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

Dr. Alexander Schubert,
Giorgos Marinou

Robotics 2 (SS 2022)

Exercise Sheet 2

Presentation during exercises in calendar week 21

Exercise 2.1 – Initial value problem and right-hand-side

We assume the following initial value problem (IVP):

$$\begin{aligned} \dot{y} &= f(t, y) \\ y(t_0) &= y_0. \end{aligned}$$

The term $f(t, y)$ is described as *right hand side* (RHS) and the problem is usually to find $y(t_f)$ of a desired t_f which is in frequent cases not possible analytically.

We want to compute until $t_f = 1$:

$$\begin{aligned} \dot{y} &= -200ty(t)^2 \\ y(-3) &= \frac{1}{901} \end{aligned}$$

A simple but yet powerful class for the solution of IVP's are the *single step methods*. They are usually written under the following form:

$$\begin{aligned} \eta_0 &:= y_0 \\ \eta_{i+1} &:= \eta_i + h \cdot \Phi(t_i, \eta_i; h; f), \\ t_{i+1} &:= t_i + h \end{aligned}$$

The term $\Phi(t, y; h; f)$ is called *increment function* and η_i holds the numerical values from previous step i. Both quantities are related to the stepsize h.

Compute $y(t)$ for the stepsize of $h = 2^{-i},\ i = 5, \ldots, 10$ and compare the values to the exact solution $u(t) = (1 + 100t^2)^{-1}$, what tendency can you observe? Visualize your result.

Exercise 2.2 – Runge-Kutta Integrator
The currently implemented single step method (the euler method) is the simplest of its type with $\Phi(t, y; h; f) = f(t, y)$. A more accurate approximation of the numerical integration in the increment function leads for example to the class of Runge-Kutta Methods - find below the method of 4th order:

$$\Phi(t, y; h; f) := \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$\begin{aligned} k_1 &:= f(t, y) \\ k_2 &:= f(t + \frac{1}{2}h, y + \frac{1}{2}hk_1) \\ k_3 &:= f(t + \frac{1}{2}h, y + \frac{1}{2}hk_2) \\ k_4 &:= f(t + h, y + hk_3) \end{aligned}$$

How does the error with respect to the exact solution change with different stepsizes? Visualize your result.

Exercise 2.3 – Forward Dynamics Integration
In this exercise we want to simulate the movement of the humanoid robot iCub. The state of the robot is described by the vector $x = [q, \dot{q}]$. The different forward dynamics functions of RBDL allow us to compute the joint acceleration $\ddot{q}$ for given torques τ and external forces f_{ext}. We can use this function to compute the motion of a robotic system by integrating the acceleration twice to get the resulting joint angle trajectories.

- Implement a RHS function that returns the derivative of the state vector $\dot{x} = [\dot{q}, \ddot{q}]$. Use the Runge-Kutta Integrator from exercise 2.1 to compute the joint angle trajectories. Simulate 10 seconds of unactuated motion and visualize the result. In this case, we can omit the time as argument for the RHS and the integrator.

Exercise 2.4 – Contact Forces and Constraint Sets in RBDL

Now, we want to fix iCub in space, as if it was hanging from a supporting device. Implement a contact force acting on the root link that counteracts gravity. How does the motion look now? Is this realistic?

Hint: The external forces interface of RBDL takes one force per body. Each force consists of a torque and a linear force. The torque can be computed by:

```
            torque = VectorCrossMatrix(pos)*force;
```

where "pos" is a 3d Vector containing the position where the force acts in global coordinates.

- Add joint friction and observe the effect. Hint: The friction force is proportional to the joint velocity.
- Add symmetric torques to the hip joints and observe the motions. Does this look as if iCub was fixed in space?
- How do the motions look if we instead of using an external force set the gravity to zero?

Now we want to fixe iCub on a pole in space. RBDL is able to handle contacts. Define a contact in space for the root link using a constraint set and the ForwardDynamicsConstraintsDirect function. How many points do you need to constrain? What exactly is constraint?

Hint: You find the object meshes for rbdl-toolkit with the exercise code. If you work from home you need to copy them into your rbdl-tookit folder to visualize the robot model.

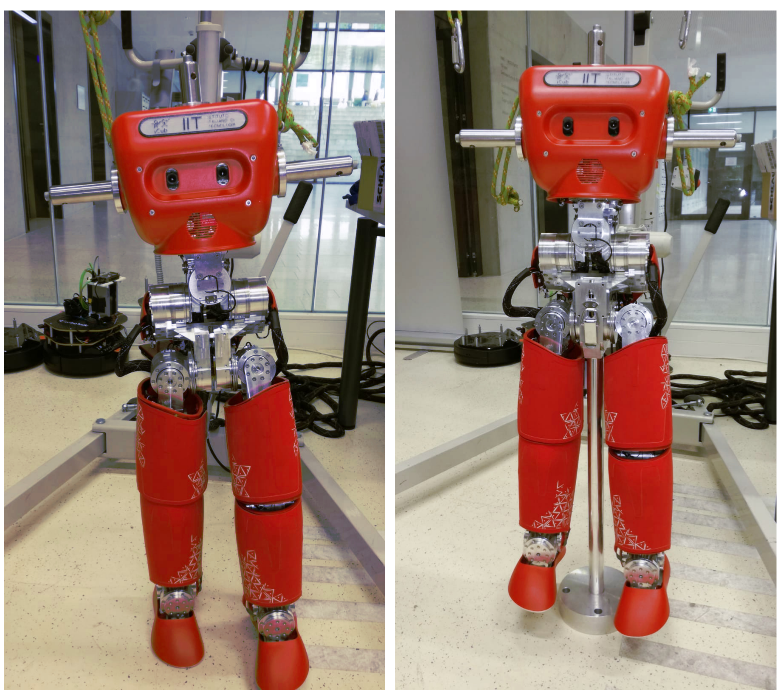

Figure 1: iCub hanging from robes and on it's pole