A study of Joint-Space Control of Non-Linear Robotics manipulators

Mohamad Qadri
Department of Computer Science, Carnegie Mellon University

Abstract

Robot manipulators have highly non-linear dynamics and as such, developing control frameworks for such systems is difficult problem. In this work, a framework was developed to control a robotic arm (The Barrett WAM 7 DOF) and perform a trajectory following task. First, the theory of Forward and Inverse Kinematics in addition to important implementation details are summarized. Then, the Equations of Motions for the WAM arm are derived using the theory of screw kinematics. Three different control strategies are studied, implemented, and tested in different settings. Then, experimental results and a comparative study are presented to illustrate the performance of each system. Finally, directions for future work are summarized.

1 Introduction

Robotic systems have various applications in a number of fields such as advanced manufacturing processes. They are used in tasks such as welding, product inspection and testing, precision machining, and assembly processing. Using such systems can lead to higher speed and better precision and accuracy when performing these tasks. A Robotic manipulator is composed of links connected by joints allowing the arm to rotate and translate. An end-effector is attached to the arm allowing it to interact with the environment. The motion of manipulators is typically controlled by a control system which calculates the torques needed to drive the arm from an initial pose to a target pose at any time. Jamshidi et al. [4] introduced decentralized stabilization using PID controllers plus a feedforward inverse dynamics compensation in the configuration space (Joint space). Khatib [3] introduced the operational space (end-effector) control formulation where the control of manipulator systems is performed with respect to the dynamic behavior of the end-effector. Nakanishi et. al performed in [2] a comparative analysis of different operational space control algorithm (velocity, acceleration and, Force control). This work focuses on joint space control as in [4]. It implements and compares different controllers and their performance for a manipulator trajectory following task. All experiments in this work were simulated using Peter Corke's Robotics's toolbox.

2 Forward Kinematics

The forward kinematics for the WAM arm is implemented using the equation $g_{st}(\theta) = e^{\hat{\xi}_1\theta_1}e^{\hat{\xi}_2\theta_2}e^{\hat{\xi}_3\theta_3}e^{\hat{\xi}_4\theta_4}e^{\hat{\xi}_5\theta_5}e^{\hat{\xi}_6\theta_6}e^{\hat{\xi}_7\theta_7}g_{st}(0)$ with $g_{st}(0) = \begin{bmatrix} 0 & -1 & 0 & 0.61 \\ 1 & 0 & 0 & 0.72 \\ 0 & 0 & 1 & 2.376 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ (given initial configuration) FK was run to update the joint angle values of a simulated arm and it was verified that the desired trajectory perfectly was reconstructed perfectly.

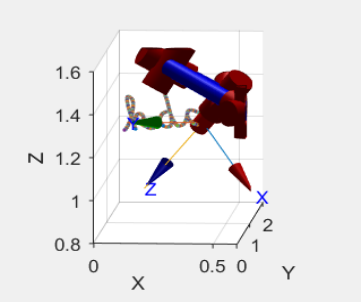

Figure 1: Reconstructed path using FK

3 Inverse Kinematics

Let the instantaneous velocity at the end-effector be the twist $\hat{V}^s_{st}$ then $\hat{V}^s_{st} = J^s_{st}(\theta)\dot{\theta}$ where $J^s_{st}(\theta)$ is the spatial manipulator Jacobian and $\dot{\theta}$ is the angular velocity of the robot's joint with $J^s_{st}(\theta) = \begin{bmatrix}\xi_1 & \xi'_2 & \xi'_3 & \xi_4 & \xi'_5 & \xi'_6 & \xi'_7\end{bmatrix}$ and $\xi'_i = Adj_{e^{\hat{\xi}_1\theta_1}\ldots e^{\hat{\xi}_{i-1}\theta_{i-1}}}$ (Adj is the adjoint operator). A differential change in the joint angle $\Delta\theta$ velocity will cause the end effector to change by ΔV with $\Delta V \approx J^s_{st}(\theta)\Delta\theta$. Define Δe as the instantaneous error between the current end effector pose and the target pose. The inverse kinematic problem aims at solving $\Delta e = J^s_{st}(\theta)\Delta\theta$. A solution $f(\Delta e) = \Delta\theta$ was computed using the damped least square method where $\Delta\theta = J^T(JJ^T + \lambda^2 I)^{-1}\Delta e$ for an appropriate choice of λ. For ease of integration with the simulation platform and address computational speed problems, the $ikine(\theta)$ provided by the robotics toolbox solves the IK problem using optimization (based on the Levenberg-Marquardt method) and was used instead of $f(\Delta e)$.

4 Equations Of Motion of the manipulator

The EOM of motions of an open-chain manipulator are $M(\theta)\ddot{\theta} + C(\theta, \dot{(\theta)})\dot{(\theta)} + N(\theta, \dot{\theta}) = \tau$ where M is the inertia matrix, C is the Coriolis matrix, N include the gravitational forces acting on the manipulator and τ is the vector of actuator torques.

4.1 Computing M for the WAM arm

Using the provided dynamics properties, a 4×3 transformation $g^i_{bj_i} = \begin{bmatrix} \mathbf{I_{3\times 3}} & q_i \\ \mathbf{zeros(1,3)} & 1 \end{bmatrix}$ was calculated to express the origin of the ith joint frame $\forall i \in [1..7]$ in terms of the base frame (q_i is the coordinate of a point lying on the joint axis of joint i). Another transformation $g^i_{jl_i} = \begin{bmatrix} \mathbf{I_{3\times 3}} & r_i^T \\ \mathbf{zeros(1,3)} & 1 \end{bmatrix}$ was found which gives the configuration of the ith link's COM in the ith joint frame where r_i is provided and denotes the location of COM of the ith link with respect to the ith joint frame. Finally, the initial configuration of the frame attached at the COM of the ith link axially aligned with the joint frame is expressed in terms of the base frame as $g^i_{bl_i} = g^i_{bj_i} g^i_{jl_i}$ $\forall i \in [1..7]$. $\mathbf{I_i}$, The inertia matrix for a frame at the COM, axially aligned with the ith joint frame is given by $Q_i diag(I_i) Q_i^T$. The body jacobian $J^b_{sl_i}(\theta)$ corresponding to $g^i_{bl_i}$ was computed for each link using the equation: $J^b_{sl_i}(\theta) = \begin{bmatrix} \xi_1^\dagger & \dots & \xi_i^\dagger & 0 & \dots & 0 \end{bmatrix}$ where $\xi_j^\dagger = Ad^{-1}_{(e^{\hat{\xi}_j\theta_j}\dots e^{\hat{\xi}_i\theta_i} g_{sl_i}(0))} \xi_j \quad \forall j \leq i$. Finally, we obtain $M(\theta) = \sum_{i=1}^n J_i^T(\theta)\mathcal{M}_i J_i(\theta)$ where $\mathcal{M}_i = \begin{bmatrix} \mathbf{I_{3\times 3}} m_i & \mathbf{zeros(3,3)} \\ \mathbf{zeros(3,3)} & \mathbf{I_i} \end{bmatrix}$ and m_i and $\mathbf{I_i}$ are the mass and inertial matrix of the ith link respectively. Note that the Matlab *inv* command was found to be slow when performing symbolic matrix inversion and therefore, Ad_g^{-1} was computed using the identity $Ad_g^{-1} = \begin{bmatrix} R^T & -R^T\hat{p} \\ 0 & R^T \end{bmatrix}$

4.2 Computing of N and C for the WAM arm

Once M is found, the C matrix can be computed as follows $C(\theta, \dot{\theta}) = \frac{1}{2}\sum_{i=1}^n (\frac{\partial M_{ij}}{\partial \theta_k} + \frac{\partial M_{ik}}{\partial \theta_j} - \frac{\partial M_{kj}}{\partial \theta_i})\dot{\theta}_k$ and $N(\theta, \dot{\theta}) = \frac{\partial V}{\partial \theta}$ where V is the potential energy of the manipulator.

Note that to speed up calculations, the *rne* method in the robotic toolbox was used to calculate the torques from the EOMs given $\theta, \dot{\theta}, \ddot{\theta}$. *rne* Computes the inverse dynamics using the recursive Newton-Euler formulation.

5 Models and Control algorithms

This section presents more details about the control algorithms. Experiments were performed on 1) a noise free system and 2) injecting a measurement noise into the first joint position measurement $\mathcal{N}(0, 0.0226 \text{ rad})$. The PIDs were all tuned on a robot model with parameters listed in project description table 3 and the control stategy was tested on a robot model with parameters in table 2.

5.1 Decentralized Joint Control

Fig 2 is the Simulink diagram for a decentralized controller in which the inertia and coupling forces are treated as disturbances in the control loop. Let $M(\theta) = \bar{M} + \Delta M(\theta)$, the EOM become: $\bar{M}\ddot{\theta} + \Delta M(\theta)\ddot{\theta} + C\dot{\theta} + N = \tau$ and treat the term $\Delta M(\theta)\ddot{\theta} + C\dot{\theta} + N$ as a disturbance d. An independent PID controller (PID = $k_p + k_i\frac{1}{s} + k_d\frac{N}{1+\frac{N}{s}}$) was designed and tuned to control each joint independently (Each pid joint control block implements one PID controller for one joint). The parameters were first tuned using Ziegler-Nichols method but it was found that better performance was achieved when tuning using the Simulink PID autotuner.

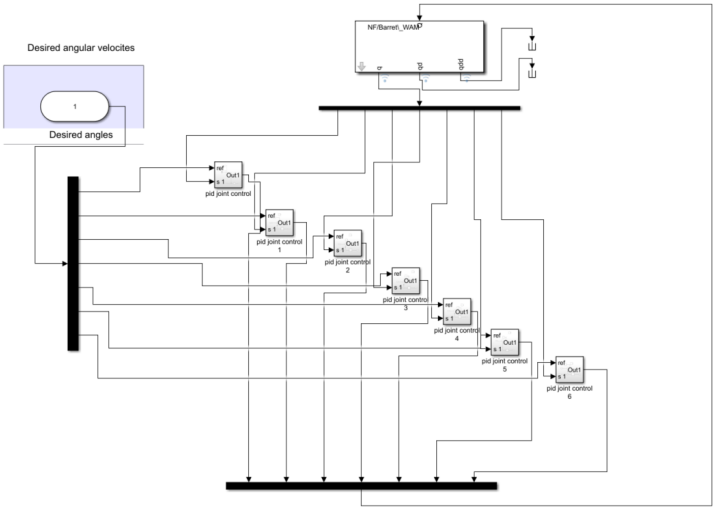

Figure 2: Decentralized Joint Controller

5.2 Feedforward Control

Fig 3 shows the plant and the implemented feedforward controller. [1] gives the equation of the generated torques:

$\tau* = M(\theta_d), \ddot{\theta}_d + C(\theta_d, \dot{\theta_d})\dot{\theta}_d + N(\theta_d) + K_v(\dot{q_d} - \dot{q}) + K_p(q_d - q) = \mathcal{D}^{-1}(\theta_d, \dot{\theta}_d, \ddot{\theta}_d) + K_v(\dot{q_d} - \dot{q}) + K_p(q_d - q)$ where $\mathcal{D}^{-1}$ is the inverse dynamics functions [inverse dynamics block in Figure 3] and K_p, K_d are gain matrices. Since the gain matrices are diagonal, and independent PID controller was designed for each joint similar to section 5.1. The feedforward control compensates for the gravity disturbance torque in pure decentralized control and incorporates computations of inertia and other dynamics into the control law [1].

5.3 Computed Torque Control

[1] and [5] describe the computed torque strategy shown in Figure 4. Computed torque control converts the nonlinear dynamics of the robot arm into linear dynam-

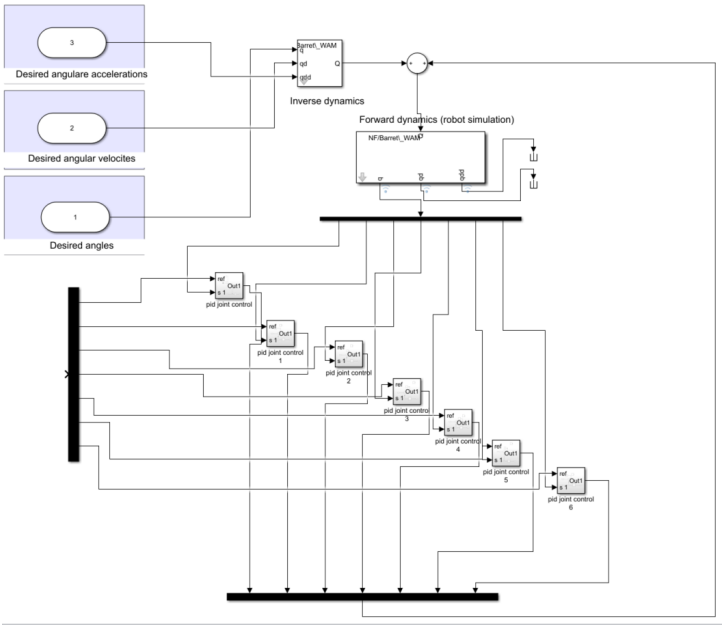

Figure 3: Feedforward Controller

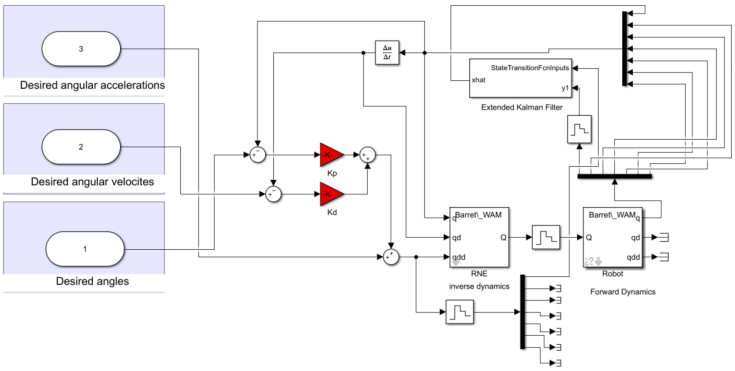

Figure 4: Computed torque controller

ics via feedback linearization (using full-state nonlinear feedback) which allows the usage of linear control strategy such as linear feedback control [5]. The control law equation is: $\tau = M(\theta)(\ddot{\theta}_d - K_d\dot{e} - K_p e) + C(\theta,\dot{\theta})\dot{\theta} + N(\theta,\dot{\theta})$ where $\ddot{\theta}_d$ is the desired angular acceleration and e is the position error term. The stability of the system was studied and established in [5]. Angular velocity was obtained by differentiating the noisy position measurements. The tested system was sensitive to noise and therefore an Extended Kalman filter (EKF) was used to filter our the additive noise from the first joint position measurements. The parameters used in the EKF are: state covariance = $diag([0 \quad 10e-5])$, measurement covariance = 0.000053. The sampling time was set to 5 times faster than the controller's sample time. The EKF measurement function is $y = x_{ik}(1)$ since the only observed state is the angle. i is the joint index and k is the time step. The EKF state transition is defined as $x_{ik} = x_{ik} + [x_{ik}(2); u_i/m_1] * dt$ where dt is the sample

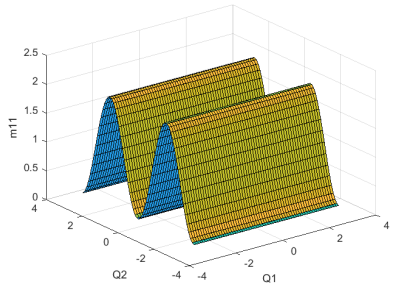

Figure 5: M(1,1) when changing J1 and J2 angles

time. m_1 is an average inertial value for the $M(1,1)$ element of the inertia matrix. This average value was found by analyzing how the value of m_{11} varied with different robot configuration and then taking an average value ($(max - min)/2$). Figure 5 shows an example of how the value of m_{11} changes when joint 1 and joint 2 angles are varied from $-\pi$ to π. *All PID parameters in addition to rise time, settling time, overshoot, gain and phase margins for the 3 controllers are specified in the appendix.*

6 Results

In this section, the results are presented for the different controllers. First a visual result overlapping the reference trajectory and the generated trajectory is presented for each controller for a perfect and noisy environment. Two error metrics are used: the per joint average error (JAE): $E(i) = \frac{1}{N}\sum_{k=1}^{N} |desired(\theta_i(k)) - response(\theta_i(k))|$ and the total averaged error (TAE): $E = \sum_{i=1}^{7}\frac{1}{N}\sum_{k=1}^{N} |desired(\theta_i(k)) - response(\theta_i(k))|$ where i is the joint index and k is the sample time.

6.1 Decentralized controller

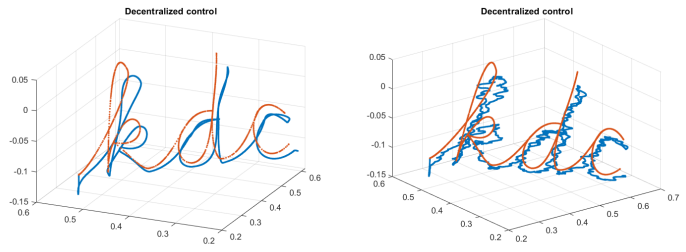

Figure 6: Decentralized controller (reference = orange). Right fig: noisy

joint index	JAE	TAE	joint index	JAE (noisy)	TAE (noisy)
1	0.0047	-	1	0.0103	-
2	0.0001	-	2	0.0002	-
3	0.0006	-	3	0.0020	-
4	0.0359	-	4	0.0354	-
5	0.0176	-	5	0.0177	-
6	0.0258	-	6	0.0260	-
7	0.0006	-	7	0.0022	-
-	-	0.0853	-	-	0.0937

Table 1: Decentralized control errors

6.2 Feedforward control

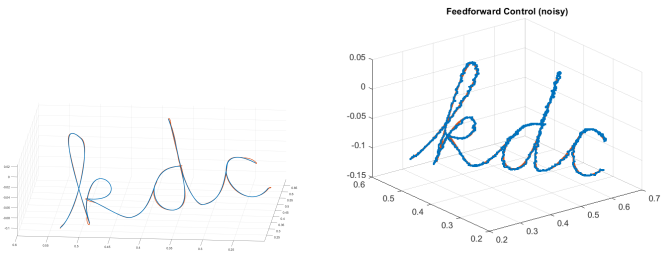

Figure 7: Feedforward control (reference = orange)

joint index	JAE	TAE	joint index	JAE (noisy)	TAE (noisy)
1	0.0013	-	1	0.0028	-
2	0.0002	-	2	0.0007	-
3	0.0002	-	3	0.0027	-
4	0.0001	-	4	0.0004	-
5	0.0002	-	5	0.0016	-
6	0.0000	-	6	0.0033	-
7	0.0006	-	7	0.0021	-
-	-	0.0028	-	-	0.0136

Table 2: Feedforward control errors

6.3 Computed Torque Control

joint index	JAE	TAE	joint index	JAE (noisy)	TAE (noisy)
1	0.0051	-	1	NA	-
2	0.0031	-	2	NA	-
3	0.0012	-	3	NA	-
4	0.0091	-	4	NA	-
5	0.0159	-	5	NA	-
6	0.0091	-	6	NA	-
7	0.0058	-	7	NA	-
-	-	0.0492	-	-	NA

Table 3: Computed torque control errors

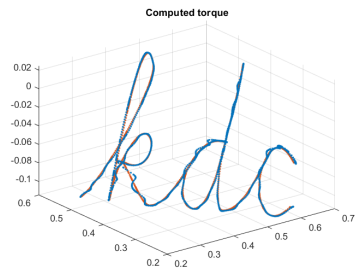

Figure 8: Computed torque control (reference = orange)

Figure 9 shows the position measurements and the reference angle for joint 1 when using computed torque control with an extended Kalman Filter in a noisy setting.

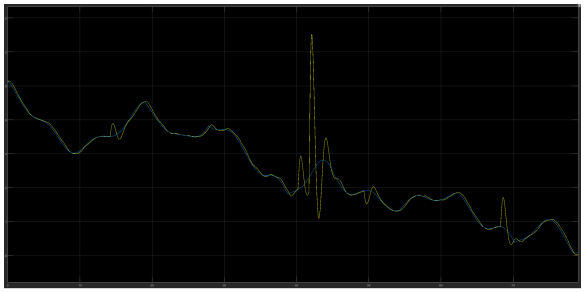

Figure 9: Reference signal in blue and measured joint 1 angle in yellow

7 Discussion

The computed torque controller was very sensitive to noise. Using an Extended Kalman Filter algorithm on top to filter our the noise resulted in a relatively better state estimate (Figure 9) over time and a smoother tracking performance. However, clear spikes in the measured joint angles can be seen, specifically when the derivative of reference signal flips signs (periods of transitions). Further analysis of the cause of these spikes is required before any potentially deploying the algorithm to a real system. Based on the TAE metric, the feedforward controller, achieved better tracking performance compared to pure joint decentralized control under both perfect ($TAE_{decentralized} = 30 * TAE_{FF}$) and noisy settings ($TAE_{decentralized} = 9 * TAE_{FF}$) which agrees with the result obtained in [4]. The feedforward controller also achieved better performance compared to the Computed Torque Control in the perfect setting ($TAE_{CT} = 15 * TAE_{FF}$). We can conclude that the feedforward controller performed best under the assumptions of our simulated environments. Further testing, better tuning of parameters, and additional theoretical analysis are needed before generalizing this conclusion. Future work for a broader comparative study is to implement and test against operational space controllers as defined in [3] and [2]. Manipulators control goals are usually formulated in Cartesian operational space and as such control strategies must be implemented in this space. Operational space also allows incorporating force control on top of position control which allows robots to perform more dexterous and a wider range of manipulation applications. Finally, It is worth noting that the topic of robotic safety (safe manipulation) was not discussed in the analysis of the different control algorithms. This should be an important aspect of future comparative studies.

References

[1] Peter I. Corke. *Robotics, Vision & Control: Fundamental Algorithms in MATLAB*. Second. ISBN 978-3-319-54413-7. Springer, 2017.

[2] Michael Mistry Jan Peters Stefan Schaal Jun Nakanishi Rick Cory. "Operational Space Control: A Theoretical and Empirical Comparison." In: *The International Journal of Robotics Research* 27 (2008), pp. 737–757. DOI: `https://doi-org.proxy.library.cmu.edu/10.1177/0278364908091463`.

[3] Oussama Khatib. "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space formulation." In: *IEEE Journal of Robotics and Automation* RA-1 (1987).

[4] Y.T.Kim M.Jamshidi H.Seraji. "Decentralized Control of Nonlinear Robot Manipulators." In: *Robotics* 3 (1987), pp. 361–370. DOI: `https://doi.org/10.1016/0167-8493(87)90053-2`.

[5] S. S. Sastry R. M. Murray Z. Li and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

8 Appendix

8.1 Decentralized Joint Control Parameters

joint index	k_p	k_i	k_d	N
1	5.44	1.23	4.85	22.7
2	1510	6860	38	297
3	73.62	181	6.76	112
4	33.8	NA	4	441
5	5	NA	0.41	6783
6	2.57	NA	0.22	1052
7	0.084	0.095	0.016	2224.66

Table 4: PID parameters (Feedforward Controller)

joint index	Rise-time	settling time	overshoot	Gain margin	phase margin
1	0.29s	2.58s	18 %	-25.3 dB @ 0.5 rad/s	64.9 deg @ 4.29 rad/s
2	0.0171s	0.244s	23.8 %	-19.6 dB @ 12.1 rad/s	78.1 deg @ 57.2 rad/s
3	0.0237s	0.217s	16.1 %	-29.8 dB @ 4.16 rad/s	90.4 deg @ 72.6 rad/s
4	0.0442s	0.33s	13.8 %	inf dB @ inf rad/s	77.6 deg @ 32.7 rad/s
5	0.0283s	0.239s	11.3 %	-74.2 dB @ 2.84 rad/s	82.5 deg @ 57.2 rad/s
6	0.0344	0.269s	13.7 %	inf dB @ inf rad/s	78.6 deg @ 43.3 rad/s
7	0.0663s	0.55s	14.6 %	-25.9 dB @ 2.41 rad/s	77.7 deg @ 22.6 rad/s

Table 5: Performance and robustness (Feedforward Controller)

8.2 Feedforward control parameters

joint index	k_p	k_i	k_d	N
1	0.21	0.009	1.15	5.46
2	105.58	49.44	15.92	1887.49
3	31.57	60.62	3.01	33.02
4	57.48	148.28	5.28	169.92
5	0.72	0.86	0.14	108.43
6	0.54	0.55	0.11	104.31
7	3.94e-6	1.33e-8	7.78e-5	13.46

Table 6: PID parameters (Feedforward Controller)

joint index	Rise-time	settling time	overshoot	Gain margin	phase margin
1	1.61s	12.8s	19.3 %	-25.9 dB @ 0.096 rad/s	65.7 deg @ 0.814 rad/s
2	0.103s	1.63s	23.4 %	-21.3 dB @ 1.94 rad/s	79.3 deg @ 14.4 rad/s
3	0.0375s	0.968s	11.6 %	inf dB @ inf rad/s	61 deg @ 33.9 rad/s
4	0.0259s	0.955s	13.8 %	-20.5 dB @ 6.02 rad/s	65.5 deg @ 48.2 rad/s
5	0.0653s	1.52s	13 %	inf dB @ inf rad/s	67.9 deg @ 20 rad/s
6	0.0688	1.59s	13.7 %	inf dB @ inf rad/s	69 deg @ 19.1 rad/s
7	12.1s	224s	12.9 %	inf dB @ inf rad/s	69 deg @ 0.118 rad/s

Table 7: Performance and robustness (Feedforward Controller)

8.3 Computed torque control parameters

$$K_p = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}$$

$$K_d = \begin{bmatrix} 1.8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 \end{bmatrix}$$