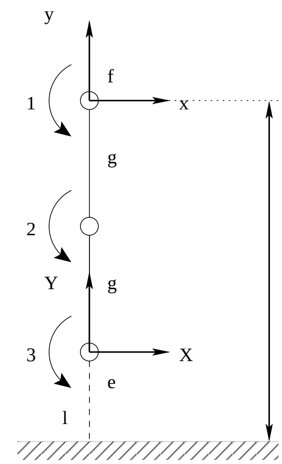
Technische Universität München **Chair of Automatic Control Engineering** Dr.-Ing. Marion Leibold **Chair of Information-Oriented Control** Dr.-Ing. Stefan Sosnowski	**Robot Control Laboratory** Practical Assignment



In this practical assignment a SCARA robot with three joints will be used. The length of the segments l are defined to be $0.30\,\mathrm{m}$. The base of the manipulator is fixed in the world frame S_0 with $h = 0.925\,\mathrm{m}$ above the ground. On the end effector a laser is attached that points into the negative y_E direction. Additionally the end effector is equipped with a camera. The camera coordinate frame S_{Kam} is shifted $0.022\,\mathrm{m}$ with respect to the end effector frame S_E in y_E direction. You will find all functions that you need to implement the exercises in the library *libCarbo.mdl*.

The shown configuration corresponds to the joint angles $[\theta_1\ \theta_2\ \theta_3]^T = [0\ 0\ 0]^T$.

Exercise 1: Control in joint space

For this exercise use the initial configuration `QINITIAL = [0 0 0]` in the $MATLAB$ script *startup.m* and execute it.

a) First the encoders for the angles of the manipulator have to be initialized. In the case that the angle θ_3 is far away from the desired initial configuration, you can move it with the programs *mvCamCW* and *mvCamCCW* in clockwise or counterclockwise direction respectively. The program *initCarbo* starts a initialization routine that brings the manipulator in the configuration shown above with the help of a sensing device.

b) Build a model, in which a trajectory in the joint space should be generated that moves the end effector vertically. Furthermore the laser should always point in negative y_0 direction and the movement should be periodic. Therefore use the blocks **Clock** as well as **Trajectory Generator joint space** from the library *libCarbo.mdl*. The frequency of the periodic movement should be $2\,\mathrm{rad/s}$ and the amplitude of the joint angle θ_1 is moving in the range from 0 to $0{,}7\,\mathrm{rad}$. (Help: What condition has to hold for this movement?)

In the next step several methods of control should be compared. For comparison you should use the mean square control error for one period and each joint $RMS = \sqrt{\sum_{i=1}^{N} \boldsymbol{e}_i^2 / N}$. Here $\boldsymbol{e}_i$ is the error of the i-th measurement and N the number of measurements.

c) To use the code in real time a few changes have to be implemented. The block **Clock** should be substituted by the block **Timer**. Extend your model from exercise a) with a PD control. Parametrize the controller as follows $K_P = [800\ 600\ 400]^T$ and $K_V = [5\ 5\ 2]^T$. The robot will be actuated using the block **Robot**. Additionally to record the control error use the block **RTAI_TO_FILE**. Prior to compiling please take a look at the indications for the compilation of a real time program at the end of the sheet. Move the robot for several oscillation periods and analyze the control error for one period.

d) In the next step the state control should be substituted with an inverse system control. Substitute the control from exercise c) with the block **rne**. If the control works reasonably well, analyze the error with the methods from exercise c).

e) Extend the control from exercise d) with a PD controller with the following parameters $K_P = [800\ 600\ 400]^T$ and $K_D = [5\ 5\ 2]^T$. Furthermore analyze the control methodology according to exercise c).

Exercise 2: Control in configuration space

To start the control of the robot in a non singular configuration, set the initial configuration in the file *startup.m* to `QINITIAL = [pi/4 -pi/2 pi/4]` and execute it.

a) Build a model that generates a periodic movement of the laser point in the horizontal plane. Choose a reasonable amplitude and frequency for the periodic movement and use the block **Timer**.

b) The block **JLaser** computes the Jacobian of the joint angles with respect to the laser point. Use the block **MATLAB Function** to compute the joint velocities from the velocities needed for the trajectory and the Jacobian.By using an integrator, you can compute the joint angles. Move the robot along the trajectory computed in a).

c) In this exercise you should generate a trajectory that moves the origin of the S_{Kam} frame circularly in the x_0-y_0 plane. Choose a reasonable frequency and amplitude.

d) The block **JCamera** computes the Jacobian with respect to the origin of the S_{Kam} frame. Move the robot along the trajectory from c) by implementing a model according to exercise b).

e) Optional: Implement according to pages 5-14 in the script a projection of the camera task from exercise d) into the null space of the laser tracking from exercise b). Therefore use the weighted pseudo inverse $\boldsymbol{J}^+ = \boldsymbol{W}^{-1}\boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^T)^{-1}$ and think about the weighting of the joints for the different tasks.

Exercise 3: Video based control

Substitute the trajectory generator for the camera movement from exercise 2e) with the block **KameraOffset** and test the result.

Notes: By using the *Real-Time Workshop* for *Matlab/Simulink*, a control in a *Simulink* model can be compiled to a program executable in real time. Therefore please follow these steps:

- Substitute all **Clock** blocks with **Timer** blocks.
- Substitute all **Scope** blocks with **RTAI_TO_FILE** blocks.
- Open the*Simulink* menu **Simulation** $\rightarrow$ **Configuration Parameters...**.
- Change the solver type to **Fixed − step**, use the solver **ode1**. For the step size use the variable `TA` and change the stop time to inf.
- In the menu **Code Generation** change the systme target file to `prt.tlc`. Subsequently change the **Target function library** to **GNU99** and check the box **classic call interface** in the menu **Interface**
- Apply the changes and close the panel.
- To compile your program press the button **BuildModel** in the top right corner of the model.