Robotics
Exercise 3

Marc Toussaint
Lecturer: Peter Englert
TAs: Matt Bernstein, Danny Driess, Hung Ngo
Machine Learning & Robotics lab, U Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany

November 9, 2016

1 Motion profiles

Construct a motion profile that accelerates constantly in the first quarter of the trajectory, then moves with constant velocity, then decelerates constantly in the last quarter. Write down the equation $\text{MP}(s):\ [0,1] \mapsto [0,1]$.

2 Verify some things from the lecture

a) On slide 02:37 there is a term $(\mathbf{I} - J^{\#}J)$ called nullspace projection. Verify that for $\varrho \to \infty$ (and $C = \varrho\mathbf{I}$) and any choice of $a \in \mathbb{R}^n$

$$\delta q = (\mathbf{I} - J^{\#}J)a \ \Rightarrow\ \delta y = 0$$

(assuming linearity of ϕ, i.e., $J\delta q = \delta y$). Note: this means that any choice of a, the motion $(\mathbf{I} - J^{\#}J)a$ will not change the "endeffector position" y.

b) On slide 02:48 it says that

$$\begin{aligned}&\operatorname*{argmin}_q \|q - q_0\|_W^2 + \|\Phi(q)\|^2 \\ &\quad\approx q_0 - (J^\top J + W)^{\text{-}1} J^\top\, \Phi(q_0) = q_0 - J^{\#}\Phi(q_0)\end{aligned}$$

where the approximation $\approx$ means that we use the local linearization $\Phi(q) = \Phi(q_0) + J(q - q_0)$. Verify this.

3 Multiple task variables

Download the code for this exercise here: https://ipvs.informatik.uni-stuttgart.de/mlr/16-Robotics/e03-code.tbz2
Unpack the code and copy the folder 'kinematics2' into 'robotics15/share/teaching/RoboticsCourse/'

The code contains the solution to last week's exercise where the robot moved his hand in a circle.

a) We've seen that the solution does track the circle nicely, but the trajectory "gets lost" in joint space, leading to very strange postures. We can fix this by adding more tasks, esp. a task that permanently tries to (moderatly) minimize the distance of the configuration q to a natural posture q_{home}. Realize this by adding a respective task.

b) Make the robot simultaneously point upward with the left hand. Use `kinematicsVec` (and `jacobianVec`) to compute the orientation of a robot shape for this.

c) Use the task spaces from question 1.a) of the last exercise to add a task such that the robot looks with his right eye towards the right hand. You can use the following code to get the gaze direction of the robot with the corresponding Jacobian:

```
arr gaze, Jgaze;
arr vec = ARR(0.,0.,1.);
S.kinematicsVec(gaze,"eyeR",&vec);
S.jacobianVec(Jgaze,"eyeR",&vec);
```