

Arm Tutorial 1

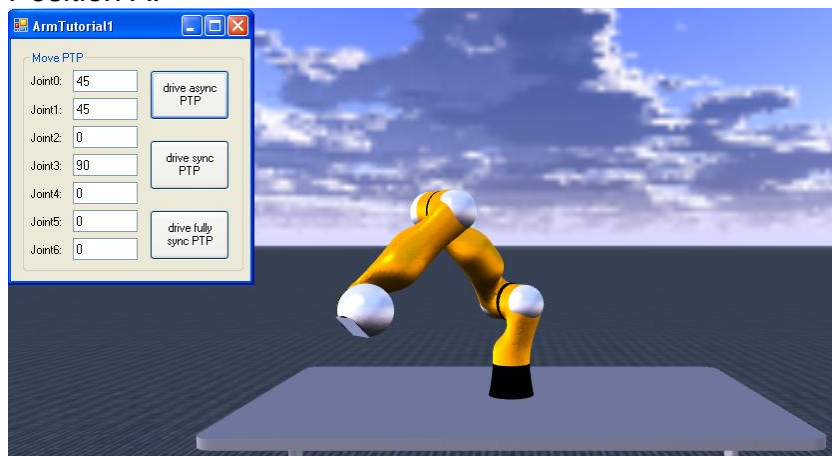
Arm Tutorial 1 project build order

1. ArticulatedArm (in abstract services)
2. Util
3. SimulatedLBR3Arm
4. KUKATutorial1MotionPlanning
5. KUKATutorial1Dashboard
6. KUKAArmTutorialSimulation

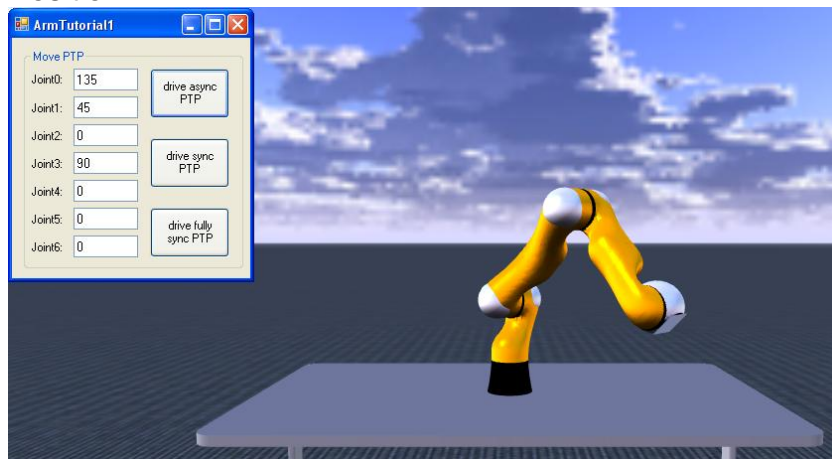
Moving the arm from point to point

To move a tool from position A to position B using a six axis robot, you have to tell each axis of the robot which positions they should assume so that the end of arm reaches the desired position and orientation.

Position A:



Position B:



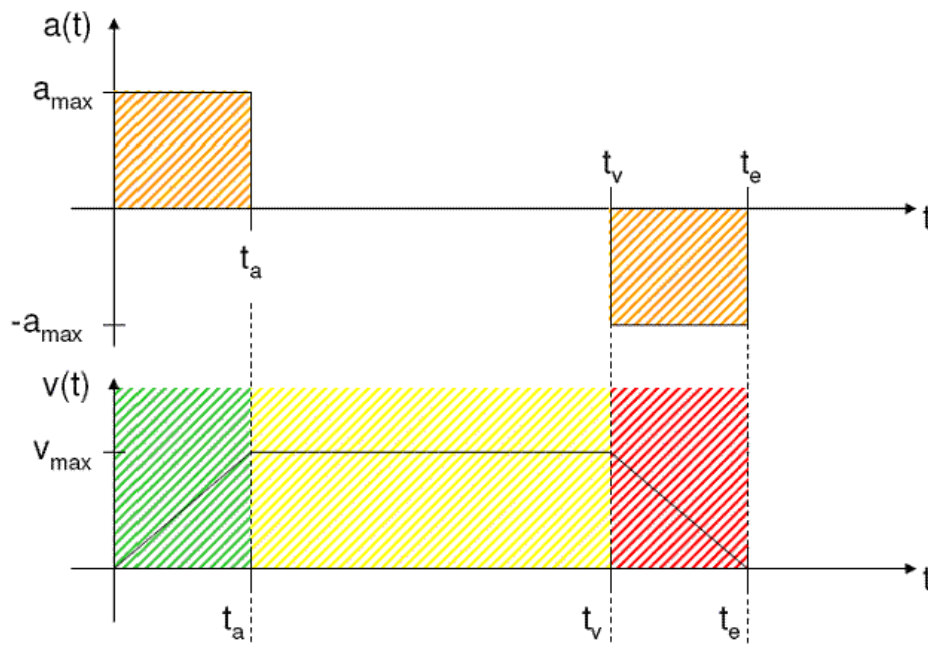
Axis specific programming:

One way to tell the robot its destination is to give it a target value for each axis.

To get to the target position the controller needs to calculate the required movements for each axis. Unless specified otherwise by the operator, the arm control takes the current axis values and calculates the shortest distances to the desired axis values to avoid large and long movements.

Movement of one axis:

To reach the target value, the drive for an axis needs to accelerate to a certain velocity, and to decelerate in time before it reaches the target. In order to plan these times correctly you need to calculate a so called velocity-profile for the axis. The following picture shows this profile for one axis – the upper curve shows the acceleration input for the drive, the lower picture shows the resulting velocity (LINK See the mathematical background here)



Green area: acceleration phase
Yellow area: constant velocity phase
Red area: deceleration phase

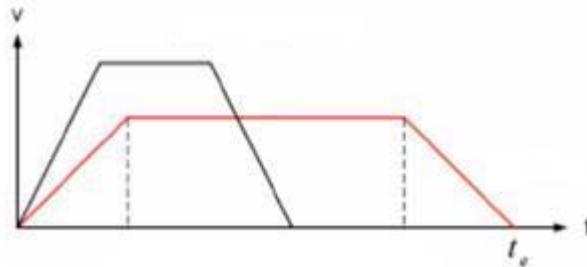
The velocity profile is used to send precise axis values at predetermined time intervals to the drive. The time interval must be well adapted to the target drive. In the MSRS simulation environment, since the drives are modeled as springs, we need to find a good interval to hide the spring nature of the drives at reasonable motion speeds. We found that we come to a nice robotic motion when using 10ms as the interval length.

Movement of multiple axes:

In order to reach the EOA target position, usually multiple axis have to move. A velocity profile has to be calculated for every axis. For the overall motion, those profiles can be coordinated in three different ways:

Asynchronous PTP motion:

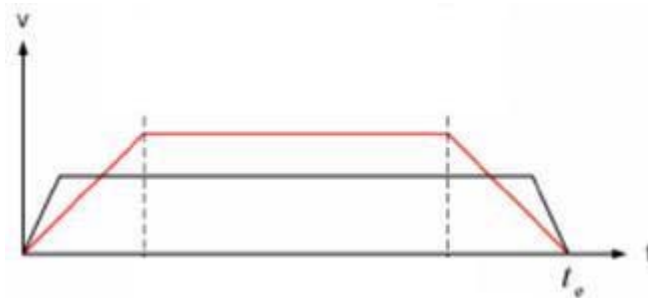
In an asynchronous PTP motion, each drive moves as fast as possible to reach the desired angle. In this case, some axis will reach their target value before others. In order to reach the final position and orientation, each axis has to have arrived at its final value. The velocity profiles of two axes could look like the following:



NOTE: In this picture, the maximum velocities are shown at different levels for the two axes since in the real world the different drives do have different maximum velocities and acceleration capabilities.

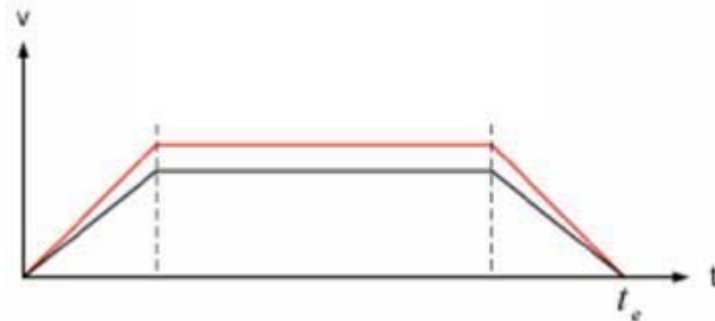
Synchronous PTP motion:

Since the final position of the EOA is not reached before each axis has reached the target angle, one axis will need the most driving time. This axis is called the leading axis. To minimize abrasion of the drives, all drives except the one for the leading axis should drive on a slower than maximum speed. So the time that is needed to reach the destination position is synchronized. The velocity profiles for the slower moving axis are recalculated, so that the times for arriving at the end positions match:



Fully synchronous PTP motion:

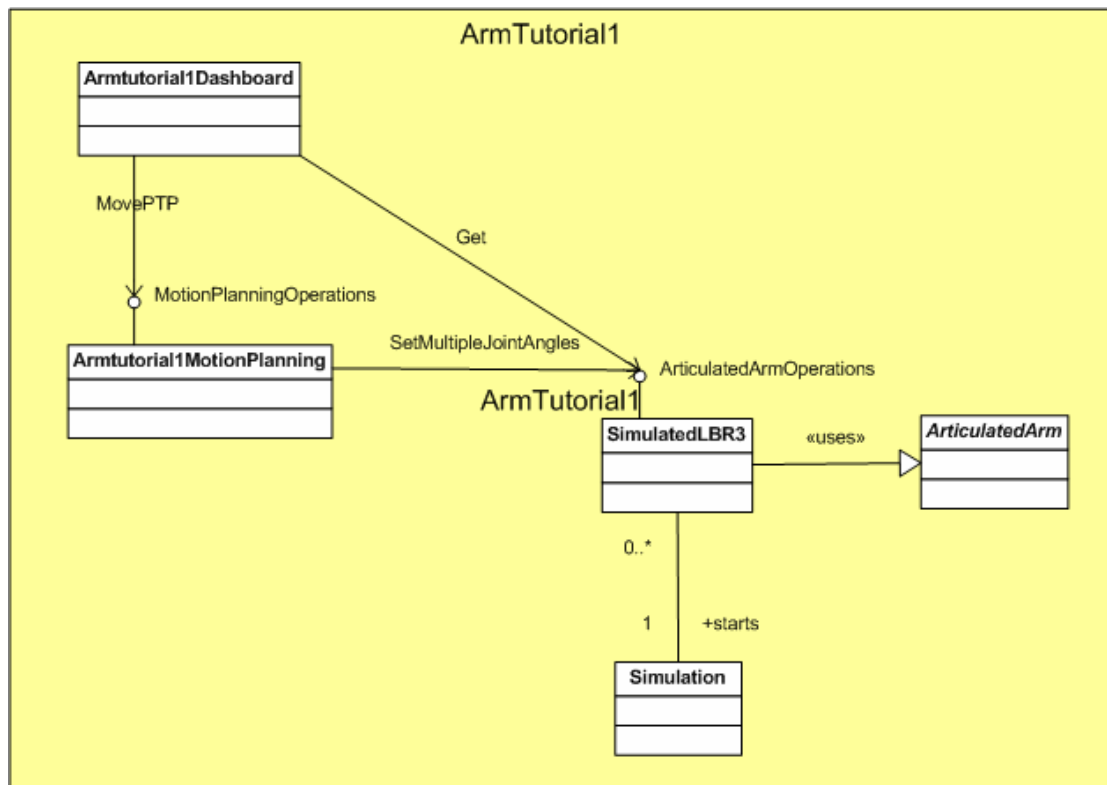
Abrasion of drives is even more prevented if not only the time to reach the destination is synchronized, but also the times for acceleration and deceleration are synchronized. This is called a fully synchronized PTP motion.



Coding for Arm Tutorial 1:

In the robotics field, calculating these profiles is called motion planning. Therefore we created a service called ArmTutorial1MotionPlanning. The '1' indicates that we have different versions of the motion planning service throughout the tutorials. This has been done for clarity reasons, to only include the necessary code for the corresponding tutorial.

The following figure shows the orchestration of the main services in ArmTutorial1:



- ArmTutorial1Dashboard: User Interface
- ArmTutorial1MotionPlanning: Calculates and executes the motion
- SimulatedLBR3: Service for the simulated LBR arm
- ArticulatedArm: Abstract Base Class for SimulatedLBR3
- KUKAArmTutorialSimulation: Startproject of the ArmTutorial1 Solution

This tutorial executes through the following steps:

- 1) The dashboard accepts target angle values and sends those through a PTP motion command to the motion planning service.
- 2) The motion planning service calculates the required times and intermediate positions for all axis and sends those values to the Simulated LBR3

The following passages show how the motion planning service calculates the different PTP types. For further information about the different PTP types please refer to (1).

Asynchronous movement (method asynchronousPTP)

The waypoint calculations $X_i[n]$ for an axis $i = 1..N$ have no dependencies to all other axes and base on the given values $\hat{v}_{m,i}$, $\hat{a}_{m,i}$ and $\hat{x}_{e,i}$.

Step 1:

First we determine the maximum amplitude for the velocity ramp. We have to check whether the distance $\hat{x}_{e,i}$ is sufficient to let the velocity grow up to $\hat{v}_{m,i}$ or not. If not, we run into the velocity triangle situation (Fig. 3) and must limit $\hat{v}_{m,i}$ down to $v_{m,\max,i}$ [8]. Therefore we evaluate [9]:

$$v_{m,i} = \min(\hat{v}_{m,i}, \sqrt{\hat{x}_{e,i} \cdot \hat{a}_{m,i}})$$

```
if (_state.MaxJointVelocity > Math.Sqrt(Math.Abs(changes[i]) * _state.MaxJointAcceleration))
    velocity[i] = (float)Math.Sqrt(Math.Abs(changes[i]) * _state.MaxJointAcceleration);
else
    velocity[i] = _state.MaxJointVelocity;
```

Step 2:

Next we compute the acceleration duration $t_{a,i}$, the deceleration start time $t_{d,i}$ and the total motion time $t_{e,i}$. We get the acceleration length $t_{a,i}$ through equation [1]. Equation [5] returns the total motion time $t_{e,i}$ and the deceleration start time $t_{d,i}$ can be calculated via [2]:

$$t_{a,i} = \frac{v_{m,i}}{\hat{a}_{m,i}}$$

$$t_{e,i} = \frac{\hat{x}_{e,i}}{v_{m,i}} + t_{a,i}$$

$$t_{d,i} = t_{e,i} - t_{a,i}$$

```
ta[i] = velocity[i] / _state.MaxJointAcceleration;
te[i] = (Math.Abs(changes[i]) / velocity[i]) + ta[i];
td[i] = te[i] - ta[i];
```

Step 3:

Now we determine the count of waypoints. The acceleration phase length $t_{a,i}$ and the entire motion process time $t_{e,i}$ are divided into time slices with a period time h [11]:

```
timeslices_e[i] = (int)Math.Round(te[i] / SAMPLERATE * 1000f);
timeslices_a[i] = (int)Math.Round(ta[i] / SAMPLERATE * 1000f);
```

Step 4

Finally we compute all waypoints via [12]:

$$X_i[n_i] = \begin{cases} \frac{1}{2} \hat{a}_{m,i} \cdot h^2 \cdot n_i^2, & 1 \leq n_i \leq n_{a,i} \\ v_{m,i} \cdot h \cdot n_i - \frac{1}{2} \cdot \frac{v_{m,i}^2}{\hat{a}_{m,i}}, & n_{a,i} < n_i \leq n_{e,i} - n_{a,i} \\ v_{m,i} \cdot (t_{e,i} - t_{a,i}) - \frac{\hat{a}_{m,i}}{2} \cdot (t_{e,i} - h \cdot n_i)^2, & n_{e,i} - n_{a,i} < n_i \leq n_{e,i} \end{cases}$$

```
if (j <= timeslices_a[i])
    currentAngleChange = 1f / 2f * _state.MaxJointAcceleration * time * time;
else if (j <= timeslices_e[i] - timeslices_a[i])
    currentAngleChange = (velocity[i] * time) - (1f / 2f * velocity[i] * velocity[i] / _state.MaxJointAcceleration);
else if (j <= timeslices_e[i])
    currentAngleChange = (velocity[i] * (te[i] - ta[i]))
        - (_state.MaxJointAcceleration / 2f * (te[i] - time) * (te[i] - time));
```

Synchronous PTP movement (method synchronousPTP)

In general the calculation for the synchronous PTP movement happen to be the same as for the asynchronous PTP movement except that the several velocities $v_{m,i}$ must be chosen in that way that all total motion durations coincide to $t_{e,\max}$:

$$t_{e,1} = t_{e,2} = \dots = t_{e,N} = t_{e,\max}$$

Step 1:

First we look for the total motion time $t_{e,\max}$. Like in the asynchronous PTP movement we compute the total movement times $t_{e,i}$ of all axes $i = 1..N$:

$$\begin{aligned} \tilde{v}_{m,i} &= \min(\hat{v}_{m,i}, \sqrt{\hat{x}_{e,i} \cdot \hat{a}_{m,i}}) \\ \tilde{t}_{a,i} &= \frac{\tilde{v}_{m,i}}{\hat{a}_{m,i}} \\ \tilde{t}_{e,i} &= \frac{\hat{x}_{e,i}}{\tilde{v}_{m,i}} + \tilde{t}_{a,i} \end{aligned}$$

The longest total motion time becomes $t_{e,\max}$:

$$t_{e,\max} = \max(\tilde{t}_{e,1}, \tilde{t}_{e,2}, \dots, \tilde{t}_{e,N})$$

```

for (int i = 0; i < _jointCount; i++)
{
    if (changes[i] == 0) continue;

    if (_state.MaxJointVelocity > Math.Sqrt(Math.Abs(changes[i]) * _state.MaxJointAcceleration))
        velocity[i] = (float)Math.Sqrt(Math.Abs(changes[i]) * _state.MaxJointAcceleration);
    else
        velocity[i] = _state.MaxJointVelocity;

    ta[i] = velocity[i] / _state.MaxJointAcceleration;
    te = (Math.Abs(changes[i]) / velocity[i]) + ta[i];
    td[i] = te - ta[i];
    if (te > temax) temax = te;
}

```

Step 2:

Next we calculate the velocities $v_{m,i}$. Each axis must finish its total motion exactly at $t_{e,max}$ [5]:

$$t_{e,max} = \frac{\hat{x}_{e,i}}{v_{m,i}} + \frac{v_{m,i}}{\hat{a}_{m,i}}$$

We solve the quadratic equation for $v_{m,i}$. From the two possible results the constraint $2 \cdot t_{a,i} \leq t_{e,max}$ annuls the larger one:

$$v_{m,i} = \frac{t_{e,max} \cdot \hat{a}_{m,i}}{2} - \sqrt{\frac{t_{e,max}^2 \cdot \hat{a}_{m,i}^2}{4} - \hat{a}_{m,i} \cdot \hat{x}_{e,i}}$$

```

velocity[i] = (float)((_state.MaxJointAcceleration * temax / 2)
    - (Math.Sqrt((_state.MaxJointAcceleration * _state.MaxJointAcceleration * temax * temax / 4) -
        (Math.Abs(changes[i]) * _state.MaxJointAcceleration))));

```

Step 3:

Since we have already the total travel time $t_{e,i} = t_{e,max}$ we just have to calculate the acceleration duration $t_{a,i}$ and the deceleration start time $t_{d,i}$. Like in the asynchronous PTP movement we get the $t_{a,i}$ and $t_{d,i}$ through the equations [1] and [2]:

$$t_{a,i} = \frac{v_{m,i}}{\hat{a}_{m,i}}$$

$$t_{d,i} = t_{e,i} - t_{a,i}$$

```

ta[i] = velocity[i] / _state.MaxJointAcceleration;
td[i] = temax - ta[i];

```

Step 4:

For the calculation of the time slices and waypoints refer to the asynchronous PTP movement description steps 3 and 4.

Fully synchronous PTP movement (method fullySynchronousPTP)

In addition to the synchronous PTP movement, in the fully synchronous PTP movement we must also synchronize the acceleration time lengths $t_{a,i}$ and the deceleration start times $t_{d,i}$ for all axes.

Step 1:

Like in the synchronous PTP movement description step 1 we compute $t_{e,\max}$.

In the same way we also compute $t_{a,\max}$ and $t_{d,\max}$:

$$\begin{aligned}\tilde{v}_{m,i} &= \min(\hat{v}_{m,i}, \sqrt{\hat{x}_{e,i} \cdot \hat{a}_{m,i}}) \\ \tilde{t}_{a,i} &= \frac{\tilde{v}_{m,i}}{\hat{a}_{m,i}} \\ \tilde{t}_{e,i} &= \frac{\hat{x}_{e,i}}{\tilde{v}_{m,i}} + \tilde{t}_{a,i} \\ \tilde{t}_{d,i} &= \tilde{t}_{e,i} - \tilde{t}_{a,i} \\ t_{e,\max} &= \max(\tilde{t}_{e,1}, \tilde{t}_{e,2}, \dots, \tilde{t}_{e,N}) \\ t_{a,\max} &= \max(\tilde{t}_{a,1}, \tilde{t}_{a,2}, \dots, \tilde{t}_{a,N}) \\ t_{d,\max} &= \max(\tilde{t}_{d,1}, \tilde{t}_{d,2}, \dots, \tilde{t}_{d,N})\end{aligned}$$

```
for (int i = 0; i < _jointCount; i++)
{
    if (changes[i] == 0) continue;

    if (_state.MaxJointVelocity > Math.Sqrt(Math.Abs(changes[i]) * _state.MaxJointAcceleration))
        velocity[i] = (float)Math.Sqrt(Math.Abs(changes[i]) * _state.MaxJointAcceleration);
    else
        velocity[i] = _state.MaxJointVelocity;

    ta = velocity[i] / _state.MaxJointAcceleration;
    te = (Math.Abs(changes[i]) / velocity[i]) + ta;
    td = te - ta;
    if (te > temax) temax = te;
    if (td > tdmx) tdmx = td;
    if (ta > tamax) tamax = ta;
}
```

Step 2:

The acceleration time lengths $t_{a,i}$, deceleration stop times $t_{d,i}$ and total motion times $t_{e,i}$ must be synchronized:

$$\begin{aligned}t_{e,1} &= t_{e,2} = \dots = t_{e,N} = t_{e,\max} \\ t_{a,1} &= t_{a,2} = \dots = t_{a,N} = t_{a,\max} \\ t_{d,1} &= t_{d,2} = \dots = t_{d,N} = t_{d,\max}\end{aligned}$$

Therefore the maximum velocity and acceleration are computed individually for each axis:

$$\begin{aligned}v_{m,i} &= \frac{\hat{x}_{e,i}}{t_{d,\max}} \\ a_{m,i} &= \frac{v_{m,i}}{t_{a,\max}}\end{aligned}$$

```
for (int i = 0; i < _jointCount; i++)
{
    velocity[i] = Math.Abs(changes[i]) / tdmx;
    acceleration[i] = velocity[i] / tamax;
}
```

Step 3:

For the calculation of the time slices and waypoints refer again to the asynchronous PTP movement description steps 3 and 4.

Math Corner

General ramp PTP movement

The point to point (PTP) ramp movement consists of three phases, acceleration, constant velocity and deceleration. The acceleration phase starts at $t = 0$ and ends at t_a , the constant velocity phase reaches up to t_d and the deceleration phase ends at t_e .

Note:

In the physics the radial acceleration a [$rad \cdot s^{-2}$] is the derivation of the radial velocity v [$rad \cdot s^{-1}$] means that the velocity equals to the area enclosed by the acceleration trajectory and the time axis. Analogous the radial velocity v is the derivation of the radial angle x [rad]. Vice versa one can write $x(t) = \int v(t)dt = \iint a(t)dt$.

E.g.: Having a constant acceleration a_0 the velocity v grows with the time

$v(t) = a_0 \cdot t$ (assuming zero velocity at $t = 0$).

Within the acceleration phase the velocity increases from zero to v_m , remains changeless in the constant velocity phase and declines in the deceleration phase back down to zero. At $t = 0$ as well at t_e of the total motion process there velocity is zero. Because the absolute value of the velocities' gain rate $|a_m|$ is the same as the absolute value of the declining rate $|-a_m|$ the durations for both acceleration and deceleration conform (Fig. 1).

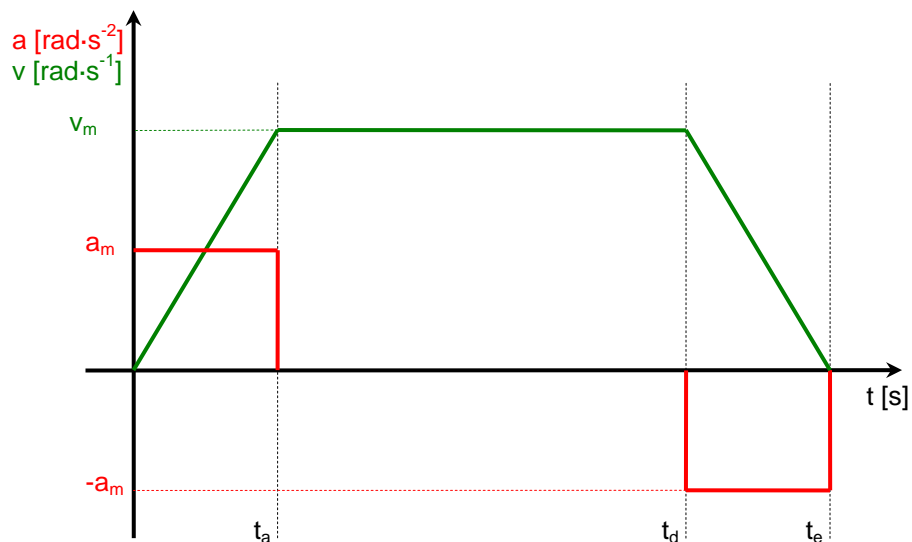


Fig. 1: Relation velocity ramp and acceleration

Presumed a constant acceleration a_m , the velocity reaches its maximum level v_m at the acceleration phase length t_a :

$$t_a = \frac{v_m}{a_m}$$

As mentioned the acceleration time length and the deceleration time length are the same, therefore the deceleration phase starts at:

$$t_d = t_e - t_a$$

[2]

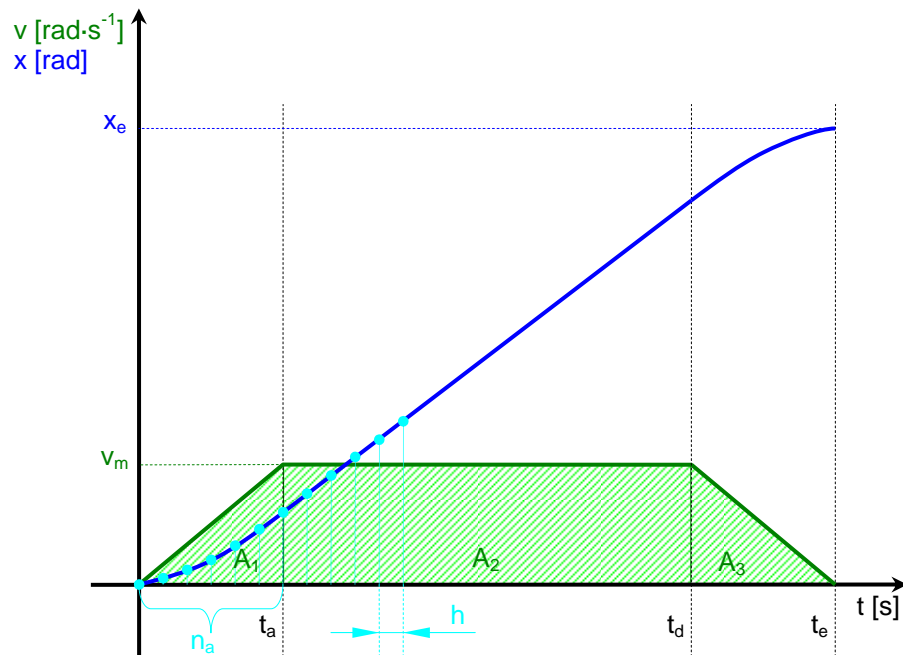


Fig. 2: Relation velocity ramp and distance

The proceeded distance equals to the area enclosed by the velocity ramp and the abscissa (Fig. 2, hatched area):

$$A_1 = \frac{v_m}{2} \cdot t_a \quad A_2 = v_m \cdot (t_d - t_a) \quad A_3 = \frac{v_m}{2} \cdot (t_e - t_d)$$

$$x_e = A_1 + A_2 + A_3 = \frac{v_m}{2} \cdot (t_d - t_a + t_e)$$

[3]

Because of the symmetry $t_d = t_e - t_a$ we state:

$$x_e = v_m \cdot (t_e - t_a)$$

[4]

Isolating [4] for the total movement duration t_e results in:

$$t_e = \frac{x_e}{v_m} + t_a = \frac{x_e}{v_m} + \frac{v_m}{a_m}$$

[5]

Now we know all parameters to compute the velocity for any point of time:

$$x(t) = \begin{cases} \frac{1}{2} a_m \cdot t^2, & 0 \leq t < t_a \\ v_m \cdot t - \frac{1}{2} \cdot \frac{v_m^2}{a_m}, & t_a \leq t < t_d \\ v_m \cdot (t_e - t_a) - \frac{a_m}{2} \cdot (t_e - t)^2, & t_d \leq t \leq t_e \end{cases}$$

[6]

At last we have to consider the case where the distance x_e isn't sufficient to let the velocity grow up to its target level v_m . Then the velocity ramp becomes a triangle (Fig. 3). The acceleration phase length is now half of the total motion duration. The side condition $t_e = 2 \cdot t_a$ applies now to the distance equation [6] and [4]. Therefore we substitute t_e for $2 \cdot t_a$ in [4]. As a result we get the equation that relates the distance to the velocity peak level:

$$x_e = t_a \cdot v_m \quad [7]$$

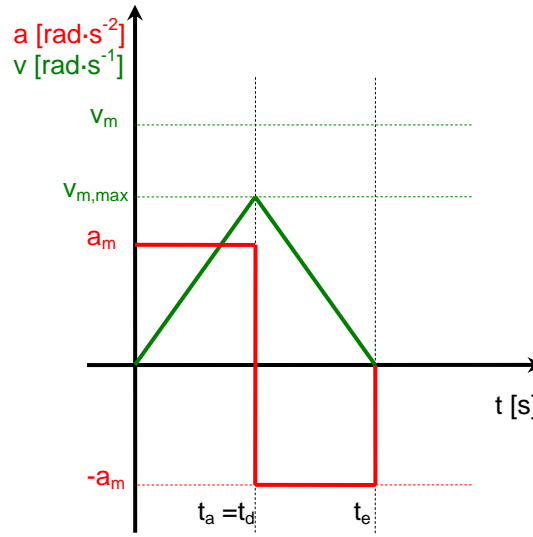


Fig. 3: Velocity triangle

Substituting $t_a = \frac{v_m}{a_m}$ into [7] and solving it for $v_{m,\max}$ leads to:

$$v_{m,\max} = \sqrt{a_m \cdot x_e} \quad [8]$$

The triangle case has to be considered when using [6]; for that reason we have to limit the given velocity v_m when it exceeds $v_{m,\max}$:

$$v_{m,\text{limited}} = \min(v_m, v_{m,\max}) \quad [9]$$

Time discretization

To enforce that the motion adheres to the velocity ramp trajectory the motion planning service must send continuously command positions to the simulated lbr3 arm service in small time intervals h (sample rate period time). Hence the total PTP movement is divided into n_e waypoints matching discrete points of time $t = n \cdot h$. We get the time discrete distance function through substituting t with $n \cdot h$ in [6] (Fig. 2):

$$X[n] = x(t) \Big|_{t \rightarrow n \cdot h, n \in N^+} \quad [10]$$

The counts of discrete time steps n_a , n_d and n_e multiplied with the period time h must match to t_a , t_d and t_e :

$$t_a = n_a \cdot h, \quad n_a \in N^+$$

$$t_d = n_d \cdot h, \quad n_d \in N^+$$

$$t_e = n_e \cdot h, \quad n_e \in N^+$$

[11]

Hence the time discrete waypoints can be calculated as follows:

$$X[n] = \begin{cases} \frac{1}{2} a_m \cdot h^2 \cdot n^2, & 1 \leq n \leq n_a \\ v_m \cdot h \cdot n - \frac{1}{2} \cdot \frac{v_m^2}{a_m}, & n_a < n \leq n_e - n_a \\ v_m \cdot (t_e - t_a) - \frac{a_m}{2} \cdot (t_e - h \cdot n)^2, & n_e - n_a < n \leq n_e \end{cases}$$

[12]

Note:

However, in practice t_a , t_d and t_e usually won't match exactly an integer multiple of h . To avoid inaccuracies some corrections have to be made for t_a, t_d, t_e, v_m and a_m . After an adjustment of t_a, t_d and t_e to satisfy [11] the velocity v_m and the acceleration a_m have to be computed again via $v_m = \hat{x}_e \cdot t_a^{-1}$ and $a_m = \hat{x}_e \cdot t_a^{-2}$. The implementation of these adjustments is left for the user as an exercise.

General approach to the distance equation:

The radial distance x can be calculated through the integration $x(t) = \int \dot{x}(t) dt = \iint \ddot{x}(t) dt$.

Function $\dot{x}(t) = \frac{d}{dt} x(t)$ describes the velocity and $\ddot{x}(t) = \frac{d^2}{dt^2} x(t)$ the acceleration trajectory.

Let's begin with the acceleration function:

$$\ddot{x}(t) = \begin{cases} a_a, & 0 \leq t < t_a \\ 0, & t_a \leq t < t_d \\ a_d, & t_d \leq t \leq t_e \end{cases}$$

[13]

The first integration results into the velocity function:

$$\dot{x}(t) = \int_0^t \ddot{x}(t) dt = \begin{cases} a_a \cdot t + \dot{x}_0, & 0 \leq t < t_a \\ a_a \cdot t_a + \dot{x}_0, & t_a \leq t < t_d \\ a_a \cdot t_a + a_d \cdot (t - t_d) + \dot{x}_0, & t_d \leq t \leq t_e \end{cases}$$

[14]

The constant \dot{x}_0 takes any prior velocity into account. The second integration results into the distance equation, where the constants \dot{x}_0 and x_0 take any prior velocity and position into account:

$$x(t) = \int_0^t \dot{x}(t) dt$$

$$= \begin{cases} \frac{a_a}{2} \cdot t^2 + \dot{x}_0 \cdot t + x_0, & 0 \leq t < t_a \\ a_a \cdot \left(t_a \cdot t - \frac{t_a^2}{2} \right) + \dot{x}_0 \cdot t + x_0, & t_a \leq t < t_d \\ a_a \cdot \left(t_a \cdot t - \frac{t_a^2}{2} \right) + a_d \cdot \left(\frac{t_d^2}{2} + \frac{t^2}{2} - t \cdot t_d \right) + \dot{x}_0 \cdot t + x_0, & t_d \leq t \leq t_e \end{cases} \quad [15]$$

Now we introduce our side conditions. In our application acceleration and deceleration have the same absolute values, therefore we state:

$$a_a = -a_d = a_m \quad [16]$$

We focus on relative movements starting with zero velocity:

$$x_0 = \dot{x}_0 = 0 \quad [17]$$

We assume that the velocity zeros again at t_e :

$$\dot{x}(t_e) = 0 \quad [18]$$

Taking the side condition [16], [17] and [18] in [14] into account we get:

$$t_d = t_e - t_a \quad [19]$$

Therefore the velocity equation [14] can be stated as:

$$\hat{x}(t) = \int_0^t \ddot{x}(t) dt = \begin{cases} a_m \cdot t, & 0 \leq t < t_a \\ a_m \cdot t_a, & t_a \leq t < t_d \\ a_m \cdot (t_e - t), & t_d \leq t \leq t_e \end{cases} \quad [20]$$

Evaluating the velocity function [20] for the time t_a results in:

$$v_m = \hat{x}(t_a) = a_m \cdot t_a \quad [21]$$

Side conditions [16] and [17] and equation [19] turn [15] into:

$$\hat{x}(t) = \int_0^t \hat{x}(t) dt = \begin{cases} \frac{a_m}{2} \cdot t^2, & 0 \leq t < t_a \\ a_m \cdot \left(t_a \cdot t - \frac{t_a^2}{2} \right), & t_a \leq t < t_d \\ a_m \cdot \left(t_a \cdot t_e - t_a^2 - \frac{t_e^2}{2} + t_e \cdot t - \frac{t^2}{2} \right), & t_d \leq t \leq t_e \end{cases} \quad [22]$$

Finally we simplify the distance equation [22] by using [19] and [21]:

$$\hat{x}(t) = \begin{cases} \frac{a_m}{2} \cdot t^2, & 0 \leq t < t_a \\ v_m \cdot t - \frac{1}{2} \cdot \frac{v_m^2}{a}, & t_a \leq t < t_d \\ v_m \cdot (t_e - t_a) - \frac{a_m}{2} \cdot (t_e - t)^2, & t_d \leq t \leq t_e \end{cases} \quad [23]$$

Legend:

t_a	: = Acceleration time length
\tilde{t}_a	: = Acceleration time length interim value
$t_{a,\max}$: = Acceleration time length all axes' maximum value
t_d	: = Deceleration start time
\tilde{t}_d	: = Deceleration start time interim value
$t_{d,\max}$: = All axes' maximum value of the deceleration start time
t_e	: = Movement end time
\tilde{t}_e	: = Movement end time interim value
$t_{e,\max}$: = All axes' maximum value of the total movement time
a_m	: = Maximum acceleration
\hat{a}_m	: = Predetermined acceleration maximum
v_m	: = Velocity ramp maximum
\hat{v}_m	: = Predetermined velocity ramp maximum
\tilde{v}_m	: = Velocity ramp maximum interim value
\hat{x}_e	: = Predetermined distance to move
h	: = Sample rate period time
N	: = Count of axes
n	: = Time discrete step indexer
$X_i[n]$: = Time discrete position

References

- (1) W. Weber, "Industrieroboter: Methoden der Steuerung und Regelung", Fachbuchverlag Leipzig, 2002