Chair of Sensor Based Robotic Systems and Intelligent Assistance Systems
School of Computation, Information and Technology
Technical University of Munich

Dynamics and Control of a 3DOF Planar Manipulator

Robot Programming and Control for Human Interaction (IN2308)

Tutors: Arne Sachtler, Michael Dyck

winter term 2024/25

Contents

1. Introduction

A short introduction to the dynamics of a 3DOF planar manipulator is treated and its dynamics model is derived. The control of the manipulator is then addressed starting from the general PD joints position control up to the Cartesian impedance controller. Further control features, like nullspace damping and singularity avoidance are considered. Finally, a collision estimator is built up.
Throughout the text, the student is introduced to the various aspects of each application. The student is asked to complete the theoretical section so as to prepare the background for the implementation of the treated algorithms. The difficulty of the problems presented in Section 2 varies widely. Simpler problems are marked by a single star '*', while higher difficulties are indicated by two stars '**' or even three stars '***'. We highly suggest to start solving the one star problems first.

The solutions to these problems allows to use the control environment presented in Section 3 and to implement a complete Cartesian impedance controller and several impedance-based control features in simulation. The successful completion of both parts gives a good overview of the control concepts for robotic applications.

1.1. Report Guidelines

Please follow this section carefully and make sure you follow the guidelines when preparing your report. You can use all a tool of your choice to prepare, but we suggest using LaTeX.

Organization

- Each student must write a report containing **their** solutions to the common parts **and** of the assigned homework part of the tutorial:
  → Common parts: (2.1 - 2.4, 3.1 - 3.4).
  → Individual HW parts: (2.5.1, 3.5.1) OR (2.5.2, 3.5.2) OR (2.5.3, 3.5.3).
  The HW will be assigned to the student by the tutor running the simulation exercise.
- If you like you can hand-write the solutions to the theoretical parts. You can them write on the respective pages of this document (pp. 3 - 8).
- The report and simulation files developed by the student (your .m and .slx/.mdl files, not the library) must be uploaded to **moodle**:
  – Report: `https://www.moodle.tum.de/mod/assign/view.php?id=3229477`
  – Simulation Files (zipped): `https://www.moodle.tum.de/mod/assign/view.php?id=3229481`

Format of the report

- Present all theoretical parts before the implementation parts.
- Keep task names as in the tutorial.
- The theoretical part can be handed in by printing the relative pages from the tutorial and completing them by hand.
- *Rules for Plots*
  – use white background,
  – use thick lines (e.g. `'linewidth'` ≥ 2),
  – state measurement units (e.g. $q_i[rad], \dot{q}_i[rad/s]$),
  – add a legend,
  – add a short caption/description.
- Not following these guidelines for the format can negatively impact the final grade.

Best Practices

- In general, every result should be commented.
- Only posting a plot without any explanation is not sufficient.
- Write concisely. There is no need to repeat concepts.
- **Always** state the value of the gains that you are using for a task, and quantify other variables like desired positions and noise.

2. Theory

2.1. Kinematics and Dynamics

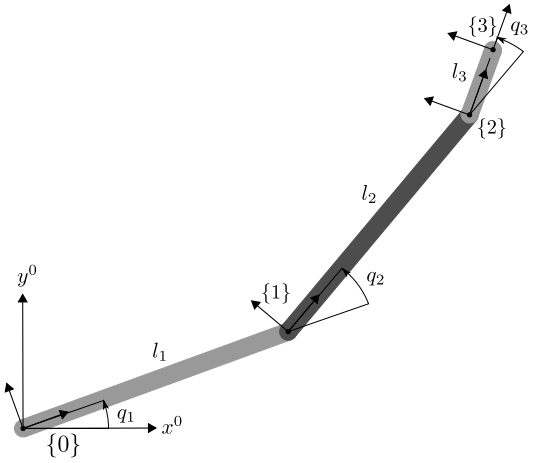


Figure 1: Kinematic structure of the 3DOF planar manipulator

To demonstrate the basic concept of impedance control, the model of a planar manipulator with three rotational degrees of freedom (DOF) will be addressed. The robot model consists of three rigid bodies which are connected via hinge joints. The kinematic structure of the robot is shown in Fig. 1. To describe the manipulator configuration, joint position variables $\mathbf{q} \in \mathbb{R}^n$ (where $n = 3$ is the number joint coordinates) are introduced. The relative position and orientation of frame $\{j\}$ w.r.t. $\{i\}$ is described by the homogeneous transformation

$$ {}^{i}\mathbf{T}_j = \begin{bmatrix} {}^{i}\mathbf{R}_j & {}^{i}\mathbf{p}_j \\ \mathbf{0} & 1 \end{bmatrix}, \tag{1} $$

where ${}^{i}\mathbf{R}_j \in SO(3)$ and ${}^{i}\mathbf{p}_j \in \mathbb{R}^3$ are the rotation matrix and position vector, respectively. By choosing a specific representation for the rotation (e.g., Cardan angles, Euler angles, etc.), Cartesian coordinates of the end-effector pose (i.e., position and orientation)

$$ \mathbf{x} = \mathbf{f}(\mathbf{q}) \in \mathbb{R}^m \tag{2} $$

can be derived from the homogeneous transformation between end-effector and base frame. The vector function $\mathbf{f}(\mathbf{q})$ gives a mapping between the n joint coordinates and the m Cartesian coordinates, where $n > m$ for redundant kinematic chains. For impedance control (and other Cartesian control schemes), the relation

$$ \dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\,\dot{\mathbf{q}} \tag{3} $$

is needed, which maps joint velocities to Cartesian velocities. Here, $\mathbf{J}(\mathbf{q}) = \frac{\partial \mathbf{f}(\mathbf{q})}{\partial \mathbf{q}} \in \mathbb{R}^{m \times n}$ is the Jacobian matrix. Notice that the Jacobian is expressed w.r.t. a coordinate frame, which depends on the choice of the reference frame for $\mathbf{f}(\mathbf{q})$. The components of $\mathbf{f}(\mathbf{q})$ and $\mathbf{J}(\mathbf{q})$ for the case of the 3DOF planar manipulator (see Fig.1) are listed in appendix A.
The dynamic model of the rigid body robot manipulator can be written:

$$ \mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \tau \tag{4} $$

Here $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n \times n}$ is the mass matrix, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^{n \times n}$ the Coriolis/centrifugal matrix, and $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^3$ the vector of gravity forces/torques. For the case of the 3DOF planar manipulator, the components of $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{g}$ are listed in the appendix A.
From a control point of view, two important properties of the matrices in (4) have to be remarked. The mass matrix is *symmetric* and *positive definite*, i.e.

$$ \mathbf{M}(\mathbf{q}) = \mathbf{M}(\mathbf{q})^T, \quad \mathbf{y}^T\mathbf{M}(\mathbf{q})\,\mathbf{y} > 0, \quad \forall \mathbf{y},\mathbf{q} \neq \mathbf{0} \in \mathbb{R}^n, \tag{5} $$

where $\mathbf{y} \in \mathbb{R}^n$ is arbitrary. This property is important for stability analyses. If $\mathbf{C}$ is derived via *Christoffel symbols*, the matrix $\dot{\mathbf{M}}(\mathbf{q}, \dot{\mathbf{q}}) - 2\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is *skew-symmetric*, i.e.

$$\mathbf{y}^T \left(\dot{\mathbf{M}}(\mathbf{q}, \dot{\mathbf{q}}) - 2\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \right) \mathbf{y} = 0, \quad \forall \mathbf{y}, \mathbf{q}, \dot{\mathbf{q}} \in \mathbb{R}^n, \tag{6}$$

which guarantees *passivity* of the dynamic system (4).

2.2. Joint Control*

The simplest way to control the configuration of a manipulator is to control directly the joint positions. Given a vector of desired joint angles $\mathbf{q}_d \in \mathbb{R}^n$, the error between the current and desired manipulator configuration is:

$$\tilde{\mathbf{q}} = \mathbf{q}_d - \mathbf{q} \tag{7}$$

Problem* 2.2.1: *Considering only the case of regulation ($\dot{\mathbf{q}}_d = 0$), write the law of the PD controller for reaching the desired joints position $\mathbf{q}_d$.*

Problem* 2.2.2: *Using the control law derived in Problem 2.2.1, write the equation of the dynamics of the controlled system.*

Neglecting the centrifugal and Coriolis forces and assuming a *quasi-stationary* variation, the controlled dynamics can be written:

$$\mathbf{M}(\mathbf{q}_0)\ddot{\tilde{\mathbf{q}}} + \mathbf{K}_d\dot{\tilde{\mathbf{q}}} + \mathbf{K}_p\tilde{\mathbf{q}} = 0 \tag{8}$$

Problem 2.2.3:** *Considering a constant stiffness matrix $\mathbf{K}_p$, derive a damping matrix $\mathbf{K}_d$ for achieving a given damping factor ζ of the system* (8)*.*

2.3. Translational Cartesian Impedance Control*

An alternative way to control the TCP pose is by means of the operational space formulation, as represented in Fig.2

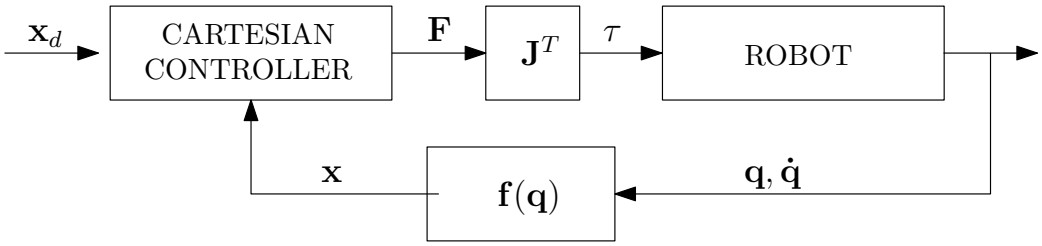


Figure 2: Scheme of cartesian control using the transpose Jacobian

Problem* 2.3.1: *The mapping from joint to Cartesian velocity space is given by (3). Derive the mapping between Cartesian forces and joint torques.*

The following steps will lead to the Cartesian impedance control law.

Problem* 2.3.2: *Considering a translational error* $\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{x}_d$ *and a constant, positive definite, symmetric stiffness matrix* $\mathbf{K}_t \in \mathbb{R}^2$*, write the law of a spring force* $\mathbf{F}_K$ *between the TCP of the manipulator and the desired point.*

Problem* 2.3.3: *Calculate the planar Cartesian force* $\mathbf{F}_D$ *resulting from a viscous damper with the constant symmetric damping matrix* $\mathbf{D}_t \in \mathbb{R}^2$*.*

Problem* 2.3.4: *The Cartesian impedance control law for the translational case can now be set up as the superposition of the solutions of the Problems 2.3.2 and 2.3.3. The transformation from Cartesian forces to joint torques was derived in Problem 2.3.1. Write the complete control law for the translational Cartesian impedance controller such that it can be commanded to the joint actuators.*

2.4. Collision Detection**

One of the features needed to allow a safe human-robot interaction is the ability to detect collisions. A possible way to detect a collision is to compare the commanded torques and the effective torques deduced from the current state of the system.

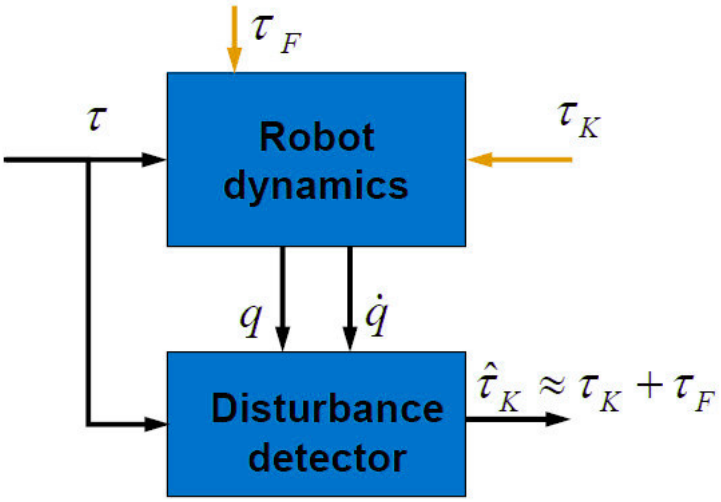


Figure 3: Scheme of estimation of the external torques

This scheme has the disadvantage that the estimated torque is the sum of both the external torques and the friction forces.

Problem* 2.4.1: *Draw a scheme for estimating only the external torques.*

The effective torque τ_e can be deduced either from the measurements or from the desired trajectory using the dynamic model of the manipulator:

$$\tau_e = \mathbf{M}(\mathbf{q}_n)\ddot{\mathbf{q}}_n + \mathbf{C}(\mathbf{q}_n, \dot{\mathbf{q}}_n)\mathbf{q}_n + \mathbf{g}(\mathbf{q}_n) \qquad \text{from measurements} \tag{9a}$$

$$\tau_e = \mathbf{M}(\mathbf{q}_d)\ddot{\mathbf{q}}_d + \mathbf{C}(\mathbf{q}_d, \dot{\mathbf{q}}_d)\mathbf{q}_d + \mathbf{g}(\mathbf{q}_d) \qquad \text{from desired trajectory} \tag{9b}$$

Problem* 2.4.2: *Explain the disadvantages of both approaches*

A better approach is to use the momentum instead of directly the torques. The momentum can be estimated by comparing the expected momentum caused by the commanded torques and the effective momentum of the system (obtained from the measurement of the state of the system). The generalized momentum of the manipulator is given by: $\mathbf{p} = \mathbf{M}(\mathbf{q})\dot{\mathbf{q}}$.

Problem 2.4.3:** *Considering the manipulator dynamics in Eq.*(4)*, derive the equation of the variation of the momentum.*

Problem 2.4.4:** *Draw the scheme of a collision estimator based on the momentum.*

Problem 2.4.5:** *Derive the equation of the dynamics of the estimator.*

Problem* 2.4.6: *Which reaction strategies can be adopted in case of a collision?*

2.5. Individual Work

2.5.1. (IK) Inverse Kinematics Control**

The robot tasks need to be specified most of the time in the operational space. A way is therefore needed in order to command the robot to reach a desired EE position in the Cartesian space.
A possible scheme of Cartesian control is represented in Fig.4. In this approach, the desired Cartesian position of the EE is inverted in order to obtain a desired joint position $\mathbf{q}_d$. The desired position $\mathbf{q}_d$ is then commanded to the robot by means of a joint controller, as developed in 2.2.

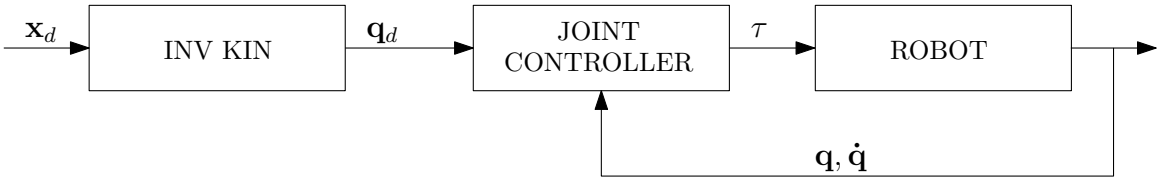


Figure 4: Scheme of Cartesian control using inverse kinematics.

Inverting the kinematics of a robot requires the solution of the equation:

$$\mathbf{x}_d = \mathbf{f}(\mathbf{q}_q) \tag{10}$$

where $\mathbf{x}_d = [x, y, \phi]^T$ is the desired EE position for the 3DOF planar robot, $\mathbf{f}(\mathbf{q})$ is the forward kinematics function, and $\mathbf{q}_d$ is the unknown joints position.
The problem of the inverse kinematics is complex and not always solvable in closed form. For complicated robot structures a numerical procedure is therefore needed in order to find one of the multiple solutions of (10).
Considering the monodimensional case of a continuos and regular function $y = f(x)$, one solution could be found by means of the Newton's method. The function $f(x)$ can be *locally* approximated by its tangent. The intersecation of this tangent and the abscissa axis gives a better approximation of the local solution x_d (i.e. such that $y_d = f(x_d)$):

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \qquad\qquad k = 0, 1, \ldots, n \tag{11}$$

Under certain hyphothesis, the iterative algorithm converges to the solution x_d. The iterations can be stopped when a convergence criterion has been satisfied:

$$|y_d - f(x)| \leq \varepsilon \tag{12}$$

Problem 2.5.1.1:** *Extend the Newton's method in order to derive an algorithm for solving the inverse kinematics problem.*

Problem* 2.5.1.2: *What happens in the vicinity of a singularity?*

Another approach for solving (10) is the gradient method. The desired joints position is found minimizing a cost function which indicates the distance of the calculated Cartesian position from the desired Cartesian position:

$$H(\mathbf{q}) = \frac{1}{2}\|\mathbf{x}_d - \mathbf{f}(\mathbf{q})\|^2 \tag{13}$$

Problem 2.5.1.3:** *Minimize equation* (13) *and derive the corresponding expression for the algorithm for solving the inverse kinematics problem.*

Problem* 2.5.1.4: *What happens in the vicinity of a singularity?*

Problem* 2.5.1.5: *Compare the two methods in terms of computational effort, behavior on singularity, convergence rate and application to redundant robots.*

Problem* 2.5.1.6: *Draw the equivalent loop scheme that implements the algorithm based on the transposed Jacobian.*

2.5.2. (NO) Nullspace Optimizations**

In Problem 2.3.4 a control law was treated which determines the translational Cartesian forces, i.e., the task space has $m = 2$ DOF. Since the considered manipulator has $n = 3$ joints, the configuration (joint angles) could change, while the end-effector is in a fixed position. The resulting motion is called a *nullspace motion*. In order to control the nullspace motion, a control law for the $r = n - m = 1$ *redundant* DOF will be derived. The complete impedance controller can be obtained by summing up the TCP impedance torque τ_i derived in Eq.(2.3.4) and a nullspace torque τ_n.

$$\tau = \tau_i + \tau_n \tag{14}$$

Problem 2.5.2.1:** *Write a control law τ_n for the nullspace and demonstrate that doesn't interfere with the TCP torque.*

Problem 2.5.2.2:** *Write the expression of the pseudo-inverse* $\mathbf{J}^+$ *and demonstrate that* $\mathbf{J}\mathbf{J}^+ = \mathbf{I}$.

Hint 1: *Use the simplest form of pseudo-inverse (Moore–Penrose)*

Problem* 2.5.2.3: *Define a damping nullspace torque* τ_0

The nullspace additional degree of freedom can be also used for making some optimizations on the configuration of the manipulator. A possible use case is the avoidance of singularities by means of additional nullspace torques that don't interfere with the TCP motion.

Problem* 2.5.2.4: *Using the mapping from Cartesian forces to joint torques* $\tau = \mathbf{J}^T\mathbf{F}$*, give a qualitative interpretation of the joint torques that result from forces acting in singular directions (how does the robot behave at a singularity?)*

An index of manipulability is $\mathbf{m}(\mathbf{q}) = \sqrt{\det(\mathbf{J}\mathbf{J}^T)}$ and for the considered 3DOF manipulator it's given by:

$$m_{\text{kin}}(\mathbf{q}) = \sqrt{l_1^2 l_2^2 \sin^2 q_2}. \tag{15}$$

As the determinant vanishes for singular $\mathbf{J}(\mathbf{q})$, this measure gives locally the distance to the singularity. A possible approach is to define a force field that repels the manipulator from singularities when the manipulability is too low.

Problem* **2.5.2.5:** *Define a quadratic potential function with a scalar coefficient k_s controlling the gain of the singularity avoidance. Use a piecewise definition to restrict the potential to the vicinity of the singularity.*

Problem** **2.5.2.6:** *Write a control torque that implements the singularity avoidance by means of the potential defined in 2.5.2.5.*

Problem* **2.5.2.7:** *Write the complete control law for the Cartesian impedance controller from Problem 2.3.4 and the nullspace optimizations (nullspace damping and singularity avoidance).*

2.5.3. (FCI) Full Cartesian Impedance Controller***

The translational Cartesian impedance controller from Section 2.3 allows only to set the position of the TCP and not its orientation. Therefore, the controller should be generalized to also allow for rotational stiffnesses. The general controller structure is maintained but extended for a rotational component. Although our 3DOF planar model can be described by $\mathbf{x} \in \mathbb{R}^3$, in the following Cartesian coordinates $\mathbf{x} \in \mathbb{R}^6$ are considered for generality. The full controller can be designed by splitting up $\tilde{\mathbf{x}}$ into two components $\tilde{\mathbf{x}}_t \in \mathbb{R}^3$ and $\tilde{\mathbf{x}}_r \in \mathbb{R}^3$ describing respectively the end effector position and orientation.

Problem* 2.5.3.1: *Write the Cartesian stiffness matrix* $\mathbf{K}_x \in \mathbb{R}^{6\times6}$*, partitioned into the translational stiffness* $\mathbf{K}_t$*, a rotational stiffness* $\mathbf{K}_r$*, and the coupling stiffnesses* $\mathbf{K}_c$*.*

The coupling stiffness $\mathbf{K}_c$ is omitted in the following, therefore $\mathbf{K}_c = 0$. In contrast to the vector difference as used in Section 2.3, the offset between the TCP frame and the desired frame is expressed by a homogeneous transformation matrix.

Problem* 2.5.3.2: *Write the transformation* ${}^{TCP}\mathbf{T}_d$ *from the TCP frame* $\{\mathrm{TCP}\}$ *to the desired frame* $\{d\}$*.*

The translational part of ${}^{TCP}\mathbf{T}_d$ can be directly used for the translational impedance controller. For the rotational stiffness, the situation is more complex. As there is no global minimal representation of $SO(3)$, the choice of orientation coordinates for use in Cartesian controllers is not straightforward. In the following, a solution based upon unit quaternions will be worked out.
Unit quaternions are a generalization of complex numbers and can represent rotations in $SO(3)$ in a similar way as complex numbers represent planar rotations. The advantage of unit quaternions is that they give a global parameterization and are therefore singularity-free in $SO(3)$. A unit quaternion consists of a scalar component η and a vector component ε. The components can be calculated by

$$\begin{aligned} \eta &= \frac{1}{2}\sqrt{1+r_{11}+r_{22}+r_{33}} \\ \varepsilon_1 &= \frac{r_{32}-r_{23}}{4\eta} \\ \varepsilon_2 &= \frac{r_{13}-r_{31}}{4\eta} \\ \varepsilon_3 &= \frac{r_{21}-r_{12}}{4\eta} \end{aligned} \tag{16}$$

where

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}.$$

The rotational error and the Cartesian rotational elastic force can then be computed as:

$$\tilde{\mathbf{x}}_r := 2\varepsilon \qquad \Longrightarrow \qquad \mathbf{F}_r = -\mathbf{K}_r\tilde{\mathbf{x}}_r \tag{17}$$

This choice of coordinates uses again only three parameters and is therefore also only a local parametrization. Its advantage is that it has no discontinuity and the only drawback is an unstable equilibrium point at $n = \pm\pi$.

Problem** **2.5.3.3:** *Derive a Jacobian for mapping the rotational forces to joint torques consistently with the quaternion representation used.*

Problem** **2.5.3.4:** *Assume a unit axis pointing along one of the Cartesian axes. Why is the choice of the quaternions for the spring torque convenient compared to a linear torque function? Draw and compare qualitatively the plots of the joint torques for both cases.*

As in the case of the PD joints controller, the damping matrix $\mathbf{D}$ shouldn't be held constant because the Cartesian mass matrix $\mathbf{M}_x(\mathbf{q})$ is not constant.

Problem* **2.5.3.5:** *Use* (3) *and the result from Problem 2.3.1 to transform the joint space dynamics* (4) *to Cartesian coordinates. Give an expression for the Cartesian mass matrix* $\mathbf{M}_x(\mathbf{q})$. *and Coriolis matrix* $\mathbf{C}_x(\mathbf{q})$

Assuming that the gravity is compensated and neglecting velocity-dependent terms, the equation of motion of the robot can be written as:

$$\mathbf{M}_x(\mathbf{q})\ddot{\mathbf{x}} = \mathbf{F} \tag{18}$$

where $\mathbf{M}_x(\mathbf{q}) \in \mathbb{R}^{6\times 6}$ is the Cartesian mass matrix and $\mathbf{F} \in \mathbb{R}^6$ is the generalized forces vector acting on the mass. Precisely, $\mathbf{F} = \mathbf{F}_c + \mathbf{F}_{\text{ext}}$ where $\mathbf{F}_c$ are the controller forces and $\mathbf{F}_{\text{ext}}$ are the external forces. In order to ensure a desired damping of the controller dynamics derived in (18), the matrix $\mathbf{D}$ should be accordingly designed.

Problem 2.5.3.6:** *Considering a constant stiffness matrix* $\mathbf{K}$*, derive a damping matrix* $\mathbf{D}$ *for achieving a given damping factor* ζ *of the complete system derived in* (18)*.*

Problem* 2.5.3.7: *Write the expression of the damping torque of the full impedance controller.*

Problem* 2.5.3.8: *Write the Full Cartesian Impedance controller, consisting of the sum of the translational and rotational stiffness, and damping derived in Problem 2.5.3.7. Define the impedances in TCP coordinates. Make sure all matrices are related to the correct frames.*

3. Simulation

This experimental section provides an introduction to the simulation environment of the 3DOF planar manipulator and the controllers presented in the theory section. Therefore, it is mandatory to solve the theory section first. The simulation should be implemented in Simulink. The library "m3dof_library.slx" is given to the students. It contains the implementations of the principal matrices needed for the implementation of the algorithms (forward kinematics, Jacobian, mass matrix, Coriolis and centrifugal matrix, gravity vector). Together with these matrices is given also a visualization utility for the 3DOF planar manipulator, which facilitates the understanding of the system behavior.

3.1. Kinematics and Dynamics**

Task 1: *Implement the dynamics of the 3DOF manipulator described in Sec. 2.1 as a standalone block. The inputs are the commanded torques τ and the outputs are the joints angles and velocities $\mathbf{q}$, $\dot{\mathbf{q}}$.*
Hint: *Add a dissipative term for simulating the friction at joints: $\tau_f = -\mathbf{K}_f \dot{\mathbf{q}}$*

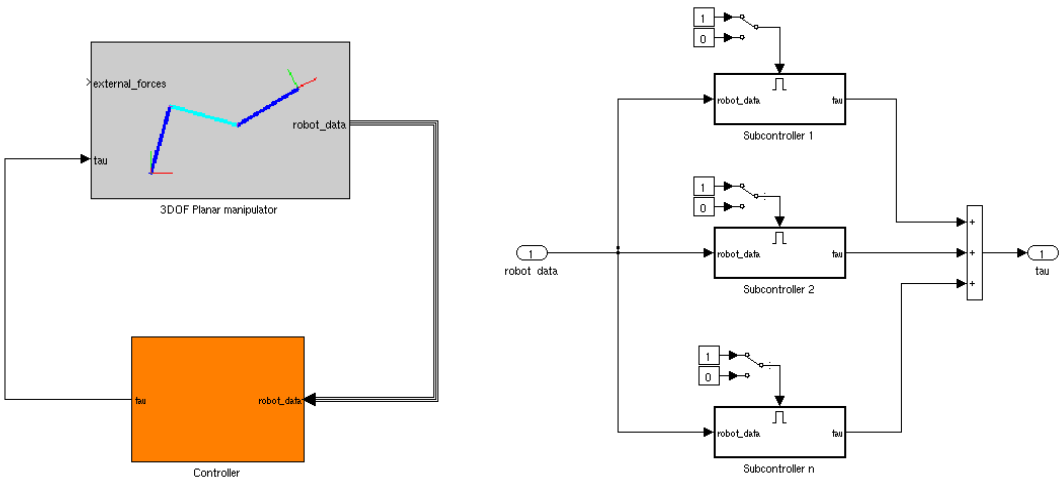
The block will be used throughout the entire simulation as the base for applying the treated control algorithms. Therefore, a correct implementation of the dynamics is required before continuing with the design of the controllers. To verify the correctness of the dynamics implementation, you can use the *Visualization* block provided to you in the library.

Task 2: *Simulate the system from the initial position $\mathbf{q}_i = [-60°, -30°, 20°]$ and no input torque $\tau = [0, 0, 0]$. Plot the joint angles and comment the results.*

Task 3: *Implement the calculation of the Cartesian pose and velocity using the forward kinematics and the Jacobian matrix and plot the TCP position and velocity over time for the same experiment as in the previous task.*

3.2. Joint Control*

For the implementation of the control laws create a "Controller" block which should include all the control algorithms considered. The input of the block is the state of manipulator (joint positions and velocities, forward kinematics, Jacobians, dynamics matrices, etc...) and the outputs are the commanded torques. The closed loop between the Dynamics block and the Controller block constitutes a simulation of the behavior of the controlled manipulator, as shown in Fig. 5(a)



(a) Scheme of simulation of the controlled manipulator (b) Controller split up in various activable subcontrollers.

It's highly suggested to implement all the following controllers as standalone subsystems and to sum up their torques in a unique signal (see Fig. 5(b)), thanks to the superposition principle for the impedances. The subsystems could be individually activated or deactivated during the simulation by means of the use of an "Enable"

input block which can be found in the "Port & Subsystems" Simulink standard library. In this way it's possible to compare the effect of each individual functionality of the controller.
The first step of the implementation is to correctly counterbalance the gravity torques acting on the manipulator.

Task 4: *Implement a gravity compensation using the gravity torques from the online model. Start the simulation again and compare the robot behavior.*

The gravity compensation is easy to implement in theory but is very important in practice. Keep the gravity compensation on throughout the tutorial.

Task 5: *Implement the joint PD controller described in* (2.2) *with a constant damping. Compare different settings for the stiffness and damping matrices* $\mathbf{K}_p$, $\mathbf{K}_d$ *and tune them to achieve a fast and well-damped response. Plot the joint angles and torques with respect to time for some representative cases of* $\mathbf{K}_p$ *and* $\mathbf{K}_d$*.*

Task 6: *Run the simulation again with the gravity compensation and the PD controller, what happens if the gravity compensation is deactivated?*

In order to achieve a desired damping factor ζ, the damping matrix $\mathbf{K}_d$ should be appropriately designed.

Task 7: *Implement the damping design with the method of the square root matrices. Plot and compare the temporal response of the joint angles for three different damping factors* ζ*.*

3.3. Translational Cartesian Impedance Control*

A translational Cartesian controller can be implemented by selecting only the translational part from the Cartesian position vector $\mathbf{x}$ and applying stiffness and damping matrices $\mathbf{K}_p$, $\mathbf{K}_d \in \mathbb{R}^{2\times 2}$.

Task 8: *Implement the translational Cartesian controller derived in the problem 2.3.4 and tune the stiffness and damping matrices. After trying various desired positions, command the position* $[x, y] = [0, 1.5]$*. What happens in this case? Show temporal responses for a reachable and for the unreachable target position.*

Task 9: *Plot the position of the TCP with respect to time for some representative cases of* $\mathbf{K}_p$ *and* $\mathbf{K}_d$ *when setting the desired TCP position to a reachable position.*

3.4. Collision Detection**

Task 10: *Implement the external torque observer derived in Problem 2.4.4. To simulate collisions, you can use the* Wall *block provided to you in the library. Make some representative plots that the observer is correctly detecting the impacts against the wall.*

Task 11: *Add source of noise on the measurements of the joints angles and velocities* $\mathbf{q}$, $\dot{\mathbf{q}}$*. Design the gain* K_I *so as to obtain a fast response of the estimator and reject the noisy components of the measurements.*

Task 12: *Plot the behavior of the observer for different cases of* K_I *and comment the differences.*

3.5. Individual Work

3.5.1. (IK) Inverse Kinematics Control**

A scheme for controlling the TCP position in the Cartesian space was given in 2.5.1 and requires the implementation of an inverse kinematics algorithm. The methods derived in 2.5.1 can be applied in case of small variations of the desired Cartesian position $\delta\mathbf{x}_d$ (e.g. in case of an interpolated trajectory). However, in this work we apply the methods directly to the resolution of a desired position $\mathbf{x}_d$, having in mind that in the real implementation particular attention on the interpolator is needed.

Task 13: *Create a separate Simulink model "invkin.slx" for implementing both inverse kinematics algorithms in a loop fashion, as derived in 2.5.1.6 for the transposed Jacobian approach.*

The convergence of the algorithms can be accelerated using the gain matrix $\mathbf{K}$. However, because of discrete time implementation issues, an upper limit of the gains $\mathbf{K}$ exists. Above this limit, the simulation becomes unstable because of the numerical noise. Find the correct trade-off between convergence rate and stability.

Task 14: *Using the model created in Task 13, calculate the joints position corresponding to the TCP position* $\mathbf{x}_d = [0.93\,m, 0.19\,m, 35°]^T$ *for two initial guesses* $\mathbf{q}_0 = [0°, -15°, 35°]^T$ *and* $\mathbf{q}_0 = [10°, 0°, 45°]^T$*. Make a plot of the error* $\mathbf{e} = \mathbf{x}_d - \mathbf{f}(\mathbf{q})$ *and the joint angles* $\mathbf{q}$ *at each iteration step for both methods and both cases.*

A control in the Cartesian space can now be implemented for the 3DOF simulation model developed in the previous sections. For this purpose, an inverse kinematics block can be used together with the joints controller developed in 3.2, as shown for example in Fig. 5.

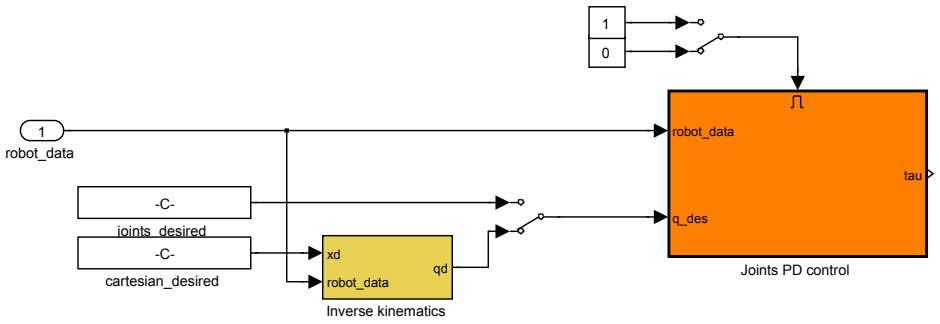


Figure 5: Scheme for commanding a cartesian position using inverse kinematics.

Task 15: *Develop a standalone inverse kinematics block and integrate it into the 3DOF simulation model. Set the initial guess joints position to the current joints position. Make a plot of the joint position and TCP position as a function of time for some representative cases.*

3.5.2. (NO) Null Space Optimizations**

The translational Cartesian controller leaves 1DOF of the manipulator out of control. If the simulation is appropriately realized, you should see the manipulator moving while the TCP remains fixed in the desired position (null space motion). This null space motion can be further damped by means of an appropriately designed null space controller.

Task 16: *Implement a null space damping to not interfere with the TCP motion.*

The additional degree of freedom could also be used for making some null space optimizations. For example, it could be used to avoid singular configurations. For the general case of singularity avoidance, the calculation of the joint torques is computationally intensive since the first derivative (w.r.t. the joint angles) of the potential function is used. In the case of the 3DOF manipulator it can be done manually, since $m_{kin}(q_2)$ is just a function of the second joint angle.

Task 17: *Write an Embedded Matlab Function to implement the results of Problem 2.5.2.*

Task 18: *Make some plots of the Cartesian error and the manipulability index in the proximity of the singularities. Compare the results with the singularity avoidance switched ON and OFF.*

3.5.3. (FCI) Full Cartesian Impedance Controller***

The additional DOF can be used to set up also the TCP orientation. The full Cartesian controller can be developed using a more convenient quaternion representation.

Task 19: *Calculate the rotational error matrix and implement an Embedded Matlab function for transforming it into the quaternion representation.*

Task 20: *Implement the full Cartesian controller derived in Problem 2.5.3.8.*

Task 21: *Implement the damping design for the full Cartesian controller.*

Task 22: *Plot the pose (position* and *orientation) of the TCP with respect to time for three different damping factors.*

4. Acknowledgement

Most of this tutorial has been written by Alessandro Giordano (alessandro.giordano@dlr.de). Part of this tutorial was taken by the material kindly furnished by Florian Petit, Dominic Lakatos of the DLR Institute of Robotics and Mechatronics. We would like to thank also Alexander Dietrich for the help in the revision.

A. Matrix components

In the following, symbolic expressions for the components of the matrices/vectors, describing the kinematic and dynamic model are listed. Therefore a set of parameters is introduced:

l_1 : Length of link 1

l_2 : Length of link 2

l_3 : Length of link 3

m_1 : Mass of link 1

m_2 : Mass of link 2

m_3 : Mass of link 3

l_{c1} : Distance to the center of mass of link 1

l_{c2} : Distance to the center of mass of link 2

l_{c3} : Distance to the center of mass of link 3

I_1 : Moment of inertia of link 1 w.r.t. $\{1\}$

I_2 : Moment of inertia of link 2 w.r.t. $\{2\}$

I_3 : Moment of inertia of link 3 w.r.t. $\{3\}$

g_0 : gravity constant

$\mathbf{e}_g = (e_{g1}, e_{g2}, e_{g3})^T$: components of the unit vector, pointing in the opposite direction of gravity

The representation of the trigonometric functions are abbreviated, e.g., $\mathrm{c}_1 = \cos q_1$, $\mathrm{s}_1 = \sin q_1$, $\mathrm{c}_{12} = \cos(q_1 + q_2)$, $\mathrm{s}_{12} = \sin(q_1 + q_2)$, etc. Hence the forward kinematic function can be written in the compact form

$$\mathbf{f}(\mathbf{q}) = \begin{bmatrix} l_1\mathrm{c}_1 + l_2\mathrm{c}_{12} + l_3\mathrm{c}_{123} \\ l_1\mathrm{s}_1 + l_2\mathrm{s}_{12} + l_3\mathrm{s}_{123} \\ q_1 + q_2 + q_3 \end{bmatrix}. \tag{19}$$

The components of the end-effector Jacobian w.r.t. to $\{0\}$ are

$${}^0\mathbf{J}(\mathbf{q}) = \begin{bmatrix} -l_1\mathrm{s}_1 - l_2\mathrm{s}_{12} - l_3\mathrm{s}_{123} & -l_2\mathrm{s}_{12} - l_3\mathrm{s}_{123} & -l_3\mathrm{s}_{123} \\ l_1\mathrm{c}_1 + l_2\mathrm{c}_{12} + l_3\mathrm{c}_{123} & l_2\mathrm{c}_{12} + l_3\mathrm{c}_{123} & l_3\mathrm{c}_{123} \\ 1 & 1 & 1 \end{bmatrix}. \tag{20}$$

For the description of the mass/Coriolis matrix, the following shortcuts are introduced:

$$\begin{aligned}
\zeta_1 &= I_1 + m_2 l_2^2 \\
\zeta_2 &= I_2 \\
\zeta_3 &= m_2 l_1 l_{c2} \\
\zeta_4 &= I_3 + m_3 \left(l_1^2 + l_2^2\right) \\
\zeta_5 &= m_3 l_1 l_{c3} \\
\zeta_6 &= m_3 l_2 l_{c3} \\
\zeta_7 &= m_3 l_1 l_2 \\
\zeta_8 &= I_3 + m_3 l_2^2 \\
\zeta_9 &= I_3
\end{aligned} \tag{21}$$

Thus the components of the mass matrix can be written in the form

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}, \tag{22}$$

where

$$\begin{aligned}
M_{11} &= \zeta_1 + \zeta_2 + \zeta_4 + 2\left(\zeta_3 + \zeta_7\right)\cos q_2 + 2\zeta_5 \cos(q_2 + q_3) + 2\zeta_6 \cos q_3 \\
M_{12} &= M_{21} = \zeta_2 + \zeta_8 \left(\zeta_3 + \zeta_7\right)\cos q_2 + \zeta_5 \cos(q_2 + q_3) + 2\zeta_6 \cos q_3 \\
M_{13} &= M_{31} = \zeta_9 + \zeta_5 \cos(q_2 + q_3) + \zeta_6 \cos q_3 \\
M_{22} &= \zeta_2 + \zeta_8 + 2\zeta_6 \cos q_3 \\
M_{23} &= M_{32} = \zeta_9 + \zeta_6 \cos q_3 \\
M_{33} &= \zeta_9
\end{aligned}$$

The components of the Coriolis matrix are given by

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} h_5 \dot{q}_3 + h_6 \dot{q}_2 & h_5 \left(\dot{q}_1 + \dot{q}_2\right) + h_6 \dot{q}_3 & h_6 \left(\dot{q}_1 + \dot{q}_2 + \dot{q}_3\right) \\ -h_5 \dot{q}_1 + h_4 \dot{q}_3 & h_4 \dot{q}_3 & h_4 \left(\dot{q}_1 + \dot{q}_2 + \dot{q}_3\right) \\ -h_6 \dot{q}_1 - h_4 \dot{q}_2 & -h_4 \left(\dot{q}_1 + \dot{q}_2\right) & 0 \end{bmatrix}, \tag{23}$$

where

$$\begin{aligned}
h_1 &= -\zeta_3 \sin q_2 \\
h_2 &= -\zeta_7 \sin q_2 \\
h_3 &= -\zeta_5 \sin(q_2 + q_3) \\
h_4 &= -\zeta_8 \sin q_3 \\
h_5 &= h_1 + h_2 + h_3 \\
h_6 &= h_3 + h_4
\end{aligned}$$

In order to describe the components of the gravity vector, the factorization

$$\mathbf{g}(\mathbf{q}) = g_0 \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix} \mathbf{e}_g \tag{24}$$

is performed, where

$$
\begin{aligned}
G_{11} &= -(m_1 l_{c1} + m_2 l_1 + m_3 l_1)\sin q_1 - (m_2 l_{c2} m_3 l_2)\sin(q_1+q_2) \\
&\quad -m_3 l_{c3}\sin(q_1+q_2+q_3) \\
G_{12} &= (m_1 l_{c1} + m_2 l_1 + m_3 l_1)\cos q_1 + (m_2 l_{c2} m_3 l_2)\cos(q_1+q_2) \\
&\quad +m_3 l_{c3}\cos(q_1+q_2+q_3) \\
G_{21} &= -(m_2 l_{c2} + m_3 l_2)\sin(q_1+q_2) - m_3 l_{c3}\sin(q_1+q_2+q_3) \\
G_{22} &= (m_2 l_{c2} + m_3 l_2)\cos(q_1+q_2) + m_3 l_{c3}\cos(q_1+q_2+q_3) \\
G_{31} &= -m_3 l_{c3}\sin(q_1+q_2+q_3) \\
G_{32} &= m_3 l_{c3}\cos(q_1+q_2+q_3) \\
G_{13} &= G_{23} = G_{33} = 0.
\end{aligned}
$$