# Robotics (IN2067)

Philipp Wulff and Jan Hansen-Palmus

28th April 2022

## Contents

# Chapter 1: Preface

This is an unofficial summary for the lecture *Robotics* (IN2067) at the Technical University of Munich (winter semester 2021/2022).
We aim to cover all the content from the exercises and supplement it with explanations from the book *Introduction to Robotics - Mechanics and Control (3rd ed.)* by John J. Craig. A lot of the content is based on the Stanford lecture *Introduction to Robotics* by professor Oussama Khatib, which you can watch on YouTube. Many of the figures in this summary are direct screenshots of his slides. Note, that the notations are not consistent throughout this summary.

Stanford lectures not covered:

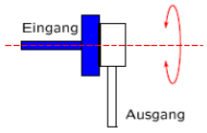
- Lecture 9: Robots and Vision
- Lecture 10: Trajectory Generation

If you find errata within this summary, you are welcome to help fix them by creating an issue or pull request in the GitHub repository for this summary.

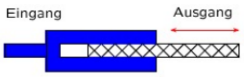
# Chapter 2: Motivation

## 2.1 Types of joints

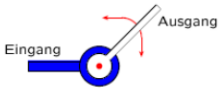
**Joints** usually have motion/position sensors, allowing to measure their relative position to neighboring links.
The **end-effector** is located at the end of the chain of links that make up the manipulator.



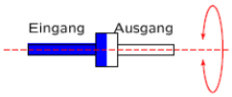

revolving joint




linear joint

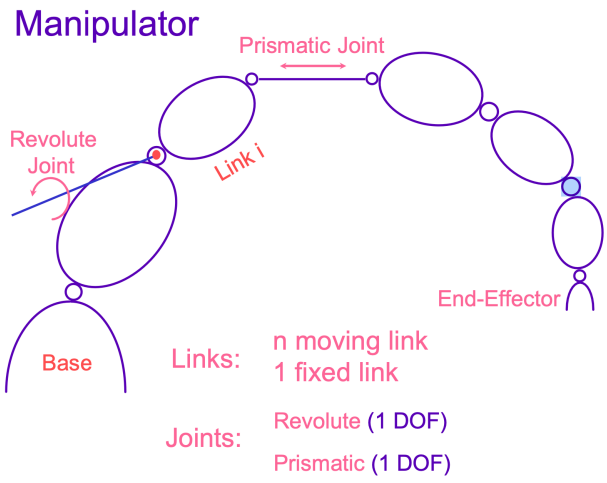



rotational joint




twistingjoint

Different types of joints connect the links of a manipulator.



# Chapter 3: Forward kinematics

## 3.1 Workspace

**Kinematics** is the science of motion that treats motion without regard to the forces which cause it. One studies position, velocity, acceleration, and all higher order derivatives of position variables.

The existence or nonexistence of a kinematic solution defines the **workspace** of a given manipulator. If a solution doesn't exist, this means that the desired position/orientation lies outside of the manipulator's workspace. In other words, the workspace consists of all points that are reachable by the end-effector.

### 3.1.1 Configuration

The **configuration** of a moving object is a specification of the position of **every** point on the object.

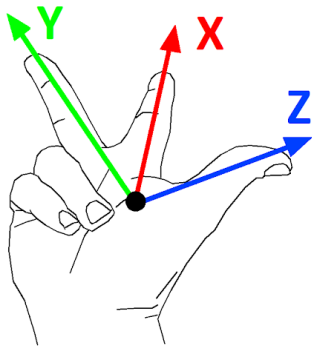
The **dimension of a config space** is the minimum number of parameters needed to specify the configuration of the object completely (also called the number of degrees of freedom of a moving object).

### 3.1.2 Degrees of freedom

The number of **degrees of freedom** that a manipulator possesses is the number of independent position variables that would have to be specified in order to locate all parts of the mechanism. E.g., industrial robotic manipulators often have as many d.o.f. as their number of joints, since each joint has one d.o.f. (and has 5 constraints).

### 3.1.3 Right-hand-rule

The signs of angles are determined by the direction of the fingers, when the thumb is pointing in the axis direction. Fingers also show the order of axis.




## 3.2 Spatial descriptions

We attach a coordinate system to a body and give a description of this coordinate system relative to the reference system. In Figure 2.2, system $B$ has been attached to the body, and its description relative to $A$ is given through its positions and orientation relative to $A$.

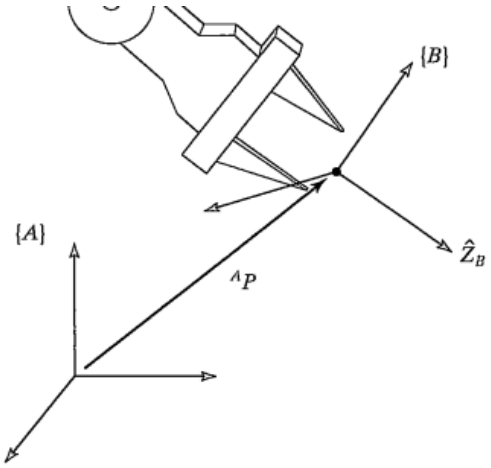



FIGURE 2.2: Locating an object in position and orientation.

### 3.2.1 Position

**Description of a position:** Since many coordinate systems will be used, one has to define to which system a vector refers, e.g.,

$$^A\vec{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix},$$

where $^A\vec{p}$ is a vector referring to coordinate system $A$.

### 3.2.2 Orientation

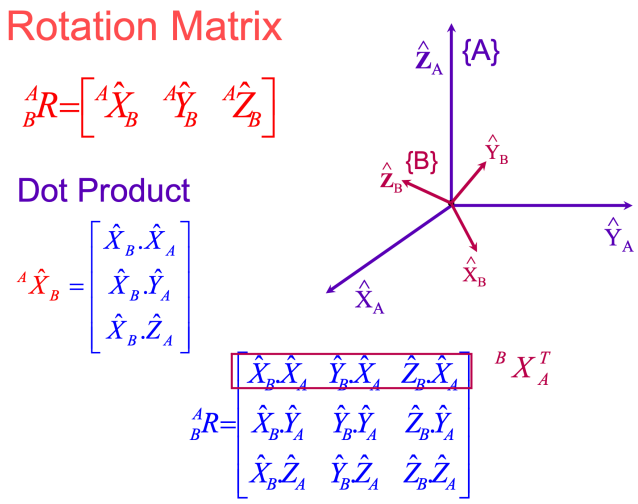
**Description of an orientation:** We stack three unit vectors (they specify the principal directions of the coord system) as columns, yielding the **rotation matrix:**

$$^A_BR = [^A\hat{X}_B \ ^A\hat{Y}_B \ ^A\hat{Z}_B] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = {}^B_AR^{-1} = {}^B_AR^T$$

The rotation matrix has three constraints:

- $|^A\hat{X}_B| = |^A\hat{Y}_B| = |^A\hat{Z}_B| = 1$
- $^A\hat{X}_B \cdot \ ^A\hat{Y}_B = \ ^A\hat{X}_B \cdot \ ^A\hat{Z}_B = \ ^A\hat{X}_B \cdot \ ^A\hat{Z}_B = 0$
- $\det R = 1$

It describes the orientation of frame $B$ relative to frame $A$, i.e. it is used as a *mapping* to **change the description of a vector from frame to frame**. Since we have unit magnitude and the vectors are orthogonal, the transposed matrix describes the orientation of system A written in B. These are orthonormal and length-preserving linear transformations. Also, rotation matrices preserve angles between vectors, i.e. $cos(\angle(\vec{p},\vec{q})) = cos(\angle(R\vec{p},R\vec{q}))$. The projection of a vector $\hat{X}_B$ in coord system $B$ into coord system $A$ is derived from the dot product with the principal directions of the coord frame of $A$, hence the graphic.



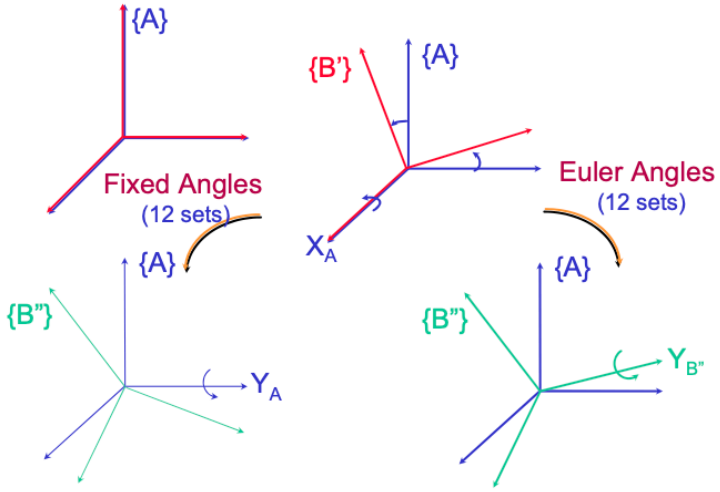

Besides mappings, another use case of rotation matrices is in the form of (rotational) *operators* to move points within the same frame.

### 3.2.3 Euler angles

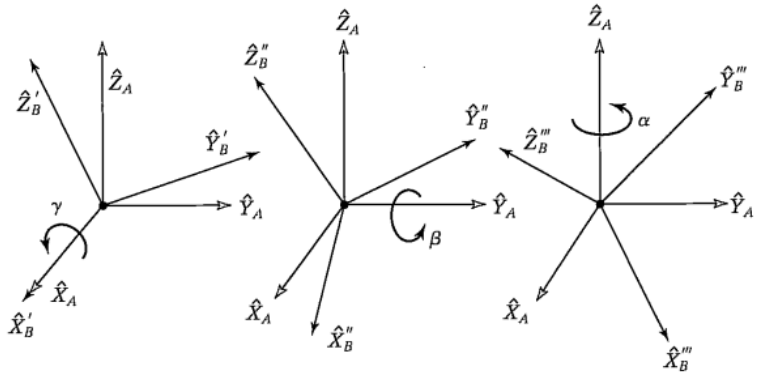
There is an issue when expressing orientations with a rotation matrix: When attempting to follow a trajectory in space by interpolating a current orientation from a start until a final orientation, the intermediary orientations are going to violate the constraints of the rotation matrix. Another possible description of a frame $B$ uses a *three-angle-representation*. It works as follows: Frame $B$ coincides with a known frame $A$. Rotate $B$ first

- about $\hat{X}_A$ by an angle $\gamma$, then about
- $\hat{Y}_A$ by an angle $\beta$, and finally about
- $\hat{Z}_A$ by $\alpha$.

If, each of the three rotations takes place about an axis in the fixed reference frame $A$, we call these angles $X$-$Y$-$Z$ *fixed angles*. An alternative representation uses angles that are relative to the previously changed coordinate frame, so-called **Euler angles**.



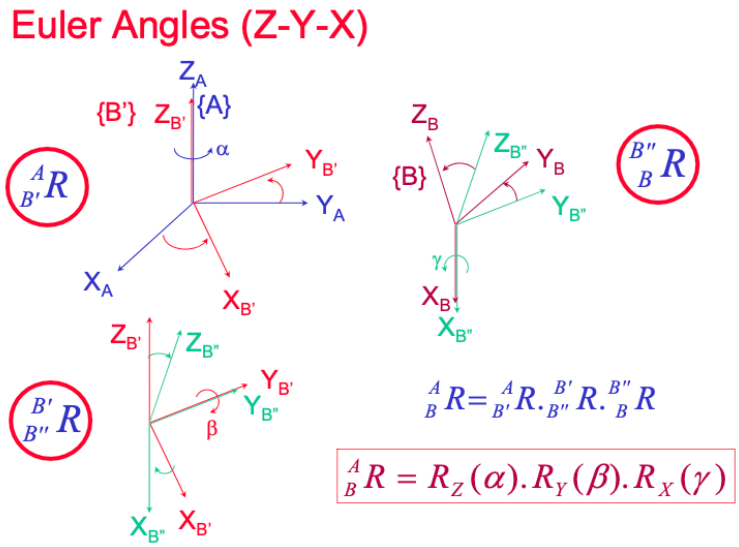
The next figure shows three subsequent rotations of a frame around the fixed axes $\hat{X}_A$, $\hat{Y}_A$ and $\hat{Z}_A$.




The combined rotation matrix is ${}^A_B R = {}^A_{B'} R \cdot {}^{B'}_{B''} R \cdot {}^{B''}_{B} R$, which is cumbersome to compute. Instead, we define:

$$
\begin{aligned}
{}^A_B R_{XYZ}(\gamma, \beta, \alpha) &= R_Z(\alpha) R_Y(\beta) R_X(\gamma) \\
&= \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \\
&= \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix}
\end{aligned}
$$

When rotating a frame with Euler angles in the order of $Z$-$Y$-$X$, they produce the same combined rotation as $X$-$Y$-$Z$ fixed angles, i.e. $R_{Z'Y'X'}(\alpha, \beta, \gamma) = R_{XYZ}(\gamma, \beta, \alpha)$ for the same values of $(\alpha, \beta, \gamma)$. This next graphic shows the rotation using Euler angles:




**Inverse problem:** Given ${}^A_B R$ find the $X$-$Y$-$Z$ fixed angles $(\alpha, \beta, \gamma)$. Let

$$
{}^A_B R_{XYZ}(\gamma, \beta, \alpha) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}
$$

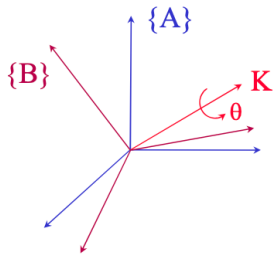
Then:

$$\begin{aligned}\beta &= \operatorname{Atan}2\left(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}\right)\\ \alpha &= \operatorname{Atan}2\left(r_{21}/\cos\beta, r_{11}/\cos\beta\right)\\ \gamma &= \operatorname{Atan}2\left(r_{32}/\cos\beta, r_{33}/\cos\beta\right)\end{aligned} \qquad \operatorname{Atan}2(a,b) = \begin{cases}\arctan\left(\frac{a}{b}\right) & \text{if } b > 0\\ \frac{\pi}{2} & \text{if } b = 0, a > 0\\ \text{undefined} & \text{if } b = 0, a = 0\\ -\frac{\pi}{2} & \text{if } b = 0, a < 0\\ \arctan\left(\frac{a}{b}\right) + \pi & \text{if } b < 0\end{cases}$$

Note that for every parameter $(\alpha, \beta, \gamma)$ there are values which cause a singularity in this computation (e.g. for $\cos\beta = 0$).

### 3.2.4 Unit quaternions (Euler parameters)

Another representation of orientation is by means of four values called the **Euler parameters**. It can be shown that every rotation can be expressed as one rotation of $\theta$ around a single axis $K = [k_x\ k_y\ k_z]^T$. This is also known as the *equivalent angle-axis representation.*




The Euler parameters are given by

$$(\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4) = (k_x \sin\frac{\theta}{2}, k_y \sin\frac{\theta}{2}, k_z \sin\frac{\theta}{2}, \cos\frac{\theta}{2})$$

and

$$\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \epsilon_4^2 = 1$$

This $4 \times 1$ vector is known as a unit quaternion, the orientation it describes could be visualized as a point on a unit hypersphere in four-dimensional space. It turns out that the orientation representation through Euler parameters does not have a singularity.

## 3.3 Spatial transformations

*Problem:* We know the definition of a vector with respect to some frame $B$ and we would like to express it with respect to another frame $A$, where the origins of the two frames are coincident. We can compute this, if we know a description of the orientation of $B$ relative to $A$. Then,

$${}^A P = {}^A_B R\ {}^B P$$

computes what a **vector that is expressed in coordinate system B "looks like" if observed from system A**.

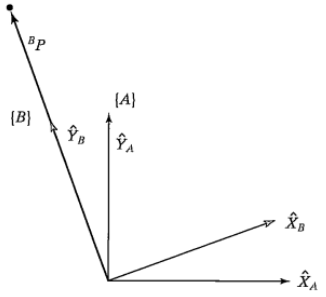



FIGURE 2.6: $\{B\}$ rotated 30 degrees about $\hat{Z}$.

Given

$$ {}^{A}_{B}R = \begin{bmatrix} 0.866 & -0.500 & 0.000 \\ 0.500 & 0.866 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}. $$

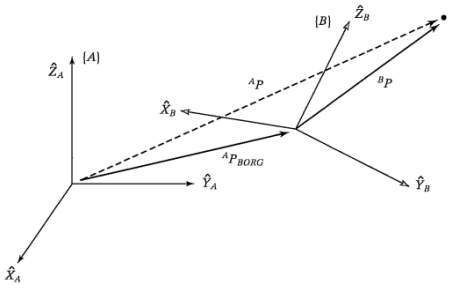
$$ {}^{B}P = \begin{bmatrix} 0.0 \\ 2.0 \\ 0.0 \end{bmatrix}, $$

we calculate ${}^{A}P$ as

$$ {}^{A}P = {}^{A}_{B}R\ {}^{B}P = \begin{bmatrix} -1.000 \\ 1.732 \\ 0.000 \end{bmatrix}. $$

Now, the **general case**, where the systems don't have the same origin, but B is shifted by $P_{BORG}$ from the origin (or ${}^{A}P_{BORG}$ when expressed in $A$):

$$ {}^{A}P = {}^{A}_{B}R\ {}^{B}P + {}^{A}P_{BORG} $$




Computing a position of a point connected by multiple subsequent links of a manipulator with individual coordinate frames, involves the multiplication of sums given by the general transform equation. This is because *rotation* and *translation* are both needed to propagate from one frame to another. To simplify this calculation, a **homogeneous transform** combines rotation and translation into a matrix ${}^{A}_{B}T$, such that: ${}^{A}P = {}^{A}_{B}T\ {}^{B}P$. Therefore, the homogeneous transform is a *general operator*.

$$ {}^{A}P = {}^{A}_{B}R\,{}^{B}P + {}^{A}P_{BORG} $$

$$ \begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} & {}^{A}_{B}R & & {}^{A}P_{Borg} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix} $$

$$ \underset{(4\times1)}{{}^{A}P} = \underset{(4\times4)}{{}^{A}_{B}T}\ \underset{(4\times1)}{{}^{B}P} $$

Due to the generality of the homogeneous transform, it can be interpreted in different ways: 1.) as a description of a frame $\{B\} = \{{}^{A}_{B}R\ \ {}^{A}P_{BORG}\}$; 2.) as a mapping of a point in frame $B$ to frame $A$; 3.) as an operator to rotate and translate a point in the same frame. This is an example for the second case:

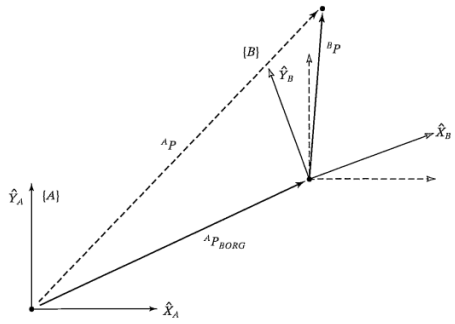



FIGURE 2.8: Frame $\{B\}$ rotated and translated.

Figure 2.8 shows a frame $\{B\}$, which is rotated relative to frame $\{A\}$ about $\hat{Z}$ by 30 degrees, translated 10 units in $\hat{X}_A$, and translated 5 units in $\hat{Y}_A$. Find $^AP$, where $^BP = [3.07.00.0]^T$.

The definition of frame $\{B\}$ is

$$^A_BT = \begin{bmatrix} 0.866 & -0.500 & 0.000 & 10.0 \\ 0.500 & 0.866 & 0.000 & 5.0 \\ 0.000 & 0.000 & 1.000 & 0.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.21)$$

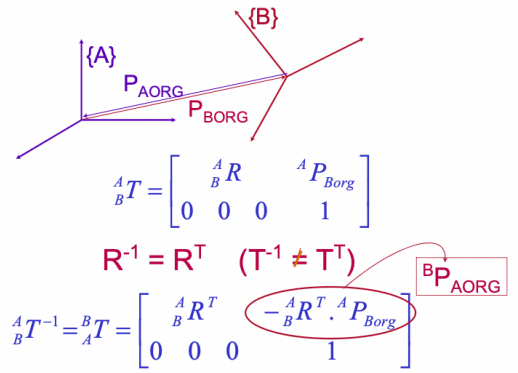
Given

$$^BP = \begin{bmatrix} 3.0 \\ 7.0 \\ 0.0 \end{bmatrix}, \quad (2.22)$$

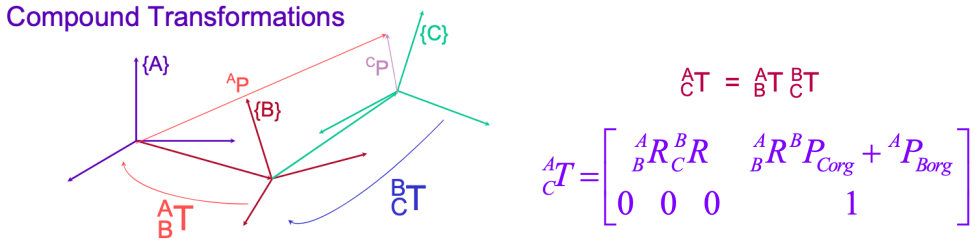
we use the definition of $\{B\}$ just given as a transformation:

$$^AP = {}^A_BT\ {}^BP = \begin{bmatrix} 9.098 \\ 12.562 \\ 0.000 \end{bmatrix}. \quad (2.23)$$

Inverse of the homogeneous transform:




Compound transformation:




Transform equation: $^A_BT\ ^B_CT\ ^C_DT\ ^D_AT = I$.

## 3.4 Forward kinematics

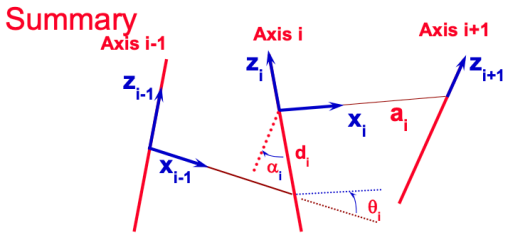
Forward kinematics solves the static geometrical problem of computing the position and the orientation of the end-effector of the manipulator. Specifically, given a **set of joint angles**, the forward kinematic problem is to compute the position and orientation of the tool frame relative to the base frame.

### 3.4.1 Denavit-Hartenberg Convention

The homogeneous transform between consecutive frames of links is represented via only 4 D-H-parameters (x-rot, x-trans, z-trans, z-rot). In general, 6 parameters are needed to represent an arbitrary rigid body transformation! Thus, a restriction of it is, that it **cannot represent a rotation around the $y$-axis and that the $y$- and $z$-position are coupled**.

Procedure for deriving the D-H-parameters:

1. Identify the joint axes and imagine (or draw) infinite lines along them. For steps 2 through 5 below, consider two of these neighboring lines (at axes $i$ and $i+1$).
2. Identify the common perpendicular between them, or point of intersection. At the point of intersection, or at the point where the common perpendicular meets the $i$th axis, assign the link-frame origin.
3. Assign the $\hat{Z}_i$ axis pointing along the $i$th joint axis.
4. Assign the $\hat{X}_i$ axis pointing along the common perpendicular, or, if the axes intersect, assign $\hat{X}_i$ to be normal to the plane containing the two axes.
5. Assign the $\hat{Y}_i$ axis to complete a right-hand coordinate system.
6. Assign $\{0\}$ to match $\{1\}$ when the first joint variable is zero. For $\{N\}$, choose an origin location and $\hat{X}_N$ direction freely, but generally so as to cause as many linkage parameters as possible to become zero.



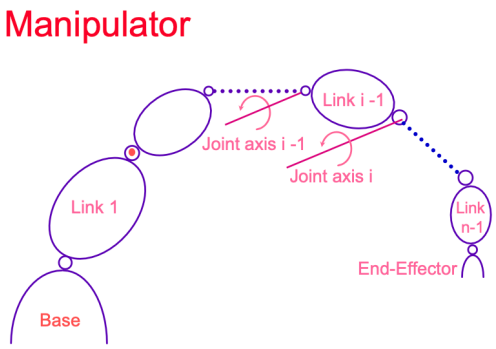

$a_i$ = *the distance from* $\hat{Z}_i$ *to* $\hat{Z}_{i+1}$ *measured along* $\hat{X}_i$;

$\alpha_i$ = *the angle from* $\hat{Z}_i$ *to* $\hat{Z}_{i+1}$ *measured about* $\hat{X}_i$;

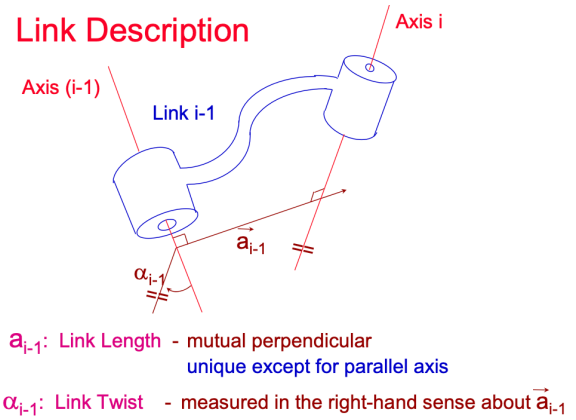
$d_i$ = *the distance from* $\hat{X}_{i-1}$ *to* $\hat{X}_i$ *measured along* $\hat{Z}_i$; and

$\theta_i$ = *the angle from* $\hat{X}_{i-1}$ *to* $\hat{X}_i$ *measured about* $\hat{Z}_i$.

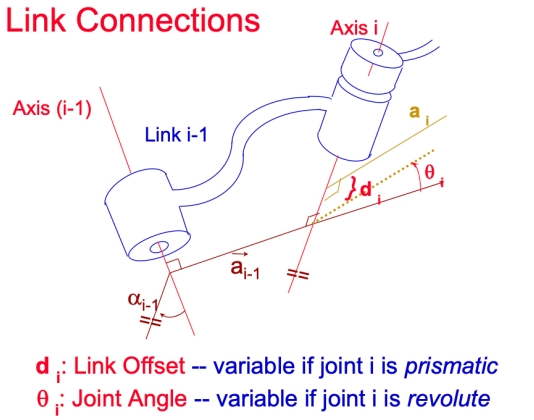
Among the four D-H-parameters are three fixed link parameters and one joint variable, which is $\theta_i$ for a revolute joint or $d_i$ for a prismatic joint. The D-H-parameters are plugged into homogenous transformation matrices, which represent the single transformations between the frames of subsequent links. Then, we can propagate through all homogeneous transformations to calculate the position and orientation of the end-effector. Hereafter, the above described procedure is detailed further:



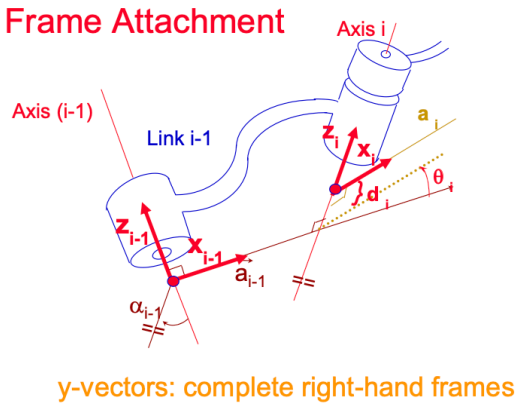

(a) Identify the joint axes; consider axes $i$ and $i-1$. By convention, a joint axis points **in the direction of the rotation/ movement** for revolute/ prismatic joints.



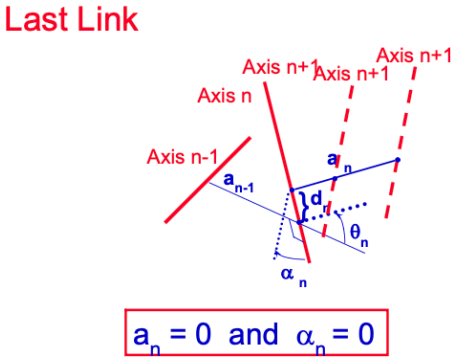

(b) Identify the common perpendicular. If the axes intersect, the common perpendicular is a normal through the plane they span and the direction of $\alpha_i$ is determined by the direction of this normal. $a_i$ **and** $\alpha_i$ **describe the** $i$**th link**. ("We rotate axis $i-1$ around the normal about $\alpha$ so that it coincides with axis $i$. This aligns the $Z$-axes.")



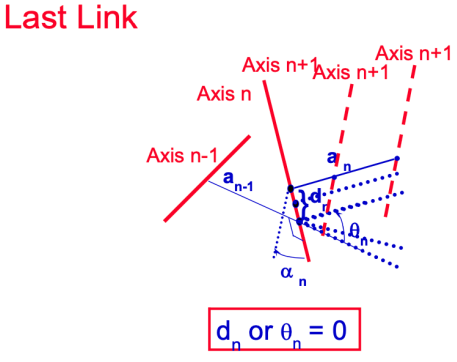

(c) $d_i$ **and** $\theta_i$ **describe the** $i-1$**th link connection**. If the axes of link $i-1$ and link $i$ are parallel, the origin of a coordinate frame $\{i-1\}$ should be attached, such that $d_i = 0$. ("$\theta_i$ aligns the $X$-axes by rotation around $Z_i$.")



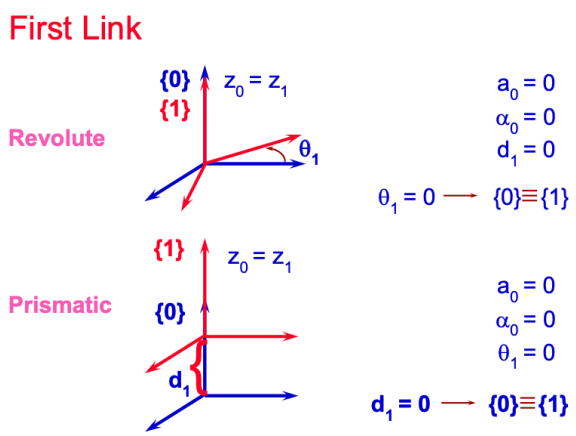

(d) Attach $i$th frame, such that $\hat{Z}_i$ is in the direction of the $i$th joint axis and $\hat{X}_i$ points along the common perpendicular (if $\hat{Z}_{i-1}$ and $\hat{Z}_i$ intersect, choose $\hat{X}_{i-1}$, such that $\alpha_{i-1} > 0$). $\hat{Y}_i$ completes the right-hand frame.



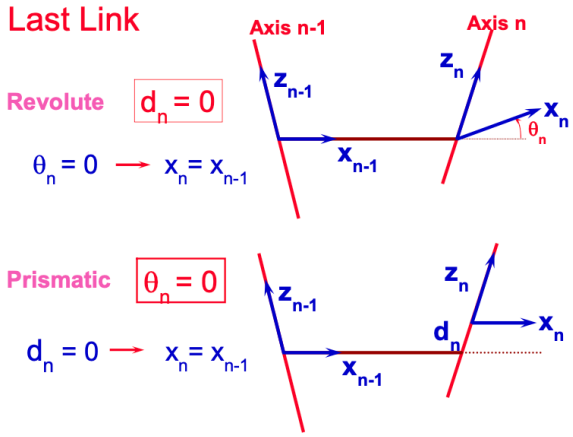

(e) $a_i$ and $\alpha_i$ depend on joint axes $i$ and $i+1$. Thus, select axes $0$ and $n+1$, such that $a_0 = a_n = 0$ and $\alpha_0 = \alpha_n = 0$ (by making axis $0$ coincident with axis $1$ and axis $n+1$ coincident with axis $n$). This simplifies the forward kinematics.




(f) $\theta_i$ and $d_i$ depend on joint axes $i$ and $i-1$. Again, select axes $0$ and $n+1$, such that depending on the joint type $a_0$ or $\theta_0 = 0$ and $a_n$ or $\theta_n = 0$ (by coinciding axes and moving the intersection point that becomes the origin of the frame so that $d = 0$ or orienting the axis so that $\theta = 0$).

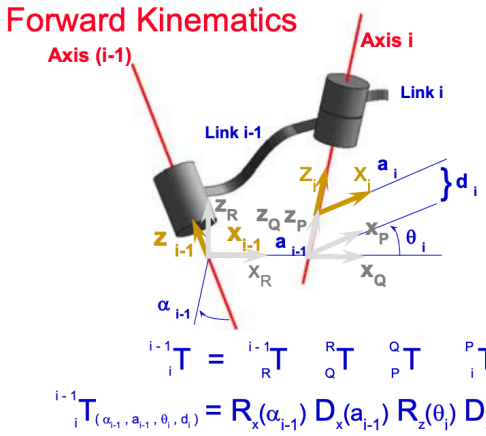



(g) $\{0\}$ is assigned, such that is equals $\{1\}$ when the first joint variable is $0$.

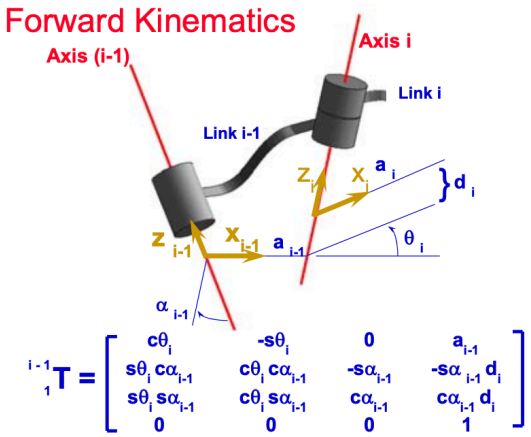



(h) Assign $\{N\}$, such that the most parameters are $0$.

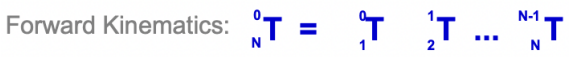
Then, the forward kinematics of a single link using the D-H-parameters derive as follows:




(a) To obtain ${}^{i-1}_{i}T$: Translate along $Z_i$ about $d_i$ ($\rightarrow \{P\}$), then rotate around $Z_i$ about $\theta_i$ ($\rightarrow \{Q\}$), then translate along $X_{i-1}$ about $a_{i-1}$ ($\rightarrow \{R\}$) and finally rotate around $X_i$ about $\alpha_i$ ($\rightarrow \{i-1\}$). This results in the homogeneous transform for a single link from $\{i\}$ to $\{i-1\}$ and also works in the other direction to compute the inverse homogeneous transform.




(b) The **homogeneous transformation from link $i$ to link $i-1$** (the formula in the slide should say $i$ instead of 1).




(c) The homogeneous transformation which transforms points from the frame of link $N$ to the frame of link 0.

Note, that the homogeneous transform ${}^{i-1}_{i}T$ cannot express arbitrary rigid body transformations, since no rotation about $\hat{Y}$ is possible. A good, practical explanation of how to place the coordinate frames at each link, derive the D-H-parameters and compute the homogeneous transformations and forward kinematics is given in this video (based on an example).

# Chapter 4: Inverse Kinematics

Recap forward kinematics: Given a joint configuration, find the pose of some part of the robot. Inverse kinematics: Given a pose, figure out the joint configurations.

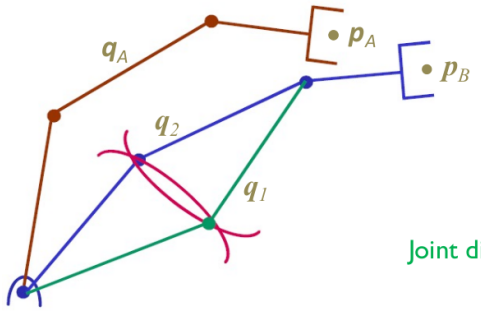
Input data for the problem is of the form:

$$T = \begin{bmatrix} R & t \\ 0\,0\,0 & 1 \end{bmatrix}$$

This is a nonlinear problem, thus it's not sure if there is a (unique/multiple/infinite/outside-of-workspace/none) solution. Inverse kinematics are not explicitly discussed the Stanford lecture and there is not particular exercise on it (albeit it is included in the lecture).

## 4.1 Multiplicity of Solutions

If there are multiple solutions (configurations), choose the closest one:




Initial configuration for $p_A$: $q_A$

New configurations for $p_B$: $q_1$, $q_2$

Is it better to choose $q_1$ or $q_2$? $q_2$

Joint distance between configurations:

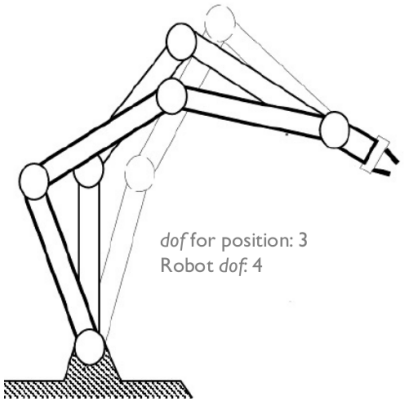
$$d_1 = \| \mathbf{q}_1 - \mathbf{q}_A \|$$
$$d_2 = \| \mathbf{q}_2 - \mathbf{q}_A \|$$

Shortest distance is preferred

In general, if there are $N$ possible configurations for $\mathbf{p}_B$, choose:

$$\mathbf{q}_b = \arg\min_{\mathbf{q}} \| \mathbf{q} - \mathbf{q}_A \| \quad \text{for} \quad \mathbf{q} \in \{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_N\}$$

### 4.1.1 Redundancy




- $n$ joints: $\mathbf{q} = (q_1, \cdots, q_n)$
- Task space: $\mathbf{x} = (x_1, x_2, \cdots, x_m)$

  $m$: dimension of the task space
- Robot is redundant with respect to this task if:

  $$n > m$$

- Example:
  - A 6 *dof* robot is redundant if only position ($m$ = 3) is considered (no orientation)
  - A 6 *dof* robot is not redundant if position and orientation ($m$ = 6) are considered

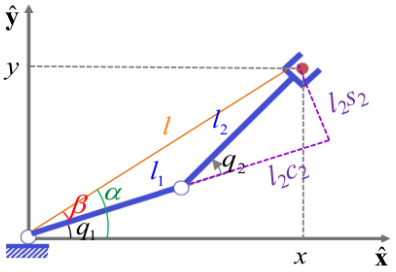
## 4.2 Analytic solutions for the inverse kinematics

### 4.2.1 Geometric solution

- only when robot has 3 or less dofs
- not a generic solution

**Example:**

Find the inverse kinematics for the position of the R-R robot using a geometric approach




**Solution**

For $q_2$:

- Using the law of cosines:

$$l^2 = l_1^2 + l_2^2 - 2l_1 l_2 \cos(180 - q_2)$$
$$x^2 + y^2 = l_1^2 + l_2^2 + 2l_1 l_2 \cos(q_2)$$
$$c_2 = \frac{x^2 + y^2 - (l_1^2 + l_2^2)}{2l_1 l_2}$$

- Using a trigonometric identity:

$$s_2^2 + c_2^2 = 1$$
$$s_2 = \pm\sqrt{1 - c_2^2}$$
$$q_2 = \text{atan2}(s_2, c_2)$$

For $q_1$ (using the geometry of the figure)

$$q_1 = \alpha - \beta$$
$$\alpha = \text{atan2}(y, x)$$
$$\beta = \text{atan2}(l_2 s_2, l_1 + l_2 c_2)$$

Inverse kinematics:

$$q_1 = \text{atan2}(y, x) - \text{atan2}(l_2 s_2, l_1 + l_2 c_2)$$
$$q_2 = \text{atan2}(s_2, c_2)$$

### 4.2.2 Algebraic solution

Solution using algebraic (and polynomial) equations.
Given: Formula to find the kinematic equations of an arm easily, given link params. E.g.:

$${}^B_W T = {}^0_3 T = \begin{bmatrix} c_{123} & -s_{123} & 0.0 & l_1 c_1 + l_2 c_{12} \\ s_{123} & c_{123} & 0.0 & l_1 s_1 + l_2 s_{12} \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This describes the wrist (manipulator) frame relative to the base frame.
We also know what is necessary to describe such a position/orientation:

$${}^B_W T = \begin{bmatrix} c_\phi & -s_\phi & 0.0 & x \\ s_\phi & c_\phi & 0.0 & y \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{\text{i}}$$

So just an x-y positon and an angle $\phi$. By equating both, we get a set of nonlinear equations, which have to be solved for $l_1, l_2, \theta$, using magical algebra:

$$c_\phi = c_{123},$$
$$s_\phi = s_{123},$$
$$x = l_1 c_1 + l_2 c_{12}$$
$$y = l_1 s_1 + l_2 s_{12}$$

### 4.2.3 Example

For a 3-*dof* robot, its end effector pose with respect to its base is given by

$$ {}^{0}\boldsymbol{T}_{3} = \begin{bmatrix} \cos(\theta_1+\theta_3) & 0 & \sin(\theta_1+\theta_3) & a_3\cos(\theta_1+\theta_3)+q_2\sin\theta_1 \\ \sin(\theta_1+\theta_3) & 0 & -\cos(\theta_1+\theta_3) & a_3\sin(\theta_1+\theta_3)-q_2\cos(\theta_1) \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

where $a_3$ and $d_1$ are constants. The desired pose for the end effector is:

$$ T_{des} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Find the inverse kinematics () of this robot as a function of the elements of $T_{des}$.

- Finding :

$$ \begin{cases} p_x = a_3c_{13} + s_1q_2 \longrightarrow p_x - a_3c_{13} = s_1q_2 \\ p_y = a_3s_{13} - c_1q_2 \longrightarrow a_3s_{13} - p_y = c_1q_2 \end{cases} $$

$$ \tan(\theta_1) = \frac{p_x - a_3c_{13}}{a_3s_{13} - p_y} = \frac{p_x - a_3n_x}{a_3n_y - p_y} \Rightarrow \theta_1 = \text{atan2}(p_x - a_3n_x, a_3n_y - p_y) $$

- Finding $\theta_3$:

$$ \frac{a_x}{n_x} = \tan(\theta_1+\theta_3) \longrightarrow \theta_1+\theta_3 = \text{atan2}(a_x, n_x) $$

$$ \theta_3 = \text{atan2}(a_x, n_x) - \theta_1 $$

- Finding $\theta_2$:

$$ \begin{cases} s_1p_x = a_3s_1c_{13} + s_1^2q_2 \\ c_1p_y = a_3s_{13}c_1 - c_1^2q_2 \end{cases} $$

$$ s_1p_x - c_1p_y = a_3(s_1n_x - n_yc_1) + q_2 $$

$$ q_2 = s_1p_x - c_1p_y - a_3(s_1n_x - n_yc_1) $$

## 4.3 Numeric solutions to the inverse kinematics

Mainly used when..

- no analytic solution
- $\infty$ solutions
- too difficult to find a solution

$$ \underbrace{\mathbf{x} = f(\mathbf{q})}_{\text{Forward kinematics}} \Rightarrow \underbrace{\mathbf{x} - f(\mathbf{q}) = 0}_{\text{Find } q \text{ (using iterations) such that the difference is zero}} $$

Uses the Jacobian matrix:

$$ J(q) = \frac{\partial f(q)}{\partial q} \quad J(q) \in \mathcal{R}^{n\times m} $$

Where $n$ is the size of $q$ (number of joints) and $m$ is the size of $x$ (size of task space)

### 4.3.1 Newton's Method

**Problem:** given a (constant) $x_d$, find $q$ such that

$$\mathbf{x}_d - f(\mathbf{q}) = 0$$

$x_d$: desired value for $x$
$f$ : forward kinematics function

**Procedure for the solution**

- First order Taylor approximation for $f(\mathbf{q})$
- Replacing:

$$f(\mathbf{q}) \approx f(\mathbf{q}_k) + J(\mathbf{q}_k)(\mathbf{q} - \mathbf{q}_k) \quad \leftarrow \quad J(\mathbf{q}_k) = \frac{\partial f(\mathbf{q}_k)}{\partial \mathbf{q}}$$

Jacobian matrix

$$\mathbf{x}_d - \left(f(\mathbf{q}_k) + J(\mathbf{q}_k)(\mathbf{q} - \mathbf{q}_k)\right) = 0$$

- Assuming $J$ is square and invertible ($n = m$):

$$\mathbf{x}_d - f(\mathbf{q}_k) = J(\mathbf{q}_k)(\mathbf{q} - \mathbf{q}_k)$$

**Final solution** ($\mathbf{q} = \mathbf{q}_{k+1}$):

$$J^{-1}(\mathbf{q}_k)\left(\mathbf{x}_d - f(\mathbf{q}_k)\right) = (\mathbf{q} - \mathbf{q}_k)$$

$$\boxed{\mathbf{q}_{k+1} = \mathbf{q}_k + J^{-1}(\mathbf{q}_k)\left(\mathbf{x}_d - f(\mathbf{q}_k)\right)}$$

**Algorithm:**

- Start with an initial $\mathbf{q}_0$ (usually the current configuration)
- Iteratively update using:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + J^{-1}(\mathbf{q}_k)\left(\mathbf{x}_d - f(\mathbf{q}_k)\right) \qquad k = 0,1,2,3,\ldots$$

- Stop when:

$$\underbrace{\| \mathbf{x}_d - f(\mathbf{q}_k) \| < \varepsilon}_{\text{Small Cartesian error}} \quad \text{or} \quad \underbrace{\| \mathbf{q}_{k+1} - \mathbf{q}_k \| < \varepsilon}_{\text{Small joint increment}} \qquad \varepsilon\text{: small value}$$

**Comments:**

- Convergence if we start with $\mathbf{q}_0$ (initial value) close to the solution
- When there is redundancy ($m < n$):

  $J$ is not square (and there is no inverse)! → we use the pseudo-inverse

- Disadvantages:
  - Computation time of the inverse (or pseudo-inverse)
  - Problems near singularities of $J$ (Jacobian matrix)
- Advantage: it is fast (quadratic convergence)

Example in slides.

### 4.3.2 Gradient Descent Method

Forward kinematics: $\mathbf{x}_d = f(\mathbf{q})$ or equivalently $\underbrace{\mathbf{x}_d - f(\mathbf{q})}_{\text{error}} = 0$

Procedure:

- Define a scalar error function: $g(\mathbf{q}) = \frac{1}{2}\| \mathbf{x}_d - f(\mathbf{q})\|^2 \longleftarrow g: \mathbb{R}^n \to \mathbb{R}$
- *Objective:* minimize the error: $\min_{\mathbf{q}} g(\mathbf{q})$
- Compute the gradient of $g(\mathbf{q})$:

$$g(\mathbf{q}) = \frac{1}{2}\left(\mathbf{x}_d - f(\mathbf{q})\right)^T \left(\mathbf{x}_d - f(\mathbf{q})\right) \quad \Rightarrow \quad \nabla g(\mathbf{q}) = -\underbrace{\left(\frac{\partial f(\mathbf{q})}{\partial \mathbf{q}}\right)^T}_{J(\mathbf{q})} \left(\mathbf{x}_d - f(\mathbf{q})\right)$$

- Apply gradient descent

$$\mathbf{q}_{k+1} = \mathbf{q}_k - \alpha \nabla g(\mathbf{q}_k)$$

$$\boxed{\mathbf{q}_{k+1} = \mathbf{q}_k + \alpha J^T(\mathbf{q}_k)\left(\mathbf{x}_d - f(\mathbf{q}_k)\right)}$$

Pros: computationally simpler (transpose instead of inverse)

Cons:

Example in slides.

## 4.4 Numeric computation of the Jacobian

For complex robots it is tedious to compute the Jacobian numerically.
**Solution:** Numeric differentiation

Approximation of the derivative of the position $x_p$ with respect to joint $q_i$

$$\frac{\partial \mathbf{x}_p}{\partial q_i} \approx \frac{\Delta \mathbf{x}_p}{\Delta q_i} = \frac{\mathbf{f}(q_1, \cdots, q_i + \Delta q_i, \cdots, q_n) - \mathbf{f}(q_1, \cdots, q_n)}{\Delta q_i}$$

Increment only for joint $q_i$

### Example

Consider the following forward kinematics (R-R Robot with $l_1 = l_2 = 1$):

$$f(\mathbf{q}) = \begin{bmatrix} f_x(\mathbf{q}) \\ f_y(\mathbf{q}) \end{bmatrix} = \begin{bmatrix} \cos(q_1) + \cos(q_1 + q_2) \\ \text{sen}(q_1) + \text{sen}(q_1 + q_2) \end{bmatrix}$$

Compute the Jacobian numerically when the joint values are $q_1 = 0.5, q_2$

### Solution

- The Jacobian is:

$$J(\mathbf{q}) = \begin{bmatrix} \frac{\partial f_x}{\partial q_1} & \frac{\partial f_x}{\partial q_2} \\ \frac{\partial f_y}{\partial q_1} & \frac{\partial f_y}{\partial q_2} \end{bmatrix}$$

- Derivatives with respect to $q_1$ (first column):

$$\frac{\partial f_x}{\partial q_1} = \frac{f_x(q_1 + \Delta q_1, q_2) - f_x(q_1, q_2)}{\Delta q_1} = \frac{\left[\cos(q_1 + \Delta q_1) + \cos((q_1 + \Delta q_1) + q_2)\right] - \left(\cos(q_1) + \cos(q_1 + q_2)\right)}{\Delta q_1}$$

$$\frac{\partial f_y}{\partial q_1} = \frac{f_y(q_1 + \Delta q_1, q_2) - f_y(q_1, q_2)}{\Delta q_1} = \frac{\left[\text{sen}(q_1 + \Delta q_1) + \text{sen}((q_1 + \Delta q_1) + q_2)\right] - \left(\text{sen}(q_1) + \text{sen}(q_1 + q_2)\right)}{\Delta q_1}$$

Derivatives with respect to $q_2$ (second column):

$$\frac{\partial f_x}{\partial q_2} = \frac{f_x(q_1, q_2 + \Delta q_2) - f_x(q_1, q_2)}{\Delta q_2} = \frac{\left[\cos(q_1) + \cos(q_1 + (q_2 + \Delta q_2))\right] - \left(\cos(q_1) + \cos(q_1 + q_2)\right)}{\Delta q_2}$$

$$\frac{\partial f_y}{\partial q_2} = \frac{f_y(q_1, q_2 + \Delta q_2) - f_y(q_1, q_2)}{\Delta q_2} = \frac{\left[\text{sen}(q_1) + \text{sen}(q_1 + (q_2 + \Delta q_2))\right] - \left(\text{sen}(q_1) + \text{sen}(q_1 + q_2)\right)}{\Delta q_1}$$

- Using $\Delta q_1 = \Delta q_2 = 0.001$:

$$\frac{\partial f_x}{\partial q_1} = \frac{\left(\cos(0.5+0.001)+\cos(0.5+0.001+1.0)\right)-\left(\cos(0.5)+\cos(0.5+1.0)\right)}{0.001} = -1.4774$$

$$\frac{\partial f_y}{\partial q_1} = \frac{\left(\text{sen}(0.5+0.001)+\text{sen}(0.5+0.001+1.0)\right)-\left(\text{sen}(0.5)+\text{sen}(0.5+1.0)\right)}{0.001} = 0.9476$$

$$\frac{\partial f_x}{\partial q_2} = \frac{\left(\cos(0.5)+\cos(0.5+1.0+0.001)\right)-\left(\cos(0.5)+\cos(0.5+1.0)\right)}{0.001} = -0.9975$$

$$\frac{\partial f_y}{\partial q_2} = \frac{\left(\text{sen}(0.5)+\text{sen}(0.5+1.0+0.001)\right)-\left(\text{sen}(0.5)+\text{sen}(0.5+1.0)\right)}{0.001} = 0.0702$$

- Jacobian using the numeric computation:

$$J(\mathbf{q}) = \begin{bmatrix} -1.4774 & -0.9975 \\ 0.9476 & 0.0702 \end{bmatrix}$$

Jacobian using the analytical approach (for comparison)

$$J(\mathbf{q}) = \begin{bmatrix} -1.4769 & -0.9975 \\ 0.9483 & 0.0707 \end{bmatrix}$$

Similar values (within rounding errors)

# Chapter 5: Jacobi Matrix ("Jacobian")

The Jacobian is the generalization of the differentiation to multi-dimensional functions, and contains all partial derivatives of function $f$'s components, namely $f_1, f_2, ..., f_m$ w.r.t its $n$ parameters:

$$J_f = \frac{\partial f}{\partial x} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

Example:

$$f : \mathbb{R}^3 \to \mathbb{R}^2, f(x,y,z) = \begin{pmatrix} x^2 + y^2 + z \cdot \sin(x) \\ z^2 + z \cdot \sin(y) \end{pmatrix} \quad J_f(x,y,z) = \begin{pmatrix} 2x + z \cdot \cos(x) & 2y & \sin(x) \\ 0 & z \cdot \cos(y) & 2z + \sin(y) \end{pmatrix}$$

## 5.1 Applications in Mathematics

### 5.1.1 Taylor's theorem

One important usage of the Jacobian is the approximation of a function f in a sufficiently small neighborhood about a point $x \in \mathbb{R}^n$, known as Taylor's theorem:

$$f(x + \delta x) \approx f(x) + \frac{\partial f}{\partial x} \cdot \delta x = f(x) + \mathbf{J}_f(x) \cdot \delta x$$

### 5.1.2 Multi-dimensional chain rule

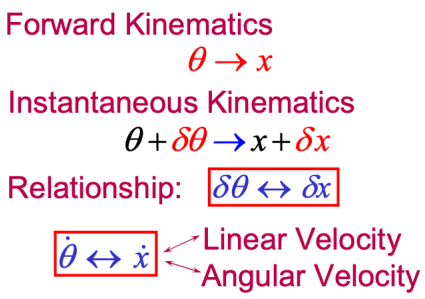
$$(f \circ g)'(t) = f'(g(t))g'(t)$$

The scalar derivatives are being replaced with Jacobi matrices:

$$\mathbf{J}_\mathbf{a}(f \circ g) = \mathbf{J}_{g(\mathbf{a})}(f)\mathbf{J}_\mathbf{a}(g), \quad \mathbf{a} = (x,y,z)^T$$

## 5.2 Jacobian as the Derivative of the Position Representation (Differential motion)

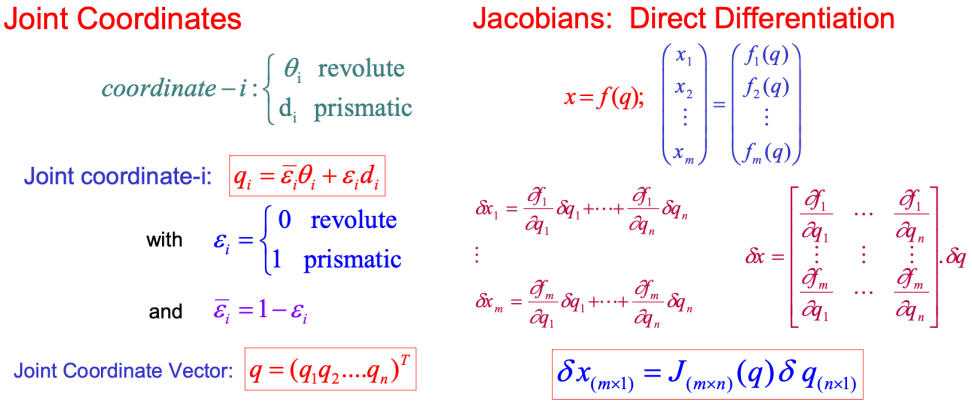
We're often not only interested in position and orientation themselves, but also how they are affected by changes in robot parameters $\delta\theta$. These changes can be either small changes (leading to approximation via Taylor's theorem), or velocities (specified as derivatives of a joint trajectory).



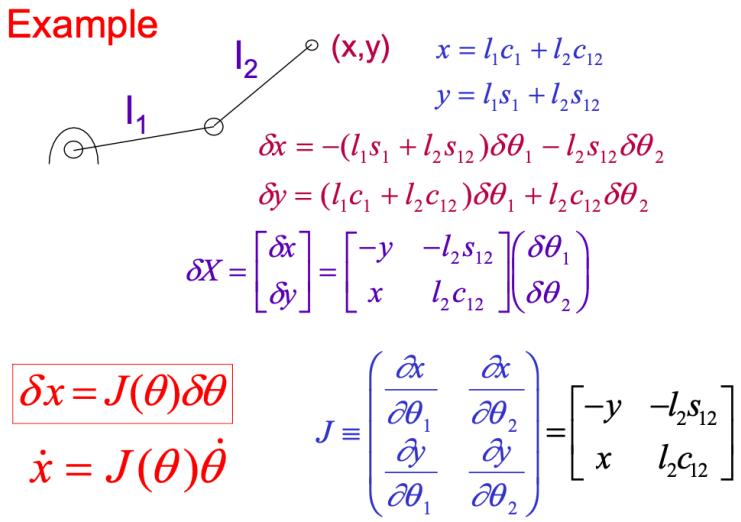

We assume that we are given a position description of a robot coordinate system in form of a function $f$ (params: joint positions, mapped to 6-dim position and orientation representation). Because this representation could use quaternions, rotation matrices or variants of Euler angle conventions, this leads to different Jacobians.

$$f : \mathbb{R}^n \to \mathbb{R}^6, (x_1, x_2, ..., x_n \mapsto (y_1, y_2, ..., y_6))$$

We define joint coordinates $q_i$, which combine revolute and prismatic joints. Then, the Jacobian is obtained from the derivative of the position representation w.r.t. the joint coordinate vector $\frac{\partial f(q)}{\partial q}$.




An example from the Stanford Lecture:




**Note that in the TUM lecture material, the joint coordinates are denoted as** $\Theta$. Since the joint angles of a robot must vary over time to cause a motion, the joint positions $\Theta_1, \Theta_2, ..., \Theta_n$ depend on time $t$. However, we usually omit the $t$ in $\Theta_i(t)$. All in all:

$$\Theta(t) : \mathbb{R} \to \mathbb{R}^n, t \mapsto (\Theta_1(t), \Theta_2(t), ..., \Theta_n(t))^T$$

Then, the position representation at time $t$ is written as follows:

$$f(\Theta(t)), \qquad \mathbb{R} \to \mathbb{R}^6$$

### 5.2.1 Craig's derivation of rotational velocities

He uses a rotation-vector $\boldsymbol{v} \in \mathbb{R}^3$ that represents the axis of rotation through its direction and the angle of rotation through its length $|\boldsymbol{v}|$. The $\mathbf{0}$ vector represents the identity.

Differentiating rotation vectors $\boldsymbol{v}$ w.r.t joint angles leads exactly to the angular velocity vectors $\boldsymbol{\omega}$ that are used in his book.

## 5.3 Jacobians and Velocities

The Jacobian has the nice property of defining a relation between linear and angular velocities in joint space and velocities in cartesian space (for example of the end-effector). Here, we are interested in derivative of the position representation $f$ w.r.t. time.

In analogy to the position representation with small changes in the link coordinates, the same Jacobian connects the derivative of the position representation w.r.t. time and the angular and linear velocities of the joint coordinates:

$$\dot{x}_{(m\times 1)} = J_{(m\times n)}(q)\ \dot{q}_{(n\times 1)}$$

where

$$J_{ij}(q) = \frac{\partial}{\partial q_j} f_i(q)$$

The proof written in the notation of the TUM lecture (the first line applies the chain rule):

$$\begin{aligned}\dot{f} = \frac{\partial f}{\partial t} &= \frac{\partial f}{\partial (x_1, ..., x_n)} \frac{\partial \Theta}{\partial t} \\ &= \frac{\partial f}{\partial (x_1, ..., x_n)} \left( \frac{\partial \Theta_1}{\partial t}, \frac{\partial \Theta_2}{\partial t}, ..., \frac{\partial \Theta_n}{\partial t} \right)^T \\ &= J \cdot \dot{\Theta}\end{aligned}$$

The individual components of $f$ are the derivative the the position $\dot{x}_P$ and of the orientation $\dot{x}_R$ and are related to the joint coordinates by individual Jacobians:

$$\begin{bmatrix} \dot{x}_P \\ \dot{x}_R \end{bmatrix}_{6\times 1} = \begin{bmatrix} J_{x_P} \\ J_{x_R} \end{bmatrix}_{6\times n} \begin{bmatrix} \dot{\Theta}_1 \\ ... \\ \dot{\Theta}_n \end{bmatrix}_{n\times 1} = J_{6\times n} \begin{bmatrix} \dot{\Theta}_1 \\ ... \\ \dot{\Theta}_n \end{bmatrix}_{n\times 1}$$

Keep in mind that the Jacobian depends on the following circumstances:

- How are positions/orientations represented? A Jacobian that is based on cartesian coordinates will look different than one based on Euler angles. This following images shows the possible representations of position and orientation:

$$X = \begin{bmatrix} x_P \\ x_R \end{bmatrix}$$

•Cartesian
•Spherical
•Cylindrical
•....

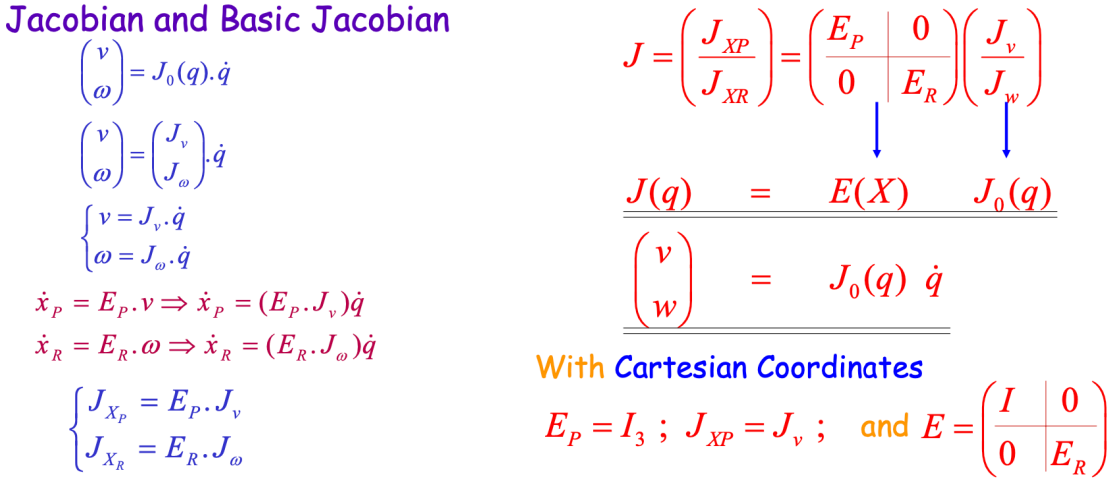
•Euler Angles
•Direction Cosines
•Euler Parameters

- What is the reference coordinate system for the Jacobian?
- How is the current joint configuration?

*Note that for this reason, the Stanford Lecture distinguishes between a Jacobian $J_x$ for the particular representation used in $X$ and a basic Jacobian $J_0$.* These two matrices are related by a matrix $E$ that relates the linear velocity to the position representation and the angular velocity to the orientation representation.

$$\begin{aligned}\dot{x}_P &= E_P(x_P)\, v \\ \dot{x}_R &= E_R(x_R)\, \omega\end{aligned}$$

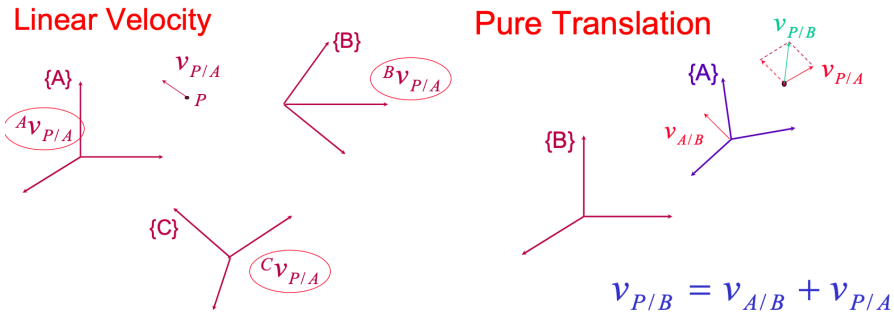
Herein, $\dot{x}_P$ and $\dot{x}_R$ are the derivatives of $f$ w.r.t. time and and $v$ and $\omega$ are the linear and angular velocity (these do not match $\dot{x}_P$ and $\dot{x}_R$ for certain representations!). The following figure shows how the Jacobian and the basic Jacobian are related:



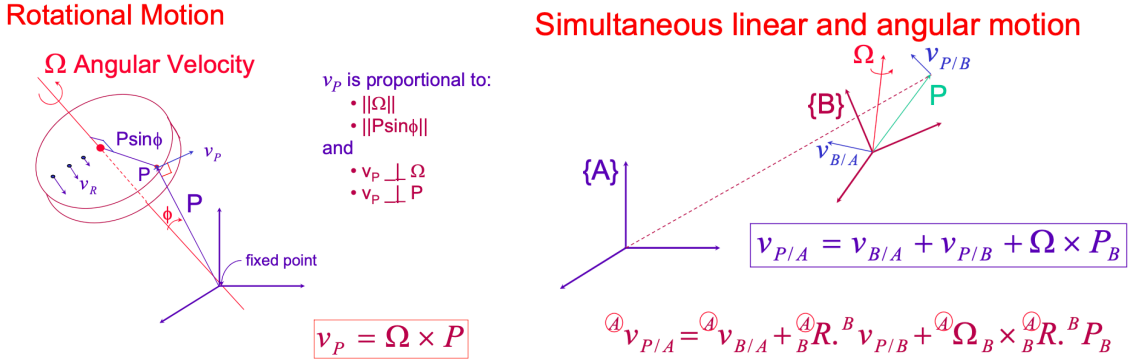

Usually, we want to know $v$ and $\omega$ and derive them from the third equation on the right side of the figure above.

## 5.4 Velocities

Linear velocities are always measured in a specific coordinate frame, e.g. the velocity of point $p$ measured in frame $\{A\}$ w.r.t. frame $\{B\}$:



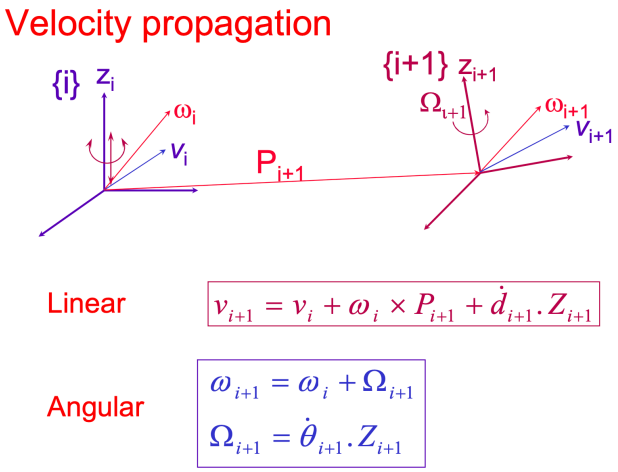

The linear velocity from rotational motion derives from the cross product of the angular velocity with the points vector, as seen in the left figure below. When combining linear and angular velocities, one must make sure to express them w.r.t. the correct coordinate frame, hence the right figure.



With these concepts, the angular and linear velocities in robot joints are propagated from the base frame to the end-effector frame.

$$^{i+1}v_{i+1} = {}^{i+1}_{i}R(^{i}v_i + {}^{i}\omega_i \times {}^{i}P_{i+1}) + \dot{d}_{i+1} \cdot {}^{i+1}\hat{Z}_{i+1}$$
$$^{i+1}\omega_{i+1} = {}^{i+1}_{i}R \cdot {}^{i}\omega_i + \dot{\Theta}_{i+1} \cdot {}^{i+1}\hat{Z}_{i+1}$$

$\dot{d}_{i+1}$ or $\dot{\Theta}_{i+1}$ for prismatic/ revolute joints (scalars).




The linear velocity w.r.t. frame $i+1$ is the sum of the linear velocity of frame $i$, the cross product of the angular velocity of frame $i$ with the distance between the two frames and the joint velocity along $\hat{Z}_{i+1}$ if $i+1$ is a prismatic joint (otherwise, $\dot{d}_{i+1} = 0$). ${}^{i+1}_{i}R$ maps the vectors from from $i$ to frame $i+1$.
The angular velocity w.r.t. frame $i+1$ is composed of the angular velocity of of the previous frame plus the angular velocity around $\hat{Z}_{i+1}$ if $i+1$ is a revolute joint.

The expressions for the linear and angular velocities at a given frame $n$ can be used to determine the full Jacobian indirectly:

$$^{n}J\dot{\Theta} = \begin{pmatrix} {}^{n}v_n \\ {}^{n}\omega_n \end{pmatrix}$$

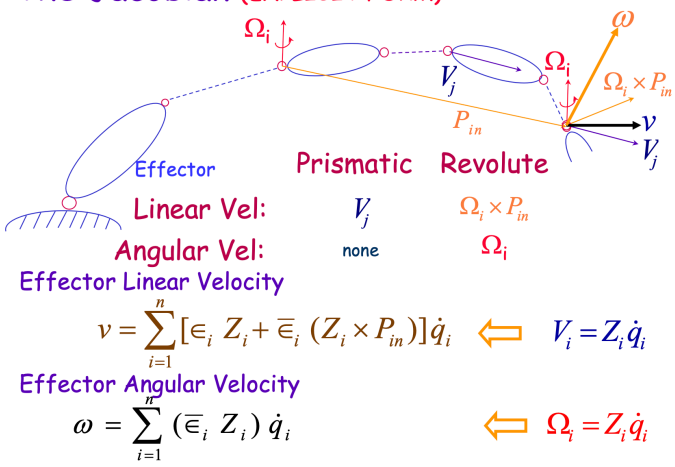
where the shape of the Jacobian on the left hand side can be determined from the shape of the linear-angular velocity vector on the right hand side (by factoring out the derivatives of the joint coordinates). Herein, the expressions for ${}^{n}v_n$ and ${}^{n}\omega_n$ are obtained from the two equations above.

An example for this is found in Lecture 6 of the Stanford course.

## 5.5 Explicit Form

This section details how one can easily *see* the Jacobian matrix in a manipulator. The next figure shows a manipulator with prismatic and revolute joints:

## The Jacobian (EXPLICIT FORM)




1. The impact of each joint type on the end-effector is given in the table.
2. The linear/ angular velocity of prismatic/ revolute joints is a function of the joints' $Z_i$ axes and joint coordinates $q_i$ ($\dot{d}$ or $\dot{\theta}$).
3. The total velocities at the end-effector are the sum over the components from all joints.

In order to derive the Jacobian, we need to factor out the $q$s:

$$v = [\epsilon_1 Z_1 + \bar{\epsilon}_1 (Z_1 \times P_{1n})]\dot{q}_1 + \cdots + [\epsilon_{n-1} Z_{n-1} + \bar{\epsilon}_{n-1} (Z_{n-1} \times P_{(n-1)n})]\dot{q}_{n-1} + \epsilon_n Z_n \dot{q}_n$$

$$v = \begin{bmatrix} \epsilon_1 Z_1 + \bar{\epsilon}_1 (Z_1 \times P_{1n}) & \epsilon_2 Z_2 + \bar{\epsilon}_2 (Z_2 \times P_{2n}) & \cdots \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{bmatrix}$$

$$\boxed{v = J_v \dot{q}}$$

$$\omega = \bar{\epsilon}_1 Z_1 \dot{q}_1 + \bar{\epsilon}_2 Z_2 \dot{q}_2 + \cdots + \bar{\epsilon}_n Z_n \dot{q}_n$$

$$\omega = \begin{bmatrix} \bar{\epsilon}_1 Z_1 & \bar{\epsilon}_2 Z_2 & \cdots & \bar{\epsilon}_n Z_n \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{bmatrix}$$

$$\boxed{\omega = J_\omega \dot{q}}$$

1. Linear velocity: Each column in $J_v$ is the joint axis if the joint is prismatic or the cross product of the axis with the vector connecting the joint to the end-effector if the joint is revolute.
2. Angular velocity: Each column in $J_\omega$ is the joint axis if the joint is revolute or the **0** vector if it is prismatic.
3. *Note:* We need to express all vectors w.r.t. the frame in which we want to define our Jacobian.

When using cartesian coordinates, we can also obtain $J_v$ directly from the derivative of the forward kinematics (or "translation" column in the homogeneous transform) w.r.t. the joint coordinates $q$. This is clearer when looking at the left side of the next figure.

The full Jacobian $J$ in some frame then consists of these $J_v$ and $J_\omega$ matrices. In order to express $J$ in reference frame $\{0\}$ we need to multiply every $Z_i$ axis with the the respective mapping ${}^0_iR$, as shown in the next figure. Note that the product ${}^0_iR\, Z_i$ is simply the third column of the rotation matrix! If we have the homogeneous transform, we can just pick the components for our Jacobian!

## Jacobian in a Frame

### Vector Representation

$$J = \begin{pmatrix} \dfrac{\partial x_P}{\partial q_1} & \dfrac{\partial x_P}{\partial q_2} & \cdots & \dfrac{\partial x_P}{\partial q_n} \\ \bar{\epsilon}_1 . Z_1 & \bar{\epsilon}_2 . Z_2 & \cdots & \bar{\epsilon}_n . Z_n \end{pmatrix}$$

### J in Frame {0}

$${}^0Z_i = {}^0_iR\, {}^iZ_i; \quad {}^iZ_i = Z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$${}^0J = \begin{pmatrix} \dfrac{\partial}{\partial q_1}({}^0x_P) & \dfrac{\partial}{\partial q_2}({}^0x_P) & \cdots & \dfrac{\partial}{\partial q_n}({}^0x_P) \\ \bar{\epsilon}_1 . ({}^0_1R . Z) & \bar{\epsilon}_2 . ({}^0_2R . Z) & \cdots & \bar{\epsilon}_n . ({}^0_nR . Z) \end{pmatrix}$$

In lecture 7 of the Stanford Lecture you can find an example to practice *seeing* the Jacobian. Finally, these two rules may help (derived in Lecture 8 of the Stanford lecture):

$$ {}^{i}J = \begin{pmatrix} {}^{i}_{j}R & 0 \\ 0 & {}^{i}_{j}R \end{pmatrix} {}^{j}J $$

$$ {}^{0}J_{e} = \begin{pmatrix} {}^{0}_{n}R & -{}^{0}_{n}R\,{}^{n}\hat{P}_{ne}\,{}^{0}_{n}R^{T} \\ 0 & {}^{0}_{n}R \end{pmatrix} {}^{n}J_{n} $$

## 5.6 Kinematic Singularity

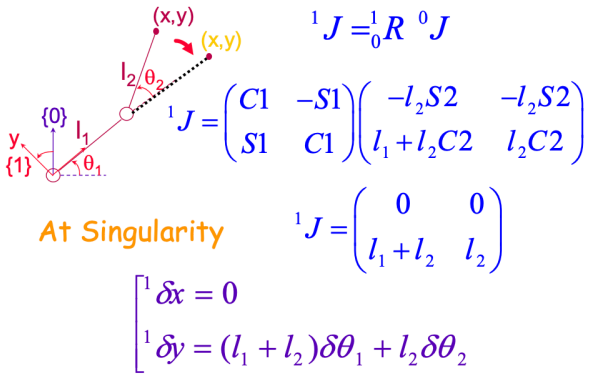
If the configuration of the robot manipulator reaches a singularity, the end-effector *locally* loses the ability to move in a direction or to rotate about a direction, e.g. because two axes of revolute joints become collinear (see this explanation).

Since the singularity happens, when $Z$ axes are aligned, the Jacobian does not have full rank in these situations (columns become dependant). In this case, the determinant is zero.

$$ \det(J) = 0 $$
$$ \det({}^{i}J) = \det({}^{j}J) $$
$$ \det(J) = \det(J^{T}J) \quad \text{Trick if } J \text{ is not square.} $$

The determinant is the same in every reference frame! We can then find all singular configurations by setting the $\det(J)$ in any reference frame to zero.

**Example (Kinematic Singularities)**




$$ {}^{1}J = {}^{1}_{0}R\ {}^{0}J $$

$$ {}^{1}J = \begin{pmatrix} C1 & -S1 \\ S1 & C1 \end{pmatrix} \begin{pmatrix} -l_2 S2 & -l_2 S2 \\ l_1 + l_2 C2 & l_2 C2 \end{pmatrix} $$

$$ {}^{1}J = \begin{pmatrix} 0 & 0 \\ l_1 + l_2 & l_2 \end{pmatrix} $$

$$ \begin{bmatrix} {}^{1}\delta x = 0 \\ {}^{1}\delta y = (l_1 + l_2)\delta\theta_1 + l_2\delta\theta_2 \end{bmatrix} $$

1. When the second joint angle is zero ($\theta_2 = 0$), the Jacobian in this frame becomes rank 1.
2. From the relationship $[\delta x, \delta y]^T = J\ \delta\Theta$ follows that no motion in the local $x$ direction is possible. This is a singularity.

## 5.7 Jacobian for Approximating very small ("infinitesimal") Movements

Taylor's theorem (which states that a function behaves like its derivative in a sufficiently small environment) can be used to approximate how the position of the end effector changes whenever small changes are made to robot parameters.

The theorem can be used to derive the following:

$$ \delta x \approx J^{-1}(f(x + \delta x) - f(x)) $$

$\delta x$ denotes a small change in $x$, or a small change in the robot parameters. Using this equation, we can relate small changes $\delta x$ in joint parameters to a small change $f(x + \delta x) - f(x)$ in position, and vice-versa.

This is the reason why singular configurations should be avoided: $J$ becomes ill-conditioned near those, meaning that computed $\delta x$ might approach infinity.

Singular positions are reached when $det(J) = 0$. Using this condition, we can figure out which joint parameters should be avoided.

## 5.8 Frame of reference of a Jacobian

Given a Jacobian in reference system B, the following equation related joint velocities to cartesian and angular velocities.

$$\begin{pmatrix} {}^{B}v \\ {}^{B}\omega \end{pmatrix} = {}^{B}J(\Theta)\dot{\Theta}$$

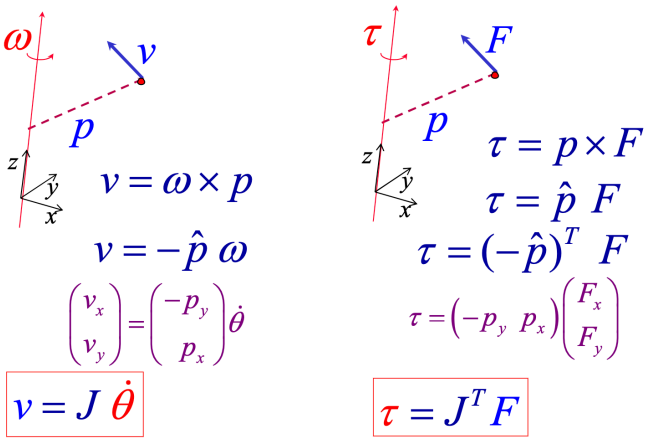
And for a Jacobian w.r.t system A:

$${}^{A}J(\Theta)\dot{\Theta} = \begin{pmatrix} {}^{A}_{B}R & \mathbf{0} \\ \mathbf{0} & {}^{A}_{B}R \end{pmatrix} {}^{B}J(\Theta)\dot{\Theta} \quad \Rightarrow \quad {}^{A}J(\Theta) = \begin{pmatrix} {}^{A}_{B}R & \mathbf{0} \\ \mathbf{0} & {}^{A}_{B}R \end{pmatrix} {}^{B}J(\Theta)$$

## 5.9 Forces and Torques in Static Manipulators

This section is based on page 156 of the book, exercise sheet 3 and the 8th lecture of the Stanford course.

In the figure below, two fundamental relations between torque exerted through motion around an axis and the force at a given point are derived.




The linear velocity $v$ is the cross product between $\omega$ and the point vector. $\omega \times p = -p \times \omega$ can be rewritten using the matrix operator $\hat{p}$ that expresses the cross product. In the right part of the figure we use the fact that $\hat{p} = (-\hat{p})^T$. When we calculate the results of the cross products, we are left with the equations in red. It turns out that, while linear and angular velocity are related via the Jacobian, torque is related to force via its transpose.

The book derives this equation in another way: in the multidimensional case, work is the dot product of a vector force or torque and a vector displacement:

$$\mathcal{F} \cdot \delta\mathcal{X} = \tau \cdot \delta\Theta$$

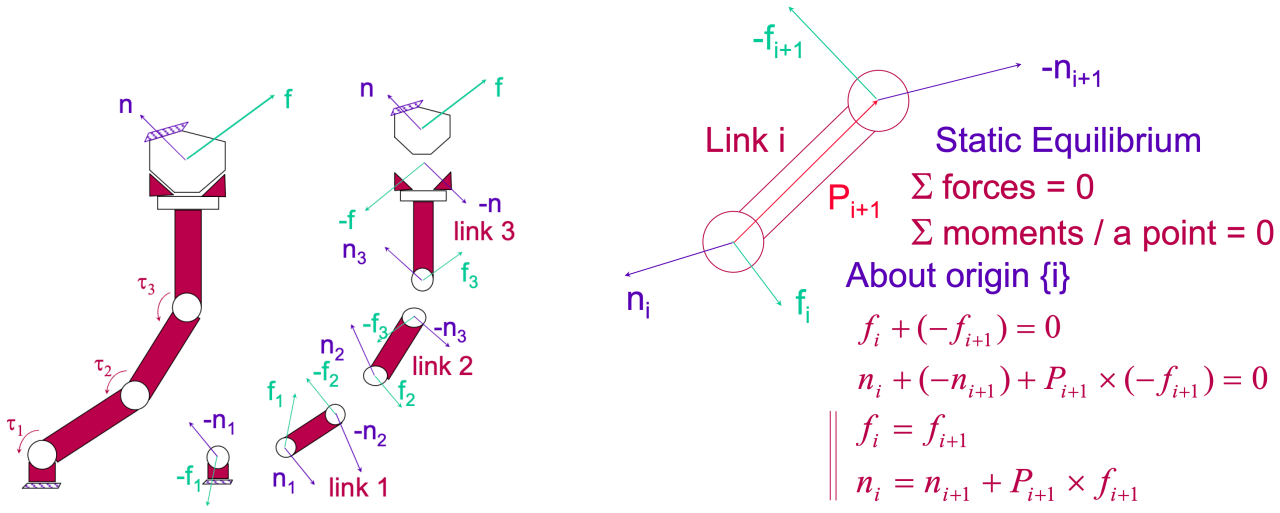
where $\mathcal{F}$ is a $6 \times 1$ cartesian force-moment vector acting at the end-effector, $\delta\mathcal{X}$ is a $6 \times 1$ infinitesimal cartesian displacement of the end-effector, $\tau$ is a $6 \times 1$ vector of torques at the joints, and $\delta\Theta$ is a $6 \times 1$ vector of infinitesimal joint displacements.

$$\begin{pmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_n \end{pmatrix} = \tau = {}^{A}J^{TA}\mathcal{F} = {}^{A}J^{T} \begin{pmatrix} {}^{A}f \\ {}^{A}n \end{pmatrix}$$

The Jacobian transpose maps cartesian forces acting at the hand into equivalent joint torques. When the Jacobian is written with respect to frame $\{0\}$, then force vectors written in $\{0\}$ can be transformed, as follows:

$$\tau = {}^{0}J^{T}\ {}^{0}\mathcal{F}$$

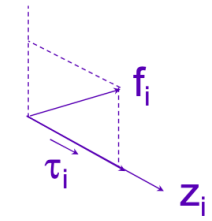
If we calculate the static equilibrium for each link in a robot manipulator, we are able to relate the forces and torques at the end effector to the forces and torques at the base. The next figure shows the (reaction) forces $f$ and torques $n$ in each link of the manipulator on the left and the static equilibrium of one link on the right.




We can then propagate the forces $f_i$ and torques $n_i$ acting on each link from the end-effector to the base:

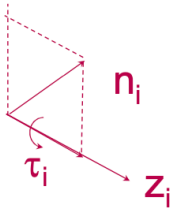
$$\begin{aligned} {}^{i}f_i &= {}^{i}_{i+1}R \cdot {}^{i+1}f_{i+1} \\ {}^{i}n_i &= {}^{i}_{i+1}R \cdot {}^{i+1}n_{i+1} + {}^{i}P_{i+1} \times {}^{i}f_i \end{aligned}$$

The torque that is exerted on prismatic/ revolute joins is given from the projection (inner product) on the joint axis, as shown in the next figure:




Prismatic Joint

$\tau_i = f_i^T Z_i$




Revolute Joint

$\tau_i = n_i^T Z_i$

$$\tau_i = {}^{i}n_i^{\mathrm{T}i}Z_i = {}^{i}n_i^{\mathrm{T}} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\tau_i = {}^{i}f_i^{\mathrm{T}i}Z_i = {}^{i}f_i^{\mathrm{T}} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

The quantities $\tau_i$ thus specify the amount of torque resp. force that is affecting the joint, and thus the amount of torque resp. force that the robot should counteract in order to remain static. The joint torques/forces $\tau_i$ are 1-dimensional quantities.

# Chapter 6: Robot Dynamics

The field of analytical dynamics, or more briefly dynamics, is concerned with the relationship between motion of bodies and its causes, namely the forces acting on the bodies and the properties of the bodies (particularly mass and moment of inertia) influencing that movement. This is in contrast to kinematics, where we were not concerned with the physical phenomena causing robot movement, but only with the movement itself.

We want to come up with equations of motion for any nDOF system, which express the force require to cause motion. The expression should have the form $\dot{q} = f(q, t)$
We can use it to choose an appropriate controller that will put our dynamical system in a desired state. This section is based on Lecture 11 and covered in exercise 4.

## 6.1 Rigid Body Dynamics

We now extend our analysis of rigid-body motion to the case of accelerations. At any instant, the linear and angular velocity vectors have derivatives that are called the linear and angular accelerations, respectively. That is,

$$ {}^B\dot{V}_Q = \frac{d}{dt}\,{}^B V_Q = \lim_{\Delta t \to o} \frac{{}^B V_Q(t + \Delta t) - {}^B V_Q(t)}{\Delta t}, \tag{6.1} $$

and

$$ {}^A\dot{\Omega}_B = \frac{d}{dt}\,{}^A\Omega_B = \lim_{\Delta t \to o} \frac{{}^A\Omega_B(t + \Delta t) - {}^A\Omega_B(t)}{\Delta t}. \tag{6.2} $$

As with velocities, when the reference frame of the differentiation is understood to be some universal reference frame, $\{U\}$, we will use the notation

$$ \dot{v}_A = {}^U\dot{V}_{AORG} \tag{6.3} $$

and

$$ \dot{\omega}_A = {}^U\dot{\Omega}_A. \tag{6.4} $$

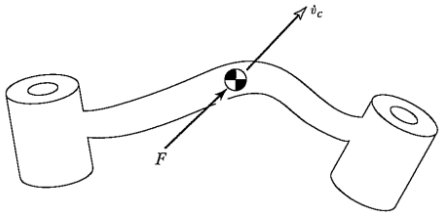
## 6.2 Newton-Euler Method

Overall, the purpose of the Newton-Euler-Method is determining the joint torques $\tau$ that are required to achieve a desired motion of the robot. More specifically, if we have a joint trajectory $\Theta(t)$ that describes the positions and velocities for a certain trajectory of the robot, then the Newton-Euler method will allow us to compute corresponding $\tau(t)$ that will, in an ideal world (no friction or other disturbances for the moving robot), cause the robot to carry out the desired trajectory.

### 6.2.1 Newton's equation

A rigid body's center of mass is accelerating with $\dot{v}_c$ when the force $F$ acts on it. The acceleration is given by

$$ F = m\dot{v}_c $$

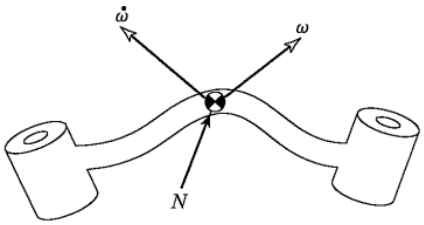



### 6.2.2 Euler's equation

The rigid body in the image below rotates with angular velocity $\omega$ and with angular acceleration $\dot{\omega}$. Then, the moment $N$, which acts on the link is:

$$N = {}^{C}I\dot{\omega} + \omega \times {}^{C}I\omega$$

where ${}^{C}I$ is the inertia tensor that is computed with respect to the center of mass in frame $C$, a $3 \times 3$ matrix that describes the distribution of masses in a link.




### 6.2.3 Linear acceleration

The linear acceleration is the derivative of the linear velocity of the joint w.r.t. time. In case of a rotational joint $i+1$, we have:

$$\begin{aligned}
{}^{i+1}v_{i+1} &= {}^{i+1}_{i}R({}^{i}v_i + {}^{i}\omega_i \times {}^{i}P_{i+1}) \\
{}^{i+1}\dot{v}_{i+1} &= {}^{i+1}_{i}R\left({}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times \left({}^{i}\omega_i \times {}^{i}P_{i+1}\right) + {}^{i}\dot{v}_i\right)
\end{aligned}$$

When joint $i+1$ is prismatic, the formula becomes more complicated:

$$\begin{aligned}
{}^{i+1}v_{i+1} &= {}^{i+1}_{i}R({}^{i}v_i + {}^{i}\omega_i \times {}^{i}P_{i+1}) + \dot{d}_{i+1} \cdot {}^{i+1}\hat{Z}_{i+1} \\
{}^{i+1}\dot{v}_{i+1} &= {}^{i+1}_{i}R\left({}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times \left({}^{i}\omega_i \times {}^{i}P_{i+1}\right) + {}^{i}\dot{v}_i\right) + 2 \cdot {}^{i+1}\omega_{i+1} \times \dot{d}_{i+1}{}^{i+1}Z_{i+1} + \ddot{d}_{i+1}{}^{i+1}Z_{i+1}
\end{aligned}$$

### 6.2.4 Angular acceleration

The angular velocities and accelerations for rotational joints are:

$$\begin{aligned}
{}^{i+1}\omega_{i+1} &= {}^{i+1}_{i}R \cdot {}^{i}\omega_i + \dot{\Theta}_{i+1} \cdot {}^{i+1}Z_{i+1} \\
{}^{i+1}\dot{\omega}_{i+1} &= {}^{i+1}_{i}R \cdot {}^{i}\dot{\omega}_i + {}^{i+1}_{i}R \cdot {}^{i}\omega_i \times \dot{\Theta}_{i+1} \cdot {}^{i+1}Z_{i+1} + \ddot{\Theta}_{i+1}{}^{i+1}Z_{i+1}
\end{aligned}$$

When joint $i+1$ is prismatic, the formulas simplifies to:

$$\begin{aligned}
{}^{i+1}\omega_{i+1} &= {}^{i+1}_{i}R \cdot {}^{i}\omega_i \\
{}^{i+1}\dot{\omega}_{i+1} &= {}^{i+1}_{i}R \cdot {}^{i}\dot{\omega}_i
\end{aligned}$$

### 6.2.5 Newton-Euler Algorithm

Each link of a manipulator is considered as a rigid body. Remember the recursive equations for the forces and torques propagated to the next link from the previous chapter? In the non-static scenario, we need to modify them, such that they regard the velocities and accelerations acting on each link.

The Newton-Euler-Method is composed of two phases: A forward phase and a backwards phase. In the first phase, the velocities and accelerations (both rotational and linear) are computed for each joint. Using these values, we compute the forces and torques that the motion exerts on each link. In the backwards phase, these forces and torques are used to compute the forces and torques that act on each joint.

The linear velocity of the center of mass for rotational and prismatic joints is:

$$ {}^i\dot{v}_{C_i} = {}^i\dot{\omega}_i \times {}^iP_{C_i} + {}^i\omega_i \times ({}^i\omega_i \times {}^iP_{C_i}) + {}^i\dot{v}_i $$

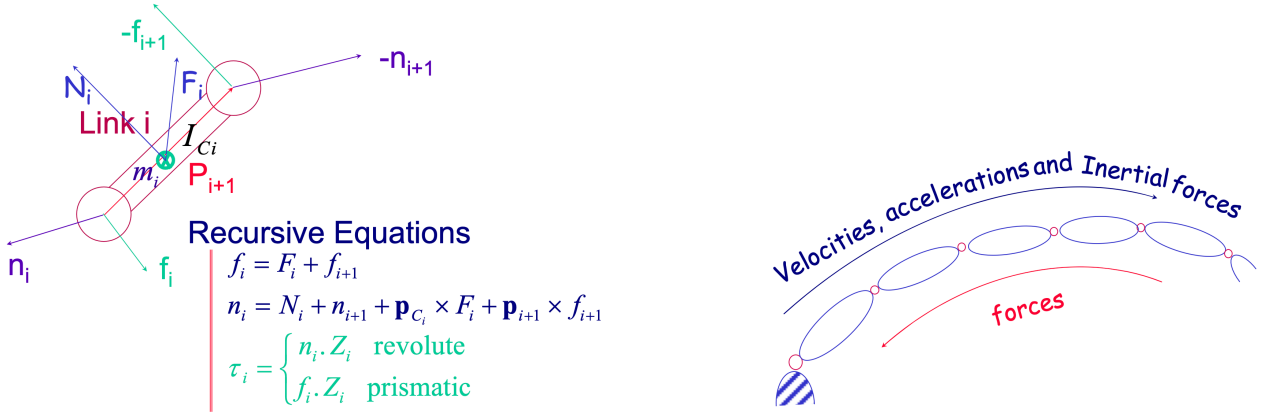
In the next step, we can compute the forces and torques that apply to the center of mass of each link from the accelerations and speeds that we have just computed:

$$ {}^iF_i = m_i \cdot {}^i\dot{v}_{C_i} $$
$$ {}^iN_i = {}^{C_i}I_i \cdot {}^i\dot{\omega}_i + {}^i\omega_i \times {}^{C_i}I_i \cdot {}^i\omega_i $$

Based on those forces and moments, we can compute $f_i$ and $n_i$ for joint $i$ of the manipulator:

$$ {}^if_i = {}^i_{i+1}R \cdot {}^{i+1}f_{i+1} + {}^iF_i $$
$$ {}^in_i = {}^iN_i + {}^i_{i+1}R \cdot {}^{i+1}n_{i+1} + {}^iP_{C_i} \times {}^iF_i + {}^iP_{i+1} \times {}^i_{i+1}R^{i+1}f_{i+1} $$

As in the static case, the values of $\tau_i$, compute as $\tau_i = {}^in_i^T \cdot {}^iZ_i$ for a rotational joint and $\tau_i = {}^if_i^T \cdot {}^iZ_i$ for a translational joint.




If we know the location of the center of mass and the inertia tensor of the link, then its mass distribution is completely characterised. In order to move the links, we must We accelerate and decelerate them. The forces required for such motion are a function of the acceleration desired and of the mass distribution of the links.

Newton's equation, along with its rotational analog, Euler's equation, describes how forces, inertias, and accelerations relate.

Exercise sheet 4 has a simple examples which demonstrates the Newton-Euler-Algorithm quite well.

## 6.3 State-Space equation (M-V-G-form)

We also need to take into account the effect of gravity on the links of the robot. This can be done by setting ${}^0\dot{v}_0 = -G$ where $G$ is the gravitational acceleration. We obtain the M-V-G-form by rearranging the equations of movement for the joint torques:

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta)$$

$M$ is a $n \times n$ matrix, and $V$ as well as $G$ are $n \times 1$ vectors. Note that in general, $M$, $V$, $G$ are complex-valued. The matrix $M$ can be determined from the dynamics equations by factorizing all summands that contain$\ddot{\theta}$. Similarily, $V$ is determined by factorizing all summands that contain $\dot{\theta}$. Finally, $G$ is determined by factorizing all summands that contain $g$. $V$ can be further decomposed into components $B$ and $C$, yielding the configuration-space equation (or M-B-C-G-form):

$$\tau = M(\Theta)\ddot{\Theta} + B(\Theta)[\dot{\Theta}\dot{\Theta}] + C(\Theta)\left[\dot{\Theta}^2\right] + G(\Theta)$$

$B(\Theta)$ is a matrix of dimension $n \times n(n-1)/2$, and $[\dot{\Theta}\dot{\Theta}]$ is an abbreviation for the vector

$$\left(\dot{\Theta}_1\dot{\Theta}_2, \dot{\Theta}_1\dot{\Theta}_3, \ldots, \dot{\Theta}_{n-1}\dot{\Theta}_n\right)$$

that has length $n(n-1)/2$. This also explains the dimension of $B$: $n(n-1)/2$ is the number of products $\Theta_i\Theta_j$ with $i \neq j$. The matrix $C$ has dimension $n \times n$, and $\left[\dot{\Theta}^2\right]$ in an abbreviation for the vector

$$\left(\dot{\Theta}_1^2, \dot{\Theta}_2^2, \ldots, \dot{\Theta}_n^2\right)^{\mathrm{T}}$$

with length $n$. The matrices $B$ and $C$ can be determined by finding the coefficients of $\Theta_i\Theta_j$ and $\Theta_i^2$, respectively.

## 6.4 Mass distribution

In systems with a single degree of freedom, we often talk about the mass of a rigid body. In the case of rotational motion about a single axis, the notion of the moment of inertia is a familiar one.

### 6.4.1 Linear Momentum

The rate of change of the linear momentum for a particle with mass $m$ is equal to the applied force.

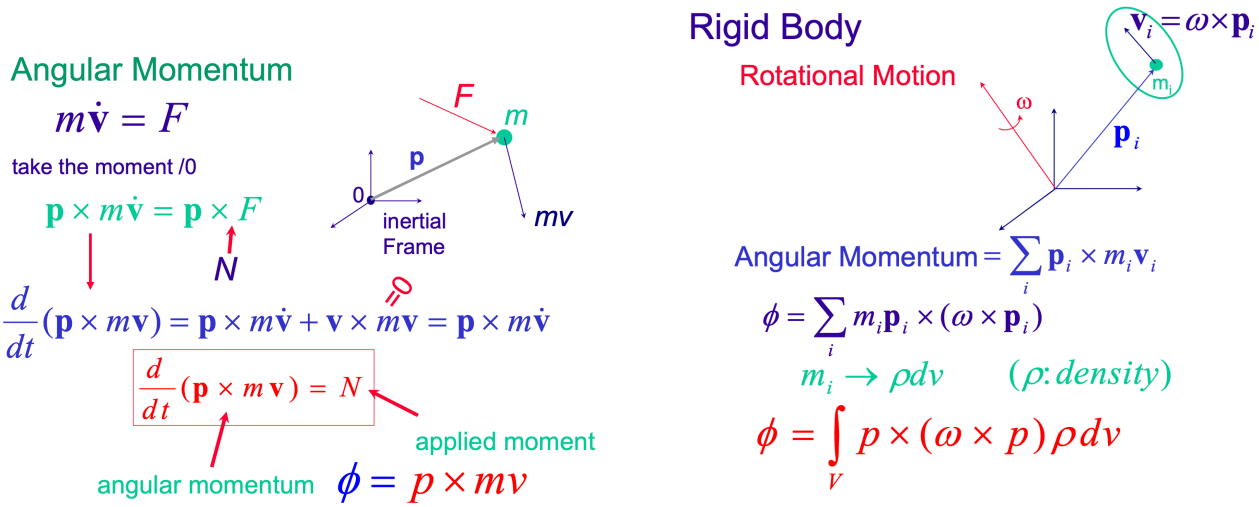
$$\frac{d}{dt}(mv) = F$$
$$\varphi = mv$$

### 6.4.2 Angular Momentum

...is the total moment of the momentum of a rigid body's constituting particles:

$$H_c = \int_{\mathcal{B}} r \times x \, dm = {}^A\mathbf{I}\omega$$

Angular momentum can be understood as the cross-product of the linear momentum $mv$ and a position vector $p$. This follows from the fact that the rate of change of the angular momentum is equal to the applied moment, hence the left image below. For a rigid body (right image), the total angular momentum $\phi$ is the sum of angular momentum for all particles. The velocity is written as $\omega \times p_i$ and the mass is the integral over the density multiplied by the volume.




Using the reformulation of the cross-product operator $\hat{\mathbf{p}} = \mathbf{p}\times$ and the fact that $\omega \times \mathbf{p} = -\mathbf{p} \times \omega$, wherein $\omega$ is shared among all points, we derive:

$$\mathbf{p} \times (\omega \times \mathbf{p}) = \hat{\mathbf{p}}(-\hat{\mathbf{p}})\omega$$

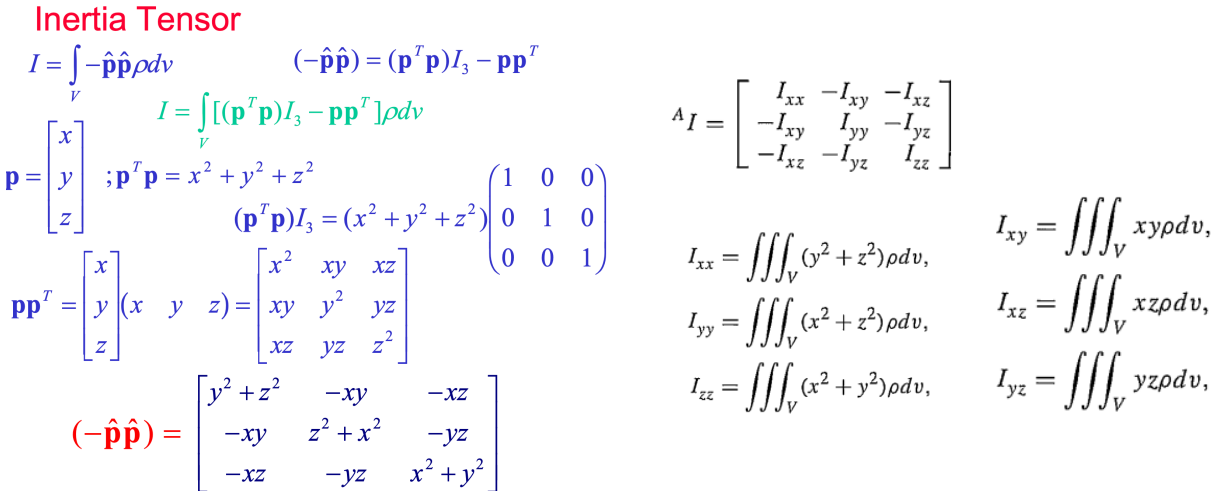
$$\phi = \int_V p \times (\omega \times p)\rho dv = \left[\int_V -\hat{\mathbf{p}}\hat{\mathbf{p}}\rho dv\right] \omega = \mathbf{I}\omega$$

$$\frac{d}{dt}(\mathbf{I}\omega) = N$$

### 6.4.3 Inertia Tensor

For a rigid body that is free to move in three dimensions, there are infinitely many possible rotation axes. In the case of rotation about an arbitrary axis, we need a complete way of characterizing the mass distribution of a rigid body.

The **inertia tensor** can be thought of as a generalization of the scalar moment of inertia of an object, relative to the frame attached to the object (rigid body). The left side of the following figure shows the derivation of the inertia tensor from the volume integral of $-\hat{\mathbf{p}}\hat{\mathbf{p}}$. Remember that $\hat{p}$ is the cross-product operator for a point on the rigid body relative to the origin. $I_{xx}, I_{yy}, I_{zz}$ are called *moments of inertia*, while $I_{xy}, I_{xz}, I_{yz}$ are called *products of inertia*.




For an example, check example 6.1 on page 169 or 6.2 on page 171.

**Some properties:**

- Inertia matrices are positive-definite, symmetric matrices.
- The inertia matrix is not in general constant and is frame-dependent:

$$\mathbf{I}_c^i = R_i^T \mathbf{I}_c R_i$$

- Any rigid body has a set of principal directions with respect to which the inertia matrix is diagonal.
- If $xy$ is the plane of symmetry, then $I_{xz} = I_{yz} = 0$ (similarly for $xz$ or $yz$).
- If the body is axissymmetric (e.g. symmetric about $z$), then the inertia matrix is diagonal and 2 of the moments of inertia equal (e.g. $I_{xx} = I_{yy}$ if z is axis of symmetry)

### 6.4.4 Parallel-Axis Theorem

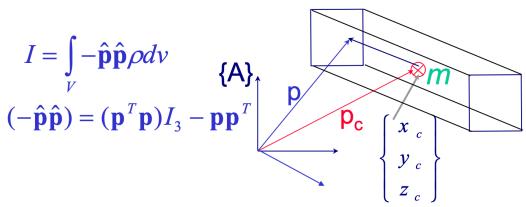
... is a way of computing how the inertia tensor changes under *translations* of the reference coordinate system (s.t. the axes remain parallel). It relates the inertia tensor in a frame with origin at the center of mass ${}^C I$ to the inertia tensor with respect to another reference frame ${}^A I$. $C$ is located at the center of mass of the body, and $A$ is an arbitrarily translated frame. The theorem can be stated as:

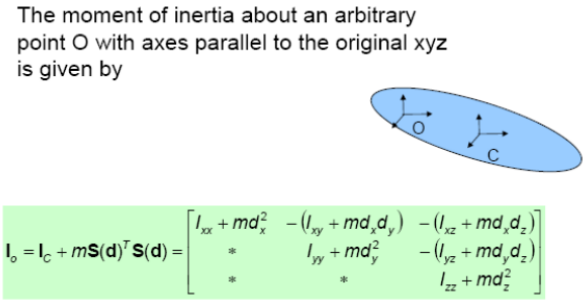
$$\begin{aligned} {}^A I_{zz} &= {}^C I_{zz} + m(x_c^2 + y_c^2) \\ {}^A I_{xy} &= {}^C I_{xy} - m x_c y_c \end{aligned}$$

Where $P_c = [x_c, y_c, z_c]^T$ locates the center of mass relative to $A$. The remaining moments and products of inertia are computed from permutations of $x, y, z$. The theorem in vector-matrix form:

$${}^A I = {}^C I + m[P_c^T P_c I_3 - P_c P_c^T]$$

with $I_3$ as the identity matrix. The left side of the following image shows the new reference frame $A$ and the center of mass $m$, as well as the connecting vector $P_c$.




## 6.5 Euler-Lagrange Equations

The Lagrangian dynamic formulation is another method for determining the dynamics of a robot which is derived from energy considerations. The Lagrange equation is:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\Theta}} - \frac{\partial L}{\partial \Theta} = \tau$$

where the Lagrangian $L = k - u$ is the difference between the kinetic energy $k$ and the potential energy $u$. Since $U = U(\Theta)$, the equation simplifies to

$$\tau = \frac{d}{dt}\frac{\partial k}{\partial \dot{\Theta}} - \frac{\partial k}{\partial \Theta} + \frac{\partial u}{\partial \Theta}.$$

The derivatives of the Lagrange equation are given the left part of the figure below. We set $G = \frac{\partial u}{\partial \Theta}$ since it only depends on $\Theta$. By calculating the derivatives, we see that the Lagrange equation is in M-V-G-form. Next, we can identify $M(q)$ from the expression $k = \frac{1}{2}\dot{q}^T M \dot{q}$. With knowledge of $M$ the expression in pink gives us $V(q, \dot{q})$.

$$\frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}}\right) - \frac{\partial K}{\partial q} = \tau - G \qquad K = \frac{1}{2}\dot{q}^T M(q)\dot{q}$$

$$\frac{\partial K}{\partial \dot{q}} = \frac{\partial}{\partial \dot{q}}\left[\frac{1}{2}\dot{q}^T M(q)\dot{q}\right] = M(q)\dot{q}$$

$$\frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}}\right) = \frac{d}{dt}(M\dot{q}) = M\ddot{q} + \dot{M}\dot{q}$$

$$\frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}}\right) - \frac{\partial K}{\partial q} = M\ddot{q} + \dot{M}\dot{q} - \frac{1}{2}\begin{bmatrix} \dot{q}^T \frac{\partial M}{\partial q_1}\dot{q} \\ \vdots \\ \dot{q}^T \frac{\partial M}{\partial q_n}\dot{q} \end{bmatrix} = M\ddot{q} + V(q, \dot{q})$$

$$M(q)\ddot{q} + V(q,\dot{q}) + G(q) = \tau$$

$$M(q): K = \frac{1}{2}\dot{q}^T M \dot{q} \qquad M(q) \Rightarrow V(q,\dot{q})$$

For each link of the robot, the kinetic energy can be computed as (and is the same for any reference frame of the velocities):

$$k_i = \tfrac{1}{2} m_i v_{C_i}^{\mathrm{T}} \cdot v_{C_i} + \tfrac{1}{2} {}^{i}\omega_i^{\mathrm{T}} \cdot {}^{C_i}I_i \cdot {}^{i}\omega_i$$

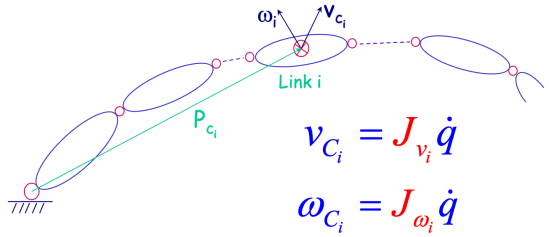
The first term corresponds to the kinetic energy caused by the linear motion of the link, and the second term corresponds to the kinetic energy caused by the rotational velocity of the link. To determine these energies, we need to compute the linear and rotational velocities of the joints. The overall kinetic energy computes then as sum of the kinetic energies of all links:

$$k = \textstyle\sum_{i=1}^{n} k_i$$

Another way to compute kinetic energy is

$$k(\Theta, \dot{\Theta}) = \tfrac{1}{2}\dot{\Theta}^{\mathrm{T}} M(\Theta)\dot{\Theta}$$

where $M$ again is the $n \times n$ mass matrix from the M-V-G form of the dynamics equations. By setting both formulas for $k$ equal to each other and expressing the velocities at the centres of mass with their Jacobians, we derive an expression for $M$, as the next figure shows. Note that these Jacobians relate the joint parameters to the velocities at the centres of mass of each link instead of at the joints. As shown in the bottom right corner, they only regard these relations up to the link $i$. All columns after this point are zero.




$$v_{C_i} = J_{v_i}\dot{q}$$

$$\omega_{C_i} = J_{\omega_i}\dot{q}$$

$$\frac{1}{2}\dot{q}^T M \dot{q} = \frac{1}{2}\sum_{i=1}^{n}\left(m_i v_{C_i}^T v_{C_i} + \omega_i^T I_{C_i}\omega_i\right)$$

$$= \frac{1}{2}\sum_{i=1}^{n}\left(m_i \dot{q}^T J_{v_i}^T J_{v_i}\dot{q} + \dot{q}^T J_{\omega_i}^T I_{Ci} J_{\omega_i}\dot{q}\right)$$

$$\frac{1}{2}\dot{q}^T M \dot{q} = \frac{1}{2}\dot{q}^T\left[\sum_{i=1}^{n}\left(m_i J_{v_i}^T J_{v_i} + J_{\omega_i}^T I_{Ci} J_{\omega_i}\right)\right]\dot{q}$$

$$M = \sum_{i=1}^{n}\left(m_i J_{v_i}^T J_{v_i} + J_{\omega_i}^T I_{Ci} J_{\omega_i}\right)$$

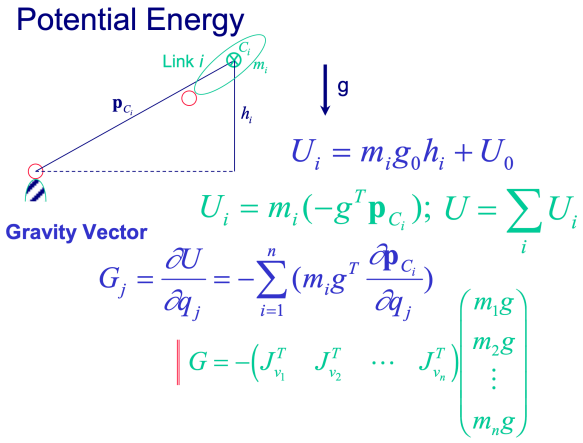
$$J_{v_i} = \left[\frac{\partial p_{C_i}}{\partial q_1} \quad \frac{\partial p_{C_i}}{\partial q_2} \quad \cdots \quad \frac{\partial p_{C_i}}{\partial q_i} \quad 0 \quad 0 \quad \cdots \quad 0\right]$$

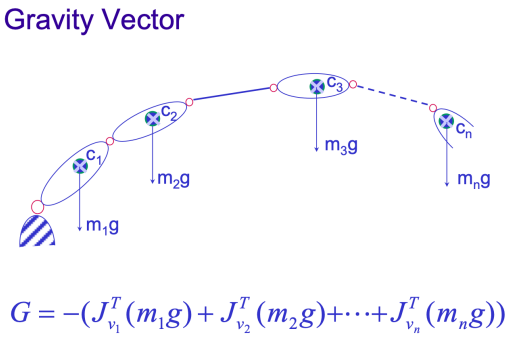
$$J_{\omega_i} = \left[\bar{\varepsilon}_1 z_1 \quad \bar{\varepsilon}_2 z_2 \quad \cdots \quad \bar{\varepsilon}_i z_i \quad 0 \quad 0 \quad \cdots \quad 0\right]$$

To compute all energies that are present in this system, we also need to take into account the potential energy:

$$u_i = -m_i \cdot {}^0g^{\mathrm{T}} \cdot {}^0P_{C_i} + u_{\mathrm{ref}_i}$$

Here, $g$ is the vector of gravity, ${}^0P_{C_i}$ denotes the centre of mass of link $i$, and $u_{\mathrm{ref}_i}$, is an arbitrary constant (the constant is added because potential energy depends on height, and a certain base height can be chosen arbitrarily). In the further computations, this constant will not play a role, since only the derivatives of the potential energy are considered - and any constant will vanish when differentiated. By differentiating the potential energy, we obtain the gravity vector $G$, which also contains the Jacobians. Thus, another way to think of the gravity vector is as the torque in each joint that is caused by a force pulling down on every link of the robot. This is shown in the right part of the figure below.




The computation of $\tau$ is finally done through the following formula:

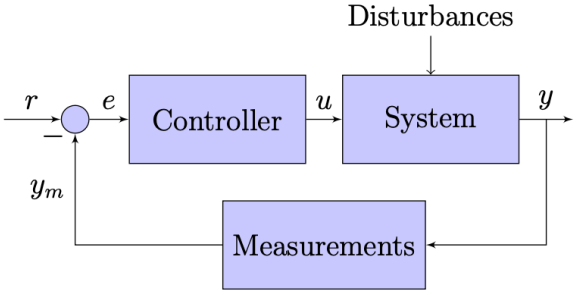
$$\tau = \frac{d}{dt}\frac{\partial k}{\partial \dot{\Theta}} - \frac{\partial k}{\partial \Theta} + \frac{\partial u}{\partial \Theta}$$

It is also possible to compute the joint torques i on a per-joint basis, which is more practical in most cases. The formula then becomes:

$$\tau_i = \frac{d}{dt}\frac{\partial k}{\partial \dot{\Theta}_i} - \frac{\partial k}{\partial \Theta_i} + \frac{\partial u}{\partial \Theta_i}$$

# Chapter 7: Control

Being able to establish the dynamics equations of a robot is still not enough to make it carry out a desired trajectory. There are several disturbances in physical systems that cannot be modelled, which means that a controlling scheme is required to alleviate the influence of those disturbances. The following schematic gives an overview of a typical control architecture:

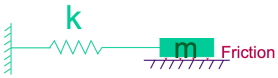



The value $r$ would be the desired state of the system. Computing the difference $e = y_m - r$ yields the error of the current state $y_m$ with respect to the desired state. The controller takes $e$ into account when generating control signals $u$, in order to correct any error introduced by disturbances.

This section is based on the explanations in exercise 5, lecture 13 and lecture 14.

## 7.1 Mass-Spring-Systems

As a prerequisite to understanding the function of a PD controller, mass-spring-systems have been studied in the lecture. The importance of these systems lies in their connection to the behaviour of the error $e$ in control problems. We will later see that the error $e$ behaves like a mass-spring-system itself.

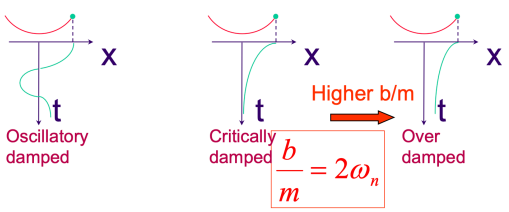



The dynamic equation of the mass-spring-system is:

$$m\ddot{x} + b\dot{x} + kx = f$$

where $m$ is the mass of the object, $b$ is the friction constant and $k$ is the spring constant. The deflection of the object from its resting position is denoted by $x$. This equation is, in the context of control, also called the *open-loop-equation*, since it does not take into account any feedback from the system (like, e.g., in the case of a robot, joint position measurements from sensors or such).

We distinguish three possible cases for the applied force $f$: Under-damping leads to oscillation of the system (spring stiffness dominates), while over-damping leads to slow convergence to the resting position (friction dominates). Critical damping leads to the fastest possible transition of the system to its resting configuration, avoiding both overdamping and underdamping (friction and stiffness are in balance).



### 7.1.1 Solution of the Dynamic Equation (Position Control)

How can we achieve critical damping? The force $f$ that we exert on the object must depend on friction and spring forces, and we can assume that it's of the form $f = -k_p x - k_v \dot{x}$ (proportional-derivative-/ PD-control; the formula is derived here in the lecture). This yields the *closed-loop equation*:

$$m\ddot{x} + (b + k_v)\,\dot{x} + (k + k_p)\,x = 0$$

We are interested in controlling the movement of the body, so we are interested in determining the function $x(t)$, which means we need to solve the differential equation. This is done by solving the characteristic equation:

$$ms^2 + (b + k_v)\,s + (k + k_p) = 0$$

The solutions to this quadratic equation are given by the well-known formula

$$s_{1,2} = \frac{-(b + k_v) \pm \sqrt{(b + k_v)^2 - 4m\,(k + k_p)}}{2m} = \frac{-b' \pm \sqrt{b'^2 - 4mk'}}{2m}.$$

It is then common to abbreviate $(b + k_v)$ as $b'$ and $(k + k_p)$ as $k'$. The solutions $s_1, s_2$ determine the trajectory $x(t)$ as follows:

$$x(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t}$$

A special case is $s_1 = s_2 \in \mathbb{R}$. This case corresponds to critical damping of the system. Note that this can be achieved by choosing $b'$ and $k'$ such that $b'^2 - 4mk' = 0$, or $b' = 2\sqrt{mk'}$ (we can assume $b'$ and $k'$ to be positive). In the case of an oscillating system, the equations can also be stated in a different form:

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0$$

with the "damping ratio" $\zeta$ and the "natural frequency" $\omega_n$

$$\zeta = \frac{b'}{2\sqrt{k'm}}$$
$$\omega_n = \sqrt{\frac{k'}{m}}.$$

Now, we design our PD-controller by setting $k_p$ and $k_v$, such that we achieve critical damping (see the experiments in lecture 14 for an example). Herein, if the damping ratio $\zeta$ is too small, the joint oscillates and if it is too large, it is overdamped (reaches the goal state slower). Moreover, the natural frequency $\omega$ is limited by an upper bound, as explained in the next section. If this upper bound is exceeded, the motion becomes unstable.

### 7.1.2 Resonance

Until now, we assumed that all parts of the mechanic systems we observed do not deform at all. But in reality, components of robots have a finite stiffness, which means that they deform minimally under stress. This can lead to the undesired effect of resonance, which leads to deformations adding up until finally components may be damaged.

A simple possibility of taking the effect of resonance into account is through enforcing the following inequality:

$$\omega_n \leq \frac{1}{2}\omega_{\text{res}}$$

Here, $\omega_{\text{res}}$ is the so-called resonant frequency of the object. If this condition holds, we can be sure that no undesired resonances occur.

### 7.1.3 Control Partitioning

Until now, we have used the following simple rule to influence the behaviour of a mass-spring system:

$$f = -k_p x - k_v \dot{x}$$

Using this rule, we were able to achieve critical damping of a mass-spring system, choosing parameters $k'$ and $b'$ according to $b' = 2\sqrt{mk'}$. However, the choice of $b'$ depends on $m$, which is acceptable for a simple mass-spring system, but makes things very complicated in more complex systems. Thus, we want to decouple the mass-dependent part from the equation, using an extended rule that reads as follows:

$$f = \alpha f' + \beta$$

This equation is known as the *model-based portion*. Through this rule, we want to achieve the following: Factors $\alpha$ and $\beta$ should be chosen such that the system, considering only $f'$ as input, behaves like a unit mass, governed by the equation

$$\ddot{x} = f'$$

Writing down the complete equation for the system, we obtain

$$m\ddot{x} + b\dot{x} + kx = \alpha f' + \beta.$$

Now comes the interesting part: How should we choose $\alpha$ and $\beta$ in order to achieve the desired effect? Obviously, $\beta = b\dot{x} + kx$ and $\alpha = m$ must hold. Note that we are now able to influence the system directly through $f'$. This is the decoupled open-loop equation. Again, be aware that for the simple case of a mass-spring system there is not much gained through rewriting the system like this, but for manipulating more complicated systems, the advantage is substantial.

Now, we move again from the open-loop form to the closed-loop form. Thus, let $f'$ again depend on spring force and friction force, such that

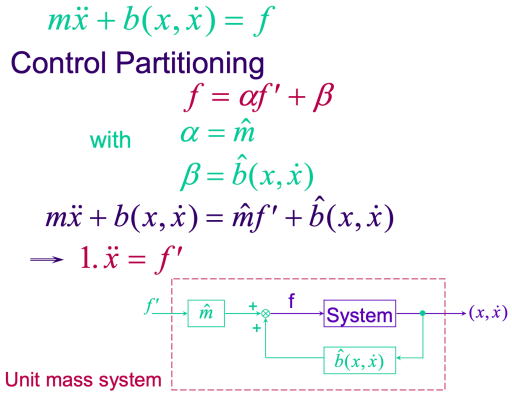
$$f' = -k'_p x - k'_v \dot{x}$$

with some new constants $k'_p$ and $k'_v$. The equation of motion becomes

$$\ddot{x} + k'_p x + k'_v \dot{x} = 0$$

In particular, we see that now $k_v$ and $k_p$ are now independent of system parameters. The system will always be critically damped if $k'_v = 2\sqrt{k'_p}$.

In other words, we want to approximate the mass-spring system with a unit mass system that is linearly scaled by $\alpha$ and $\beta$. Herein, $\alpha$ compensates for the mass ($m$ or the mass matrix $M$) and $\beta$ compensates for $b\dot{x} + kx$ (or other terms like centrifugal, coreolis and gravitational forces with $V$ and $G$). All terms that depend on $m$ cancel out and we are left with the equation for the unit mass system, for which we then design $k'_v$ and $k'_p$ independent of the mass.




In the slide on the left, the non-linear system $m\ddot{x} + b(x,\dot{x}) = f$ is approximated (through control partitioning) with the factors $\alpha$ and $\beta$. This equation is equivalent to $f' = \ddot{x}$ (the other terms cancel out), which is the unit mass system. The control loop shows how the unit mass system is linearly scaled with the factor $\hat{m}$ and $\hat{b}(x,\dot{x})$.

### 7.1.4 Trajectory following/ Motion Control

Now we no longer assume that we are simply interested in achieving critical damping of a mass-spring system, but instead we want the system to carry out a certain trajectory. The computed trajectory shall be denoted by $x_d(t)$, and we assume that $x_d(t)$ is a twice continuously differentiable function. Let $e(t) = x_d(t) - x(t)$ denote the difference between the actual position and the desired position. Movement along the desired trajectory can now be achieved by employing the following control rule (*servo-portion* for trajectory following/ "servo control law"):

$$f' = \ddot{x}_d + k_v \dot{e} + k_p e$$

Substituting this rule into the partition scheme, we obtain:

$$\ddot{x} = \ddot{x}_d + k_v \dot{e} + k_p e \Leftrightarrow \ddot{e} + k_v \dot{e} + k_p e = 0$$

We see that the error $e$ now behaves like a mass-spring system! This means that we can achieve critical damping of the error through appropriate choice of $k_p$ and $k_v$. This means that the error will tend towards 0 , and it will approach 0 with a speed depending on $k_v$ and $k_p$ - critical damping will thus provide the fastest possible convergence of the error towards 0.

**Motion Control**

$m\ddot{x} + b(x, \dot{x}) = f \underset{f = mf' + b}{\Rightarrow} 1.\ddot{x} = f'$

**Goal Position ($x_d$):**

Control: $f' = -k_v'\dot{x} - k_p'(x - x_d)$

Closed-loop System: $1.\ddot{x} + k_v'\dot{x} + k_p'(x - x_d) = 0$

**Trajectory Tracking**

$x_d(t)$; $\dot{x}_d(t)$; and $\ddot{x}_d(t)$

Control: $f' = \ddot{x}_d - k_v'(\dot{x} - \dot{x}_d) - k_p'(x - x_d)$

**Closed-loop System:**

$(\ddot{x} - \ddot{x}_d) + k_v'(\dot{x} - \dot{x}_d) + k_p'(x - x_d) = 0$

with $e \equiv x - x_d$

$\ddot{e} + k_v'\dot{e} + k_p'e = 0$

The slide on the left summarises the control rules used for position control and trajectory tracking (controlling the error). The respective control force then leads to the closed-loop equation.

Note that this is only true if there are no further unmodeled effects present in the system - in practice, one usually employs a PID controller (PD controller with additional integral part) to assure that the error always approaches 0. What we have discussed here is a mathematical treatment of the PD-controller. Above derivation explains the typical oscillating behaviour of PD-controllers. Furthermore, we see that it is possible to compute the proportional and differential factors directly.

### 7.1.5 Multi-dimensional systems

The decoupling method might not seem very useful at first sight, because it does not really help much with the simple case of a mass-spring-system with only one object. The main advantage of using this partitioning scheme is that it greatly simplifies the control problem for multi-dimensional problems. Assuming that $x$ is a multi-dimensional quantity, the equations of motion without partitioning will look like this:

$$M\ddot{x} + B\dot{x} + Kx = f$$

Here we can as well adopt the approach of using system feedback, in hope of achieving critical damping for that system. We set the controlling force

$$f = -K_v\dot{x} - K_p x$$

with some matrices $K_v$ and $K_p$, and we end up with:

$$M\ddot{x} + (B + K_v)\,\dot{x} + (K + K_p)\,x = 0$$

This is a multi-dimensional linear differential equation, and it is quite difficult to handle and to analyze. Determining $K_v$ and $K_p$ in order to achieve critical damping becomes a **major** problem when using this approach!

Now let's find out how this works when the partitioning scheme is in use. We set $f = \alpha f' + \beta$, where $\alpha = M$, and $\beta = B\dot{x} + Kx$. We end up with:

$$M\ddot{x} + B\dot{x} + Kx = Mf' + B\dot{x} + Kx$$

Assuming that $M$ is invertible, this transforms into

$$\ddot{x} = f'$$

just as it did in the one-dimensional case. Further setting $f' = -K_v\dot{x} - K_p x$, we end up with the following system:

$$\ddot{x} + K_v\dot{x} + K_p x = 0$$

Choosing $K_p$ and $K_v$ as diagonal matrices with entries $k_{pi}, k_{v_i}$, this becomes a series of decoupled differential equations:

$$\begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \vdots \\ \ddot{x}_n \end{pmatrix} + \begin{pmatrix} k_{v_1} & 0 & \dots & 0 \\ 0 & k_{v_2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & k_{v_n} \end{pmatrix} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{pmatrix} + \begin{pmatrix} k_{p_1} & 0 & \dots & 0 \\ 0 & k_{p_2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & k_{p_n} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \Leftrightarrow$$

$$\begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \vdots \\ \ddot{x}_n \end{pmatrix} + \begin{pmatrix} k_{v_1}\dot{x}_1 \\ k_{v_2}\dot{x}_2 \\ \vdots \\ k_{v_n}\dot{x}_n \end{pmatrix} + \begin{pmatrix} k_{p_1}x_1 \\ k_{p_2}x_2 \\ \vdots \\ k_{p_n}x_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Achieving critical damping of this system is extremely easy: Set $k_{vi} = 2\sqrt{k_{pi}}$, and we're done!

## 7.2 Manipulator Control

By using either the Newton-Euler or the Lagrange method, we are already able to determine equations of motion of a robot. The equations, formulated in state-space form, generally look like this:

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta)$$

This is a multi-dimensional system, but it's not even linear. It turns out that dealing with a nonlinear system is no problem at all when applying the partitioning scheme. Partitioning works exactly like before (let $\alpha = M(\Theta), \beta =$"everything else") and leads to an easily controllable system. Applying the well-known partitioning scheme

$$\tau = \alpha\tau' + \beta$$

with $\alpha = M(\Theta), \beta = V(\Theta, \dot{\Theta}) + G(\Theta)$ to our system, we can use the servo control law

$$\tau' = \ddot{\Theta}_d + K_v\left(\dot{\Theta}_d - \dot{\Theta}\right) + K_p\left(\Theta_d - \Theta\right)$$

to control our robot. Here, $\Theta_d$ is the vector of desired joint positions. The expression $(\Theta_d - \Theta)$ will now be abbreviated as $E$. Inserting above values into the state space equation, we obtain:

$$\begin{aligned} M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) &= M(\Theta)\left(\ddot{\Theta}_d + K_v\dot{E} + K_pE\right) + V(\Theta, \dot{\Theta}) + G(\Theta) \\ 0 &= M(\Theta)\left(\ddot{\Theta}_d - \ddot{\Theta} + K_v\dot{E} + K_pE\right) \\ 0 &= \ddot{E} + K_v\dot{E} + K_pE \end{aligned}$$

This is the so-called error equation. Again, we choose $K_v$ and $K_p$ to be diagonal:

$$K_v = \begin{pmatrix} k_{v1} & 0 & 0 & \cdots & 0 \\ 0 & k_{v2} & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & k_{vn} \end{pmatrix}, \quad K_p = \begin{pmatrix} k_{p1} & 0 & 0 & \cdots & 0 \\ 0 & k_{p2} & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & k_{pn} \end{pmatrix}$$

As before, we end up with a couple of independent error equations, and each one of those can be seen as a mass-spring model that we whish to damp critically. The equations would be

$$\ddot{e}_i + k_{v_i}\dot{e}_i + k_{p_i}e_i = 0.$$

When talking about natural frequencies in the context of manipulator control, we are referring to the natural frequency associated with those error equations. This means that the following simple relationship holds:

$$\omega_{ni} = \sqrt{k_{pi}}.$$

Finally, critical damping can be achieved as usual by letting

$$k_{vi} = 2\sqrt{k_{pi}}$$

# Chapter 8: Formula Cheat Sheet

Cross product: $\mathbf{a}\times\mathbf{b} = \begin{bmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{bmatrix}$

Trigonometric Identities:
$$\sin(\alpha)^2 + \cos(\alpha)^2 = 1$$
$$\sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \cos\alpha\sin\beta$$
$$\cos(\alpha \pm \beta) = \cos\alpha\cos\beta \mp \sin\alpha\sin\beta$$

Determinant:

$$|A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$$

$$|A| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a\begin{vmatrix} e & f \\ h & i \end{vmatrix} - b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix}$$
$$= aei + bfg + cdh - ceg - bdi - afh$$

A manipulator may have special configurations, called "isotropic points", that are characterized by the Jacobi matrix having orthogonal columns of equal length, thus $J^TJ = \delta I$ for some $\delta \in \mathbb{R}$.

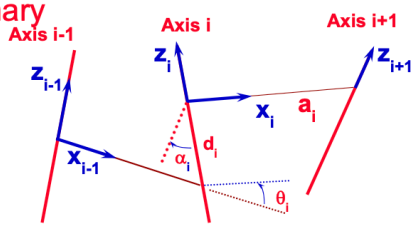
## 8.1 Denavit-Hartenberg Parameters

DH-table (link-index $i$):

| $i$ | $a_{i-1}$ | $\alpha_{i-1}$ | $d_i$ | $\theta_i$ |
|---|---|---|---|---|
| 1 | ... | ... | ... | ... |
| ... | ... | ... | ... | ... |

1. Shift $Z_{i-1}$ by $a_{i-1}$ along $X_{i-1}$.
2. Rot. $Z_{i-1}$ by $\alpha_{i-1}$ about $X_{i-1}$ & shift by $d_i$ along $Z_i$.
3. Rot. $X_{i-1}$ by $\theta_i$ about $Z_i$ & move it to $Z_i$.

Summary




$a_i$ = *the distance from* $\hat{Z}_i$ *to* $\hat{Z}_{i+1}$ *measured along* $\hat{X}_i$;
$\alpha_i$ = *the angle from* $\hat{Z}_i$ *to* $\hat{Z}_{i+1}$ *measured about* $\hat{X}_i$;
$d_i$ = *the distance from* $\hat{X}_{i-1}$ *to* $\hat{X}_i$ *measured along* $\hat{Z}_i$; and
$\theta_i$ = *the angle from* $\hat{X}_{i-1}$ *to* $\hat{X}_i$ *measured about* $\hat{Z}_i$.

Homogeneous transformation from link $i$ to $i-1$:
$$\overset{i-1}{i}T = \left[\begin{array}{ccc|c} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i\ c\alpha_{i-1} & c\theta_i\ c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}\ d_i \\ s\theta_i\ s\alpha_{i-1} & c\theta_i\ s\alpha_{i-1} & c\alpha_{i-1} & d_i\ c\alpha_{i-1} \\ \hline 0 & 0 & 0 & 1 \end{array}\right].$$

Inverse of the homogeneous transform:
$$\overset{A}{B}T^{-1} = \overset{B}{A}T = \left[\begin{array}{ccc|c} & \overset{A}{B}R^T & & -\overset{A}{B}R^T\,{}^AP_{Borg} \\ \hline 0 & 0 & 0 & 1 \end{array}\right].$$

## 8.2 Jacobian

*Singularity*: the end-effector locally looses at least 1 DOF. This happens, when $Z$-axes are aligned $\Leftrightarrow$ the Jacobian does not have full rank $\Leftrightarrow$ $det(J) = 0$. Small end-effector motions require large joint motions near singularities.

### 8.2.1 Velocity Propagation

Linear and angular velocities at joint $i+1$ (with scalar $\dot{d}_{i+1}$ or $\dot{\Theta}_{i+1}$ for prismatic/ revolute joints):

$${}^{i+1}\omega_{i+1} = \overset{i+1}{i}R \cdot {}^i\omega_i + \dot{\Theta}_{i+1} \cdot {}^{i+1}\hat{Z}_{i+1}$$
$${}^{i+1}v_{i+1} = \overset{i+1}{i}R({}^iv_i + {}^i\omega_i \times {}^iP_{i+1}) + \dot{d}_{i+1} \cdot {}^{i+1}\hat{Z}_{i+1}$$

Then read off the Jacobian from:
$$\begin{bmatrix} \dot{x}_P \\ \dot{x}_R \end{bmatrix}_{6\times1} = \begin{bmatrix} J_{x_P} \\ J_{x_R} \end{bmatrix}_{6\times n} \dot{\boldsymbol{\Theta}} = J_{6\times n} \begin{bmatrix} \dot{\Theta}_1 \\ ... \\ \dot{\Theta}_n \end{bmatrix}_{n\times1}.$$

### 8.2.2 Force/ Torque Propagation

Force $f$ and torque $n$ at joint $i$:

$$
\begin{aligned}
{}^{i}f_i &= {}^{i}_{i+1}R \cdot {}^{i+1}f_{i+1} \\
{}^{i}n_i &= {}^{i}_{i+1}R \cdot {}^{i+1}n_{i+1} + {}^{i}P_{i+1} \times {}^{i}f_i
\end{aligned}
$$

The force/ torque exerted on the $i$th joint:

$$
\tau_i = {}^{i}f_i^{\mathrm{T}\,i}Z_i = {}^{i}f_i^{\mathrm{T}} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \qquad \tau_i = {}^{i}n_i^{\mathrm{T}\,i}Z_i = {}^{i}n_i^{\mathrm{T}} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}
$$

Then read off the Jacobian from:

$$
\begin{pmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_n \end{pmatrix} = \tau = {}^{A}J^{T\,A}\mathcal{F} = {}^{A}J^{T} \begin{pmatrix} {}^{A}f \\ {}^{A}n \end{pmatrix},
$$

where $\mathcal{F}$ is a $6 \times 1$ force-torque vector in frame $\{A\}$.

### 8.2.3 Explicit Form

Jacobian in frame $\{0\}$:

$$
{}^{0}J = \begin{bmatrix} \frac{\partial}{\partial q_1}\left({}^{0}x_P\right) & \frac{\partial}{\partial q_2}\left({}^{0}x_P\right) & \cdots & \frac{\partial}{\partial q_n}\left({}^{0}x_P\right) \\ \bar{\epsilon}_1 \cdot \left({}^{0}_{1}R \cdot Z\right) & \bar{\epsilon}_2 \cdot \left({}^{0}_{2}R \cdot Z\right) & \cdots & \bar{\epsilon}_n \cdot \left({}^{0}_{n}R \cdot Z\right] \end{bmatrix}
$$

with ${}^{0}Z_i = {}^{0}_{i}R\,{}^{i}Z_i$; ${}^{i}Z_i = Z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$. The indicator variable $\bar{\epsilon}_i$ is 1 if joint $i$ is revolute, otherwise it is 0. Then, rotate ${}^{0}J$ into the required reference frame:

$$
{}^{A}J(\Theta) = \begin{pmatrix} {}^{A}_{B}R & \mathbf{0} \\ \mathbf{0} & {}^{A}_{B}R \end{pmatrix} {}^{B}J(\Theta)
$$

## 8.3 Newton-Euler Method

Propagate lin. and ang. velocities and accelerations foward (account for gravity by setting ${}^{i}\dot{v}_i = -G$):

| | |
|---|---|
| Rotational joint $i+1$ | $\begin{aligned} {}^{i+1}\omega_{i+1} &= {}^{i+1}_{i}R \cdot {}^{i}\omega_i + \dot{\Theta}_{i+1} \cdot {}^{i+1}Z_{i+1} \\ {}^{i+1}\dot{\omega}_{i+1} &= {}^{i+1}_{i}R \cdot {}^{i}\dot{\omega}_i + {}^{i+1}_{i}R \cdot {}^{i}\omega_i \times \dot{\Theta}_{i+1} \cdot {}^{i+1}Z_{i+1} + \ddot{\Theta}_{i+1}\,{}^{i+1}Z_{i+1} \\ {}^{i+1}\dot{v}_{i+1} &= {}^{i+1}_{i}R \left({}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times \left({}^{i}\omega_i \times {}^{i}P_{i+1}\right) + {}^{i}\dot{v}_i\right) \end{aligned}$ |
| Prismatic joint $i+1$ | $\begin{aligned} {}^{i+1}\omega_{i+1} &= {}^{i+1}_{i}R \cdot {}^{i}\omega_i \\ {}^{i+1}\dot{\omega}_{i+1} &= {}^{i+1}_{i}R \cdot {}^{i}\dot{\omega}_i \\ {}^{i+1}\dot{v}_{i+1} &= {}^{i+1}_{i}R \left({}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times \left({}^{i}\omega_i \times {}^{i}P_{i+1}\right) + {}^{i}\dot{v}_i\right) \\ &\quad + 2 \cdot {}^{i+1}\omega_{i+1} \times \dot{d}_{i+1}\,{}^{i+1}Z_{i+1} + \ddot{d}_{i+1}\,{}^{i+1}Z_{i+1} \end{aligned}$ |

At the center of mass we have (revolute/ prismatic joints): ${}^{i}\dot{v}_{C_i} = {}^{i}\dot{\omega}_i \times {}^{i}P_{C_i} + {}^{i}\omega_i \times \left({}^{i}\omega_i \times {}^{i}P_{C_i}\right) + {}^{i}\dot{v}_i$.
Next, compute the forces and torques at the center of mass of each link (applied by the motion):

$$
\begin{aligned}
{}^{i}F_i &= m_i \cdot {}^{i}\dot{v}_{C_i} \\
{}^{i}N_i &= {}^{C_i}I_i \cdot {}^{i}\dot{\omega}_i + {}^{i}\omega_i \times {}^{C_i}I_i \cdot {}^{i}\omega_i
\end{aligned}
$$

Propagate forces and moments $f_i$ and $n_i$ backward:

$$
\begin{aligned}
{}^{i}f_i &= {}^{i}_{i+1}R \cdot {}^{i+1}f_{i+1} + {}^{i}F_i \\
{}^{i}n_i &= {}^{i}N_i + {}^{i}_{i+1}R \cdot {}^{i+1}n_{i+1} + {}^{i}P_{C_i} \times {}^{i}F_i + {}^{i}P_{i+1} \times {}^{i}_{i+1}R^{i+1}f_{i+1}
\end{aligned}
$$

The values of $\tau_i$, compute as $\tau_i = {}^{i}n_i^{T} \cdot {}^{i}Z_i$ for a revolute joint and $\tau_i = {}^{i}f_i^{T} \cdot {}^{i}Z_i$ for a prismatic joint.

### 8.3.1 Parallel-Axes Theorem

How the inertia tensor changes under *translations* of the reference coordinate system (s.t. the axes remain parallel). It relates the inertia tensor w.r.t. the center of mass ${}^{C}I$ to the inertia tensor w.r.t. another reference frame ${}^{A}I$:

$$ {}^{A}I = {}^{C}I + m[P_c^T P_c I_3 - P_c P_c^T] $$

where $P_c = [x_c, y_c, z_c]^T$ locates the center of mass relative to $A$. $I_3$ is the identity matrix.

## 8.4 Lagrange Method

Kinetic energy of link $i$ (requires ${}^{0}v_{C_i}$ (from ${}^{0}P_{C_i}$) and $\omega_i$):

$$ k_i = \underbrace{\frac{1}{2} m_i v_{C_i}^{\mathrm{T}} \cdot v_{C_i}}_{\text{transl. energy of } C_i} + \underbrace{\frac{1}{2} {}^{i}\omega_i^{\mathrm{T}} \cdot {}^{C_i}I_i \cdot {}^{i}\omega_i}_{\text{rot. energy about } C_i} \qquad \text{or alternatively} \qquad k(\Theta, \dot{\Theta}) = \frac{1}{2} \dot{\Theta}^{\mathrm{T}} M(\Theta) \dot{\Theta} $$

where ${}^{C_i}I_i$ is the inertia of link $i$ in $\{C_i\}$. Then compute the total kinetic energy, $k = \sum_{i=1}^{n} k_i$. Potential energy of link $i$:

$$ u_i = -m_i \cdot {}^{0}g^{\mathrm{T}} \cdot {}^{0}P_{C_i} + u_{\mathrm{ref}_i} $$

Finally, use the Lagrangian $L = k - u$ to compute the joint torques $\tau$:

$$ \tau = \frac{d}{dt}\frac{\partial L}{\partial \dot{\Theta}} - \frac{\partial L}{\partial \Theta} = \frac{d}{dt}\frac{\partial k}{\partial \dot{\Theta}} - \frac{\partial k}{\partial \Theta} + \frac{\partial u}{\partial \Theta} \qquad \text{or per-joint:} \qquad \tau_i = \frac{d}{dt}\frac{\partial k}{\partial \dot{\Theta}_i} - \frac{\partial k}{\partial \Theta_i} + \frac{\partial u}{\partial \Theta_i} $$

### 8.4.1 M-V-G-form (State-Space-form)

$$ \tau = \underbrace{M(\Theta)}_{\substack{n \times n \text{ matrix} \\ \text{All coefficients of } \ddot{\Theta}}} \ddot{\Theta} + \underbrace{V(\Theta, \dot{\Theta})}_{\substack{n \times 1 \text{ vector} \\ \text{All summands with } \dot{\Theta}}} + \underbrace{G(\Theta)}_{\substack{n \times 1 \text{ vector} \\ \text{All summands with } g}} $$

$V$ can be decomposed into $B$ and $C$, yielding the M-B-C-G-form (configuration-space equation):

$$ \tau = M(\Theta)\ddot{\Theta} + \underbrace{B(\Theta)}_{n \times \frac{n(n-1)}{2} \text{ matrix}} \overbrace{\left[\dot{\Theta}\dot{\Theta}\right]}^{=\left(\dot{\Theta}_1\dot{\Theta}_2, \dot{\Theta}_1\dot{\Theta}_3, \ldots, \dot{\Theta}_{n-1}\dot{\Theta}_n\right)^T} + \underbrace{C(\Theta)}_{n \times n \text{ matrix}} \overbrace{\left[\dot{\Theta}^2\right]}^{=\left(\dot{\Theta}_1^2, \dot{\Theta}_2^2, \ldots, \dot{\Theta}_n^2\right)^{\mathrm{T}}} + G(\Theta) $$

The matrices $B$ and $C$ can be determined by finding the coefficients of $\Theta_i\Theta_j$ and $\Theta_i^2$, respectively.

## 8.5 Control

### 8.5.1 Mass-Spring-System

PD-control force for critical damping

Open-loop equation: $m\ddot{x} + b\dot{x} + kx = f \quad \overbrace{= -k_p x - k_v \dot{x}}$

Closed-loop equation: $m\ddot{x} + (b + k_v)\,\dot{x} + (k + k_p)\,x = 0$

Solve the characteristic equation to determine $x(t)$: $ms^2 + (b + k_v)\,s + (k + k_p) = 0$

$$ s_{1,2} = \frac{-b' \pm \sqrt{b'^2 - 4mk'}}{2m} \quad \Rightarrow \quad x(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t} \underbrace{=}_{\text{if } s_{1,2} = \lambda \pm \mu i} e^{\lambda t}\left(c_1 \cos(\mu t) + c_2 \sin(\mu t)\right) $$

If $s_1 = s_2 \in \mathbb{R}$ the system is critically damped ($\Rightarrow b'^2 - 4mk' = 0$ or $b' = 2\sqrt{mk'}$, where $b', k' > 0$). For an oscillating system, the equations can also be stated as:

$$ s^2 + 2\zeta\omega_n s + \omega_n^2 = 0 \qquad \text{with } \textit{damping ratio } \zeta = \frac{b'}{2\sqrt{k'm}} \text{ and } \textit{natural frequency } \omega_n = \sqrt{\frac{k'}{m}} $$

### 8.5.2 PD Control

*Control law partitioning*: separate model dependant parameters like mass, friction, gravitation from the ideal unit mass system: $\tau = \alpha\tau' + \beta$. The system appears as a unit mass system to the controller.

| | **Mass-Spring-System** | **Multi-Body-System** |
|---|---|---|
| Control law partitioning | Into system dependant & servo part:<br>$m\ddot{x} + b\dot{x} + kx = f = \alpha f' + \beta.$<br>with $\alpha = m$ and $\beta = b\dot{x} + kx$. Unit mass system: $\ddot{x} = \tau'$. | $M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) = \tau = \alpha\tau' + \beta$<br>with $\alpha = M(\Theta)$ and $\beta = V(\Theta, \dot{\Theta}) + G(\Theta)$. Unit mass system if $M^{-1}$ exists: $\ddot{\Theta} = \tau'$. |
| Position control | Control law & equation of motion:<br>$f' = -k_v\dot{x} - k_p x$<br>$\ddot{x} + k_p x + k_v\dot{x} = 0$ | $\tau' = -K_v\dot{\Theta} - K_p\Theta$<br>$\ddot{\Theta} + K_v\dot{\Theta} + K_p\Theta = 0$ |
| Trajectory following | Control law & equation of motion:<br>$f' = \ddot{x}_d + k_v\dot{e} + k_p e$<br>$\ddot{x} = \ddot{x}_d + k_v\dot{e} + k_p e$<br>$\Leftrightarrow 0 = \ddot{e} + k_v\dot{e} + k_p e$ | $\tau' = \ddot{\Theta}_d + K_v\left(\dot{\Theta}_d - \dot{\Theta}\right) + K_p\left(\Theta_d - \Theta\right)$<br>$= \ddot{\Theta}_d + K_v\dot{E} + K_p E$<br>where $\Theta_d$ is the vector of desired joint positions. Insert $\tau'$ for the error equation:<br>$\tau = M(\Theta)\left(\ddot{\Theta}_d + K_v\dot{E} + K_p E\right) + V(\Theta, \dot{\Theta}) + G(\Theta)$<br>$\Leftrightarrow 0 = M(\Theta)\left(\ddot{\Theta}_d - \ddot{\Theta} + K_v\dot{E} + K_p E\right)$<br>$\Leftrightarrow 0 = \ddot{E} + K_v\dot{E} + K_p E$<br>with diagonal $K_v$ and $K_p$. |
| Natural frequency & critical damping | Max. freq.: $\omega_n = \sqrt{k_p} \leq 0.5\omega_{\text{res}}$<br>Critical damping for: $k_v = 2\sqrt{k_p}$ | $\omega_{ni} = \sqrt{k_{pi}}$<br>$k_{vi} = 2\sqrt{k_{pi}}$ |

### 8.5.3 Controller Block Diagram

General form (use the expressions for $\alpha$ and $\beta$). Complete the diagram according to the equation for $\tau'$.