Robotics
Exercise 5

Marc Toussaint
Lecturer: Duy Nguyen-Tuong
TAs: Philipp Kratzer, Janik Hager, Yoojin Oh
Machine Learning & Robotics lab, U Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany

November 20, 2018

1 Controlling an arm to follow a trajectory (8 points)

1. Please update the version of the repository by running the following.

```
git pull
git submodule update
make -C rai dependAll
make -j4
```

2. For python you can run: *'jupyter-notebook py/03-dynamics/03-dynamics.ipynb'*
3. For C++ run: *'cd cpp/03-dynamics'*, *'make'*, *'./x.exe'*

In the main.cpp we provide a dynamics simulation of a robot arm that simulates the system for $T = 5$ seconds. The task is to write a controller that moves the arm from the initial position to a desired position $q^*_T = 0$ and velocity $\dot{q}^*_T = 0$ with a sine motion profile (see slide 02:40)

$$q^*_t = q_0 + \frac{1}{2}[1 - \cos(\pi t/T)](q^*_T - q_0).$$

a) Compute the desired velocity $\dot{q}^*_t$ and acceleration $\ddot{q}^*_t$ of the motion profile. (2P)

b) Implement the motion profile in the for-loop of the code, such that at each time step t a desired position, velocity and acceleration is available. (1P)

c) Implement direct PD control to follow the trajectory and try to find parameters K_p and K_d (potentially different for each joint) to reach the goal position. (2P)

d) Try to do the same with a PID controller that also includes the integral error (1P)

$$u = K_p(q^* - q) + K_d(\dot{q}^* - \dot{q}) + K_i \int_{s=0}^{t} (q^* - q(s))\ ds\ .$$

e) Now use the knowledge of M and F (slide 03:29) in each time step. The matrices M and F of the current robot state can be computed using 'M, F = K.equationOfMotion()' (python) or 'K.equationOfMotion(M, F)' (C++). Use inverse dynamics feedforward control to determine the necessary u (slide 03:33). (2P)

Optional:

Try different starting positions.

Try the same controllers for the arm in `pegArm2.ors`.

Play with `noise` parameter and check stability.

2 Local linearization (4 points)

Slide 04:02 describes the cart pole dynamics, which is similar to the Segway-type system of Exercise 4, but a little simpler. The state of the cart-pole is given by $x = (p, \dot{p}, \theta, \dot{\theta})$, with $p \in \mathbb{R}$ the position of the cart, $\theta \in \mathbb{R}$ the pendulums angular deviation from the upright position and $\dot{p}, \dot{\theta}$ their respective temporal derivatives. The only control signal $u \in \mathbb{R}$ is the force applied on the cart. The analytic model of the cart pole is

$$\ddot{\theta} = \frac{g \sin(\theta) + \cos(\theta) \left[-c_1 u - c_2 \dot{\theta}^2 \sin(\theta) \right]}{\frac{4}{3} l - c_2 \cos^2(\theta)} \tag{1}$$

$$\ddot{p} = c_1 u + c_2 \left[\dot{\theta}^2 \sin(\theta) - \ddot{\theta} \cos(\theta) \right] \tag{2}$$

with $g = 9.8ms^2$ the gravitational constant, $l = 1m$ the pendulum length and constants $c_1 = (M_p + M_c)^{-1}$ and $c_2 = l M_p (M_p + M_c)^{-1}$ where $M_p = M_c = 1kg$ are the pendulum and cart masses respectively.

Derive the local linearization of these dynamics around $x^* = (0, 0, 0, 0)$. The eventual dynamics should be in the form

$$\dot{x} = Ax + Bu$$

Note that

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ \frac{\partial \ddot{p}}{\partial p} & \frac{\partial \ddot{p}}{\partial \dot{p}} & \frac{\partial \ddot{p}}{\partial \theta} & \frac{\partial \ddot{p}}{\partial \dot{\theta}} \\ 0 & 0 & 0 & 1 \\ \frac{\partial \ddot{\theta}}{\partial p} & \frac{\partial \ddot{\theta}}{\partial \dot{p}} & \frac{\partial \ddot{\theta}}{\partial \theta} & \frac{\partial \ddot{\theta}}{\partial \dot{\theta}} \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ \frac{\partial \ddot{p}}{\partial u} \\ 0 \\ \frac{\partial \ddot{\theta}}{\partial u} \end{pmatrix}$$

where all partial derivatives are taken at the point $p = \dot{p} = \theta = \dot{\theta} = 0$.