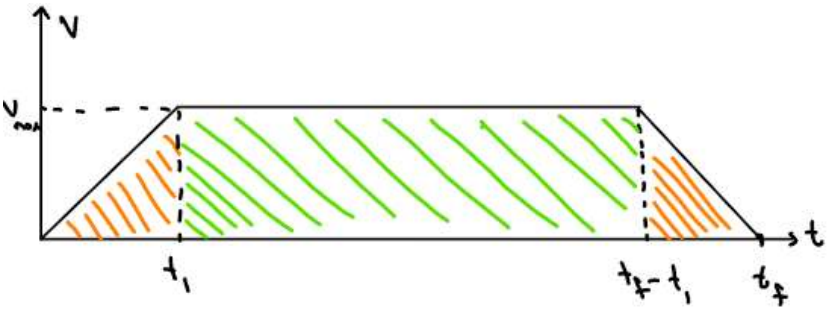
Trapezoidal Velocity Profile

Serdar Hakan Argüz, 2022

Velocity Profile



We can formulate the velocity profile as a piecewise function of time. It will also require 3 constant inputs: a_{max}, V_{max} and D.

func $vel\_trap(a_{max}, V_{max}, D, t)$:

a_{max} : maximum acceleration / deceleration to use
V_{max} : maximum velocity to reach
D : total distance to cover. **Remark:** total distance should be enough to give enough space to accelerate + decelarate.
t : current value of time, between 0 and t_f. This function returns the value of the velocity at this time.

- $t_1 = V_{max} \,/\, a_{max}$

$$D = D_1 + D_2 = V_{max} t_1 + (t_f - 2t_1) V_{max}$$
$$V_{max}(t_f - t_1) = D$$
$$t_f - t_1 = D / V_{max}$$

- $t_f = D/V_{max} + t_1$

- $$V(t) = \begin{cases} a_{max} t & 0 \le t \le t_1 \\ V_{max} & t_1 < t \le (t_f - t1) \\ V_{max} - a_{max}(t - t_f + t_1) & (t_f - t1) < t \le t_f \end{cases}$$

Here is an example implementation of this function in MATLAB.

```
function [vel_trap] = vel_trap(amax,Vmax, D, t)
t1 = Vmax / amax;
tf = D / Vmax + t1;

    if t >= 0 && t <= t1
        vel_trap = amax * t;
    elseif t > t1 && t <= (tf - t1)
        vel_trap = Vmax;
    elseif t > (tf-t1) && t <= tf
        vel_trap = Vmax - amax * (t - tf + t1);
    else
        vel_trap = 0;
    end

end
```

Such velocity profile functions are often used in numeric integrals to generate trajectories, such as the values for a joint variable.
It is certainly true that arbitrary velocity profiles (with smoothed ends, etc.) may require this numerical integration approach.
However, it is not needed for this simple trapezoidal velocity profile. Because this profile can directly be integrated.
Doing so gives a closed-form, fully analytical solution for the position level trajectory.

Position Profile

Trapezoidal profile is a piecewise function. If you integrate it over these parts, and solve for the integration constants to satisfy the continuity constraints, you will end-up with the following formulation.

- $t_1 = V_{max} \,/\, a_{max}$
- $t_f = D/V_{max} + t_1$
- $x(t) = \begin{Bmatrix} x_1(t) & 0 \le t \le t_1 \\ x_2(t) & t_1 < t \le (t_f - t1) \\ x_3(t) & (t_f - t1) < t \le t_f \end{Bmatrix}$

Where
$x_1(t) = 0.5\, a_m\, t^2$
$x_2(t) = V_{max} t \; - \; 0.5\, a_m\, t_1$
$x_3(t) = \; -0.5\, \mathrm{a}_{max} \left(t - t_f\right)^2 \; + D$

Here is an example implementation in MATLAB.

```
function [trajectory] = trajectory_trapvel(amax, Vmax, D)
% Generates a position trajectory using trapezoidal velocity.
% uses a closed form analytical formulation for the position.
t1 = Vmax / amax;
tf = D/Vmax + t1;

T = 0:0.1:tf;
X = linspace(0,0,length(T));

for i = 1:length(T)
t = T(i);
    if(t < t1)
        X(i) = 0.5*amax*t^2;

    elseif( t >= t1 && t< (tf - t1))
        X(i) = Vmax*t - 0.5*Vmax*t1;

    elseif ( t >= (tf-t1) && t <= tf)
        X(i) = Vmax*t - 0.5*Vmax*t1 - 0.5*amax*t^2 - amax*t*t1 + amax*t*tf + amax*t1*(tf-t1) -
amax*tf*(tf-t1) + 0.5*amax*(tf-t1)^2

    else
        X(i) = X(i-1);
    end
end

trajectory = [T' , X'];

end
```

Having a fully closed form solution allows for interesting possibilities.
For example, you can solve the inverse of the position function to solve for time, in terms of position (or archlength).

If you are given a path, you can compute its archlength along each of its indices.
Then, using the invers formulation, you can solve for the time values each of these indices should have.
This allows you to get a path, and convert it into a trajectory by finding appropriate time values for each index.

Trajectory Generation

You are given a path in plane or space with some coordinates. First, compute its arclength numerically at each index.

Arc-Length

On a plane,	**In space,**
If the path is some kind of polynomial, you will already have an analytical formula for dy/dx. For example; $$dS = \sqrt{1 + \left(\frac{dy}{dx}\right)^2}\, dx$$ Or, if you only have the X and Y arrays, then: $$\Delta S = \sqrt{\Delta x^2 + \Delta y^2}$$	Let $i = \ index\ number$ to serve as the parametrization. $$\frac{dS}{di} = \sqrt{\left(\frac{dx}{di}\right)^2 + \left(\frac{dy}{di}\right)^2 + \left(\frac{dz}{di}\right)^2}$$ Numerically, $\Delta S = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$

Time Parametrization

Now, we are ready to solve for time for some known arch-length. The parameters V_{max} and a_{max} are given.

- $D = \max(S) = S(end)$
- $t_1 = V_{max} \,/\, a_{max}$
- $t_f = D/V_{max} + t_1$

We can also compute the arclengths at the times t_1 , $(t_f - t_1)$ and t_f.

- $S_1 = 0.5\, a_{max}\, t_1^2$
- $S_2 = D - S_1$
- $S_3 = D$

Now, what is left is to express time as a function of archlength.

- $$T(S) = \begin{Bmatrix} T_1(S) & 0 \le S \le S_1 \\ T_2(S) & S_1 < S \le S_2 \\ T_3(S) & S_2 < S \le S_3 \end{Bmatrix}$$

Where

$$T_1(S) = \sqrt{\frac{2S}{a_{max}}}$$

$$T_2(S) = t_1 + \frac{S - S_1}{V_{max}}$$

$$T_3(S) = t_f - \sqrt{\frac{2(D-S)}{a_{max}}}$$

Here is an example implementation in MATLAB.

```
function [T] = parametrize(amax,Vmax,S)
% T: time
D = S(end);
t1 = Vmax / amax;
tf = D/Vmax + t1;

S1 = 0.5*amax*t1^2;
S2 = D - S1;
S3 = D;

T = linspace(0,0, length(S));
for i = 1:length(S)
    s = S(i);
    if( s <= S1)
        T(i) = sqrt( 2 * s / amax);

    elseif( s > S1 && s <= S2)
        T(i) = t1 + (s-S1)/Vmax;

    elseif( s > S2 && s <= S3)
        T(i) = tf - sqrt( 2 * (D - s) / amax);

    else
        T(i) = -1; % cannot enter this case.
    end
end

end
```