Generate stepper motor S-shaped motion profiles using sigmoid functions

Patrick Sorn

April 23, 2019

Contents

1 Trapezoidal speed ramp

In order to understand how stepper motor ramp profiles work, we have to dive into the most commonly used and most applicable profile type: the trapezoidal ramp profile. It consists in general of 3 phases: An acceleration phase T_a, a constant velocity phase T_c and a deceleration phase T_d. The sum of the 3 phases constitutes the total time T (see Fig. 1). The trapezoidal ramp profile is the simplest solution, as it can be represented by the standard physics formula for velocity with constant acceleration.

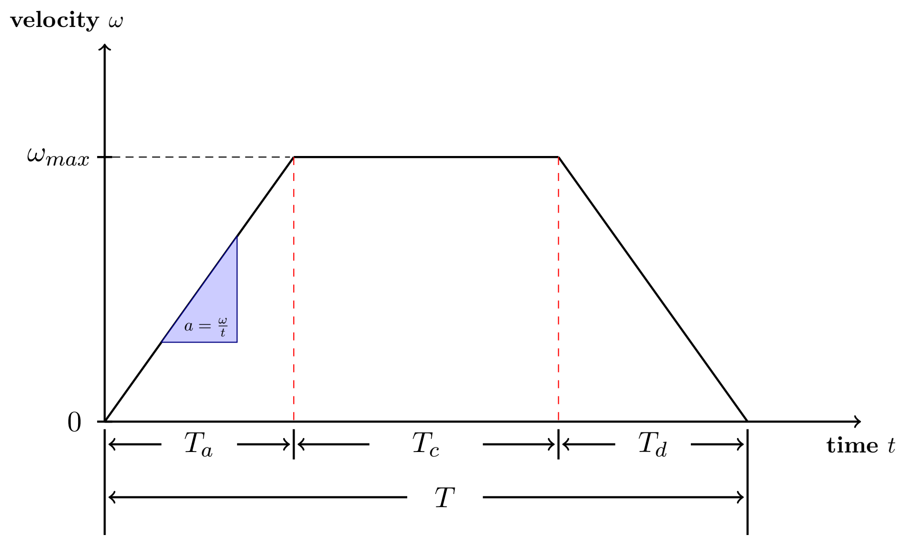


Figure 1: Trapezoidal ramp in detail.

1.1 The velocity ω

Due to the fact, that we have a constant increase in velocity respectively constant acceleration, we can use the standard physics formula for constant acceleration a during T_a, which is as follows:

$$\omega(t) = a \cdot t \qquad \textbf{[Equation 1]}$$

1.2 Calculation of motor shaft angle θ

In order to get the position of the motor at a certain point in time, we have to integrate [**Equation 1**]. See Figure 2 for detail.

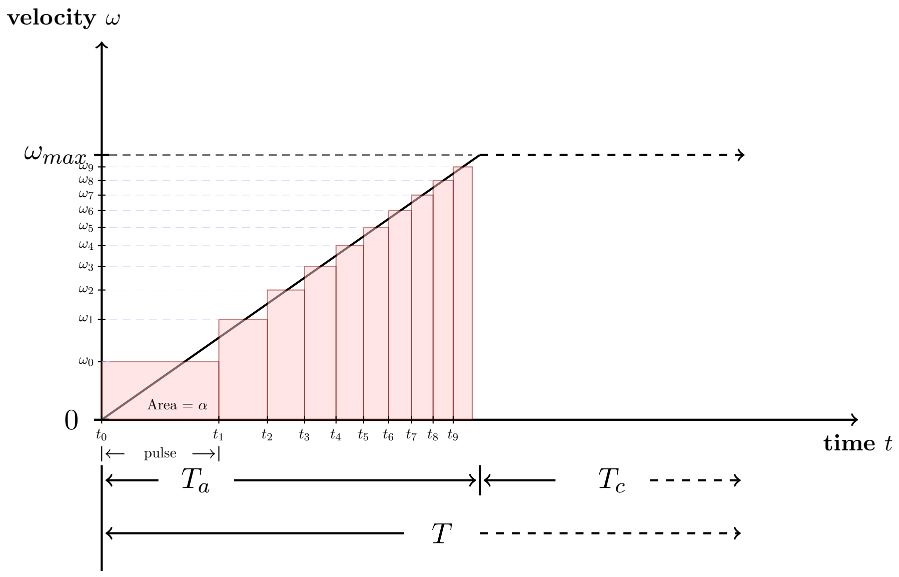


Figure 2: Ramp geometry showing n=10 steps.

$$\theta(t) = \int_0^t \omega(\tau) d\tau = \frac{a \cdot t^2}{2} = n \cdot \alpha \qquad \textbf{[Equation 2]}$$

where $n \geq 1$ is the step number and α the step angle in radians.

1.3 Calculation of timer delay t_n

To get the time point of step number n, we need to solve [**Equation 2**] for t:

$$t_n = \sqrt{\frac{2 \cdot n \cdot \alpha}{a}} \qquad \text{[\textbf{Equation 3}]}$$

To get the timer delay between two time points t_{n+1} and t_n, we just subtract them from one another:

$$c_n = t_{n+1} - t_n \qquad \text{[\textbf{Equation 4}]}$$

1.4 Calculation of the number of steps n

Lastly, we need to calculate the number of steps, which are necessary to reach maximum velocity. We do that by combining [**Equation 1**] and [**Equation 2**].

From [**Equation 1**] we do get:

$$t = \frac{\omega}{a}$$

Inserting this into [**Equation 2**]:

$$\frac{a \cdot \left(\frac{\omega}{a}\right)^2}{2} = n \cdot \alpha$$

And solving for n:

$$n = \frac{\omega^2}{2 \cdot \alpha \cdot a} \qquad \text{[\textbf{Equation 5}]}$$

2 S-shaped speed ramp

In general, trapezoidal ramp profiles (Fig. 1) are used for stepper motor acceleration. The problem using trapezoidal ramp profiles is that the system experiences full acceleration right from the beginning, which can either lead to misstepping (signal does not last long enough for the stepper motor to execute a single step) or reaching a lower maximum speed. To prevent this behavior we can use a S-shaped ramp profile (Fig. 3). S-shaped profiles consist of a concave period and a convex period. During the concave period acceleration increases to a certain point, also called the inflection point, and then slowly decreases during concave period until maximum velocity is reached. Therefore the system experiences maximum acceleration at the inflection point.

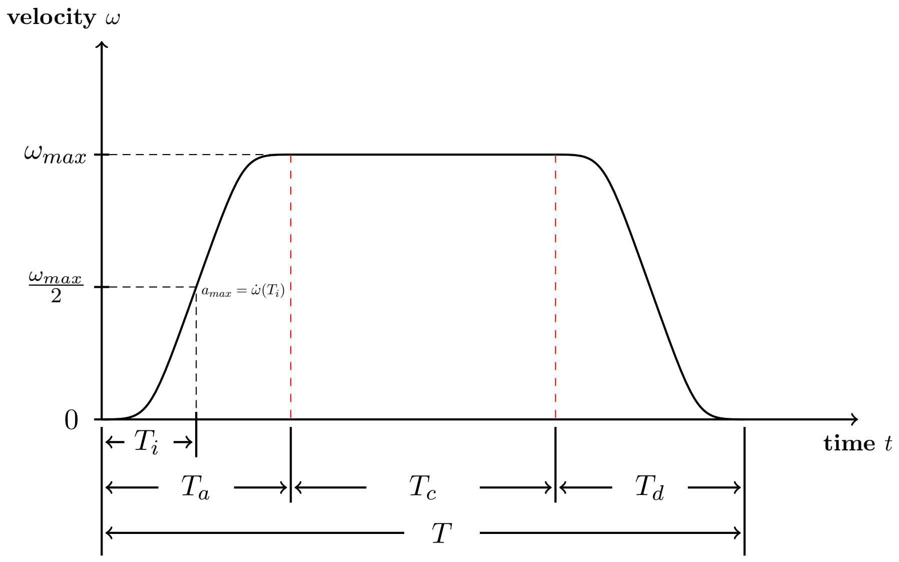


Figure 3: S-shaped ramp in detail.

To represent the S-shaped acceleration profile ($0 \leq t \leq T_a$) we can use a special case of a sigmoid function, the logistic function, which has the form:

$$S(t) = \frac{1}{1 + \mathrm{e}^{-t}}$$

2.1 Finding a suitable velocity function ω

To be able to use this function, we have to modify it to meet certain criteria:

$$
\begin{aligned}
S(T_a) &= \omega_{max} \\
S(T_i) &= \frac{\omega_{max}}{2} \\
\dot{S}(T_i) &= a_{max}
\end{aligned}
$$

where T_i is the time period to the inflection point, T_a is the total acceleration period and ω_{max} is the maximum velocity.

Using this criteria we can generate a final modified version of the logistic function to represent velocity ω on a S-shaped ramp profile:

$$\omega(\tau) = \frac{\omega}{\mathrm{e}^{\left(\frac{4 \cdot a}{\omega}\right) \cdot (T_i - \tau)} + 1} \qquad [\textbf{Equation 1}]$$

where ω is the maximum velocity and a is the maximum acceleration (respectively the acceleration at the inflection point T_i) that can be reached.

2.2 Calculation of motor shaft angle θ

Integration of $\omega(\tau)$ gives the motor shaft angle $\theta(t)$:

$$\theta(t) = \int_0^t \omega(\tau) d\tau = \frac{\omega^2 \left(\ln\left(\mathrm{e}^{\frac{4 \cdot a \cdot t}{\omega}} + \mathrm{e}^{\frac{4 \cdot a \cdot T_i}{\omega}} \right) - \ln\left(\mathrm{e}^{\frac{4 \cdot a \cdot T_i}{\omega}} + 1 \right) \right)}{4 \cdot a} = n \cdot \alpha \quad [\textbf{Equation 2}]$$

where $n \geq 1$ is the step number and α the step angle in radians.

2.3 Calculation of timer delay t_n

To get an exact timer delay (pulse width) for the stepper motor STEP signal, we need to solve [**Equation 2**] for t:

$$
\begin{aligned}
\frac{\omega^2\left(\ln\left(\mathrm{e}^{\frac{4\cdot a\cdot t}{\omega}}+\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}}\right)-\ln\left(\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}}+1\right)\right)}{4\cdot a} &= n\cdot\alpha \\
\ln\left(\mathrm{e}^{\frac{4\cdot a\cdot t}{\omega}}+\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}}\right)-\ln\left(\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}}+1\right) &= \frac{4\cdot a\cdot n\cdot\alpha}{\omega^2} \\
\ln\left(\frac{\mathrm{e}^{\frac{4\cdot a\cdot t}{\omega}}+\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}}}{\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}}+1}\right) &= \frac{4\cdot a\cdot n\cdot\alpha}{\omega^2} \\
\mathrm{e}^{\ln\left(\frac{\mathrm{e}^{\frac{4\cdot a\cdot t}{\omega}}+\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}}}{\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}}+1}\right)} &= \mathrm{e}^{\frac{4\cdot a\cdot n\cdot\alpha}{\omega^2}} \\
\frac{\mathrm{e}^{\frac{4\cdot a\cdot t}{\omega}}+\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}}}{\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}}+1} &= \mathrm{e}^{\frac{4\cdot a\cdot n\cdot\alpha}{\omega^2}} \\
\mathrm{e}^{\frac{4\cdot a\cdot t}{\omega}} &= \left(\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}}+1\right)\left(\mathrm{e}^{\frac{4\cdot a\cdot n\cdot\alpha}{\omega^2}}\right)-\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}} \\
\ln\left(\mathrm{e}^{\frac{4\cdot a\cdot t}{\omega}}\right) &= \ln\left(\left(\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}}+1\right)\left(\mathrm{e}^{\frac{4\cdot a\cdot n\cdot\alpha}{\omega^2}}\right)-\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}}\right) \\
\frac{4\cdot a\cdot t}{\omega} &= \ln\left(\left(\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}}+1\right)\left(\mathrm{e}^{\frac{4\cdot a\cdot n\cdot\alpha}{\omega^2}}\right)-\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}}\right) \\
t &= \omega\frac{\ln\left(\left(\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}}+1\right)\left(\mathrm{e}^{\frac{4\cdot a\cdot n\cdot\alpha}{\omega^2}}\right)-\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}}\right)}{4\cdot a}
\end{aligned}
$$

After simplification we obtain:

$$
t_n=\omega\frac{\ln\left(\left(\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}}+1\right)\cdot\left(\mathrm{e}^{\frac{4\cdot a\cdot n\cdot\alpha}{\omega^2}}\right)-\mathrm{e}^{\frac{4\cdot a\cdot T_i}{\omega}}\right)}{4\cdot a}
$$

[**Equation 3**]

The exact delay between the n-th and $(n+1)$-th step pulse can then be calculated as follows:

$$
c_n = t_{n+1} - t_n
$$

[**Equation 4**]

2.4 Calculating the number of steps n

Finally, we need to calculate the number of steps that are necessary for the stepper motor to reach maximum velocity. As our defined velocity function (see [**Equation 1**])only reaches its maximum value while t being infinity, we increase the maximum velocity ω by 0.5% to get a finite number of steps. We can combine [**Equation 1**] and [**Equation 2**] to get the number of steps.

Solving [**Equation 1**] for t, we get:

$$t = T_i - \frac{\ln\left(1.005 \cdot \omega - 1\right)}{\frac{4 \cdot a}{\omega}}$$

Now we can insert t into [**Equation 2**] and obtain:

$$n = \frac{\omega^2 \left(\ln \left(\mathrm{e}^{\frac{4 \cdot a \cdot \left(T_i - \frac{\ln\left(\frac{\omega}{0.995 \cdot \omega} - 1\right)}{\frac{4 \cdot a}{\omega}} \right)}{\omega}} + \mathrm{e}^{\frac{4 \cdot a \cdot T_i}{\omega}} \right) - \ln \left(\mathrm{e}^{\frac{4 \cdot a \cdot T_i}{\omega}} + 1 \right) \right)}{4 \cdot a \cdot \alpha} \qquad \text{[\textbf{Equation 5}]}$$

3 Implementation

3.1 Pseudocode

Next, we want to see how such a motion profile of a stepper motor can be implemented. Therefore, we have this pseudocode snippet:

Algorithm 1 Calculate ramp profile

```
1:  $c \leftarrow$ empty list
2:  $n \leftarrow calculateNumberOfSteps()$
3:  $i \leftarrow 0$
4:  while $i < n$ do
5:      $t_{n+1} \leftarrow calculateT(n+1)$
6:      $t_n \leftarrow calculateT(n)$
7:      $c_n \leftarrow t_{n+1} - t_n$
8:      $c[i] \leftarrow c_n$
9:      $i \leftarrow i + 1$
10: end while
```

3.2 Implementation in Python

Below is an implementation example of a trapezoidal speed ramp in Python.

```python
import math

step_angle = 1.8     # Step Angle of stepper motor in degrees
microsteps = 2       # Microstepping mode
travel_per_rev = 5   # Axis travel per stepper motor revolution
accel = 80.0         # Acceleration in [mm/s^2]

sqrt = math.sqrt
spr = 360.0 / step_angle * microsteps
steps_per_mm = spr / travel_per_rev
step_angle_in_rad = 2 * math.pi / spr
cf = vm / 60 * steps_per_mm * step_angle_in_rad
accel_fact = accel * steps_per_mm * step_angle_in_rad
num_steps = int(round(cf * cf / (2 * step_angle_in_rad * accel_fact)))
c0 = sqrt(2 * step_angle_in_rad / accel_fact)
c = [c0]
for i in range(1, num_steps):
    cn = c0 * (sqrt(i+1) - sqrt(i))
    c.append(cn)
c_total = sum(c)
```

Algorithm 1: Python example of trapezodial speed ramp calculation

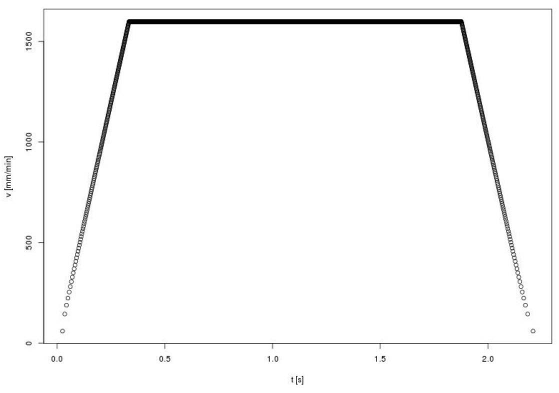
Next, we have an implementation example of a sigmoidal acceleration ramp in Python.

```python
import math

step_angle = 1.8     # Step Angle of stepper motor in degrees
microsteps = 2       # Microstepping mode
travel_per_rev = 5   # Axis travel per stepper motor revolution
accel = 80.0         # Acceleration in [mm/s^2]

sqrt = math.sqrt
spr = 360.0 / step_angle * microsteps
steps_per_mm = spr / travel_per_rev
step_angle_in_rad = 2 * math.pi / spr
cf = vm / 60 * steps_per_mm * step_angle_in_rad
accel_fact = accel * steps_per_mm * step_angle_in_rad

T_2 = 0.4
# pre-calculated values
a_func = math.e**(4 * accel_fact * T_2 / cf)
t_tmp = T_2 - math.log(cf/(0.995*cf)-1)/(4*accel_fact/cf)

num_steps = int(round(cf**2 * (math.log(math.e**(4*accel_fact*t_tmp/cf) +
    a_func) - math.log(a_func + 1)) / (4 * accel_fact * step_angle_in_rad)))
c = []
for i in range(num_steps):
    cn_i_plus_one = cf * math.log((a_func + 1) * (math.e**(4 * accel_fact * (
    i+1) * step_angle_in_rad / cf**2)) - a_func) / (4 * accel_fact)
    cn_i = cf * math.log((a_func + 1) * (math.e**(4 * accel_fact * i *
    step_angle_in_rad / cf**2)) - a_func) / (4 * accel_fact)
    cn = cn_i_plus_one - cn_i
    c.append(cn)
c_total = sum(c)
```

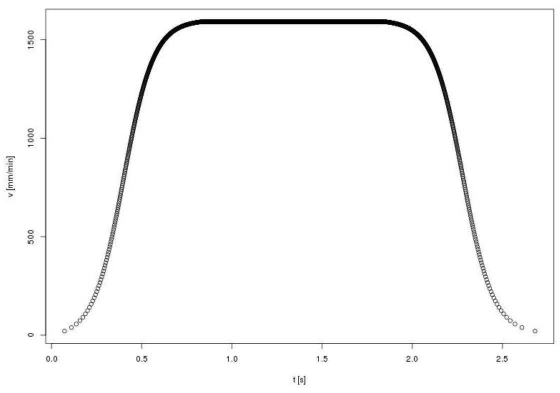
Algorithm 2: Python example of sigmoidal speed ramp calculation

3.3 Output

Below are some examples of calculated profiles with the code above. Every point represents a step signal.



(a) trapezoidal



(b) S-shaped

Figure 4: Comparison of ramp profiles.