Optimization, Robotics & Biomechanics (ORB)
Institute of Computer Engineering (ziti)
Heidelberg University

Prof. Dr. Katja Mombaur,
Benjamin Reh,
Silvan Lindner

Robotics 1 (WS 2018/2019)

Exercise Sheet 7

Presentation during exercises in calendar week 51

Exercise 7.1 – Kinematic Motion Planning Part I

This assignment is the first part of a small collection of real world kinematic motion planning functions for the Kuka robot.

As the first form of motion we choose the simple PTP motion. PTP stands for Point-to-Point or better Pose-to-Pose. Given an initial pose $\vec{q}(t = 0)$ and a final pose $\vec{q}(t = t_{\text{end}})$ the axes angles q_i are interpolated using a velocity ramp function. The position of the end effector (TCP) is not of interest.

Ramp function First we only look at the motion of a single axis q_i that moves from $q_i(0)$ to $q_i(t_{\text{end}})$. The velocity $v = \dot{q}_i$ of the joint should be increased by the constant acceleration $a = \ddot{q}$ until the maximum velocity is reached at time t_1. Starting from t_2 the velocity has to be decreased again such that the end position is reached when the motion has stopped ($v = 0$). This case is shown in fig. 1a. A special case occurs, if $v_{\max}$ cannot be reached because the distance to travel is too small for the given acceleration. This is shown in 1b.

a) Find the critical distance s_{crit} for a given $v_{\max}$ and a to distinguish between the two cases in fig. 1.

   a) $s \geq s_{\text{crit}} \rightarrow v \leq v_{\max}$

   b) $s < s_{\text{crit}} \rightarrow v < v_{\max}$

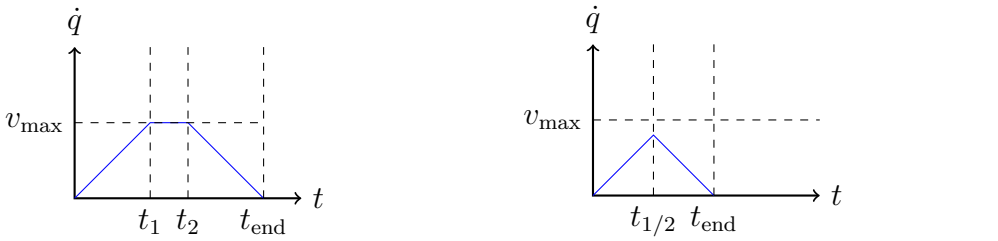
   Hints:

   - What is $t_{1/2}$ for the corner case $s = s_{\text{crit}}$ is exactly reached?
   - Look at the integral $s_{1/2} = \int_0^{t_{1/2}} a \cdot dt^2$. What is the relation between $s_{1/2}$ and s_{crit} ?

b) Now we have a criterion to decide, which of the two cases will occur for any given s. Give t_1,t_2 and t_{end} for the first case and $t_{1/2}$ and t_{end} for the second case for a general s with given a and $v_{\max}$.

   Hint: It might help to look at the integral $s = \int v(t)dt$ from a geometric point as the "area under the curve".

c) Implement the results into the class `Rampfunction` which is prodived for this sheet. Fill the `S(t)` function that interpolates the distance with time $0 \leq t \leq t_{\text{end}}$ using the velocity ramps described above.



(a) Velocity reaching $v_{\max}$ (b) Distance too small to reach $v_{\max}$

Figure 1: Two examples for velocity curves

When executing the programm, an automated test is run to verify the correctness of the class. Use this to find any mistakes in your implentation or calculation.

PTP motion

The transistion from the initial pose $\vec{q}_{\text{start}}$ to the final pose $\vec{q}_{\text{end}}$ shall be interpolated. The joint angels have to travel the distances $\Delta\vec{q} = \vec{q}_{\text{end}} - \vec{q}_{\text{start}}$. For simplicity we assume that the acceleration a and the maximal velocity v_{vmax} are the same for all six axes.

a) Implement the PTP function a using the ramp function for all six axes. A sample motion is defined in the code snippet. Visualize your result in meshup to verify your implementation. Consider the following point:

- In general, one axis has the largest distance (angle) to travel and therefore defines the duration t_{end} of the whole motion. We call this axis with index k the leading axis: $|\Delta\vec{q}_k| \geq |\Delta\vec{q}_{i\neq k}|$.
- The motion of the other axes should end at the same time t_{end} when the leading axes stops to move. As fig. 2 illustrates, the motion of the other axes have to be slowed down / stretched to achieve that.

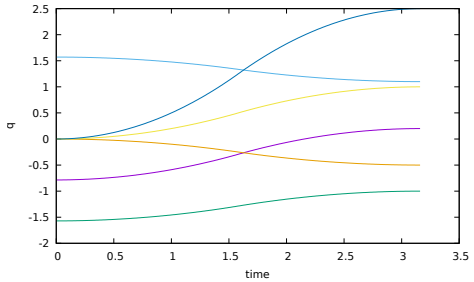


Figure 2: Angle of the joint axes for a sample PTP motion. The leading axis is blue.

- Assuming we set up the ramp function $R_k(t)$ to deal with the leading axis k, the PTP interpolation for the other joints can be written as:

$$\vec{q}(t) = \vec{q}_{\text{start}} + \Delta\vec{q} \cdot \left(\frac{R_k(t)}{|\Delta\vec{q}_k|}\right) \tag{1}$$

with $0 \leq t \leq t_{\text{end}}$.

Exercise 7.2 – Newton-Euler Formalism

In the lecture you have learned about the Newton-Euler formalisms to compute equations of motion and inverse dynamics. It is an inherently recursive method and provides a formalism to systematically deal with arbitrary complex rigid multi-body systems. Thus, it is mostly used in implementations for forward- and inverse dynamics. It consists of a two step recursion:

- During forward iteration, the positions, velocities, and accelerations of each link are propagated from the base to the tip.
- In the backward iterations the forces and moments experienced by each link are propagated from the tip to the base.

While we will not apply Newton-Euler exactly in this task, we will proceed similarly by separating the multi-body system and computing quantities for each separate body.

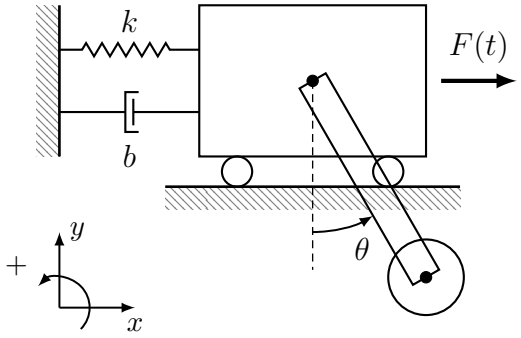


Figure 3: Spring-damped cart-pendulum.

Consider the following example:

A cart with a pendulum, as shown in figure 3, is moving on a horizontal surface. A spring with spring constant k, a damper with damper constant b and an external force $F(t)$ are acting upon the cart. The mass of the cart is given as m_C. The pendulum is connected to the center of the cart and can freely rotate. It consists of a uniform solid cylindrical rod of length l, diameter d and mass m_R and a spherical mass connected to the end of the rod with radius r and mass m_S. Friction is neglected.

Virtually separate the segments from each other by introducing "cutting forces" and set up Newtons equation for each body. Afterwards derive the full equations of motion.

Exercise 7.3 – Forward & Inverse Dynamics

When simulating a multi-body system one usually proceeds similar to exercise 6.2 (double pendulum): An ordinary differential equation of order 2 is obtained using a forward dynamics operator

$$\mathrm{FD}\left(q, \dot{q}, \tau\right) \longmapsto \ddot{q} \qquad \text{Forward Dynamics Operator} \tag{2}$$

and integrated using numerical integration (e.g. Runge-Kutta).

The problem of inverse dynamics on the other hand calculates joint torques and forces τ from given robot states which is usually necessary in robot control:

$$\mathrm{ID}\left(q, \dot{q}, \ddot{q}\right) \longmapsto \tau \qquad \text{Inverse Dynamics Operator} \tag{3}$$

Both operators solve the equations of motion

$$\mathrm{M}\left(q\right)\ddot{q} + N\left(q, \dot{q}\right) = \tau. \tag{4}$$

Imagine you are using a framework that only provides a black-box inverse dynamics (3) implementation. The dimensions of q and τ are known. How could you use the ID implementation to construct a forward dynamics operator (2) without any additional knowledge about the multi-body system? Formulate your answer as pseudo-code.

Hint: Try to extract information about your system in terms of (4) by calling (3) with special values of $\ddot{q}$.

Note: In RBDL FD and ID operators are implemented using *ABA* (2) and *RNEA* (3) - refer to Featherstone et. al.