

Robotics 1 - Summary

In this summary we will recapture the most important results of the lecture Robotics 1. There will be some self-explanatory example tasks. **This summary of the lecture may not be viewed as complete, nor as correct.**

1 Cheat Sheet

Position	s	1 m	Angle	φ	1 rad
Mass	m	1 kg	Inertia	I	1 kg · m ²
Velocity	$v = \dot{s} = \frac{d}{dt}s$		Angular Velocity	$\omega = \dot{\varphi} = \frac{d}{dt}\varphi$	
Acceleration	$a = \ddot{s} = \frac{d}{dt^2}s$		Angular Acceleration	$\dot{\omega} = \ddot{\varphi} = \frac{d}{dt^2}\varphi$	
Eq. of Motion	$s = \frac{1}{2}at^2 + v_0t + s_0$		Eq. of Motion	$\varphi = \frac{1}{2}\dot{\omega}t^2 + \omega_0t + \varphi_0$	
Momentum	$p = mv$		Angular Momentum	$L = I\omega$	
Kinetic Energy	$E_{\text{kin,trl}} = \frac{1}{2}mv^2$		Kinetic Energy	$E_{\text{kin,rot}} = \frac{1}{2}I\omega^2$	
Potential Energy	$E_{\text{pot}} = mgh$				

2 Robot Configurations

Before we can start with anything else, we need to consider that a robot is just a concatenation of masses. Since we do usually not only consider point masses, we need to investigate on rigid bodies and their inertias as well.

2.1 Rigid Bodies

In contrast to point masses, rigid bodies can rotate in space. Hence, we need additional degrees of freedom to describe their states, namely rotations. In the course of the lecture we got in touch with multiple ways to express rotations. Most notably the homogeneous transformation, which also includes translations, and the angle axis representation.

2.1.1 Homogeneous Transformation

There exist rotation matrices that describe rotations around the x-,y-, and z-axis, respectively. These look as follows

$$\mathbf{R}_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{pmatrix} \quad (1)$$

$$\mathbf{R}_y = \begin{pmatrix} \cos \alpha & 0 & \sin \alpha \\ 0 & 1 & 0 \\ -\sin \alpha & 0 & \cos \alpha \end{pmatrix} \quad (2)$$

$$\mathbf{R}_z = \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (3)$$

Rotation matrices do not commute, hence, finding the inverse of a concatenation of rotations is strongly dependent on the order in which they were applied. We therefore need conventions on how to apply rotations. One example are the **Euler angles**. In task 2 of sheet 3, we found that the so called **Gimbal lock** may occur for this formulation of rotations, by means that we effectively lose one degree of freedom. A workaround is the angle axis representation in section 2.1.2. In the section 2.5 on kinematics, we will see that it is helpful to combine rotations and translations within one matrix. This is where **homogeneous transformations** $\mathbf{H}_{x/y/z}$ come into play. They are constructed with an additional translational vector $\mathbf{t} = (t_x, t_y, t_z)^T$ as follows

$$\mathbf{H}_{x/y/z} = \begin{pmatrix} \mathbf{R}_{x/y/z} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \quad (4)$$

If we apply this transformation to a point $(\mathbf{p}, 1) = (p_x, p_y, p_z, 1)^T$ in space, we find

$$\mathbf{H}_{x/y/z} \mathbf{p} = \begin{pmatrix} \mathbf{R}_{x/y/z} \mathbf{p} + \mathbf{t} \\ 1 \end{pmatrix} \quad (5)$$

We therefore can formulate translations and rotations within one matrix multiplication by this new formalism.

2.1.2 Angle Axis Representation

An additional way to describe rotations is the **angle axis representation**. The name itself is self-explanatory. We can define a rotation in space by a normalized rotation axis $\hat{\omega}$ and the angle α we rotate by, which is encoded into the the rotation axis' length $\alpha \hat{\omega} = \omega$, see figure 1. The rotation itself is then expressed in terms of the **Rodrigues' formula**

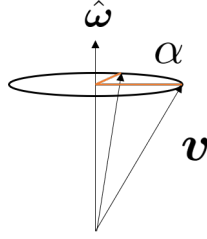


Figure 1: Angle axis representation of rotation.

$$\exp([\omega]_{\times}) = I + \frac{\sin(\alpha)}{\alpha} [\omega]_{\times} + \left(\frac{1 - \cos(\alpha)}{\alpha} \right) [\omega]_{\times}^2$$

$$\text{with } [\omega]_{\times} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \quad (6)$$

2.2 Inertia

We have now learned about rotations, so let's have a look at a rotating system. We will try to understand the inertia of a system, given the following example.

Example 1: Inertia A ball that rotates around the z-axis, shall be displaced by a length l from the axis of rotation, see figure 2. How does the angular velocity $\dot{\varphi}$ change, if the length l changes? What about the system's total energy?

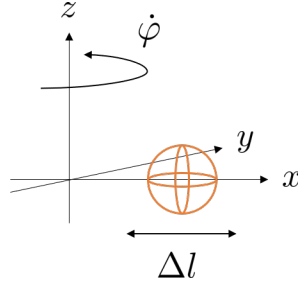


Figure 2: Conservation of angular momentum.

From the **conservation of angular momentum** we know that

$$\begin{aligned} \mathbf{L}_1 &= \mathbf{L}_2 \\ I_1 \dot{\varphi}_1 &= I_2 \dot{\varphi}_2 \end{aligned} \quad (7)$$

So what about the moment of inertia? From **Steiner's theorem** we know

$$\begin{aligned} I_1 &= I_s + m_s l_1^2 \\ I_2 &= I_s + m_s l_2^2 \end{aligned} \quad (8)$$

Of course, the system's **energy is conserved**. But, the potential energy plays an important role here. As we can imagine, a centrifugal force acts upon the rotating ball. Hence, if we try to pull it closer to the center, we need to use energy for that. Namely the integral of the centrifugal force along the path of distance change. This decrease in potential energy is converted into rotational and kinetic energy.

2.3 Concatenation of Rigid Bodies

Now that we have a rough idea about rigid bodies, let's have a look at the concatenation of them, or simply put, a robot. Rigid bodies can be connected in a number of ways by different joints. In the lecture we learned about **revolute**, **prismatic**, **helical**, **cylindrical**, **universal**, and **spherical** joints. They can be defined by their **DOF** and the number of **constraints** they put onto the links they connect, table 1. These concatenation of links with joints form a

Type	DOF	Constraints
Revolute	1	5
Prismatic	1	5
Helical	1	5
Cylindrical	2	4
Universal	2	4
Spherical	3	3

Table 1: Different joint types.

kinematic chain. What we are most interested in is the number of DOF of our kinematic chain. **Grübler's formula** is a suitable tool for this, and we will try to understand it in the following example.

Example 2: Grübler's Formula First of all, the formula is given by

$$\text{DOF} = m(N - 1 - J) + \sum_{i=1}^J f_i, \quad (9)$$

where m is the number of DOF in our space, N the number of links, J the number of joints, f_i the number of DOF provided by joint i . Consider the kinematic chain from figure 3. Recalling that a joint by definition connects 2 links, the revolute joint from figure 3 that connects 3 links needs to be counted twice. Therefore, we have 8 revolute joints, 1 prismatic joint ($J = 9$), and 7 links plus ground ($N = 8$). Every joint has 1 DOF ($f_i = 1 \quad \forall i$). This gives the number of DOF for the chain

$$\text{DOF} = 3(8 - 1 - 9) + \sum_{i=1}^9 1 = 3 \quad (10)$$

This, and further instructive examples can be found in [Modern Robotics](#).



$$\text{dof} = 3(8 - 1 - 9) + 9(1) = 3$$

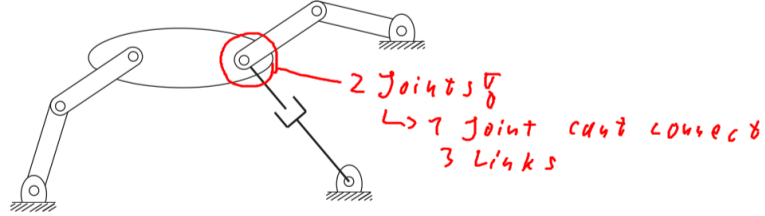


Figure 3: A planar mechanism with two overlapping joints.

2.4 Configuration and Work Space

With Grübler's formula, we found a powerful tool to determine the DOF of a robot. Now we want to try and understand different spaces, a robot can be represented in. Probably the most natural description space is the space we live in, the **work space**, which is the **Cartesian space**, constrained to positions that the robot can reach in at least one configuration with its end effector. The **operational space** describes the same positions as the work space, and further includes all possible orientations to reach those points. But of course there exists the space that may seem most natural to the robot, the **configuration space**, which has at least the dimensionality of the robot's number of DOF. We already know these spaces from task 3 of sheet 2, where we had a closer look at a 2-revolute-link planar robot, figure 4. All these spaces equip us with a tool

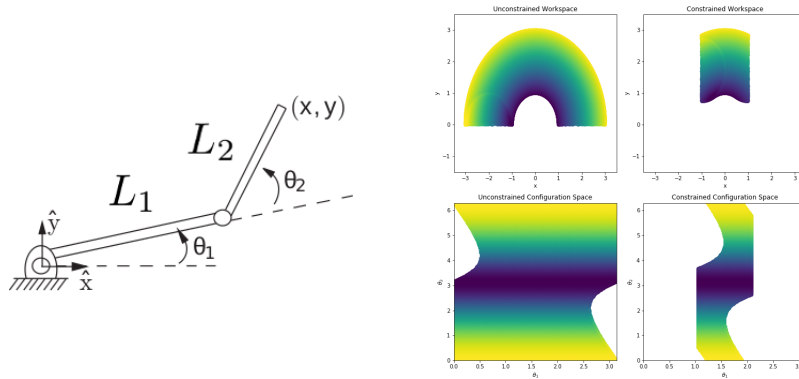


Figure 4: Configuration and workspace of a 2-revolute-link planar robot.

to think about redundancy. A manipulator is **kinematically redundant**, if it has more DOF than the task we want to describe. Further, a manipulator is **intrinsically redundant**, if the dimension of the operational space is smaller than the configuration space. No worries if you don't understand figure 4 yet, you will by the end of section 2.6.

2.5 Kinematics

The urge to find the solution for **kinematics**, now naturally arises from the need to switch between the work space and the configuration space. It turns out that is actually easy to determine the solution for **forward kinematics**, but that analytical solutions for **inverse kinematics** are not always to obtain.

2.5.1 Forward Kinematics

Homogeneous transforms from section 2.1.1 provide us with a powerful tool to determine the forward kinematics of a system. We will try to understand it with the following example.

Example 3: Forward Kinematics Consider the kinematic chain from figure 5, where we want to find the transform from the end effector frame b to frame s . We can find the transformation from the end effector to a reference frame by successively applying homogeneous transforms. Also, often we are not interested in the position of the end effector, but in the position of the so called **tool center point, TCP** for short. The TCP is just the position of the tool we want to use with respect to the end effector.

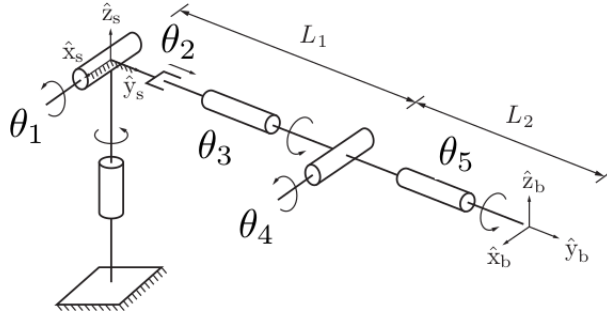


Figure 5: RRPRRR kinematic chain.

$$\mathbf{TCP}_s = \mathbf{H}_{x,1}^s \mathbf{H}_{y,2}^1 \mathbf{H}_{x,3}^2 \mathbf{H}_{y,b}^3 \mathbf{TCP}_b, \quad (11)$$

where we transform from the end effector frame b , to frame 4, to frame 3, to frame 2, to frame 1, to the reference frame s . The different homogeneous

transforms can be found to be

$$\begin{aligned}
H_{y,b}^3 &= \begin{pmatrix} R_y(\theta_5) & (0, L_2, 0)^T \\ \mathbf{0} & 1 \end{pmatrix} \\
H_{x,3}^2 &= \begin{pmatrix} R_x(\theta_4) & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} \\
H_{y,2}^1 &= \begin{pmatrix} R_y(\theta_3) & (0, L_1, 0)^T + (0, \theta_2, 0)^T \\ \mathbf{0} & 1 \end{pmatrix} \\
H_{x,1}^s &= \begin{pmatrix} R_x(\theta_1) & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix}
\end{aligned} \tag{12}$$

So given the configuration space of the kinematic chain $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$, we can now determine the position of the TCP with respect to frame s , the work space of the robot.

2.5.2 Inverse Kinematics

Finding the forward kinematics of a system turned out to be rather simple. Finding an analytical solution of the inverse kinematics on the other hand is not always possible, so we won't consider it. But, we will try to understand how we can use the forward kinematics to iteratively find a solution to the inverse kinematics. And in fact, we have already done this in task 1 of sheet 4. RBDL uses an iterative method to find the inverse kinematics of a problem. Without going into too much detail, here is how. Our goal is to find the joint angles θ , such that the forward kinematics result in our desired end effector position x .

$$x \stackrel{!}{=} \text{FK}(\theta) \tag{13}$$

RBDL uses a Levenberg-Marquardt algorithm, but to get the idea, having a look at a Taylor expansion instead will be sufficient. Now the Taylor expansion of equation 13 around the angle θ_{init} is

$$x \approx \text{FK}(\theta)|_{\theta=\theta_{\text{init}}} + \left. \frac{\partial \text{FK}(\theta)}{\partial \theta} \right|_{\theta=\theta_{\text{init}}} (\theta - \theta_{\text{init}}) + \text{higher orders} \tag{14}$$

If we name $\Delta\theta = \theta - \theta_{\text{init}}$, and recognize that the term $\left. \frac{\partial \text{FK}(\theta)}{\partial \theta} \right|_{\theta=\theta_{\text{init}}}$, is the Jacobian of our system $J(\theta_{\text{init}})$, we can rewrite equation 14

$$\begin{aligned}
J(\theta_{\text{init}})\Delta\theta &= x - \text{FK}(\theta_{\text{init}}) \\
\Delta\theta &= J(\theta_{\text{init}})^{-1}(x - \text{FK}(\theta_{\text{init}}))
\end{aligned} \tag{15}$$

Hence, we can find an update $\Delta\theta$ for θ_{init} , such that eventually our desired end effector position x can be reached by our robot's configuration θ . In contrast to the forward kinematics, the inverse kinematics is **not unique**. But if our desired position x does not change too much between time steps, then we are likely to find a configuration θ that behaves well.

2.6 Tree Representations

In the last lecture we have briefly touched tree representations of the work space and the configuration space. A tree naturally arises from problems where we have to take decisions for each time step. Lets consider that at time t_i our robot is in configuration θ_i . Then, every feasible configuration θ_{i+1} for time step t_{i+1} , creates a new node. Each new node is connected to our initial node over an edge, which represents the cost to reach the respective new node. This process creates a tree. We can search this tree to find the cheapest path by our means. A possible way to search a tree is the **A* search**. The A* search is quite suitable for robots, as we take a **heuristic** into account that we can define for ourselves. A suitable heuristic for our purposes usually is the **Euclidean distance**. The search algorithm will therefore prefer decision that will minimize the Euclidean distance, or put in other words, will get us closer to the final goal. We will try to search a simple tree using the A* algorithm in the following.

Example 4: A* Search For those of you who are familiar with the Dijkstra algorithm, the A* search is simply that with an additional metric. Figure 6 shows a possible path finding problem that can be solved by an A* search, and further the connection to a tree representation. The agent in this problem is in principle allowed to got up/down, as well as left/right. The algorithm itself always takes the next step that minimizes the total cost. It may therefore happen that the algorithm jumps to nodes that are not directly connected to the current one. If the used heuristic satisfies the triangle inequality, the algorithm is guaranteed to find the optimal solution. In the tree representation of figure 6, the nodes show the total cost rather than the cost for a single step. Each step is assumed to have equal cost of 1. Having this said, we can understand figure

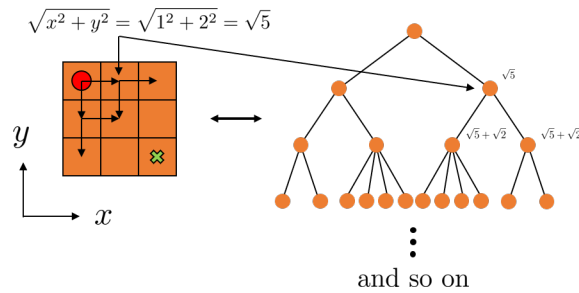


Figure 6: A* search.

4. Here we started with possible angles for the configuration space. To obtain the workspace then, we used forward kinematics and an exhaustive tree search for all possible angles within the configuration space.

3 Equations of Motion

We found that we can derive the equations of motion for several systems using either the **Euler-Lagrange** formalism, or the **Newton-Euler** formalism. Although the Euler-Lagrange method is conceptually very beautiful, the Newton-Euler formalism can be implemented easier for any kinematic chain. This is why RBDL uses the Newton-Euler method rather than the Euler-Lagrange method. Also, the Newton-Euler method can be used recursively, by means of **forward recursion** and **backward recursion**. Forward recursion yields the positions, velocities, and accelerations of successive links, given the generalized coordinates and their derivatives. The backward recursion yields the generalized forces, given the external forces and inertial forces. Together it implements **inverse dynamics**, see section 4. In the following we will try to find equations of motion for the same system using both methods. The system we will consider is the physical pendulum.



Example 5: Euler-Lagrange Method We identify our generalized coordinates as φ , the angle of displacement from the resting position, and its angular velocity $\dot{\varphi}$, see figure 7. For the physical pendulum, the kinetic energy and the

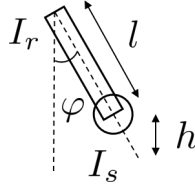


Figure 7: Physical pendulum.

potential energy in terms of the generalized coordinates are given as follows

$$\begin{aligned}
 T &= E_{\text{kin}} + E_{\text{rot}} = \frac{1}{2}(m_r v_r^2 + m_s v_s^2) + \frac{1}{2}I\dot{\varphi}^2 \\
 &= \frac{1}{2} \left[m_r \left(\frac{l}{2}\dot{\varphi} \right)^2 + m_s l^2 \dot{\varphi}^2 \right] + \frac{1}{2}(I_r + I_s)\dot{\varphi}^2 \\
 U &= mgh = (m_r + m_s)g(1 - \cos \varphi) \frac{1}{m_r + m_s} \left(\frac{l}{2}m_r + lm_s \right) \\
 &= g(1 - \cos \varphi) \left(\frac{l}{2}m_r + lm_s \right)
 \end{aligned} \tag{16}$$

The **Lagrangian**, and the **Lagrange equations** for the system can therefore be formulated as

$$\begin{aligned}
\mathcal{L} &= T - U \\
\frac{\partial \mathcal{L}}{\partial \varphi} &= \frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{\varphi}} \\
\frac{\partial \mathcal{L}}{\partial \varphi} &= -g \sin \varphi \left(\frac{l}{2} m_r + l m_s \right) \\
\frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{\varphi}} &= \frac{d}{dt} \left[m_r \left(\frac{l}{2} \right)^2 \dot{\varphi} + m_s l^2 \dot{\varphi} + (I_r + I_s) \dot{\varphi} \right] \\
&= \left[m_r \left(\frac{l}{2} \right)^2 + m_s l^2 \right] \ddot{\varphi} + (I_r + I_s) \ddot{\varphi}
\end{aligned} \tag{17}$$

The resulting equation of motion therefore is

$$\left[m_r \left(\frac{l}{2} \right)^2 + m_s l^2 + I_r + I_s \right] \ddot{\varphi} + g \sin \varphi \left(\frac{l}{2} m_r + l m_s \right) = 0 \tag{18}$$

3.1 Cutting Forces

Cutting forces are introduced to cut down kinematic chains into their single rigid bodies. We will need to introduce a cutting force for every constraint between the two connected bodies. Lets briefly recall table 1. A revolute joint for example introduces 5 constraints to the bodies it connects. Therefore, with only one DOF left in a 3D space, we need to introduce 3 cutting forces, and 2 cutting torques. The 3rd cutting torque that we do not need, is the DOF of the revolute joint. The **single body diagram**, see figure 8, that we obtain from this formalism are perfectly suited for the Newton-Euler method.

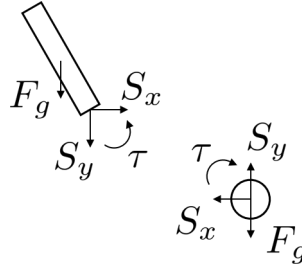


Figure 8: Single body diagram of physical pendulum.

Example 6: Newton-Euler Method We again look at the physical pendulum, but this time we use a single body diagram of the system to obtain the

equation of motion.

$$\begin{aligned} \left(I_r + m_r \left(\frac{l}{2} \right)^2 \right) \ddot{\varphi} + S_x \cos \varphi - S_y \sin \varphi + m_r g \frac{l}{2} \sin \varphi + \tau &= 0 \\ (I_s + m_s l^2) \ddot{\varphi} - S_x \cos \varphi + S_y \sin \varphi + m_s g l \sin \varphi - \tau &= 0 \end{aligned} \quad (19)$$

Solving for the torque τ yields

$$\left[m_r \left(\frac{l}{2} \right)^2 + m_s l^2 + I_r + I_s \right] \ddot{\varphi} + g \sin \varphi \left(\frac{l}{2} m_r + l m_s \right) = 0, \quad (20)$$

which is the exact same solution that we already obtained from the Euler-Lagrange formalism.

3.2 Order Reduction

Since our equations of motion usually do not have an analytical solution, we can use techniques like **order reduction** to transform **ordinary differential equations (ODE's)** of higher order into a system of ODE's of first order, and then use an integrator to solve our **initial value problem**. We introduced order reduction already in task 1 of sheet 2. Although the physical pendulum, that we were looking at, has an analytical solution, let's try to reduce its order so we could solve it numerically.

Example 7: Order Reduction Using the small angle approximation $\sin \varphi \approx \varphi$, equations 18 and 20 become

$$\begin{aligned} \left[m_r \left(\frac{l}{2} \right)^2 + m_s l^2 + I_r + I_s \right] \ddot{\varphi} + g \varphi \left(\frac{l}{2} m_r + l m_s \right) &\approx 0 \\ a \ddot{\varphi} + b \varphi &\approx 0 \end{aligned} \quad (21)$$

This equation, similar to task 3 on sheet 2, has the form of a single pendulum. If we now substitute $z_0 = \varphi$, $z_1 = \dot{\varphi}$, and $z_2 = \ddot{\varphi} = -\frac{b}{a} z_0$, we have transformed our 2nd order ODE into a system of 1st order ODE's. Given initial values for $z_0 = \varphi$ and $z_1 = \dot{\varphi}$, we can now use an integrator to solve the problem.


4 Dynamics

To manipulate the outer world, we are not only interested in the positions of the links, but also in the forces that our links apply onto objects. This is where we need to consider **forward dynamics** and **inverse dynamics**. A possible way to implement the inverse dynamics is by a recursive Newton-Euler algorithm. First, we propagate θ , $\dot{\theta}$, and $\ddot{\theta}$ forwards to obtain x , \dot{x} , and \ddot{x} . Then, we propagate backwards to obtain the torques and forces that act onto our system.

4.1 Jacobian

The **Jacobian** relates the angular velocity of the joints to the velocity of the links in Cartesian space. Kinematic redundancy delivers joint velocities $\dot{\theta}$ that result in an end effector velocity that is zero. This is the so called **Null space**.

Example 8: Derive Jacobian Now lets see how we can derive the Jacobian for a system. We know that the forward kinematics relate the configuration space and the work space. To obtain the velocity of the end effector, we therefore need to determine the time derivative of the forward kinematics. We do this for the example of a 2-revolute-link robot, as shown in figure 4.

$$\begin{aligned}\dot{\mathbf{x}} &= \frac{d}{dt} \text{FK}(\boldsymbol{\theta}) \\ &= \frac{\partial \text{FK}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \frac{d\boldsymbol{\theta}}{dt} \\ &= \mathbf{J} \dot{\boldsymbol{\theta}},\end{aligned}\tag{22}$$


where the forward kinematics is known

$$\mathbf{x} = \text{FK}(\boldsymbol{\theta}) = \begin{pmatrix} L_1 \cos \theta_1 + L_2 \cos(\theta_1 + \theta_2) \\ L_1 \sin \theta_1 + L_2 \sin(\theta_1 + \theta_2) \end{pmatrix},\tag{23}$$

of which the derivative is

$$\mathbf{J} = \frac{\partial \text{FK}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \begin{pmatrix} \frac{\partial x_1}{\partial \theta_1} & \frac{\partial x_1}{\partial \theta_2} \\ \frac{\partial x_2}{\partial \theta_1} & \frac{\partial x_2}{\partial \theta_2} \end{pmatrix} = \begin{pmatrix} -L_1 \sin \theta_1 - L_2 \sin(\theta_1 + \theta_2) & -L_2 \sin(\theta_1 + \theta_2) \\ L_1 \cos \theta_1 + L_2 \cos(\theta_1 + \theta_2) & L_2 \cos(\theta_1 + \theta_2) \end{pmatrix}\tag{24}$$

Further, during the lecture we learned about a more algorithmic approach to determine the Jacobian of a system for the **Denavit-Hartenberg** parametrization. As we have already seen in task 3 of sheet 5, the homogeneous transform for the Denavit-Hartenberg parametrization is given as

$$T_i^{i-1}(\theta_i) = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & L_i \cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) & 0 & L_i \sin(\theta_i) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\tag{25}$$

Then the entries of the Jacobian for revolute joint can be found to be

$$\begin{aligned}J_{P_i} &= \mathbf{z}_{i-1} \times (\mathbf{x}_e - \mathbf{x}_{i-1}) \\ J_{O_i} &= \mathbf{z}_{i-1}\end{aligned}\tag{26}$$

where \mathbf{x}_e is the end-effector position in the world frame, \mathbf{x}_i the origin of the i -th DH frame in world coordinates and \mathbf{z}_i the rotation axis of joint i represented in the world frame. For prismatic joints we have

$$\begin{aligned}J_{P_i} &= \mathbf{z}_{i-1} \\ J_{O_i} &= 0\end{aligned}\tag{27}$$

We therefore have

$$\begin{aligned} \mathbf{x}_1 &= T_1^0 \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} = \begin{pmatrix} L_1 \cos \theta_1 \\ L_1 \sin \theta_1 \\ 0 \end{pmatrix} \\ \mathbf{x}_e &= T_1^0 T_2^1 \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} = \begin{pmatrix} L_1 \cos \theta_1 + L_2 \cos(\theta_1 + \theta_2) \\ L_1 \sin \theta_1 + L_2 \sin(\theta_1 + \theta_2) \\ 0 \end{pmatrix} \end{aligned} \quad (28)$$

Now, with equation 26 we find

$$\begin{aligned} \begin{pmatrix} \dot{\mathbf{x}}_e \\ \dot{\theta}_e \end{pmatrix} &= \begin{pmatrix} J_{P_1} & J_{P_2} \\ J_{O_1} & J_{O_2} \end{pmatrix} \boldsymbol{\theta} = \begin{pmatrix} \mathbf{z}_0 \times (\mathbf{x}_e - \mathbf{x}_0) & \mathbf{z}_1 \times (\mathbf{x}_e - \mathbf{x}_1) \\ \mathbf{z}_0 & \mathbf{z}_1 \end{pmatrix} \boldsymbol{\theta} \\ &= \begin{pmatrix} -L_1 \sin \theta_1 - L_2 \sin(\theta_1 + \theta_2) & -L_2 \sin(\theta_1 + \theta_2) \\ L_2 \cos \theta_1 + L_2 \cos(\theta_1 + \theta_2) & L_2 \cos(\theta_1 + \theta_2) \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{pmatrix} \boldsymbol{\theta}, \end{aligned} \quad (29)$$

which is, if we neglect the dimensions in which the robot can't be moved, the exact same solution that we found before in equation 24.

4.2 Singularities

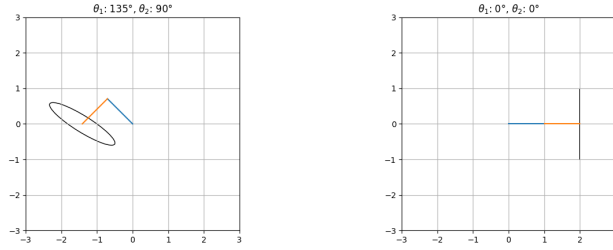
Singularities are robot configurations at which we lose the ability to move along certain directions. The Jacobian loses full rank at these points, such that the Null space increases there. The inverse kinematics may have an infinite number of solutions then. There exist measures that predict how close we are to singularities.

4.3 Manipulability Ellipsoids

Manipulability ellipsoids describe the effort to move the end effector of the robot along certain directions. These manipulability ellipsoids are described by the eigenvectors and eigenvalues λ_i of JJ^T . The eigenvectors determine the orientation of the manipulability ellipsoid. The square root of the eigenvalues describe the semi-axis of the ellipsoids. We have learned that we can define **manipulability measures** that describe how close we are to singularities, and how well the robot can be moved in general. We got to know the condition number

$$\begin{aligned} \mu_1 &= \sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} \\ \mu_2 &= \frac{\lambda_{\max}}{\lambda_{\min}} \\ \mu_3 &= \sqrt{\lambda_{\max} \lambda_{\min}} \end{aligned} \quad (30)$$

μ_1 and μ_2 diverge close to singularities. Having them close to 1 is what we desire. μ_3 describes the volume of the manipulability ellipsoids, and therefore it approaches zero close to a singularity, see figure 9. In general, we want to maximize the volume of our manipulability ellipsoids.



(a) Well behaving configuration.

(b) At a singularity.

Figure 9: Manipulability ellipsoids of a 2-revolute-link planar robot.