Report

Robot Control

Performed at the:
University of Applied Science Carinthia

University course: Systems Design

under the guidance of:
Andrjia Rojko

by

Tobias Karg Bsc.
Studentnumber: 1110527027

Contents

List of Figures

1 Properties of a Robot

In this chapter the basic properties of the robot will be described, therefore the dynamic equations of the motor will be given and the behavior of the robot will examined with some basic simulations.

1.1 Dynamic equation

The robot arm, which can be seen in figure 1.1, has two joints within the same plain.

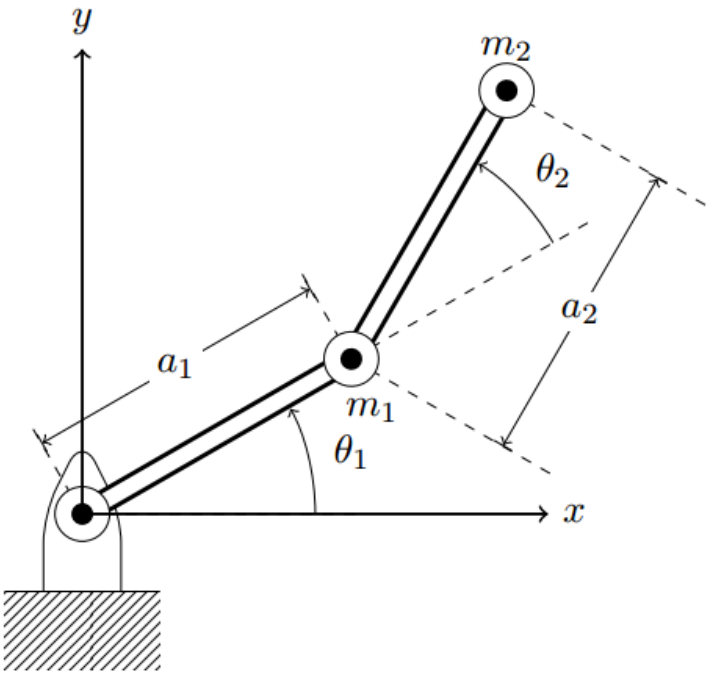

Figure 1.1: Schmetics of the 2DOF Robot arm

The basic differential equation of the robot can be seen by (1.1).

$$\tau = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{v}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) + \tau_D \tag{1.1}$$

where:

τ = input torque vector
$\mathbf{M}(\mathbf{q})$ = inertia matrix
$\mathbf{v}(\mathbf{q}, \dot{\mathbf{q}})$ = centrifugal/corriolis vector
$\mathbf{g}(\mathbf{q})$ = gravitation vector
τ_D = disturbance vector

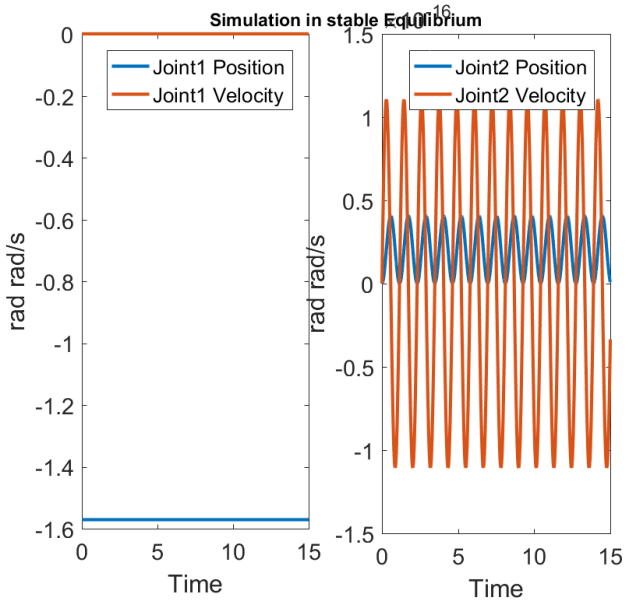

Figure 1.2: Simulation with initial condition in stable equilibrium

With the concept from Lagrange the dynamic equation of this system can be calculated:

$$\tau = \begin{pmatrix} {a_1}^2 m_1 + {a_1}^2 m_2 + {a_2}^2 m_2 + 2a_1 a_2 m_2 \cos(q_2) & a_2 m_2 \left(a_2 + a_1 \cos(q_2)\right) \\ a_2 m_2 \left(a_2 + a_1 \cos(q_2)\right) & {a_2}^2 m_2 \end{pmatrix} \begin{pmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{pmatrix}$$
$$+ \begin{pmatrix} -a_1 a_2 m_2 \dot{q}_2 \sin(q_2) \left(2\dot{q}_1 + \dot{q}_2\right) \\ a_1 a_2 m_2 \dot{q}_1^2 \sin(q_2) \end{pmatrix}$$
$$+ \begin{pmatrix} g m_2 \left(a_2 \cos(q_1 + q_2) + a_1 \cos(q_1)\right) + a_1 g m_1 \cos(q_1) \\ a_2 g m_2 \cos(q_1 + q_2) \end{pmatrix}$$

1.2 Simulation of the robot dynamics

To check if the previous derived dynamic equations are reasonable, some basic simulations can be performed. In these simulations the robot arm will be taken into some specific initial conditions and the behavior will be simulated without any external force applied to the system.
There are three different initial conditions chosen. The first one is the robot arm points down and it rests in its stable equilibrium state. The next one is keeping the robot in its unstable equilibrium by pointing straight up. The last initial condition is an arbitrary setup, outside of the previous given equilibrium states.
In figure 1.2 it can be seen that the robot arm is staying at rest and does not start to move. On one joint there can be some minor oscillation seen, which can be explained by the numerical error of the simulation.
The next simulation can be seen in figure 1.3, here the initial condition of the robot arm was within the unstable equilibrium. Theoretically the robot will stay at rest within this state, unfortunately by some numerical error, caused by the simulation,

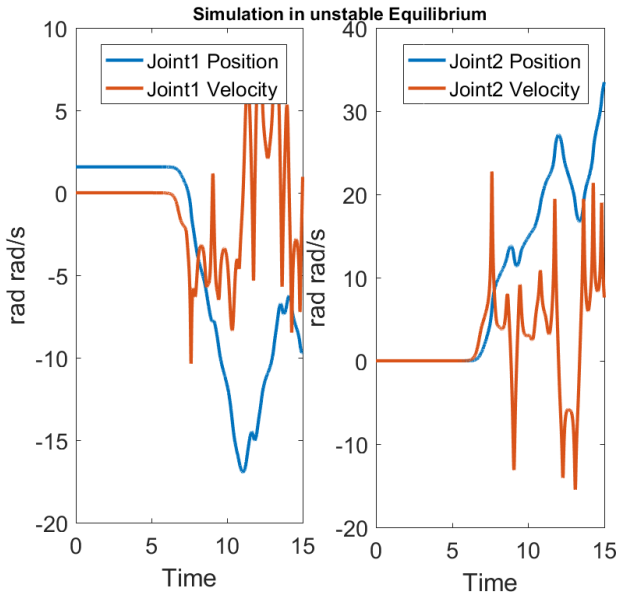

Figure 1.3: Simulation with initial condition in unstable equilibrium

the robot arm starts to move after around 7 seconds and continuous to swing with some chaotic pattern. Since the numerical error within the system gets integrated over time and there is no friction within the system, the movement of the robot is not expected to stop in anytime.
The last simulation is the initial condition outside of any equilibrium and the movement of all joints are expected to take place immediately; in comparison to the last simulation, which take 7 seconds to start moving visibly. The results of the simulation can seen in figure 1.4. The joints move in a chaotic way, as it has been expected.
All results of the simulations show expected behavior and we can assume that the model, which has been used, is correct.

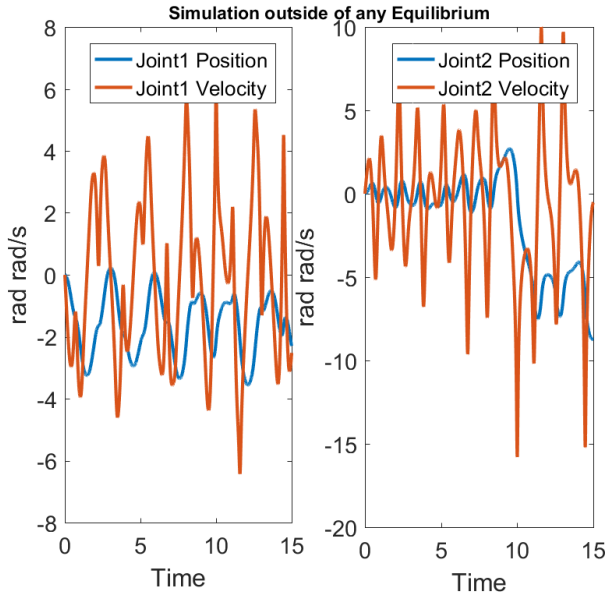

Figure 1.4: Simulation with initial condition outside of any equilibrium

2 Computed Torque controllers

In this chapter multiple variations of computed torque controllers will be developed and applied to the just verified model of the robot. In general computed torque controllers are nonlinear ones, which are linearizing the errors of the system, so that a linear controller can be applied. Therefore the previously deducted model will be used. First the error system needs to be defined:

$$\begin{aligned}\mathbf{e} &= \mathbf{q}_D - \mathbf{q} \\ \dot{\mathbf{e}} &= \dot{\mathbf{q}}_D - \dot{\mathbf{q}} \\ \ddot{\mathbf{e}} &= \ddot{\mathbf{q}}_D - \ddot{\mathbf{q}}\end{aligned}$$

where:

$\mathbf{e}$: error vector
$\mathbf{q}_D$: vector with reference trajectories for each joint

Substituting the robot dynamic equation (1.1) into the acceleration vector $\ddot{\mathbf{q}}$, results in:

$$\ddot{\mathbf{e}} = \ddot{\mathbf{q}}_D - \mathbf{M}^{-1}\left(\tau - \mathbf{v} - \mathbf{g}\right) + \mathbf{M}^{-1}\tau_D$$

Defining $\mathbf{u} = \ddot{\mathbf{q}}_D - \mathbf{M}^{-1}\left(\tau - \mathbf{v} - \mathbf{g}\right)$ and $\mathbf{w} = \mathbf{M}^{-1}\tau_D$ results in the following linear system

$$\begin{pmatrix} \dot{\mathbf{e}} \\ \ddot{\mathbf{e}} \end{pmatrix} = \begin{pmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{e} \\ \dot{\mathbf{e}} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{I} \end{pmatrix} \mathbf{u} + \begin{pmatrix} \mathbf{0} \\ \mathbf{I} \end{pmatrix} \mathbf{w}$$

As the actual input of the robot arm is not the input $\mathbf{u}$, but the motor torque τ, $\mathbf{u} = \ddot{\mathbf{q}}_D - \mathbf{M}^{-1}\left(\tau - \mathbf{v} - \mathbf{g}\right)$ needs to be solved for τ:

$$\tau = \mathbf{M}(\ddot{\mathbf{q}}_D - \mathbf{u}) + \mathbf{v} + \mathbf{g} \tag{2.1}$$

The (2.1) is the inner control low, which linearizes the robot to a linear double integrator. The closed loop dynamics of the robot can now be defined by choosing a proper control law for $\mathbf{u}$. In the figure 2.1 the control structure can be seen, with the outer controller and the inner one, which is linearizing.

2.1 PD-Controller

For the PD-Controller a controller of the structure

$$\mathbf{u} = -\mathbf{K}_d\dot{\mathbf{e}} - \mathbf{K}_p\mathbf{e}$$

where:

$\mathbf{K}_d$: diagonal matrix with derivative gains
$\mathbf{K}_p$: diagonal matrix with proportional gains

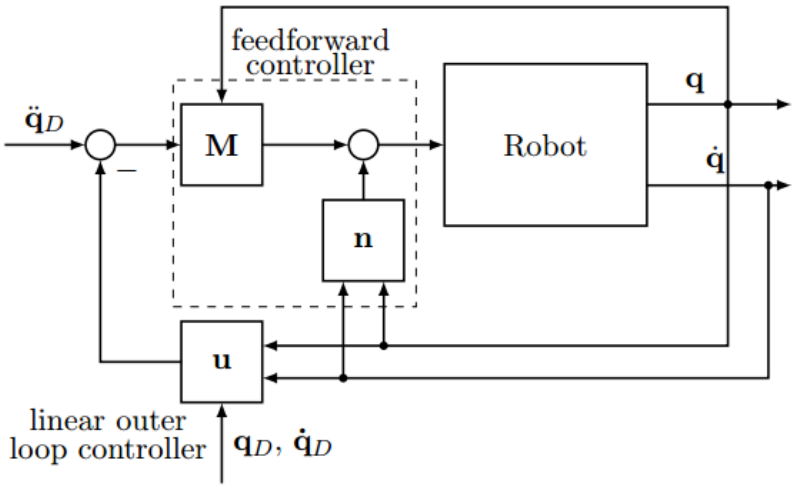

Figure 2.1: Schmetics of the 2DOF Robot arm

is chosen. It can be shown that the characteristic polynomial of the closed loop answer is:

$$\Delta(s) = s^2 + k_d s + k_p = 0$$

By comparing this polynomial to a general second order system:

$$p(s) = s^2 + 2\zeta\omega_n s + {\omega_n}^2$$

it can be seen, that $k_d = 2\zeta\omega_n$ and $k_p = {\omega_n}^2$. For the frequency of the system ω_n the value 10 has been chosen. The damping factor will be changed over different simulations to show the influence of over- and under-damping. At last another simulation will be discussed, with the initial settings for the controller but with an external distortion. In table 2.1 the different settings for the four simulations can be seen.

	k_p	k_d	τ_D	According Figure
$\zeta = 1$:	100	20	0 N m	2.2
$\zeta = 0.1$:	100	2	0 N m	2.3
$\zeta = 10$:	100	200	0 N m	2.4
$\zeta = 1$:	100	20	1 N m	2.5

Table 2.1: Controller parameters for simulations witch PD outer loop controller

In figure 2.2 the simulation results the critical damping. The error of the system for the first joint starts at zero and has a small peak around 0.01 rads; the second joint has its error peak right at the beginning with 0.1rads. The torque has rather high

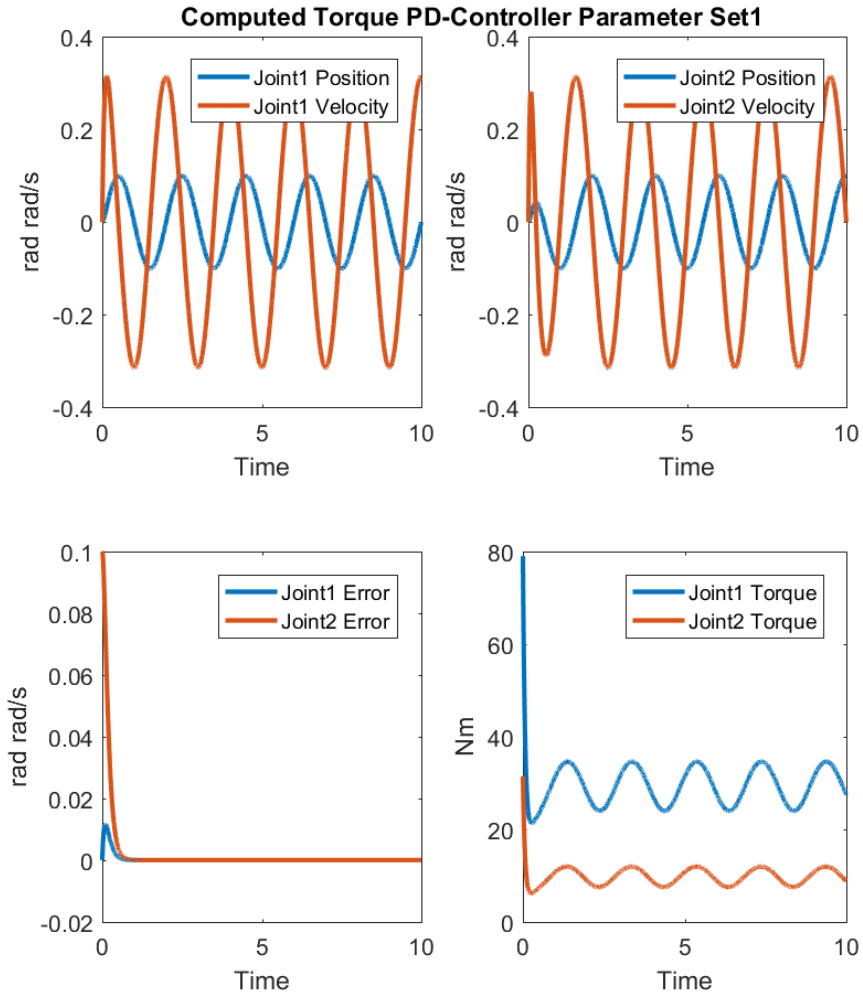

Figure 2.2: Computed Torque PD Controller

peaks right at the beginning but is reduced almost immediately to values around 30Nm for the first and 10Nm for the seconds joint. Wrapped up this simulation result shows that the robot arm behaves as expected. The error of the system is reduced to a minimum pretty fast, as it has been expected by a critically damped system.

The next simulation result 2.3 show the behavior of the system in combination with an under-damped controller. Per definition an under-damped system oscillates with a frequency, slightly different to the undamped case, until the amplitude of the oscillation gradually reaches zero. This is exactly the behavior of the robot arm. When the error is taken in observation, it can be seen that there are quite severe oscillations, in comparison the the previous simulation. The robot arm needs around 5 seconds to finally move without an position error.

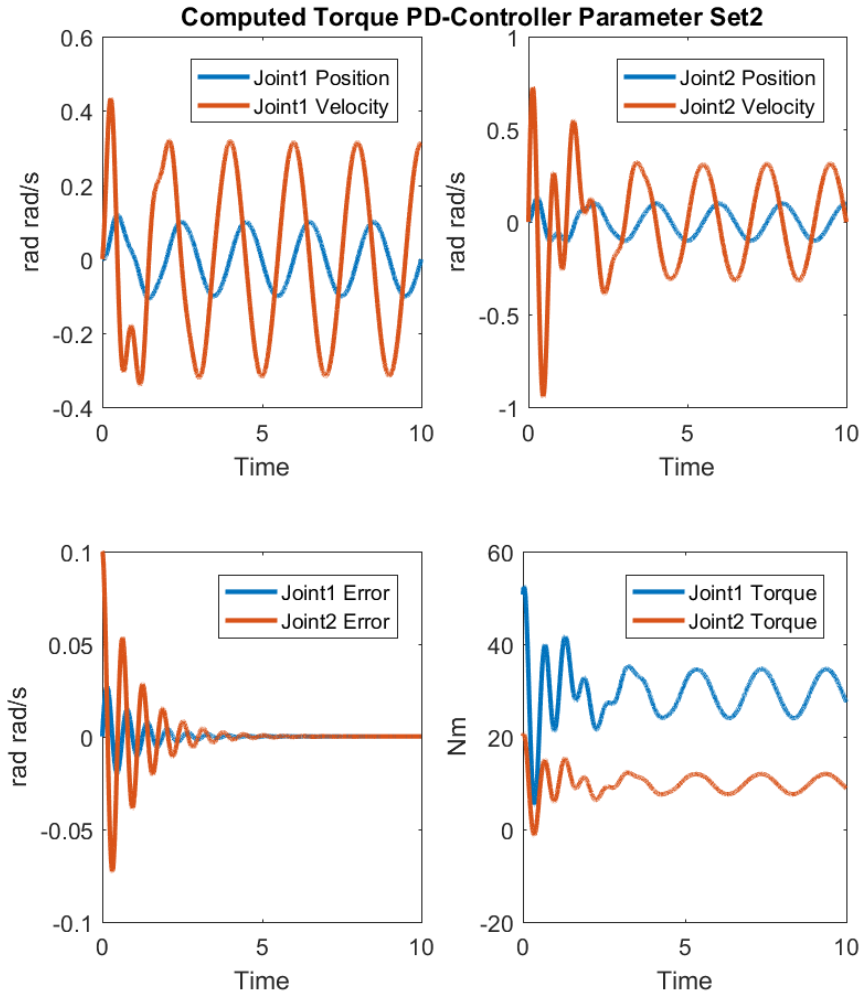

Figure 2.3: Computed Torque PD Controller with under-damping

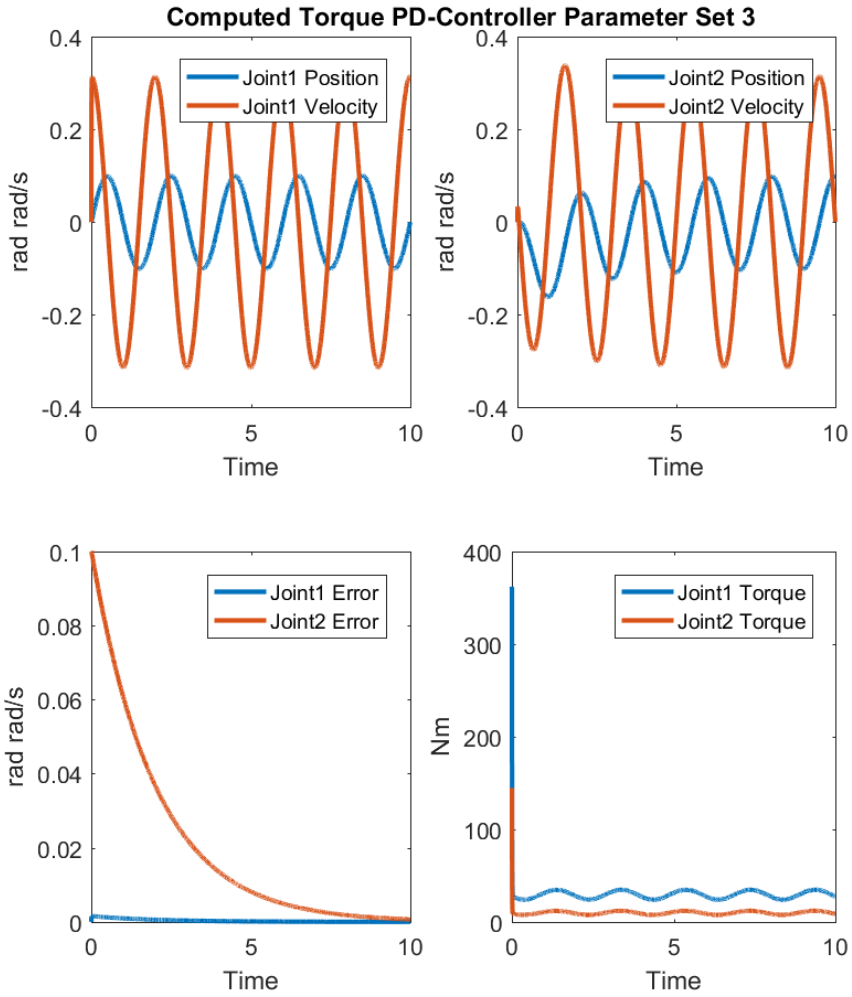

Figure 2.4: Computed Torque PD Controller with over-damping

The third simulation, which can be seen in figure 2.4, has a controller, which is over-damped. An over-damped controller drives the system to steady state with exponential decay without any oscillations. Exactly this behavior can be seen within the simulation results. To force the system on this trajectory, there are high torque values needed (up to 350Nm), which makes this approach not very useful for a real world application, since there would be huge motors needed, to drive this torques.

The last simulation uses the parameter set from the first simulation, so the critically damped controller. Although here is a distortion torque applied to each joint. Since the controller is based on a PD-setup, the closed loop answer is not expected to have no steady-state error. In figure 2.5 exactly this behavior can be seen. The other signals are visually the same as the initial simulation with the same parameter set. To avoid the steady-state error another control approach needs to be chosen.

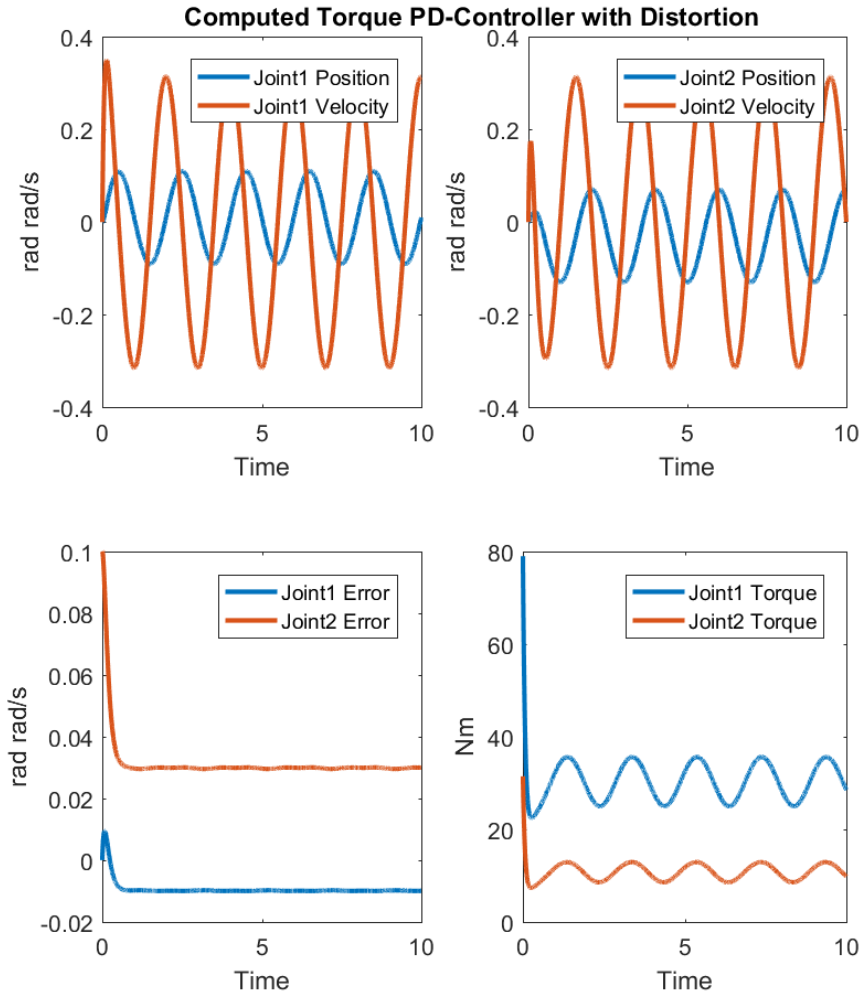

Figure 2.5: Computed Torque PD Controller with distortion

2.2 PID-Controller

Similar to the previous approach a controller can be designed using a PID control structure.

$$\mathbf{u} = -\mathbf{K}_d\dot{\mathbf{e}} - \mathbf{K}_p\mathbf{e} - \mathbf{K}_i\varepsilon$$

where:

$$\varepsilon \int \mathbf{e}(\tau)\mathrm{d}\tau$$

It can shown that the characteristic polynomial of the system is given by:

$$\Delta(s) = s^3 + k_d s^2 + k_p s + k_i = 0$$

To calculate the conditions, for which this polynomial only contains negative roots, the Routh-Hurwitz stability criterion is used.

$$\begin{array}{ccc} 1 & k_p & 0 \\ k_d & k_i & 0 \\ \dfrac{k_d k_p - k_i}{k_d} & 0 & 0 \\ k_i & 0 & 0 \end{array}$$

In order, that the first row has no changes in sign, the following conditions need to be fulfilled:

$$\begin{aligned} k_d &> 0 \\ k_i &> 0 \\ k_p &> \frac{k_i}{k_d} \end{aligned}$$

The controller gain k_i is given with 500. By parameter coefficient comparison with the characteristic polynomial this leads to $k_p = 2520$ and $k_d = 100.2$.

The robot arm is simulated with this type controller and a disturbance of 1Nm at each joint. The results of this simulation can be seen in figure 2.6. The values of torque and the behavior of the error dynamics is comparable to the critically damped PD-Controller, although this system is able to handle the distortion of the joints, since it was an integrator within the controller. This integrator allows the closed loop answer of the system to have no steady-state error.

The simulation of figure 2.7 uses the same controller setup and parameter as the previous simulation, although the robot model itself has been changed, by applying an known payload to it. The controller is not aware of this change and this can be seen within the dynamics of the system. The error of the seconds joint starts to oscillate, where the previous simulation show no steady state error at all.

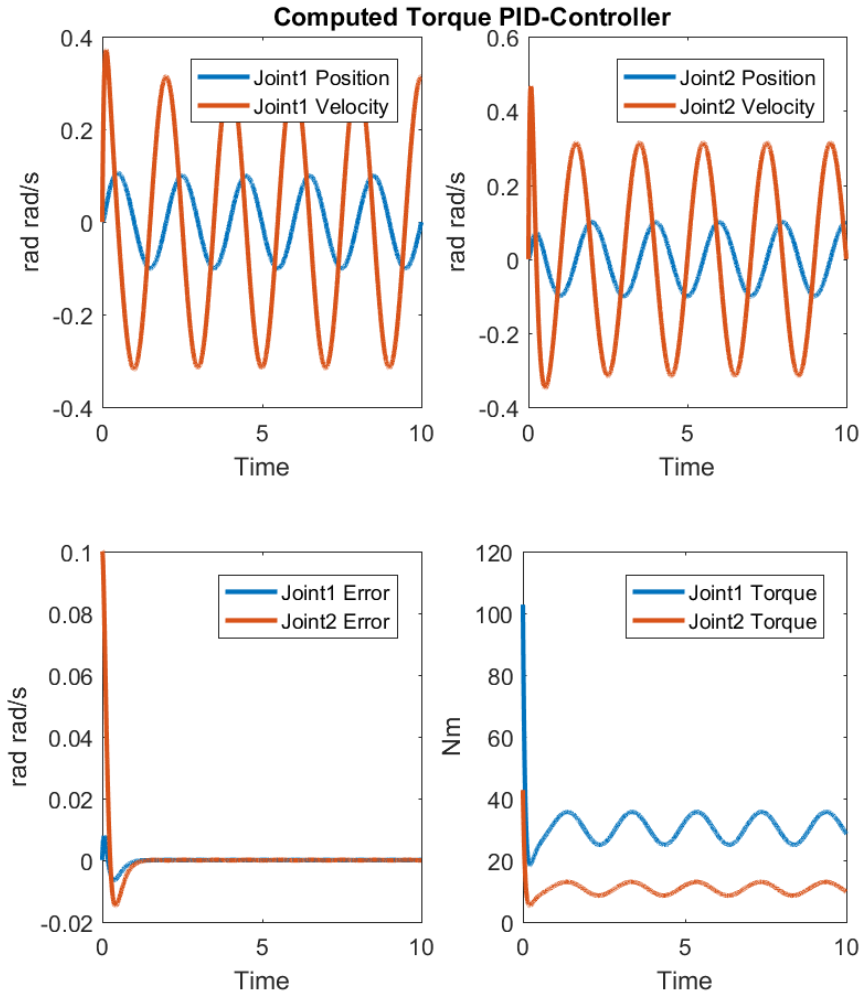

Figure 2.6: Computed Torque PID Controller

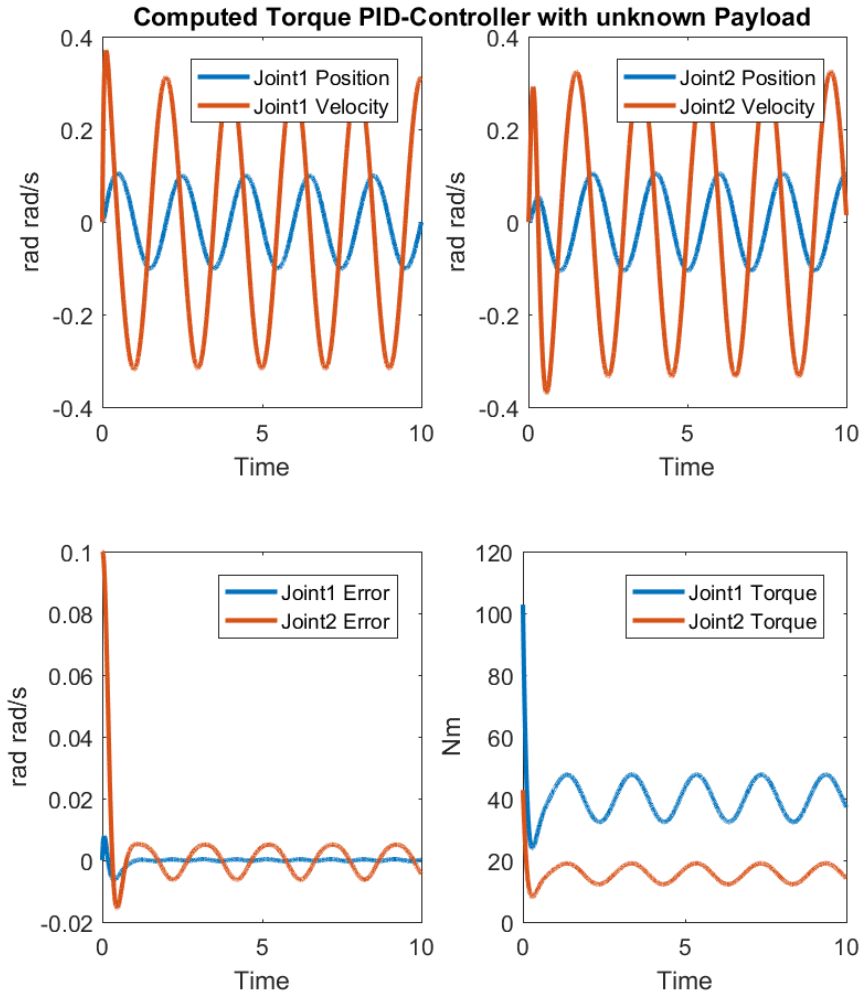

Figure 2.7: Computed Torque PID Controller with unknown Payload

3 Computed Torque like controllers

The previous chapter did present some implementations of computed controllers. Those controllers are relying on a reasonable model of the robot and if the robot's dynamics are changed, the controllers start to perform worse. More over the robot's dynamics do include a lot of terms where the deduction of a mathematical description is a hard task e.g. friction models.
Now a type of controllers are presented, which are a simplified version of the real computed torque controllers. Those controllers do not take the complete system dynamics of the robot into consideration, the only take a small portion of it and linearize the model with this term.

3.1 PD controller with gravity compensation

For this computed torque like controller the inertial and the centrifugal/coriolis term of the robot dynamics are neglected, only leaving the gravitational term. This leaves for the control law only the term:

$$\tau_c = -\mathbf{u} + \mathbf{g}(\mathbf{q})$$

The control scheme is given in figure 3.1
The controller is designed similar to the Computed Torque PD-Controller, with a coefficient comparison of the characteristic polynomial. For the damping a critical damping is required and for the frequency of the system ω_n two different values are chosen, 100 $\frac{rad}{s}$ and 10 $\frac{rad}{s}$.
Figure 3.2 shows the simulation with $100\frac{rad}{s}$. The system reaches zero steady-state error within 0.2seconds. Therefore the initial torque has a peak of around 1kNm, which is in a range, where it is unreasonable to use motors, which are supporting

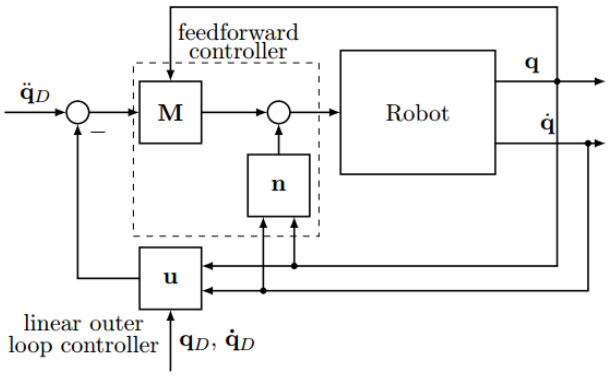

Figure 3.1: Controller overview of Computed Torque like controller with gravity compensation

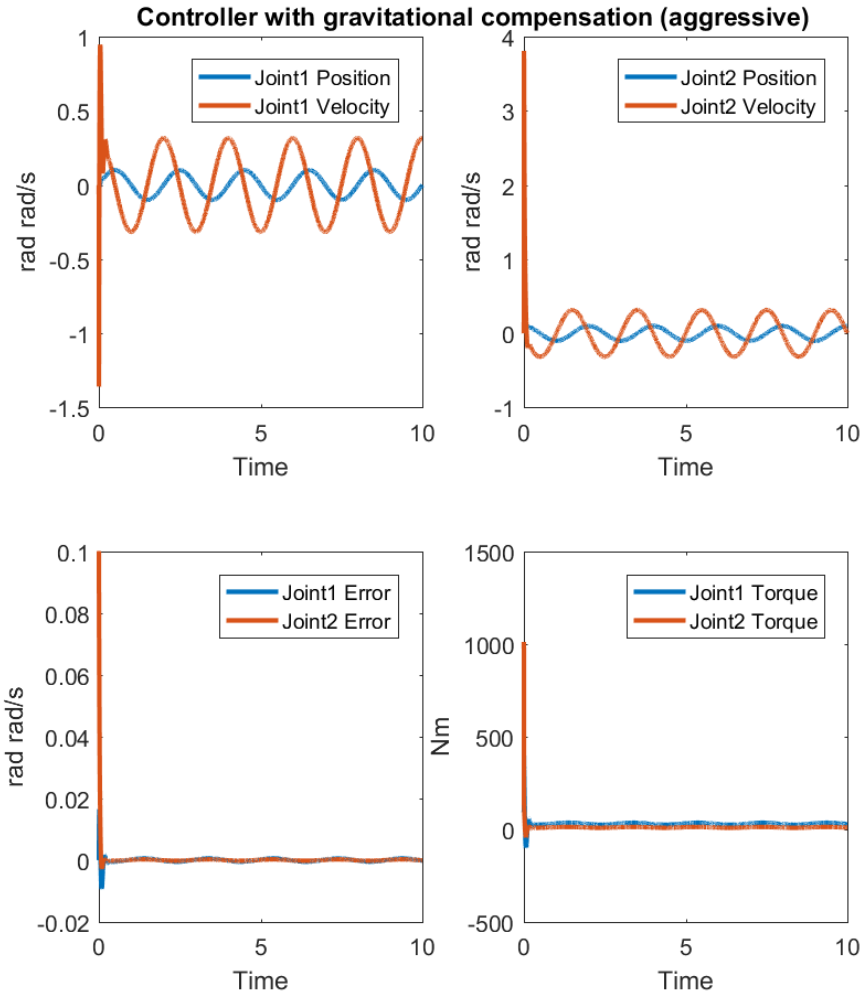

Figure 3.2: Aggressive Computed Torque like controller with gravitational compensation

this high torque.
In comparison the controller with $\omega_n = 10\frac{rad}{s}$ which is shown in figure 3.3 does show reasonable values for the torque. The torques needed for this controller oscillate around 30Nm/10Nm with an amplitude of 20Nm/10Nm.
Although a lot of system components have not been used for the controller design, the controller show reasonable results, which are accurate enough for the most real world tasks.

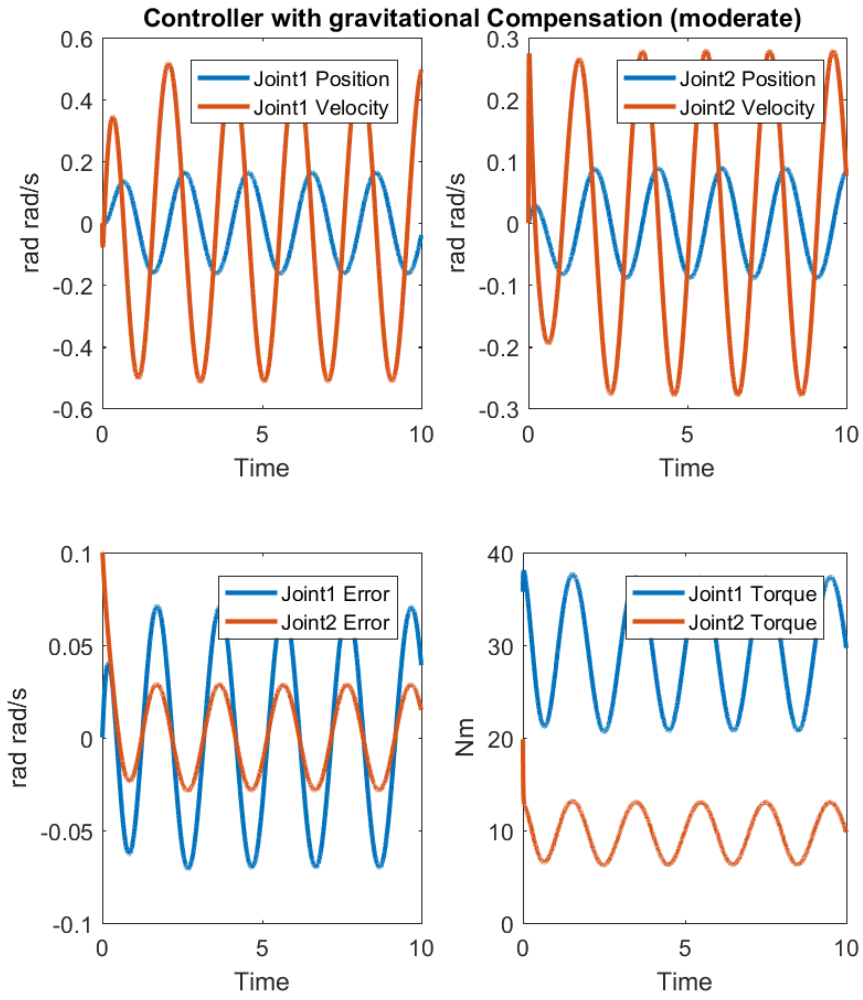

Figure 3.3: Moderate Computed Torque like controller with gravitational compensation

3.2 Classical PD Joint Control

The concept of classical joint controller further simplifies the model and even removes the gravitational model from the system. So the control law is given by:

$$\tau_c = -\mathbf{u}$$

The controller consists of two decoupled controller, one for each joint, they do not share any information or states, which might affect each offer. This approach is useful if the motor of robot arm uses a high gear ratio, since the gear reduces the influence of the non linear parts of the system. For motors with a high gear ration the problem from controlling a robot arm shifts to the controlling of the motor, which is easier to accomplish, since the motor itself is not changed, if the arm has some unknown disturbance.
For the controller design of the the PD-Controller a general second order polynomial is used. Again, critical damping is assumed during the controller design. The last free parameter ω_n is varied over different simulations. The used set of parameters is given in the following table:

	$k_{p,i}$	$k_{d,i}$	According Figure
$\omega_n = 10\ \frac{\text{rad}}{\text{s}}$:	100	20	3.4
$\omega_n = 25\ \frac{\text{rad}}{\text{s}}$:	625	50	3.5
$\omega_n = 50\ \frac{\text{rad}}{\text{s}}$:	2500	100	3.6

Table 3.1: Controller parameters for simulations witch classical PD controller

The first simulation, figure 3.4, show the results with $\omega_n = 10$. Here the error of the joints is relatively large. This huge error is caused by absence of he gravity within the design process. Therefore the torques of the joints do not show any peak.

The next parameter set, figure 3.5, does provide way less position error of the axes. Although the torques do have peaks at the beginning of around 60 Nm. After the initial peak the values of the torque decreases to same level as the previous simulation.
The last simulation uses the highest gains. Here an even higher peak torque is expected in comparison to the previous simulation. This expectation is valid, if the results from the last simulation is observed 3.6. The initial peak of the torque is around 250Nm. But this controller provides with its more aggressive parameter set an even more accurate tracking of the reference and the position error is reduced to arounf 0.01rads.

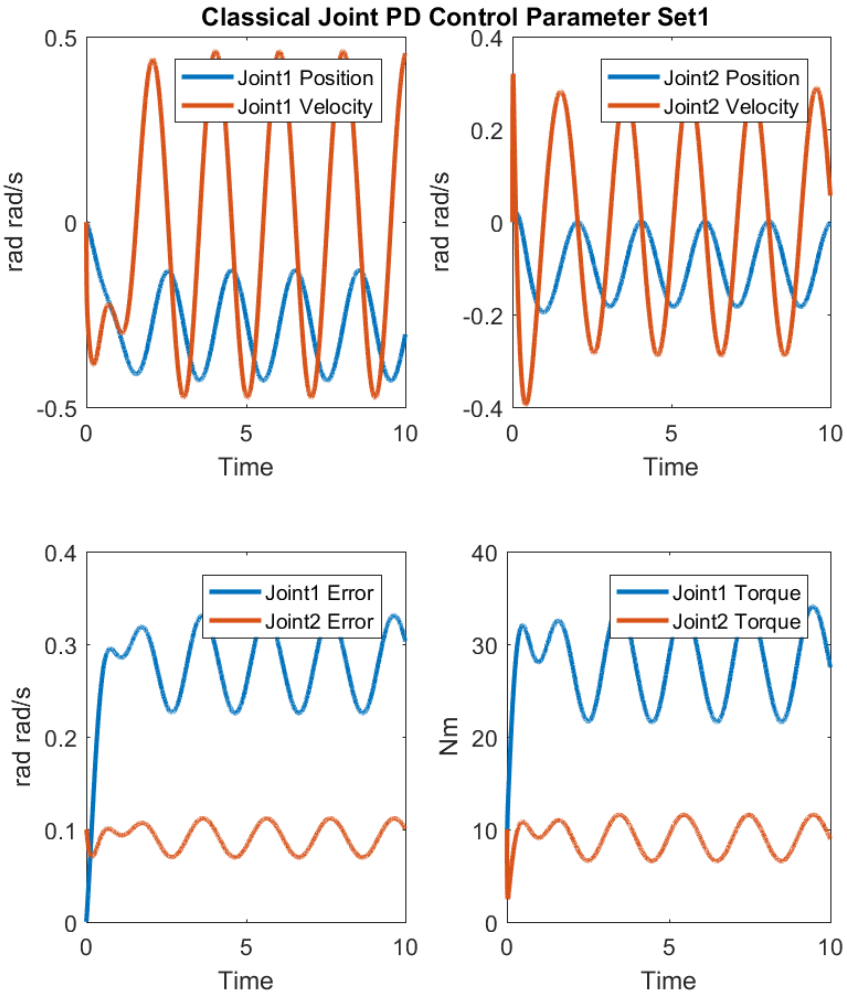

Figure 3.4: Classical PD Joint Control $\omega_n = 10$

3.3 Classical PID Joint control

All controllers from the previous section are not able to reach a steady-state error of zero. All controllers have an error unequal to zero even after longer time. This can be explained with the absence of the gravitational force within the design process. Anyway this errors, can be reduced taken into consideration and an integrator can be added to the controller. This integrator is able to take care of the gravitational term and compensate it.

By taking the characteristic polynomial of this controller:

$$\Delta(s) = J_i s^3 + (B_i + k_{d,i})s^2 + k_{p,i}s + k_{i,i}$$

To find the stability region for the controller parameters, the Routh-Hurwitz stability

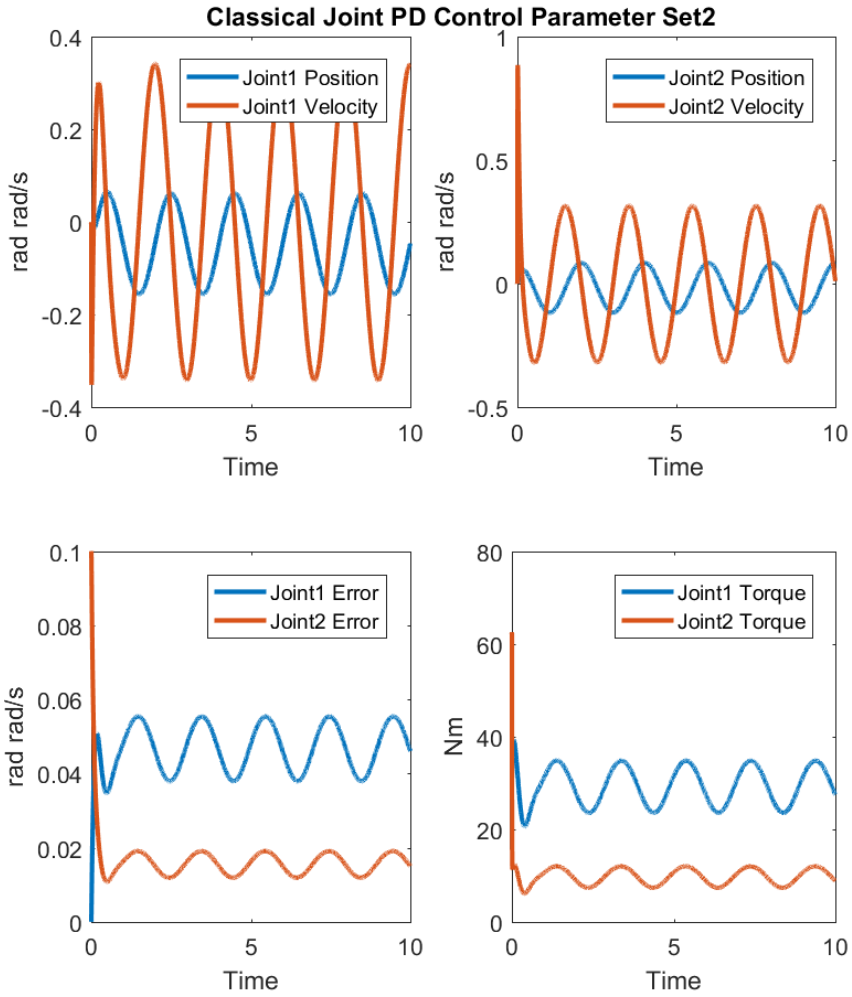

Figure 3.5: Classical PD Joint Control $\omega_n = 25$

criterion is applied.

$$\begin{matrix} J & k_p & 0 \\ B + k_d & k_i & 0 \\ \dfrac{(B + k_d)k_p - Jk_i}{B_i + k_d} & 0 & 0 \\ k_i & 0 & 0 \end{matrix}$$

Since the inertia is always positive the following conditions have to be fulfilled so that the closed loop controller is stable.

$$\begin{aligned} B + k_d &> 0 \\ k_i &> 0 \\ (B + k_d)\frac{k_p}{J} &> k_i \end{aligned}$$

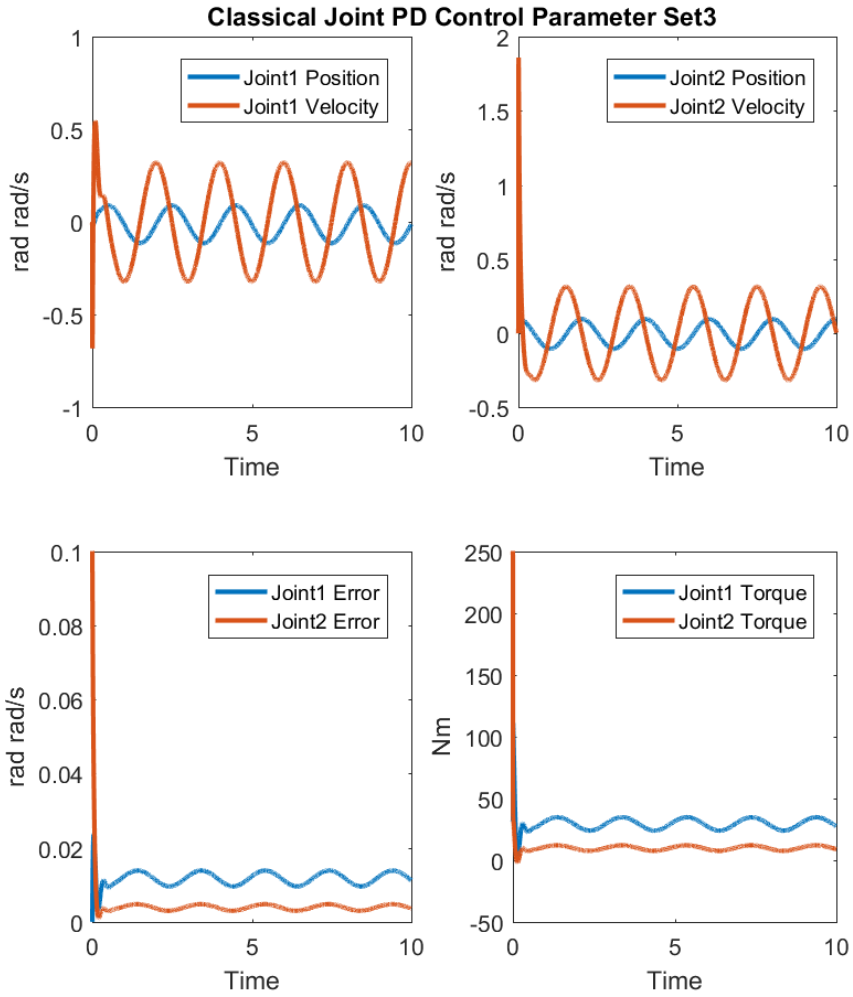

Figure 3.6: Classical PD Joint Control $\omega_n = 50$

This controller is used with 3 different set of gains. First with some moderate controller gains, second some aggressive controller gains and last the aggressive controller with some saturation within the torque. The saturation is simulated to match a real world hardware better, which is not able to supply infinite values of torque. The overview over the parameters is given within the next table:
In figure 3.7 the simulation with the moderate parameter set can be seen. Here the constant offset of the error has been removed and the error oscillates around zero. Although the amplitude of the error is rather high with 0.1rads.
In the next simulation which can be seen within figure 3.8 the dynamics of the error are greatly improved in comparison to the previous simulation. This can be explained by the way more aggressive parameter set. The drawback of this parameter set is the high initial peak within the torque. The max value of the torque is around 250Nm. Which makes this controller not really suitable for a real world implementation. The last simulation of the PID joint controllers, will utilize

	k_p	k_d	saturation	According Figure
$\omega_n = 10\,\frac{\text{rad}}{\text{s}}$:	300	30	$\pm\infty\,\text{N m}$	3.7
$\omega_n = 50\,\frac{\text{rad}}{\text{s}}$:	2540	100.4	$\pm\infty\,\text{N m}$	3.8
$\omega_n = 50\,\frac{\text{rad}}{\text{s}}$:	2540	100.4	$\pm 35\,\text{N m}$	3.9

Table 3.2: Controller parameters for simulations witch classical PID controller

the previous parameter set, which has shown suitable system dynamics, although it has high torque peaks. This simulation will saturate the torque signal at ±35Nm. The saturation is clearly visible within the simulation results. Comparing the error dynamics of the last simulation with the error from the previous one, there is almost no difference visible. So it can be said that the PID joint control is a suitable method for the control of a robot arm. The saturation itself does not provide any drawback and is usually automatically implemented within the motor driver.

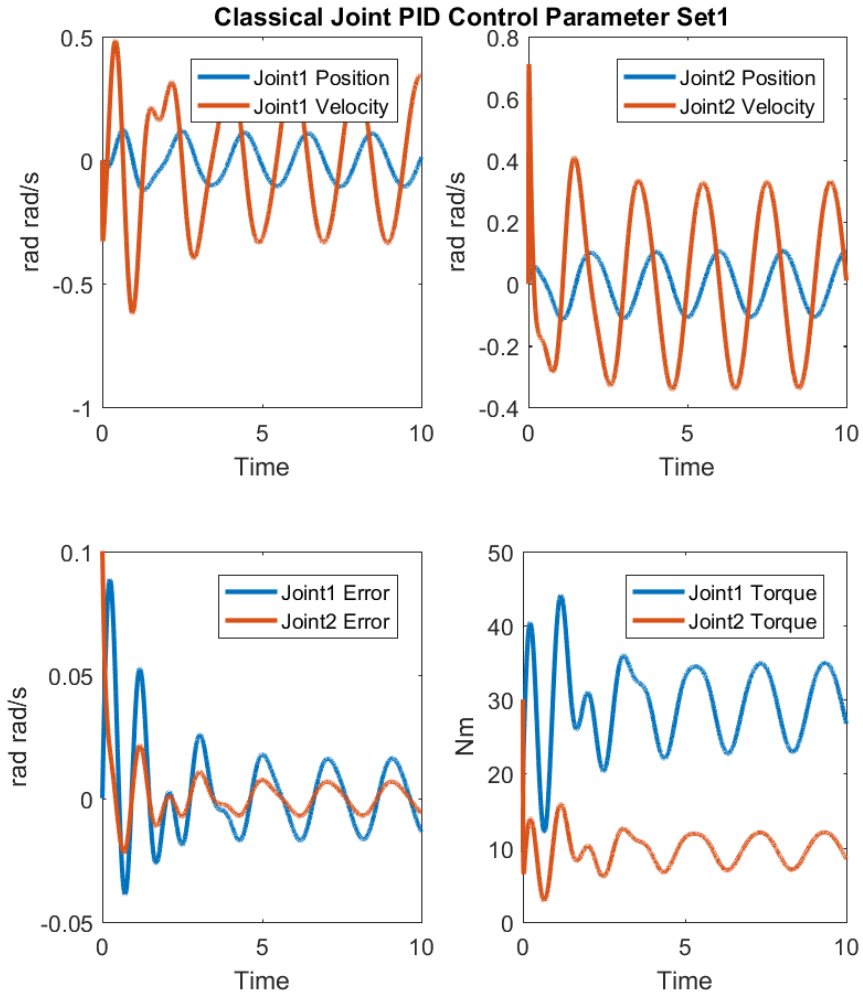

Figure 3.7: Classical PD Joint Control $\omega_n = 10$

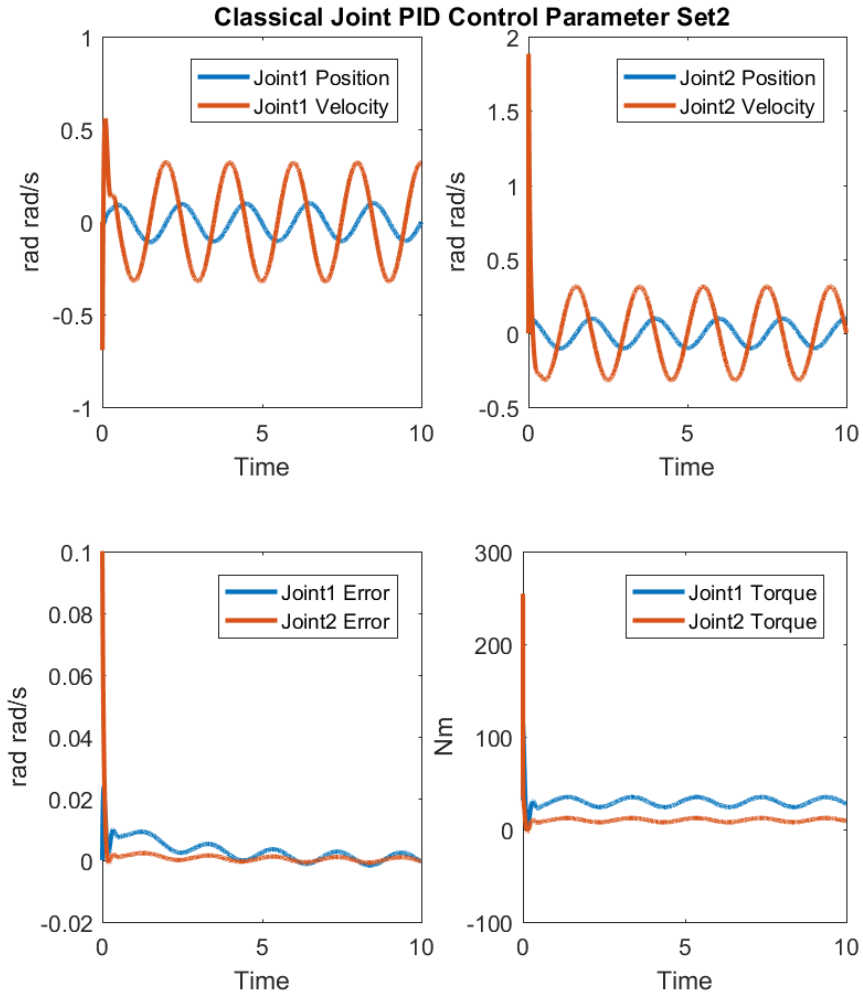

Figure 3.8: Classical PID Joint Control $\omega_n = 50$

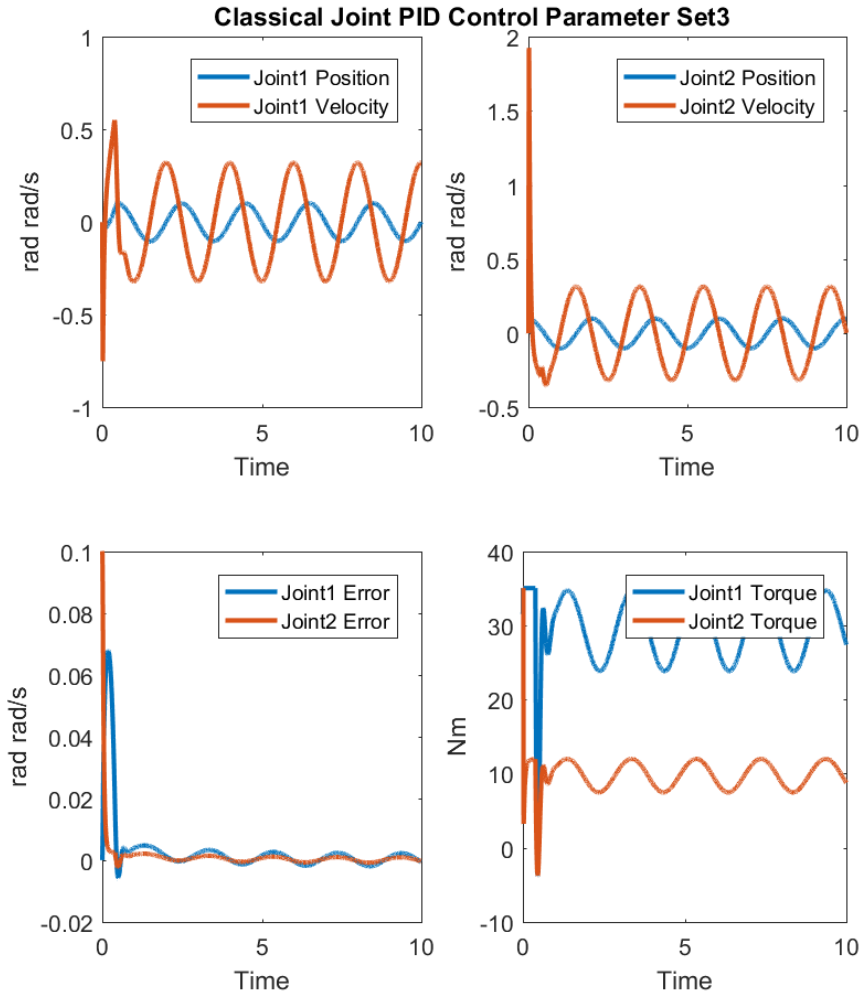

Figure 3.9: Classical PID Joint Control $\omega_n = 50$ with saturation

4 Practical issues at the realisation of robot control

When a controller for a robot needs to be implemented often a digital controller is needed. For the design of a discrete CT Controller the dynamics of the robot needs to be discretized, which leads to a very complicated system description and a even more complicated controller design. Since the CT Controller is a non-linear one, you cannot just discretize the controller and expect the same result. The only reasonable method is to use the discretized version of the CT controller as an approximation. For the following simulation and discretized version of a computed torque PD-controller will be used in a digital implementation. The robot arm itself will be still simulated with a variable step solver to contain the continuous behavior of the robot. The real world robot model is also a continuous one, so the behavior of the controller can be simulated within a good framework.
The parameter of the controller will be the same as the computed torque PD-Controller with critical damping 2.2. For this simulation the sampling time of the controller will be changed from 1ms, 50ms and 100ms. The rest of the simulation model will run with a variable step solver.
The first simulation, which can be seen in figure 4.1, show about the same behavior as the continuous implementation. The next simulation with a sampling time of 50ms, figure 4.2, starts to show some differences to the continuous implementation. The error signal has some minor oscillations. All of the are still pretty small and neglect-able for the most application.
If the sample time of the controller is further increased the controller starts to be unstable. This can be seen in figure 4.3. Here the controller itself is too slow to react on any changes of the model. Until the controller is calculated again the model has already moved so far, that the control action needs to be inverted. This leads to an oscillation with increasing amplitude.

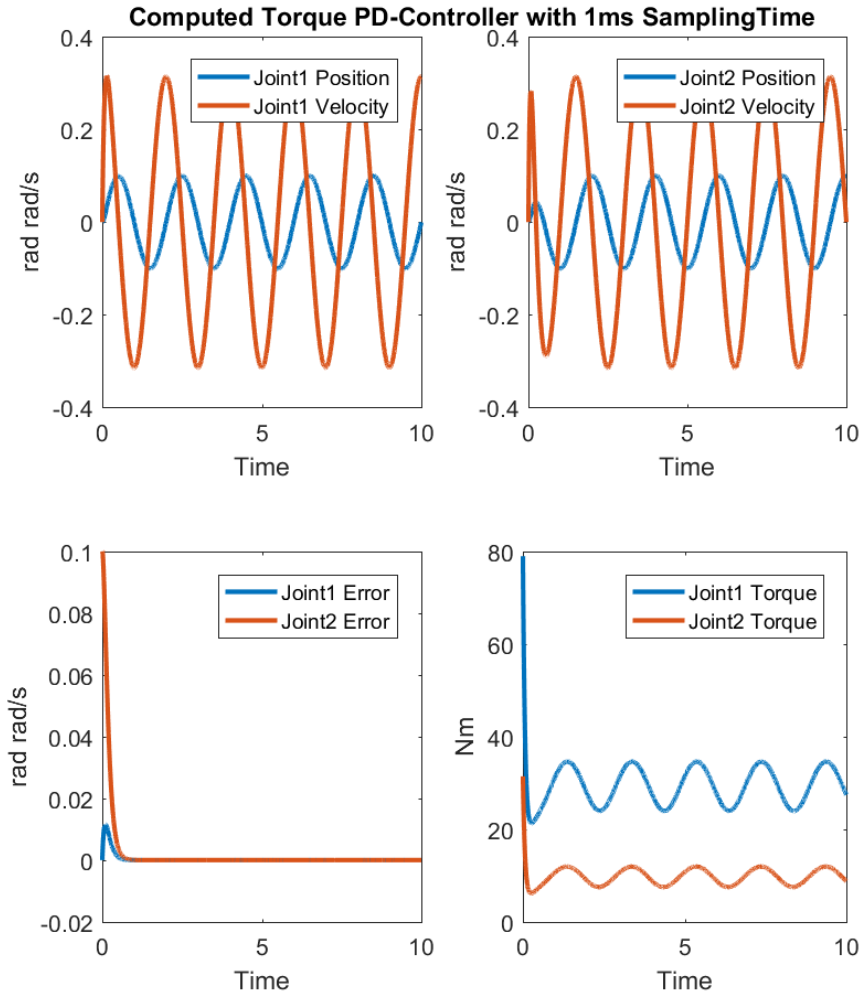

Figure 4.1: Discrete CT-PD Controller with $T_s = 1$ms

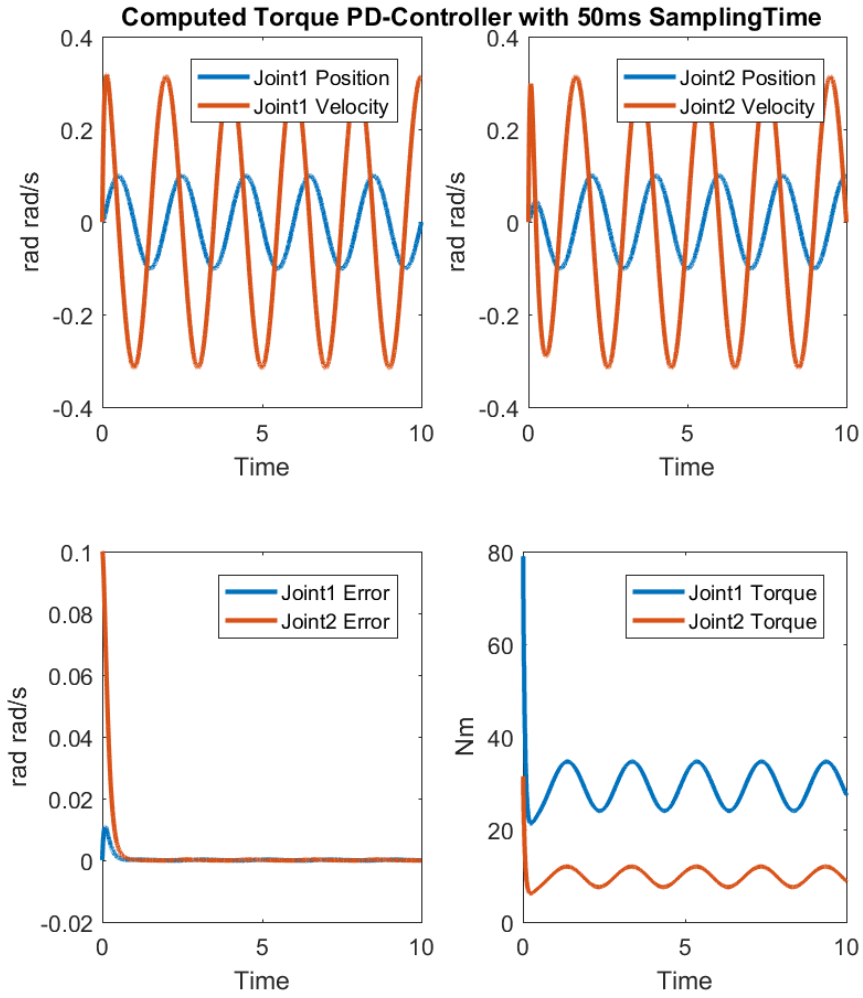

Figure 4.2: Discrete CT-PD Controller with $T_s = 50$ms

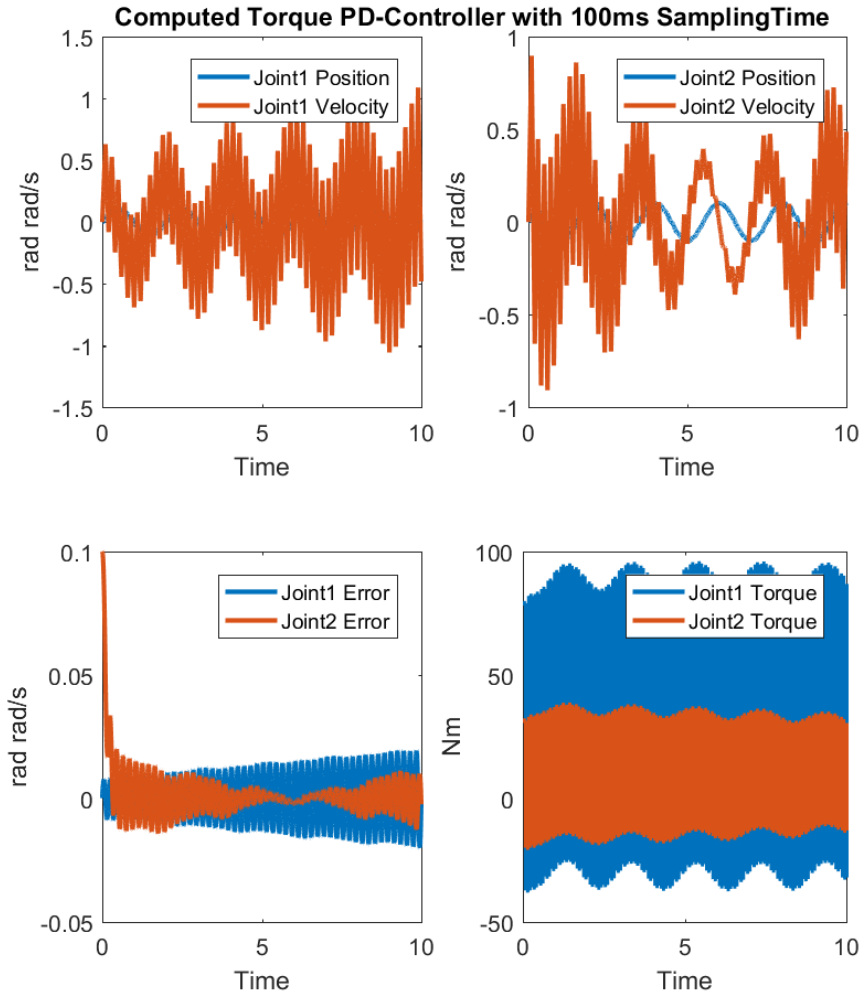

Figure 4.3: Discrete CT-PD Controller with $T_s = 100$ms

5 Computed Torque like Controller with disturbance estimator

Similar to the classical joint control, this controller is a completely decoupled controller, which control each joint independent and separately. In comparison to the classical joint control, which neglected all nonlinearities, this controller estimates the non linearities within the system for each joint. The controller consists of the average moment of inertia of the corresponding joint, a PD controller for the stabilization and the disturbance estimator to compensate for unknown or unmodelled system dynamics.

$$\tau_{c,i} = \bar{M}_i \ddot{q}_{calc} + \hat{w}(q, \dot{q})$$

where:

$\bar{M}_i$: average moment of inertia of the joint
$\ddot{q}_{calc}$: $\ddot{q}_D + k_d \dot{e} + k_p e$
$\hat{w}(q, \dot{q})$: estimated disturbances

For the estimation of the disturbances, a PI estimator of the following form is used:

$$\hat{w}(q, \dot{q}) = l(\dot{q}_{calc} - \dot{q})$$

where:

l : estimator parameter

The controller structure can be seen in figure 5.1.
For the simulation of this controller several different parameter sets are used, which can be seen within table 5.

l_i	$\frac{\omega_{estim}}{\omega_n}$	According Figure
20	2	5.2
100	2	5.3
100	10	5.4

Table 5.1: Controller parameters for simulations witch CT like controller with disturbance estimator

Figure 5.2 show the first simulation result. Here the controller needs around 20 seconds to settle and the initial peak of the torque is around 100Nm.

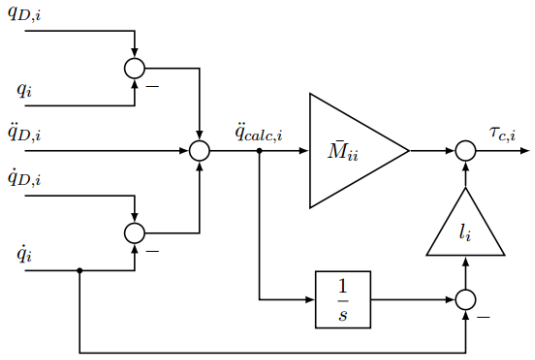

Figure 5.1: Controller overview of Computed Torque like controller with distortion estimator

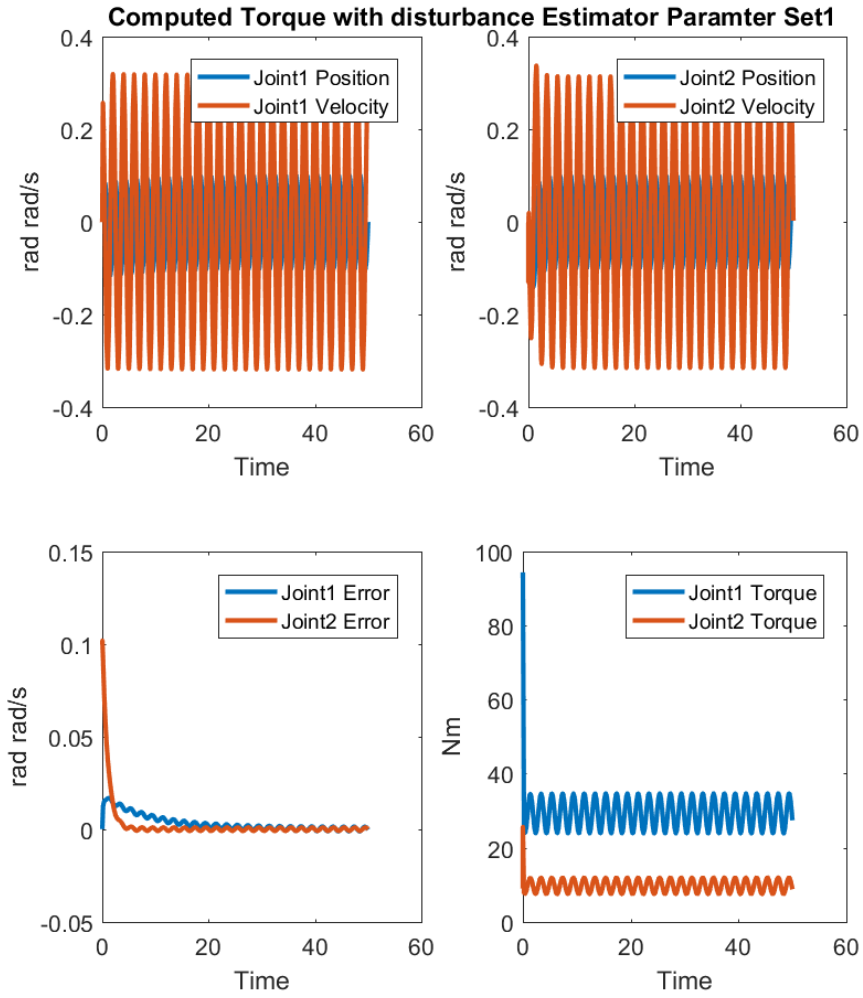

Figure 5.2: CT like Controller with disturbance estimator

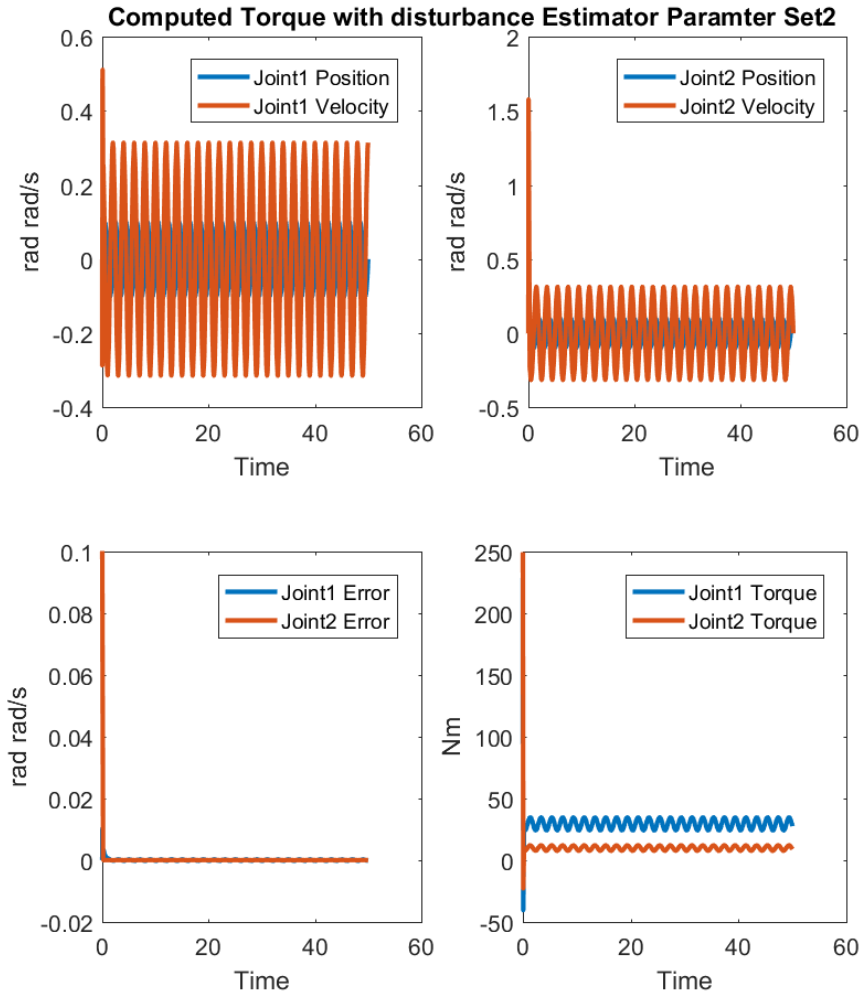

Figure 5.3: CT like Controller with disturbance estimator

The next simulation uses parameters with higher gains, which lead to a settling time of around 2seconds; figure 5.3. To reach the faster result a higher peak of the torque is needed.
The last parameter set combines the advantages from the previous simulations. It has a fast settling at around 5seconds, which is short compared to the first simulation. And the torque of the system does not show similar peaks as the previous simulation with a maximum of around 100Nm.

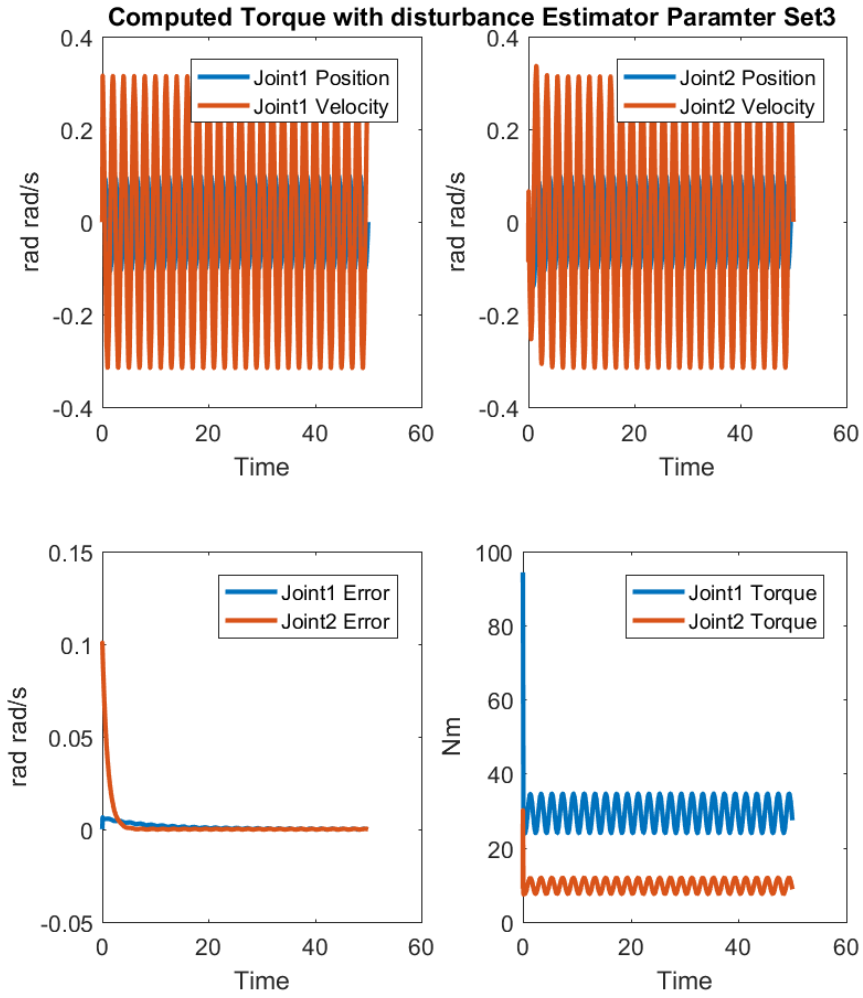

Figure 5.4: CT like Controller with disturbance estimator

6 Sliding Mode controller

The idea behind sliding mode is to force the states of a system to a so called sliding manifold and let it slide along this surface to the origin. When the states are on the surface the system show reduced order dynamics. To keep the system on the manifold a high frequency switching signal is needed. This can be obtained easily with a sign. As this would need to infinite fast switching , which cannot be realized, a boundary layer is introduced. This layer is achieved by using, instead of sign-function, another function like tanh.
To stabilize the error system of the robot arm the following sliding manifold is used:

$$\mathbf{s} = \dot{\mathbf{e}} + \Lambda \mathbf{e}$$

where:

Λ : diagonal matrix with time constants of the systems, when in sliding mode

This system corresponds to a first order differential equation. To be stable the eigenvalues of this system needs to be negative. To control the robot the following control law is used:

$$\tau_c = \hat{\mathbf{M}}\ddot{\mathbf{q}}_s + \hat{\mathbf{V}}_m\dot{\mathbf{q}}_s + \hat{\mathbf{g}} + \mathbf{K}sign(\mathbf{s})$$

where:

$\hat{\mathbf{M}}$: estimated inertia matrix
$\hat{\mathbf{V}}_m$: estimated centrifugal/coriolis matrix
$\dot{\mathbf{q}}_s$: $\Lambda\mathbf{e} + \dot{\mathbf{q}}_D$
$\hat{\mathbf{g}}$: estimated gravitation vector
$\mathbf{K}$: diagonal matrix with gains

The parameter $\mathbf{K}$ is used to compensate for the unknown or unmodelled dynamics. The better the model is the smaller this parameter can be chosen. For the simulation following parameter sets have been chosen:
In figure 6.1 it can be seen that the error converges after 0.5 to 1 second. The torques show the typical high frequency switching of the sliding mode control law.
The simulation, which can be seen in figure 6.2, has the linear gain of the switching function reduced, compared to the previous simulation. The result of the less aggressive controller, can be seen in the convergence time of the error, which is about twice the time compared to the previous parameter set. The values of the torques are also reduced, since the controller does react less aggressive.
In figure 6.3 the simulation results are shown for a controller with increased values λ. The time to drive the error to zero is even more increased, compared to the previous simulations.

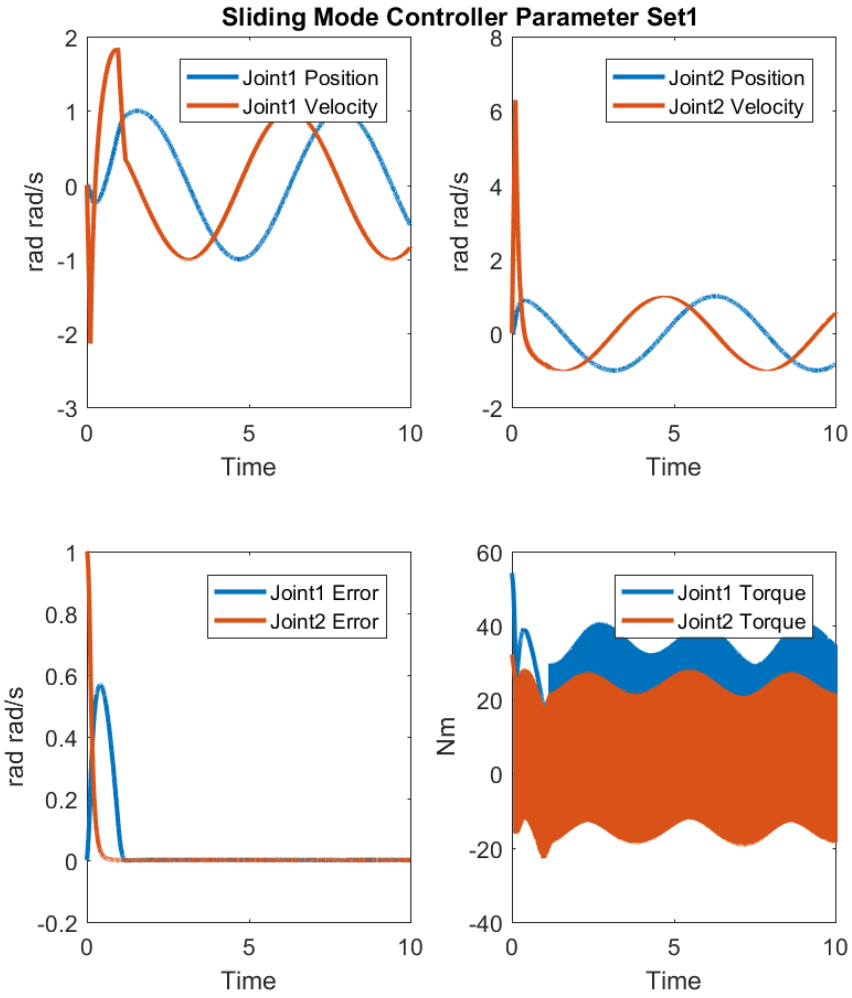

Figure 6.1: Sliding Mode Controller with $\lambda = 10, k = 20, \hat{\mathbf{g}} = 0.75\mathbf{g}$

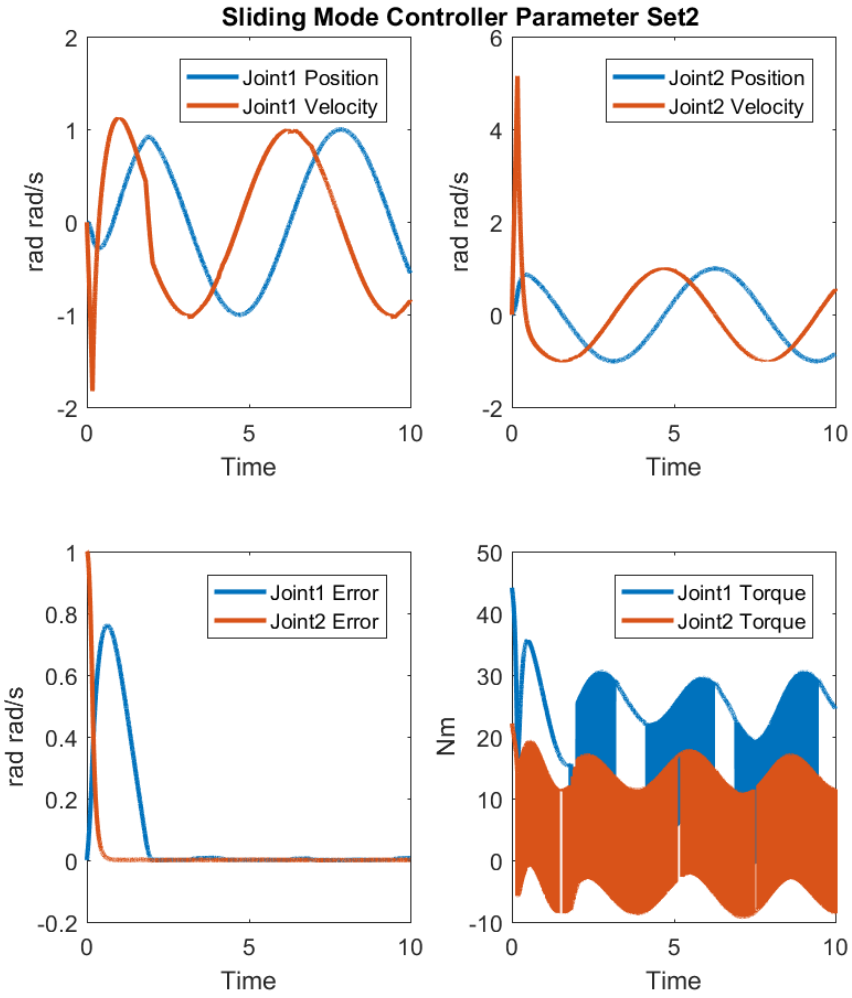

Figure 6.2: Sliding Mode Controller with $\lambda = 10, k = 10, \hat{\mathbf{g}} = 0.75\mathbf{g}$

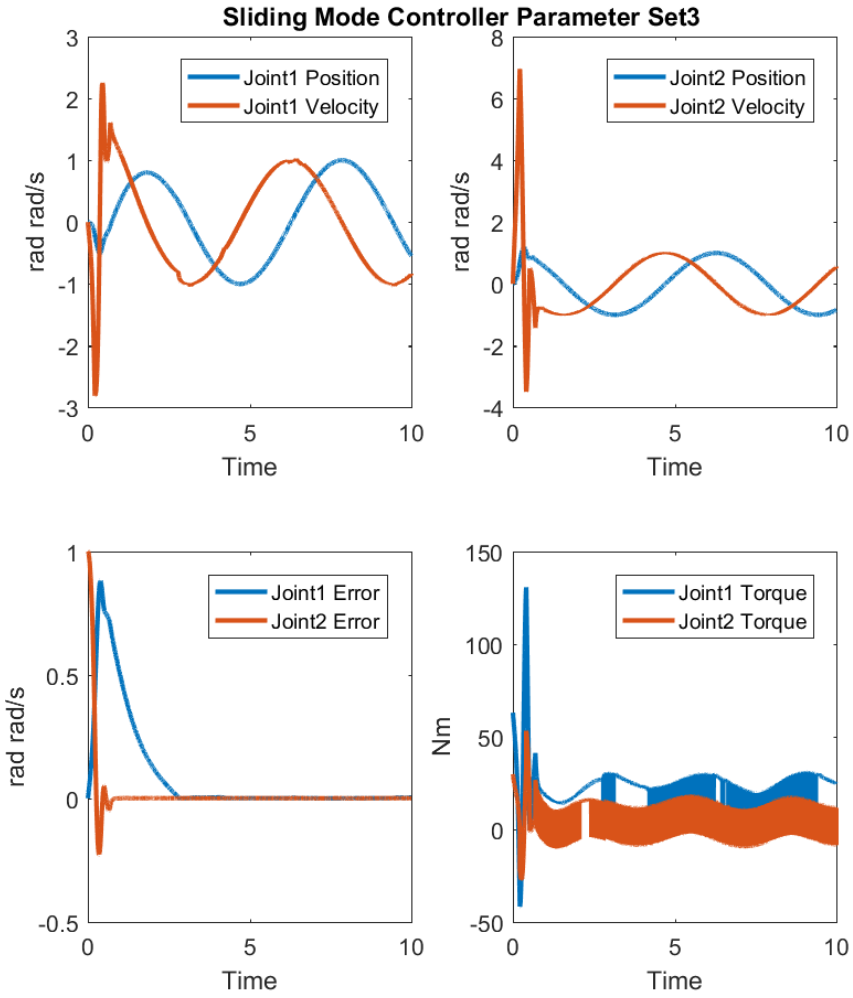

Figure 6.3: Sliding Mode Controller with $\lambda = 25, k = 20, \hat{\mathbf{g}} = 0.75\mathbf{g}$

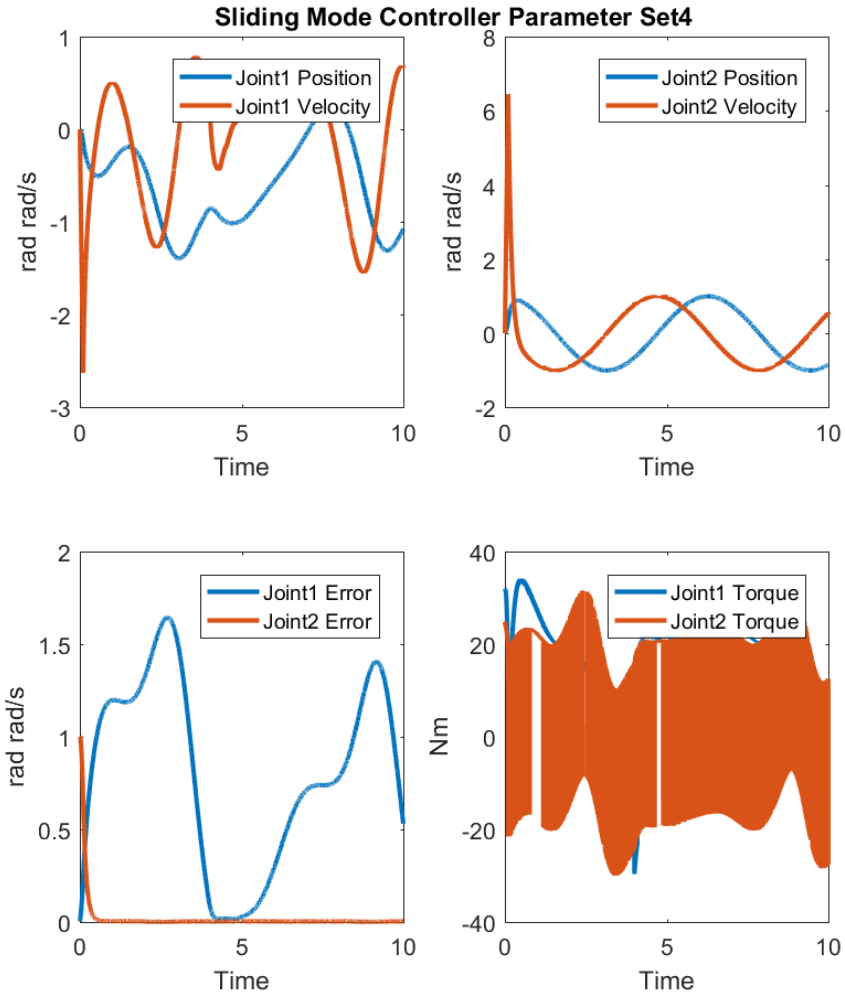

Figure 6.4: Sliding Mode Controller with $\lambda = 10, k = 20, \hat{\mathbf{g}} = 0\mathbf{g}$

λ	k	$\hat{\mathbf{g}}$	Switching Function	According Figure
10	20	$0.75\mathbf{g}$	sign(s)	6.1
10	10	$0.75\mathbf{g}$	sign(s)	6.2
25	20	$0.75\mathbf{g}$	sign(s)	6.3
10	20	$0\mathbf{g}$	sign(s)	6.4
10	30	$0\mathbf{g}$	sign(s)	6.5
10	20	$0\mathbf{g}$	tanh(s)	6.6
10	40	$0\mathbf{g}$	tanh(5s)	6.7

Table 6.1: Controller parameters for simulations with sliding mode controller

The next two simulation will show the results, if the gravitational component in the controller is ignored. The results in 6.4 show that the error of the first joint does not converge anymore.
The next simulation has no gravitational term within the controller, although the parameters are tuned for an optimal behavior, which lead to $\lambda = 10, k = 30$. It can be seen that the controller is able to drive the error to zero after 3 seconds.
To check the influence of the switching function itself the following two simulations have been done with a non-ideal switching function, which introduces a boundary layer around the switching manifold. In this case the *tanh* function has been used. In figure 6.6 it can be seen that the asymptotically stability of the error function did disappear and some oscillations can be seen. Therefore the torque itself does not show any high frequency switching anymore.
In figure 6.7 the results can be seen with a more aggressive switching function by scaling the sliding function itself, this leads to a narrower boundary layer around the switching manifold. The torque values still do not show the high frequency switching, but the convergence of the error shows better behavior and is almost asymptotically stable.

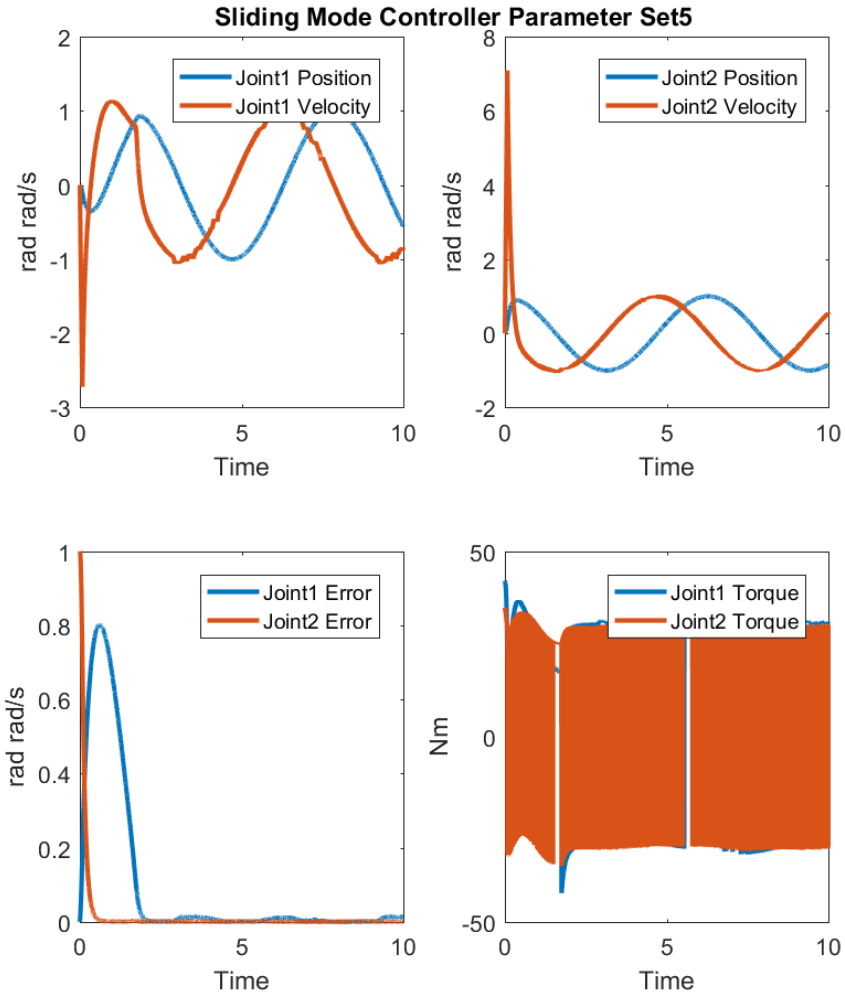

Figure 6.5: Sliding Mode Controller with $\lambda = 10, k = 30, \hat{\mathbf{g}} = 0\mathbf{g}$

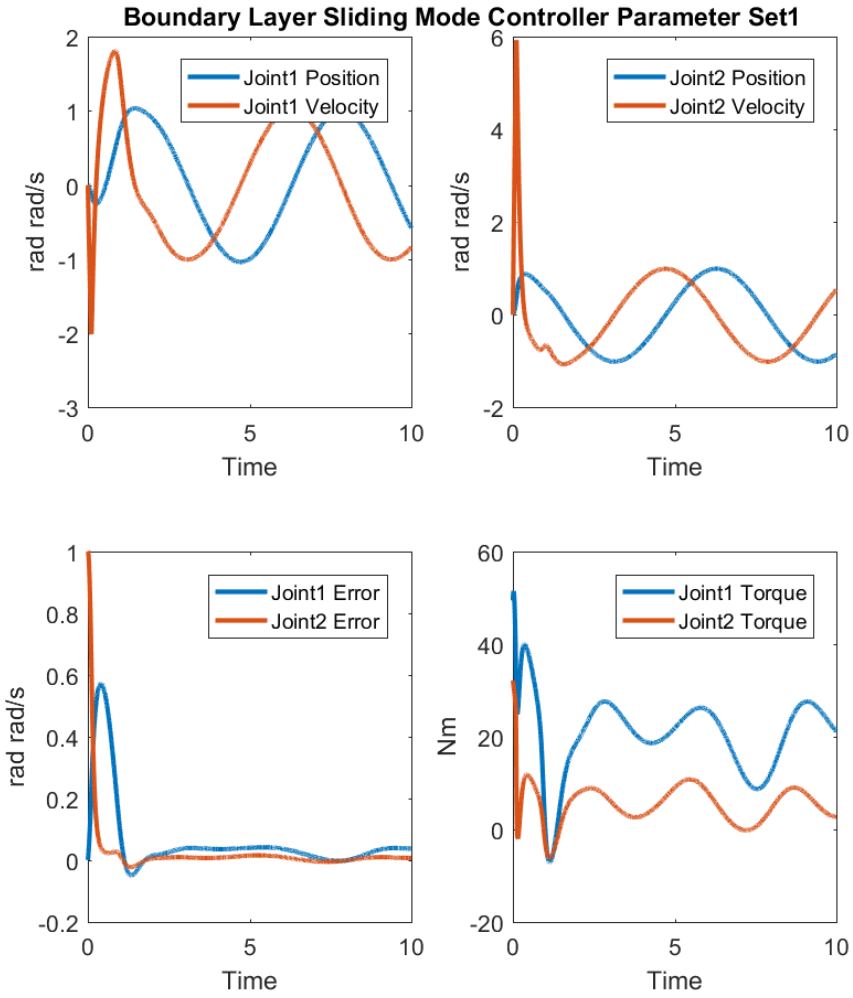

Figure 6.6: Boundary Layer Sliding Mode Controller with $\lambda = 10, k = 20, \hat{\mathbf{g}} = 0\mathbf{g}$

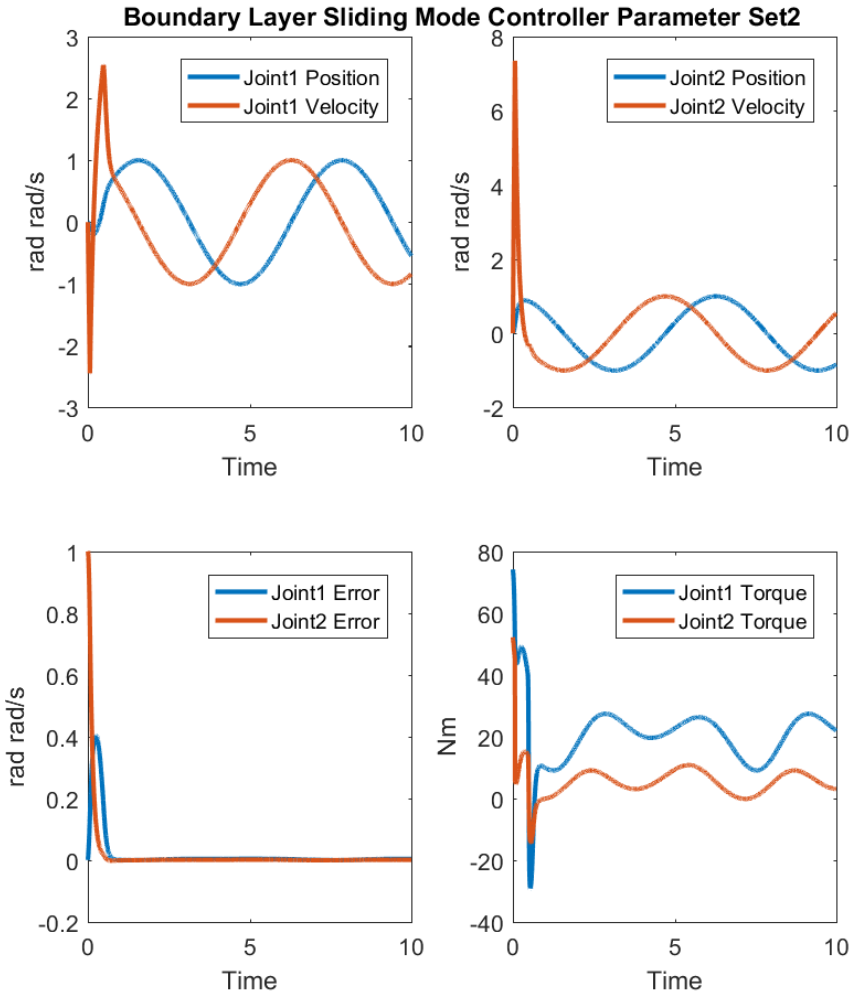

Figure 6.7: Boundary Layer Sliding Mode Controller with $\lambda = 10, k = 40, \hat{\mathbf{g}} = 0\mathbf{g}$

7 Adaptive Control

An adaptive controller uses some adaption rule to change the parameters of the controller, with the target the drive the error of the system to zero. In this approach the dynamic equation of the robot will the separated into a know part $\mathbf{Y}(.)$ and a part, which have to be adapted φ. In this case the masses of the system are assumed to be unknown. The following equation the known part can be calculated:

$$\mathbf{Y}(.)\varphi = \mathbf{M}(\ddot{\mathbf{q}}_D + \Lambda\dot{\mathbf{e}}) + \mathbf{V}_m(\dot{\mathbf{q}}_D + \Lambda\mathbf{e}) + \mathbf{g}$$

Since the masses are unknown the parameter vector φ consists of two masses and the rest of the dynamic equations is put into $\mathbf{Y}(.)$, which leads to:

$$\begin{aligned}
Y_{11} =& \left(\ddot{q}_{D,1} - \lambda_1\left(\dot{q}_1 - \dot{q}_{D,1}\right)\right) {a_1}^2 + g\cos(q_1)\, a_1 \\
Y_{12} =& g\left(a_2\cos(q_1+q_2) + a_1\cos(q_1)\right) + \left(\ddot{q}_{D,1} - \lambda_1\left(\dot{q}_1 - \dot{q}_{D,1}\right)\right)\left({a_1}^2 + 2\cos(q_2)\, a_1 a_2 + {a_2}^2\right) \\
&+ a_2\left(\ddot{q}_{D,2} - \lambda_2\left(\dot{q}_2 - \dot{q}_{D,2}\right)\right)\left(a_2 + a_1\cos(q_2)\right) - 2a_1a_2\dot{q}_2\sin(q_2)\left(\dot{q}_{D,1} - \lambda_1\left(q_1 - q_{D,1}\right)\right) \\
&- a_1a_2\dot{q}_2\sin(q_2)\left(\dot{q}_{D,2} - \lambda_2\left(q_2 - q_{D,2}\right)\right) \\
Y_{21} =& 0 \\
Y_{22} =& {a_2}^2\left(\ddot{q}_{D,2} - \lambda_2\left(\dot{q}_2 - \dot{q}_{D,2}\right)\right) + a_2\left(\ddot{q}_{D,1} - \lambda_1\left(\dot{q}_1 - \dot{q}_{D,1}\right)\right)\left(a_2 + a_1\cos(q_2)\right) + a_2 g\cos(q_1+q_2) \\
&+ a_1a_2\dot{q}_1\sin(q_2)\left(\dot{q}_{D,1} - \lambda_1\left(q_1 - q_{D,1}\right)\right)
\end{aligned}$$

Additional a function for the tracking error is needed:

$$\mathbf{r} = \Lambda\mathbf{e} + \dot{\mathbf{e}}$$

Λ is again a diagonal matrix which is positive definite. With these components, the control law can be built:

$$\tau_c = \mathbf{Y}(.)\hat{\varphi} + \mathbf{K}_v\mathbf{r}$$

where:

$\hat{\varphi}$: approximated system parameters
$\mathbf{K}_v$: diagonal gain matrix, positive definite

As a last component, the adaptation rule for the unknown parameters is needed:

$$\dot{\hat{\varphi}} = \Gamma\mathbf{Y}^T(.)\mathbf{r}$$

where:

Γ : diagonal matrix which defines adaptation speed

The simulation results in figure 7.1 are showing that the controller needs around 2 seconds to adapt to the right value of the masses. Caused by the initial wrong values of the masses the torque shows a very high peak.
The results in figure 7.2 are showing that the controller needs 3 seconds to adapt. Since the controller is reacting faster the peak within the error signal is quite higher.

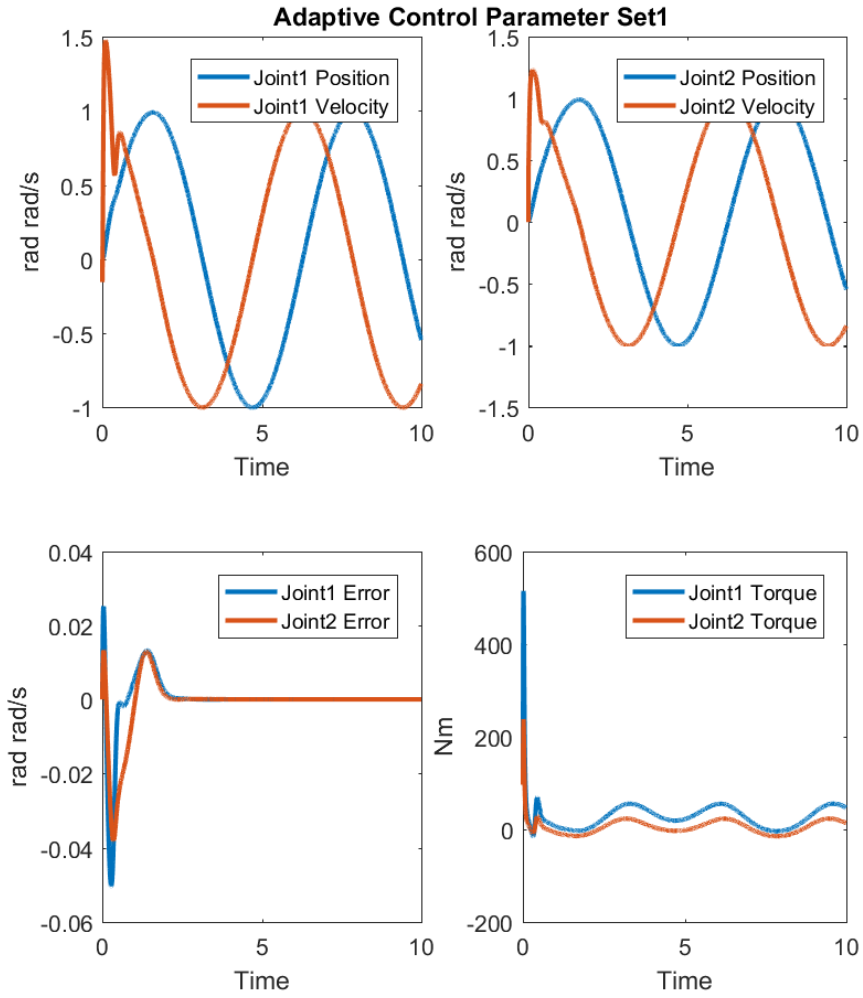

Figure 7.1: Adaptive Control with $\lambda = 5, \gamma = 10$ and $k_v = 100$

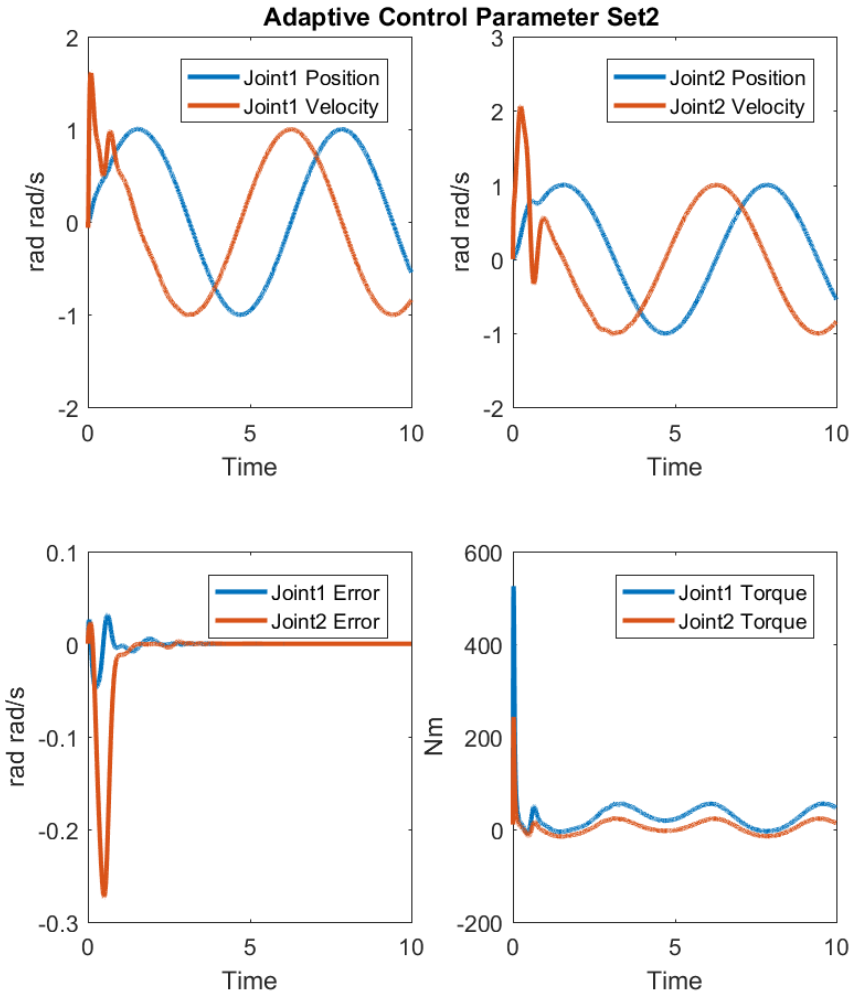

Figure 7.2: Adaptive Control with $\lambda = 5, \gamma = 10$ and $k_v = 10$