Technische Universität München **Chair of Automatic Control Engineering** Dr.-Ing. Marion Leibold **Chair of Information-Oriented Control** Dr.-Ing. Stefan Sosnowski	**Robot Control Laboratory** 4. Assignment

- All Simulink models must be executable without any use of further models, functions or scripts. The only exception is a script to generate the SerialLink object "Rob" for exercise 2 and 3.
- Do not generate models on your own. Only use the models available on Moodle or copies of those models.
- In the downloaded models you can find modified blocks from the *Robotics Toolbox*. To solve the exercises you will only need standard Simulink blocks. Therefore it is not allowed to use any blocks from the *Robotics Toolbox* .
- For intruductionary purposes take a look at examples of Simulink models from the *Robotics Toolbox* in the PDF robot.pdf.

Exercise 1: Position Control

In the following exercises different control laws for robots will be implemented. These laws can be found in the lecture notes in chapter VIII.1. First the model of a decoupled revolute joint $\ddot{q} = u + z$ (VIII.1) will be used. The disturbance z is a step function and defined as $z = 0.3 \cdot \sigma(t)$ for the following exercises. Implement a controller for the revolute joint. The joint angle q [rad] should be saved in the workspace. Therefore additional information will be provided in each exercise.

a) First the state controller described in equation VIII.2 in the lecture notes should be implemented.

1. Download the model *ssc_p.slx* from Moodle. Solve the exercise using **only** this model with a double integrator.
2. Use the block **Step** to generate a desired trajectory q^d. The desired trajectory of the joint angle is defined as a step function with $q^d = \pi/2 \cdot \sigma(t)$. Additionally the disturbance z can be observed. Both step functions should start at time $t = 0$. The initial angle is $q_0 = 0$ and the initial disturbance is $z_0 = 0$.
3. Implement the state control from the lecture notes. Use the blocks **Gain** and **Sum**, as well as the values $K_p = 5$ for the P term and $K_v = 4$ for the damping. To add the disturbance to the model use the block **Sum**.
4. Save the resulting angle q [rad] of the double integrator using the block **To Workspace**. The variable name should be $\boldsymbol{Q}$ and use the option *Save format: Timeseries*. Do not change the other options!

(6)	$\boldsymbol{Q}$ = ssc_p()	SIM
Input:	—	
Output:	$\boldsymbol{Q}$ Data structure from the **To Workspace** block	

Simulate the model for $10\,\text{s}$. Plot the desired trajectory q^d and the trajectory of the system q together in one **Scope**. Therefore you have to generate a vector from both trajectories using the **Mux** block. Compare your result with the figure VIII.2 (a) of the lecture notes with respect to the steady state error.

b) Next the joint angle q should increase from $0°$ to $90° = \pi/2$ within $10\,\text{s}$. Furthermore the PD controller described in equation VIII.4 of the lecture notes will be used.

1. Download the model *ssc_pd.slx* from Moodle. Solve this exercise **only** with this model and use the given blocks.
2. Use the block **jtraj** to generate the desired trajectories q^d, $\dot{q}^d$ and $\ddot{q}^d$.
3. Use the PD control mentioned above with the parameters $K_p = 5$ for the P term and $K_d = 4$ for the D term.
4. Add the disturbance.
5. Save the resulting joint angle $\boldsymbol{q}$ [rad] of the double integrator using the block **To Workspace**. The name of the variable should be $\boldsymbol{Q}$. Furthermore set the option *Save format: Timeseries*. Do not change any other options!

(4)	$\boldsymbol{Q} =$ ssc_pd()	SIM
Input:	—	
Output:	$\boldsymbol{Q}$ Data structure from the **To Workspace** block	

Simulate the model for $10\,\text{s}$. Plot the desired trajectories q^d, $\dot{q}^d$ and the resulting trajectories q, $\dot{q}$ using one or two **scope** blocks.

c) Now a PID controller according to the control law in equation VIII.6 of the lecture notes should be implemented.

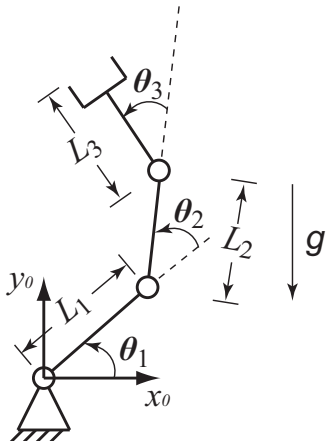
1. Make a copy of the model ssc_pd.slx and change the name of the copy to **ssc_pid.slx**. Solve this exercise using **only** this model and the predefined blocks.
2. Change the controller from PD to PID using the parameters $K_p = 5$ for the P term, $K_i = 5$ for the I term and $K_d = 4$ for the D term. For the I term you should also use the block **Integrator**.
3. Add the disturbance to the model.
4. Save the resulting joint angle $\boldsymbol{q}$ [rad] of the double integrator using the block **To Workspace**. The name of the variable should be $\boldsymbol{Q}$. Furthermore set the option *Save format: Timeseries*. Do not change any other options!

(4)	$\boldsymbol{Q} =$ ssc_pid()	SIM
Input:	—	
Output:	$\boldsymbol{Q}$ Data structure from the **To Workspace** block	

Simulate the model for $10\,\text{s}$. Plot the desired trajectories q^d, $\dot{q}^d$ and the resulting trajectories q, $\dot{q}$ using one or two **scope** blocks. Compare the results to those of the exercises a) and b) with respect to a steady state error.

Exercise 2: Cartesian Position Control

Given is the planar robot with $N = 3$ segments from the last assignments, see figure below. The robot has three revolute joints θ_1, θ_2 and θ_3. The length of the segments is given by $L_1 = 4\,\text{m}$, $L_2 = 3\,\text{m}$ and $L_3 = 2\,\text{m}$. The robot should move from its starting position (8, 0, 0) [m m rad] around an ellipse within 10 s. The rotation should happen counter clockwise for the robot in the figure below. One focal point should be located on the x axis and have an radius of 8 m and the other one on the y axis with a radius of 6 m. Additionally the orientation of the end effector should change in a constant manner, i.e. $\phi = \frac{2\pi}{10\,\text{s}}t$, where t denotes the time and ϕ the orientation.



The position of the end effector defines the desired position. Implement a control for the current position of the end effector. Therefore use the Jacobi Controller from figure VIII.6 a) (Jacobi Control Laws) in the lecture notes for cartesian position control. The system will be described by a double integrator. The joint angles $\boldsymbol{q}$ [rad, rad, rad] should be saved to the workspace.

1. Download the model *cartControl.slx* from Moodle. Solve the exercise using **only** this model and the given blocks. Do **not** use any other blocks from the *Robotics Toolbox*.
2. For the simulation a SerialLink object "Rob" has to be in the workspace. Therefore build this object using the functions $\mathbf{Link()}$ and $\mathbf{SerialLink()}$ in a Matlab script. This script has not to be submitted. Execute your script.
3. Use the blocks $\boxed{\mathbf{Clock}}$ and $\boxed{\mathbf{Fcn}}$ from the Simulink library to compute the desired position and orientation.
4. Use the robot kinematic model with the block $\boxed{\mathbf{fkine}}$, to generate the homogeneous transformation matrix from the joint angle output of the block $\boxed{\mathbf{DoubleIntegrator}}$.
5. Use the block $\boxed{\mathbf{T2xyz}}$ to get the x, y and z position of the end effector from the homogeneous transformation matrix.
6. Compute the orientation by summing up the joint angles.
7. Generate vectors of the current and desired position using the $\boxed{\mathbf{Mux}}$ block from Simulink library.
8. Compute the error and use the blocks $\boxed{\mathbf{jacob}}$ and $\boxed{\mathbf{ijacob}}$ to compute the Jacobian and its inverse as well as the given controller $\boxed{\mathbf{Control}}$ and the block $\boxed{\mathbf{Product}}$ from the Simulink library to implement the Jacobi Control.
9. Save the resulting joint angle $\boldsymbol{q}$ [rad, rad, rad] of the double integrator using the block $\boxed{\mathbf{To\ Workspace}}$. The name of the variable should be $\boldsymbol{Q}$. Furthermore set the option *Save format: Timeseries*. Do not change any other options!

(10)	$\boldsymbol{Q}$ = cartControl()	SIM
Input:	—	
Output:	$\boldsymbol{Q}$ Data structure from the **To Workspace** block	

Simulate the model for 10 s and plot the cartesian desired and current positions of the end effector. If you save $\boldsymbol{Q}$ to the workspace, a visualization of the robot is generated through a callback with the function **plot()**. There the behavior of the robot can be seen.

Exercise 3: Computed Torque Control

Given is the planar robot from exercise 2. The robot consists of three revolute joints θ_1, θ_2 and θ_3. The length of the segments is defined as $L_1 = 4\,\text{m}$, $L_2 = 3\,\text{m}$ and $L_3 = 2\,\text{m}$. The segments are assumed to be rigid bodies with a homogeneous mass distribution. The mass of each segments is defined as $m_1 = 20\,\text{kg}$, $m_2 = 15\,\text{kg}$ and $m_3 = 10\,\text{kg}$. The inertias are given by $I_{zz1} = 0.5\,\text{kgm}^2$, $I_{zz2} = 0.2\,\text{kgm}^2$ and $I_{zz3} = 0.1\,\text{kgm}^2$ (the rest does not have an effect on the dynamics). It is assumed that the center of mass coincides with the geometric center of the segments. Set the gear ratio to 1 and and the motor inertia to zero. The gravitational acceleration is pointing into negative vertical direction.

In this exercise the control of the robot using the Computed Torque Control should be implemented. Use the control law from equation VIII.4 of the lecture notes and additionally consider figure VIII.4. The joint angles of the robot $\boldsymbol{q}$ [rad, rad, rad] should be saved to the workspace.

1. Download the model *ctc.slx* from Simulink. Solve the exercise using **only** this model and the given blocks.
2. For the simulation a SerialLink object "Rob" has to be in the workspace. Therefore build this object using the functions $\mathbf{Link()}$ and $\mathbf{SerialLink()}$ in a Matlab script. Please note that this time the dynamic parameters have to be considered. This script has not to be submitted. Execute your script.
3. Use the block $\boxed{\mathbf{jtraj}}$ to generate the desired trajectories. Use the following parameters: initial joint angles $(0, 0, 0)^T$ [rad, rad, rad], final joint angles $(\pi/2, -\pi/2, \pi/2)^T$ [rad, rad, rad] and the time $10\,\text{s}$.
4. To generate the inverse dynamics, use the functions **itorque()**, **coriolis()** and **gravload()**. To use functions in Simulink, use the $\boxed{\mathbf{Interpreted\ MATLAB\ Function}}$ block. The name of the SerialLink object is defined as "Rob".
5. The P term of the controller should be set to 100, whereas the D term parameter should be set to 1.
6. Save the resulting joint angle $\boldsymbol{q}$ [rad, rad, rad] of the double integrator using the block $\boxed{\mathbf{To\ Workspace}}$. The name of the variable should be $\boldsymbol{Q}$. Furthermore set the option *Save format: Timeseries*. Do not change any other options!

Additional Note: The used model of the robot has an added disturbance. Therefore there will be observable differences to the dynamic model you are using. These differences will be reflected in joint angle errors

(10)	$\boldsymbol{Q} = \text{ctc}()$	SIM
Input:		—
Output:		$\boldsymbol{Q}$ Data structure from the $\boxed{\mathbf{To\ Workspace}}$ block
Forbidden Routines:		rne()

Simulate the model for $10\,\text{s}$. Plot the desired trajectories q^d and the current trajectories q with several $\boxed{\mathbf{scope}}$ blocks. Only small differences should be visible.
The block $\boxed{\mathbf{rne}}$ from the *Robotics Toolbox* could be used instead of the functions **itorque()**, **coriolis()** and **gravload()**, however, its use is not allowed. Compare the computed torques from your model to those of the $\boxed{\mathbf{rne}}$ block to check whether both approaches lead to the same result. At the end of the simulation the movement of the robot can be seen again, if you saved $\boldsymbol{Q}$ to the workspace.
Important: Delete the block $\boxed{\mathbf{rne}}$ from your model before submitting it due to the fact that its usage is not allowed.