controller in use. In addition, its feedback component adds to its consistency and makes it an even more attractive control algorithm.

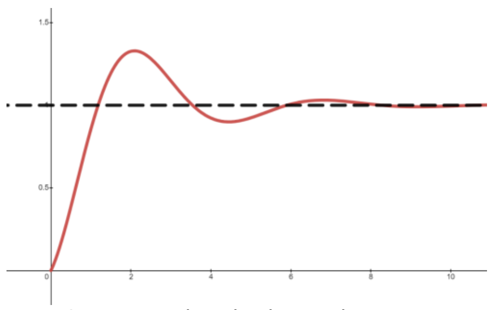

Fig. 2 Typical underdamped PID controller response.

However, even if it has potential to be used in almost any mechanism's control system, this can also pose a disadvantage. The controller's gains can be sensitive to external variations like changes in the motor's temperature, the robot's load, inconsistencies in traction between the wheels and the field, and sometimes even the slightest mechanical improvements can significantly affect the PID controller's consistency. Therefore, moving a tuned PID controller to a new environment can greatly impact its performance. It might even render autonomous routines useless and harmful as excessive oscillations might damage mechanisms or field elements and unexpected movements might break competition rules (like crossing autonomous lines in some competitions).

In addition to this, tuning the PID gains can be a time-consuming process. Even if it is possible to tune the PID gains in the new environment, time constraints during the competition can be detrimental, and last-minute mechanical fixes can undo the progress done after tuning the PID controller.

B. *Motion Profile as an alternative*

Motion profiling can serve as an alternative to PID in some situations. Given a path, motion profiles function as a time-scaler that assign different speeds to points along the path. The reason it is called a time-scaler is because the combination of speed and position can be used to determine the robot position at a time instant.

Motion profiling focuses on the kinematics of the robot, as opposed to the PID controller which focuses more on the dynamics of the system. This helps keep the autonomous routines consistent even if there are external variations between the practice environment and the competition environment.

In addition, motion profile parameters are much more intuitive and easily adjustable compared to the PID gains. These parameters have physical interpretations. For the trapezoidal motion profile, these are: distance, speed, and acceleration. If there are variations between the expected behavior and the real behavior, the parameters can be easily adjusted without worrying about oscillations or potentially damaging mechanisms. Meanwhile, the PID gains in its ideal (and most used) form have a weak relationship to the controller's physical behavior. Even using the PID's standard form, it might be difficult to understand the gain's physical interpretation for some subsystems. [1]

Motion profiles can be as simple or as complicated as needed for the application. They can simply be made up of two stages with two straight lines, they can be composed of seven second-order polynomials for smoother motion, or they can be composed of customized curves and lines that achieve a specific desired behavior.

Overall, Motion Profiles are not appropriate for every scenario. Using motion profiles for situations that require to hold a position or follow a reference that continuously changes might not be as effective. However, Motion Profiling is well-suited for applications that have a fixed reference, just like travelling a fixed distance from a fixed initial position.

III. Conceptual Overview of Motion Profiles

How does the motion profiling achieve all these goals? In this section, the advantages and limitations of motion profiles will be illustrated using Trapezoidal Motion Profiles as used for 1-dimensional robot position control, where the path is a straight line.

As mentioned before, motion profiles focus on the kinematics of the system being controlled. Therefore, the equations used to control the robot are derived from equations 1 and 3, with the help of equation 4. The process of deriving these equations are left to the Appendix A. For now, we can use the following Desmos Project as an interactive visualization of what we will be explaining.

A) The Trapezoidal Speed Profile

The protagonist in our story is the speed profile: a relationship between how much time has elapsed since the movement began and the desired speed we want to move at. To achieve a smooth motion, we want a smooth transition between the robot's two main states: travelling at full speed and sitting completely still. The Trapezoidal Motion Profile achieves this by separating the robot's motion into three separate stages.

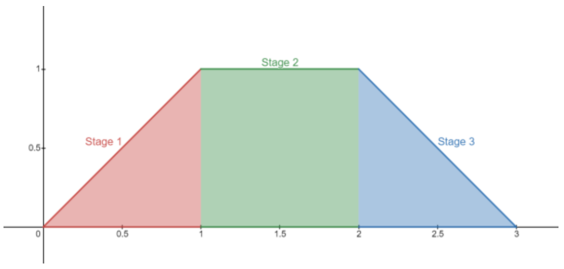

Fig 3. Trapezoidal Speed Profile decomposed into its different stages.

During the first stage of the motion the robot's speed increases at a constant rate. As seen in figure 3, the robot goes from a speed of 0 (sitting completely still) and goes up to a speed of 1 (travelling at maximum speed). Geometrically this transition is done by forming a straight line going from the origin to a maximum speed sometime after. In a more physical sense, this is just the application of equation 1 when $v_0 = 0$.

This first stage is particularly important, as it is what gives Motion Profile one of its biggest advantages over the PID controller. A gradual and controlled increase in speed prevents the robot's drive from losing traction with the field or causing other unpredictable behaviors.

Once the robot reaches its maximum travel speed, it moves on to the second stage of the motion. As seen in figure 3, this stage is characterized by a period of travel with constant speed, geometrically seen as a horizontal line. This is when the robot will travel the fastest, and therefore is the stage we want to spend the most time travelling in. The PID achieves a somewhat similar behavior to this stage immediately after starting, when the error is so big it saturates the motors.

Finally, some time before reaching our destination, the robot needs to start decelerating to prevent a sudden stop (which could cause overshoots and other undesired behavior). This is when we reach the third stage of the motion: when the robot is decelerated at a constant rate until it fully stops. This has the advantage of preventing overshoots and helping make the motion much more predictable.

The PID controller has a similar mechanism when the error approaches 0. The actuator's input is decreased as the robot reaches its destination.

As seen by the simple description above, most external variables that affect the robot's motion are abstracted way in the form of a few select parameters that shape the Trapezoid seen in figures 1 and 3. These parameters are:

1. The initial speed v_0
2. The initial acceleration a_0
3. The maximum speed V
4. The final speed v_f
5. The final acceleration a_f
6. The motion's total duration T

Other parameters such as the travel distance D and stage durations t_0, t_v, and t_f are all derived from these six basic parameters. Yet, this does not mean they do not play a crucial role within the speed profile.

The number of parameters might sometimes start out as overwhelming but notice how v_0 and v_f are mostly going to be 0 since we want to perform a point-to-point motion. These will start from a stand-still $(v_0 = 0)$ and stop at the destination $(v_f = 0)$. There is not always a need to specify a_0 and a_f separately, so they might as well be equal to a variable A, which creates a symmetric trapezoid shape.

As for the other parameters, when using the PID controller we already need to define a desired travel distance D (which takes the form of a setpoint, or a reference signal) and desired maximum speed V. This leaves the programmer with the task of determining the acceleration A and the motion's total duration T. Even this last parameter will be automatically calculated in the program given analytical constraints found in the Appendix A.

A note before moving on to the position profile is that the second stage is the stage that we want to spend the most time travelling in since this will bring the robot to its destination in the shortest amount of time. It leaves extra time during the competition for other tasks during the autonomous routine. To maximize the time spent in this stage we only need to reduce the duration of stages 1 and 3 by increasing both the initial acceleration and final deceleration. However, this comes at a cost. The bigger the acceleration, the more problems might arise due to slippage and other factors. It is the programmer's job to intuitively (or sometimes analytically) estimate the robot's limits and manipulate these parameters accordingly.

B) The Trapezoidal Position Profile

What if we now want to know not only the desired speed at a given time, but also the desired position at that same instant of time? This is exactly what the position profile will tell us!

We can use a little bit of calculus to integrate the speed profile and obtain the position profile seen in figure 4. Otherwise, check the appendix A for the derivation using the kinematic formulas.

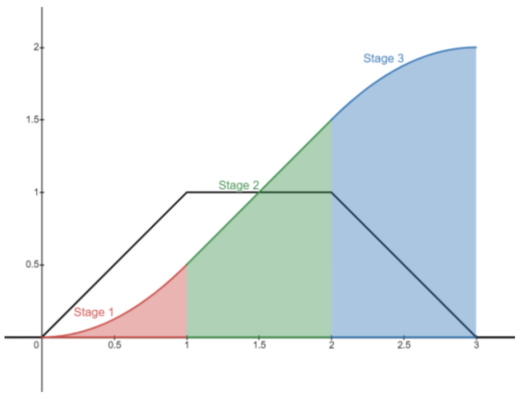

Fig 4. Trapezoidal Position Profile decomposed into its multiple stages.

Once again, we see that the position profile is separated into the same 3 stages as before. The same colors are used in figure 3 and figure 4 to represent the same stages. This time, the polynomials involved to create this shape involve quadratics, but they all still boil down to equations 1 and 3.

At the end of the motion, the position profile shows that the robot arrived at the desired location, and that it arrives there without stopping abruptly. The smooth transition that stage 3 provides from moving at full speed to a complete stop is evident in figure 4. Additionally, the parameter D, which represents the total distance travelled is clearly seen as the maximum height the position profile reaches.

C) *Generalizing*

We can repeat the exact same reasoning as before to generalize for other motion profiles. Sometimes it might be convenient to simplify the problem and use less stages for computational performance reasons. Figure 5 shows an example of a two-stage speed profile, also called a triangular speed profile.

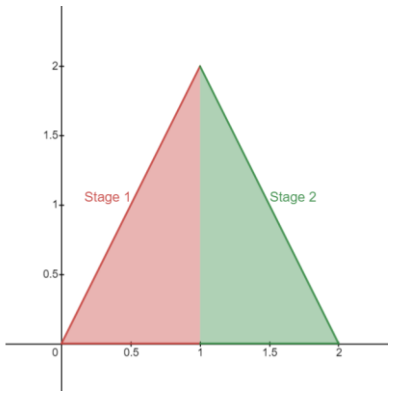

Fig 5. Triangular Speed Profile decomposed into its multiple stages.

The triangular speed profile only has an acceleration stage and a deceleration stage. As a tradeoff for its simplicity, the robot’s motion will be less efficient and will take longer than when the trapezoidal speed profile is used.

Note that this triangular speed profile is a specific case of our trapezoidal speed profile, but with stage 2 eliminated. Therefore, $t_v = 0\,s$.

On the other hand, more stages can be added to the motion profile, incorporating additional parameters such as jerk (rate of change of acceleration). Figure 6 shows an example of a seven stage Motion Profile, also called an S-Curve.

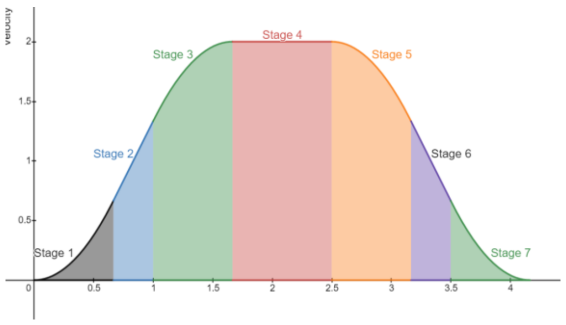

Fig 6. S-Curve Speed Profile decomposed into its seven stages

The S-Curve is much more complicated, but it also brings its own set of benefits. Being able to control the parameter of jerk is useful for situations where vibrations can cause stability issues with the mechanism. In addition, converting the S-Curve to a Trapezoid Speed Profile is as easy as increasing this jerk to a very big number, effectively eliminating stages 1, 3, 5, and 7.

However, these additional advantages come at a cost. The added complexity of the S-Curve makes the coding process much more error prone. Also, it more than doubles the number of stages we must work with, which also means that the code will be more than twice as long. The code will be more than twice as long even without considering the additional terms that the quadratic polynomials now include. It becomes even worse when calculating the position profile, which now involves cubic polynomials.

The added advantages of the S-Curve might outweigh the additional problems it might create in some situations. Control over the snap (rate of change of jerk) of the motion might even be necessary at times, but these kinds of profiles are instead approximated through the use of numerical methods [7]. However, for the purpose of this article, we will stay focused on the Trapezoidal Speed Profile for the sake of simplicity.

IV. Conceptual Implementation using Feedback

On the 1-D path case, the controller used is an open loop control algorithm. This means that the motion profile does not use sensors or feedback to correct for external disturbances. Motion profiles only depend on how much time has passed since the motion began. External disturbances can take the form of unpredictable drifts due to lack of traction with the field, deviating from the path after bumping with field elements, and others. In our case we incorporate feedback using Encoder Odometry [4] to track the robot's X position in the field.

A) Position Profile Inverse for feedback.

Given that the position profile is strictly increasing from 0 to T, it must be one-to-one or invertible. Its inverse's plot is shown in figure 7.

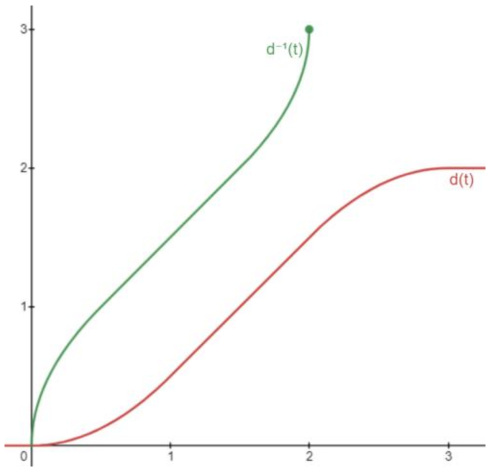

Fig 7. Position profile (red) and its inverse (green)

This inverse position profile now has a time-position relationship. Given a particular position, it computes the time instance it should theoretically be in that specific position.

The time instant t can now be calculated as a function of the position p. In other words, whereas before we had $d(t)$, now we have $t(d)$. This time instant can now be used to calculate the speed the robot will travel at.

Beforehand we only had $v(t)$, where the speed v only depended on the time elapsed from the start of the motion. Now we can now calculate t in terms of p, so now we have $v\big(t(d)\big)$, which could ultimately be expressed as $v(d)$.

In other words, by using the inverse of the position profile (shown in green in figure 7) we can now determine the speed we will travel at using our real progress and taking into account real-life disturbances.

Let us illustrate this through an example. If we start at position $24\,in$ and want to go to position $36\,in$, the total distance to travel D is $36 - 24 = 12$. Applying the inverse position profile to that initial position of 24, the value of time instant is 0, and therefore the output velocity is 0. Moving forward just a few inches to $d = 26$ will now result in a greater value for t when applying the inverse position profile. Now the robot realizes that it is operating in its first stage and still accelerating, so it increases its speed. As the robot gets to its maximum speed, its position increases more rapidly, but now t increases linearly given the relationship of the inverse position profile.

If during this motion the robot loses traction with the field and slows down momentarily, the position measurements will now reflect that. The position will momentarily increase slower. Applying the inverse position profile produces a value of t that reflects this external disturbance, and the algorithm will keep moving the robot at the ideal t given the disturbance.

Finally, when the robot reaches the third stage and starts decelerating, the linear momentum it carries due to its mass might cause it to decelerate slower than anticipated. The position measured might increase at a bigger rate than expected. Applying the inverse position profile now reflects that the value of t is greater than it would be if a simple timer was used. Now that t is closer to the end of the motion, the smaller speed might cause the motor controllers to actually start breaking, which makes the robot decelerate more. If the deceleration parameter was chosen correctly, by the time reaches the position of 36 the value of t will be T and the robot's real speed will be 0.

In conclusion, the inverse position profile can be used with a simple 1-d path to create a custom closed-loop position controller even if this is not the intended use for a motion profile.

V. References

[1] https://en.wikipedia.org/wiki/PID_controller#Pseudocode

[2] https://en.wikipedia.org/wiki/Bang-bang_control

[3] https://www.tutorialspoint.com/fuzzy_logic/fuzzy_logic_introduction.htm

[4] https://thepilons.ca/wp-content/uploads/2018/10/Tracking.pdf

[5] https://okapilib.github.io/OkapiLib/md_docs_tutorials_concepts_twodmotionprofiling.html

[6] https://www.baucomrobotics.com/domino-robot-posts/trajectory-generation

[7] https://www.sciencedirect.com/science/article/abs/pii/S0094114X18317890

Appendix A

Derivation of Trapezoidal Motion Profile Equations

The derivation for the formulas of the trapezoidal speed profile is a simpler than the derivation for the formulas of the position profile. The speed profile also sets up some parameters that are required by the position profile. In addition, it might be simpler to derive the formulas for the position profile by simply integrating the speed profile. However, for those who are not so mathematically oriented, we derive the position profile only using the kinematic formulas taught at introductory physics courses.

It is note-worthy that no single function will perfectly create the shape we desire. Therefore, we will be using piecewise functions, or functions that change behavior based on the input value they are given. One especially important objective is that the parts of these piecewise functions must be continuous, which is going to be a recurring observation we need to point out and will affect some of the computations.

A) *Speed Profile*

The kinematic formula that will be repeatedly used throughout this section is shown in equation A.1.

$$v_{final}(t) = a\Delta t + v_{initial} \qquad \text{(A.1)}$$

During stage 1 we have the parameters of initial speed v_0 and initial acceleration a_0. Also, Δt is as simple as $\Delta t = t - 0$ since the change in time for this stage is with respect to $t = 0$. Therefore, we can substitute these in equation A.1 to obtain the first part of the speed profile:

$$v_1(t) = a_0 t + v_0$$

At the end of this first stage, we will have that $v_1(t_0) = V$. The reason is that we start with velocity v_0 and once we reach the target of V, we will not need to keep accelerating. Therefore, once $t \geq t_0$, we will proceed to the second stage. This observation in conjunction with equation A.1 can be used to calculate t_0:

$$\begin{aligned} V &= a_0(t_0 - 0) + v_0 \\ \Rightarrow \quad V - v_0 &= a_0 t_0 \\ \Rightarrow \quad t_0 &= \frac{V - v_0}{a_0} \end{aligned}$$

Now that we have the formula for the first stage and its duration in the form of t_0, we will proceed with stage 2. This new stage simply consists of a constant velocity V, meaning that the acceleration is 0. Substituting in equation A.1 we will obtain the second part of the speed profile:

$$v_2(t) = V$$

As for the duration of this stage, it will greatly depend on the travel distance D. The parameter D will be defined by the programmer, but we will not have a formula for D until after deriving the formulas for the position profile. Therefore, we'll proceed to stage 3 without computing how long the second stage will last.

We'll begin to calculate the third stage of the speed profile by calculating its duration t_f using equation A.1. At the beginning of the stage, we start with a speed of V and finish with a speed of v_f. Our acceleration will be $-a_f$ since we need to decrease our speed, and the stage will only last t_f, so $\Delta t = t_f$. Similar to how we derived t_0:

$$\begin{aligned} v_f &= -a_f t_f + V \\ \Rightarrow \quad v_f - V &= -a_f t_f \\ \Rightarrow \quad t_f &= \frac{V - v_f}{a_f} \end{aligned}$$

We have t_f, but we don't know when the second stage ends in order to calculate $\Delta t = t - t(end_2)$. Because of this we will create a new variable T that will meet the following relationship:

$$T = t_0 + t_v + t_f$$

Notice that we called the second stage's duration t_v. Additionally, we created the variable T out of seemingly nowhere to determine how much time the full motion will take. We will derive a concrete formula for T after calculating the position profile.

Proceeding to calculate the third stage of the speed profile, $v_{initial} = V$, $a = -a_f$, and $\Delta t = t - (t_0 + t_v) = t - (T - t_f)$. Substituting into equation A.1 and simplifying:

$$\begin{aligned} v_3(t) &= V - a_f\left(t - (T - t_f)\right) \\ &= V - a_f t + a_f\left(T - \frac{V - v_f}{a_f}\right) \\ &= V - a_f t + a_f T - (V - v_f) \\ &= -a_f t + a_f T + v_f \\ &= a_f(T - t) + v_f \end{aligned}$$

In conclusion, the piecewise function that defines the trapezoidal speed profile is:

$$v(t) = \begin{cases} 0 \le t < t_0 & a_0 t + v_0 \\ t_0 \le t < T - t_f & V \\ T - t_f \le t \le T & a_f(T - t) + v_f \end{cases}$$

B) Position Profile

The kinematic formulas used to derive the position profile equations throughout this section are:

$$\Delta x = v_{initial}\Delta t + \frac{1}{2}a\Delta t^2 \qquad \text{(A.2)}$$

$$v_{final}^2 = v_{initial}^2 + 2a\Delta x \qquad \text{(A.3)}$$

Now that we have formulas for t_0 and t_f, there is no need to calculate the durations for any of the stages t_0, t_v, and t_f. We can proceed to directly calculate the formulas for the position profile.

For the first stage, the acceleration is a_0 and the initial velocity is v_0. The Δt parameter is as simple as $\Delta t = t - 0$ since the stage begins at $t = 0$, and $\Delta x = d_1(t) - 0$ since this stage always begins at an initial position of 0. Substituting in equation A.2:

$$d_1(t) = \frac{1}{2}a_0 t^2 + v_0 t$$

The second stage has an acceleration of 0 and a constant velocity V. However, $\Delta t = t - t_0$ and Δx will now be $d_2(t) - d_1(t_0)$. This time, the time and distance travelled in the first stage must be considered. Substituting in equation A.2 and simplifying:

$$\begin{aligned} &d_2(t) - d_1(t_0) = V(t - t_0) \\ &\Rightarrow \quad d_2(t) = V\left(t - \frac{V - v_0}{a_0}\right) + d_1(t_0) \end{aligned}$$

Using equation A.3 to evaluate $d_1(t_0)$:

$$\begin{aligned} &V^2 = v_0^2 + 2a_0(d_1(t_0) - 0) \\ &\Rightarrow \quad d_1(t_0) = \frac{V^2 - v_0^2}{2a_0} \end{aligned}$$

Substituting back:

$$\begin{aligned} d_2(t) &= Vt - V\frac{V - v_0}{a_0} + \frac{V^2 - v_0^2}{2a_0} \\ &= Vt + \frac{V^2 - v_0^2 - 2(V^2 - Vv_0)}{2a_0} \\ &= Vt + \frac{-V^2 + 2Vv_0 - v_0^2}{2a_0} \\ &= Vt - \frac{(V - v_0)^2}{2a_0} \end{aligned}$$

The position profile for the third and final stage comes out to be more complicated. The expansion and simplification process becomes more tedious than the previous calculations. Therefore, some intermediate steps might be skipped by using MATLAB to reduce the amount of work.

To determine the position profile for the final stage, we set the acceleration to $-a_f$, the initial speed is V since we're starting to decelerate starting from that speed. The change in time will be $t - (t_0 + t_v)$, which, as seen in the previous section, this will give us that $\Delta t = t - (T - t_f)$. The change in distance is $\Delta x = d_3(t) - d_2(T - t_f)$. Substituting these parameters into equation A.2:

$$\begin{aligned} &d_3(t) - d_2(T - t_f) \\ &= V\left(t - (T - t_f)\right) + \frac{1}{2}(-a_f)\left(t - (T - t_f)\right)^2 \end{aligned}$$

Evaluating the first term of the right-hand side:

$$\begin{aligned} V\left(t - (T - t_f)\right) &= Vt - VT + Vt_f \\ &= Vt - VT + \frac{V(V - v_f)}{a_f} \\ &= Vt - VT + \frac{V^2 - v_f V}{a_f} \end{aligned}$$

Evaluating the second term of the right-hand side:

$$\begin{aligned} &\frac{1}{2}(-a_f)\left(t - (T - t_f)\right)^2 \\ &\qquad = -\frac{a_f}{2}\left(t - T + \frac{V - v_f}{a_f}\right)^2 \\ &\qquad \cdots \\ &= -\frac{1}{2}a_f t^2 + (v_f - V - Ta_f)t \\ &\qquad - \frac{(v_f - V + Ta_f)^2}{2a_f} \end{aligned}$$

Finally, evaluating the second term of the left-hand side of the expression:

$$d_2(T - t_f) = V(T - t_f) - \frac{(V - v_0)^2}{2a_0}$$

Now we collect like terms, cancel whatever terms can be cancelled, and simplify. The position profile for the third stage turns out to be:

$$\begin{aligned} d_3(t) = -\frac{1}{2}a_f t^2 + (v_f + Ta_f)t - \frac{(V - v_0)^2}{2a_0} \\ - \frac{(v_f - V + Ta_f)^2}{2a_f} \end{aligned}$$

In conclusion, the piecewise function for the position profile is:

$$d(t) = \begin{cases} 0 \le t < t_0 & \frac{1}{2}a_0 t^2 + v_0 t \\ t_0 \le t < T - t_f & Vt - \frac{(V - v_0)^2}{2a_0} \\ T - t_f \le t \le T & -\frac{1}{2}a_f t^2 + (v_f + Ta_f)t - \frac{(V - v_0)^2}{2a_0} - \frac{(v_f - V + Ta_f)^2}{2a_f} \end{cases}$$

C) *Formulas for D and T*

We successfully derived the formulas for the speed profile and the position profile. However, there are still two main unknown variables. The first one is D, which is the total distance we want to travel). The second variable is T, which is the total amount of time the trapezoidal motion profile will take. Calculating D is as simple as evaluating the position profile at time T.

The function is long, and the procedure is tedious, so once again I'll employ the help of MATLAB to help substitute and simplify the expression. It turns out that:

$$D = d(T) = TV - \frac{(V - v_f)^2}{2a_f} - \frac{(V - v_0)^2}{2a_0}$$

D is determined by a very pretty formula! (Unlike the position profile's third stage, anyways)

D is defined in terms of T, but the programmer will define D when creating the paths for the robot's motion. Therefore, it is much more convenient to calculate T, as we can see it is used multiple times during the speed profile and the position profile. Solving for T:

$$\begin{aligned} D &= TV - \frac{(V - v_f)^2}{2a_f} - \frac{(V - v_0)^2}{2a_0} \\ \Rightarrow \quad TV &= D + \frac{(V - v_f)^2}{2a_f} + \frac{(V - v_0)^2}{2a_0} \\ \Rightarrow \quad T &= \frac{D}{V} + \frac{(V - v_f)^2}{2Va_f} + \frac{(V - v_0)^2}{2Va_0} \end{aligned}$$

Ta-da! We finally have formulas for the two unknown variables we had left over. It's just a matter of computing the value of T before calculating the speed profile and the position profile.

D) Restrictions for V

We're not done yet! We can't simply plug in any value we want. If, for example, we have a very small acceleration a_0 but a very high maximum speed V, it might take a long time before reaching that maximum speed V. By that point we might have missed and overshot our desired position D, which is less than desirable. Because of this we must create restrictions for each of the variables. Let's start with the formulas for t_0 and t_f.

Time must always be positive, so we can constrain these two to be greater than or equal to 0.

$$t_0 \geq 0 \quad \Rightarrow \quad \frac{V - v_0}{a_0} \geq 0$$

$$t_f \geq 0 \quad \Rightarrow \quad \frac{V - v_f}{a_f} \geq 0$$

For the sake of simplicity, assume that $v_0 \geq 0$, $a_0 \geq 0$, $v_f \geq 0$, and $a_f \geq 0$. Therefore $V \geq \max(v_0, v_f)$ must be true for $t_0 \geq 0$ and $t_0 \geq 0$. We only have lower bounds on these seven variables, but as we will see later, this is all we need. More constraints than these might over-complicate the profiles and reduce some of their flexibility.

Afterwards, let's consider what constraints the rest of the variables need. T and t_v are still time, so they must be positive. On the other hand, D should have the same sign as V since that sign will indicate what direction the robot will move. Therefore, $D > 0$.

Let's now consider t_v more deeply. $t_v = T - t_0 - t_f$. Expanding this expression using MATLAB once again gives us:

$$t_v = \frac{D}{V} - \frac{V^2 - v_f^2}{2a_f} - \frac{V^2 - v_0^2}{2a_0} \geq 0$$

This formula gives us a lot of flexibility on what to restrict. Let's consider the possibilities.

If we restrict D with strict boundaries, the distance the programmer wants the robot to move will <u>not</u> always be followed. If the distance is instead provided by another algorithm, this can be even more problematic. Therefore, we should let D be a free variable.

A similar logic can be applied to a_0 and a_f. If we determined a specific value for these accelerations, anything greater might create problems with wheel slippage and anything smaller might create a movement that's too slow.

In the case of v_0 and v_f, we might not always have control over them. If the mechanism is currently moving and we want to start applying motion profiles and can't freely set v_0 and v_f, the robot might experiment jerks and other behavior that might damage the mechanisms in the long term.

Therefore, the remaining parameter V is the parameter we would want to constrain, which makes sense. If V has an upper bound, we can set V in between $\max(v_0, v_f)$ and that upper bound. V is involved in all the formulas that constrain how long each stage will last, so the different sections of the profile will be directly affected. In addition, it affects how fast we arrive at the destination, and would be affected by the acceleration parameters. If the acceleration parameters are big enough, the second stage will have an opportunity to predominate. On the other hand, if the acceleration parameters are small, the upper bound for V will be smaller and we won't overshoot our desired position.

Solving $t_v \geq 0$ for the parameter V using MATLAB:

$$V \leq \sqrt{\frac{a_f v_0^2 + a_0 v_f^2 + 2D a_0 a_f}{a_0 + a_f}}$$

Now that the parameters are constrained, we can finally say that we finished deriving the Trapezoidal Motion Profile.

E) Summary

1) D – Total travel distance
2) V – Maximum motion velocity

$$\max(v_0, v_f) < V \leq \sqrt{\frac{a_f v_0^2 + a_0 v_f^2 + 2Da_0 a_f}{a_0 + a_f}}$$

3) T – Total travel time

$$T = \frac{D}{V} + \frac{\left(V - v_f\right)^2}{2Va_f} + \frac{(V - v_0)^2}{2Va_0} \geq 0$$

4) a_0 – Initial acceleration > 0
5) a_f – Final acceleration > 0
6) v_0 – Initial speed ≥ 0
7) v_f – Final speed ≥ 0
8) t_0 – Time taken by 1st stage

$$t_0 = \frac{V - v_0}{a_0} \geq 0$$

9) t_v – Time taken by 2nd stage

$$t_v = T - t_0 - t_f = \frac{D}{V} - \frac{V^2 - v_f^2}{2a_f} - \frac{V^2 - v_0^2}{2a_0} \geq 0$$

10) t_f – Time taken by 3rd stage

$$t_f = \frac{V - v_f}{a_f} \geq 0$$

Speed Profile

$$v(t) = \begin{cases} 0 \leq t < t_0 & a_0 t + v_0 \\ t_0 \leq t < T - t_f & V \\ T - t_f \leq t \leq T & a_f(T - t) + v_f \end{cases}$$

Position Profile

$$d(t) = \begin{cases} 0 \leq t < t_0 & \frac{1}{2}a_0 t^2 + v_0 t \\ t_0 \leq t < T - t_f & Vt - \frac{(V - v_0)^2}{2a_0} \\ T - t_f \leq t \leq T & -\frac{1}{2}a_f t^2 + \left(v_f + Ta_f\right)t - \frac{(V - v_0)^2}{2a_0} - \frac{\left(v_f - V + Ta_f\right)^2}{2a_f} \end{cases}$$

F) Final note

See the `TrapezoidProfile.m` Matlab script and its extensive comments to see how we can use Calculus to derive these exact same formulas. Section 4 (starting in line 59) is the most relevant. The results of the speed profile are stored in $v(t)$ as a piecewise function. The results of the position profile are stored in $d(t)$ also as a piecewise function. Conditions for each one of the variables is summarized when the `assumptions` command is run.

Octave will NOT work as intended since there the commands like `piecewise` and `assumeAlso` that don't exist. At least you can retrieve each stage individually by using `v_1(t)`, `v_2(t)`, and `v_3(t)` (the same with `d_1(t)`, `d_2(t)`, and `d_3(t)`)