Robotics
Exercise 4

Marc Toussaint
Machine Learning & Robotics lab, U Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany

November 4, 2014

1 Verify some things from the lecture

a) On slide 02:46 it says that

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

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

b) On slide 02:34 there is a term $(\mathbf{I} - J^\# J)$ called nullspace projection. Verify that for $\varrho \to \infty$ 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 motion $(\mathbf{I} - J^\# J)a$ will not change the "endeffector position" y.

c) On slide 02:27 we derived the basic inverse kinematics law. Verify that (assuming linearity of ϕ) for $C \to \infty$ the desired task is fulfilled exactly, i.e., $\phi(q^*) = y^*$. By writing $C \to \infty$ we mean that C is a matrix of the form $C = \varrho C_0$, $\varrho \in \mathbb{R}$, and we take the limit $\varrho \to \infty$.

2 Multiple task variables

Consider again the last week's exercise where the robot moved his hand in a circle (Exercise 3.2d).

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 to the code given on the next page.

b) Make the robot simultaneously point upward with the left hand. Use `kinematicsVec` and `jacobianVec` for this.

```
void multiTask(){
  Simulator S("man.ors");
  arr q,q_home,y_target,y,J,W,Phi,PhiJ;
  uint n = S.getJointDimension();

  S.getJointAngles(q);
  W.setDiag(1.,n); //we define W as identity matrix
  q_home = q; //we store the initial posture as q_home

  for(uint i=0;i<10000;i++){
    Phi.clear();
    PhiJ.clear();

    //1st task: track with right hand
    y_target = ARR(-0.2, -0.4, 1.1);
    y_target += .2 * ARR(cos((double)i/20), 0, sin((double)i/20));
    S.kinematicsPos(y,"handR");
    S.jacobianPos  (J,"handR");

    Phi .append( 1e2 * (y - y_target) );  //rho = 1e4 (cp. slide 02:45)
    PhiJ.append( 1e2 * J );

    //add the "stay close to home" task here

    //add another task for the left hand here, if you like

    //compute joint updates
    q -= inverse(~PhiJ*PhiJ + W)*~PhiJ* Phi; //(cp. slide 02:46)
    S.setJointAngles(q);
  }
}
```