LECTURES 1 & 2

Review of robotics

Paolo Rocco

Agenda

- Introduction
- Robot kinematics
- Robot dynamics
- Motion planning
- Motion control

Agenda

- **Introduction**
- Robot kinematics
- Robot dynamics
- Motion planning
- Motion control

What is a robot?

A reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through various programmed motions for the performance of a variety of tasks.

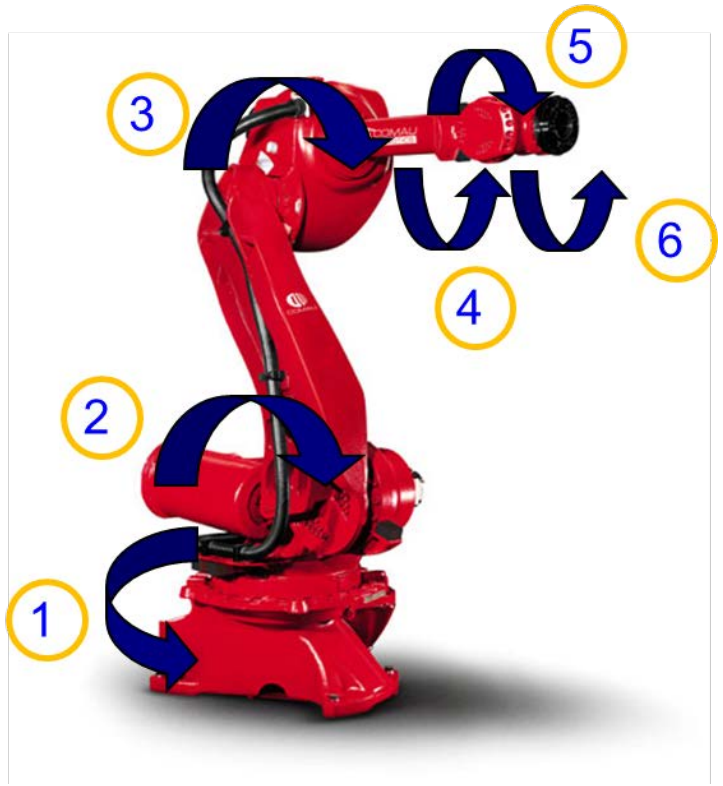
(Robot Institute of America, 1980)

The robot is not just a mechanical device…

COMAU SMART NH3

A robot and its control unit

Mechanics



Intelligence

The mechanical system

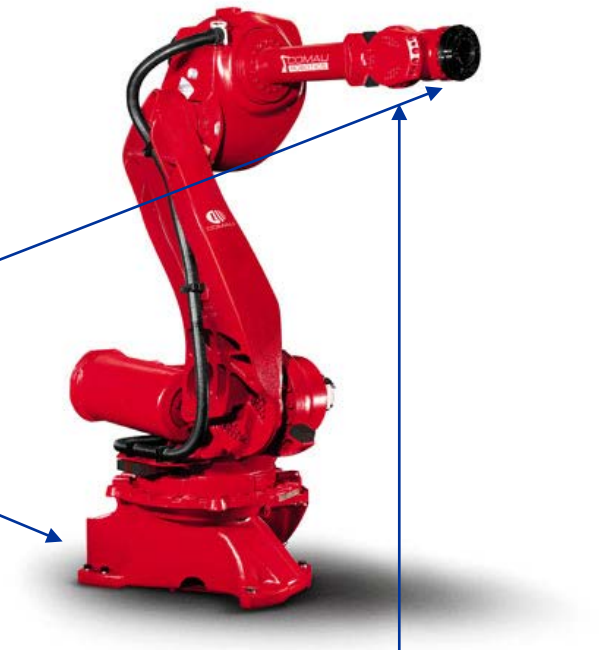
The manipulator consists of a series of rigid bodies (links) connected by joints.

One end of this chain makes the BASE, usually fixed to the floor.

At the other end we have the END EFFECTOR where the gripper or tool is mounted.

Usually manipulators have six links:

- the first three make the positioning
- the last three (WRIST) make the orientation



Common robot configurations (1/2)

Anthropomorphic

- The typical structure of the robot manipulator
- Dexterous structure
- Mechanical stiffness is a function of configuration

SCARA

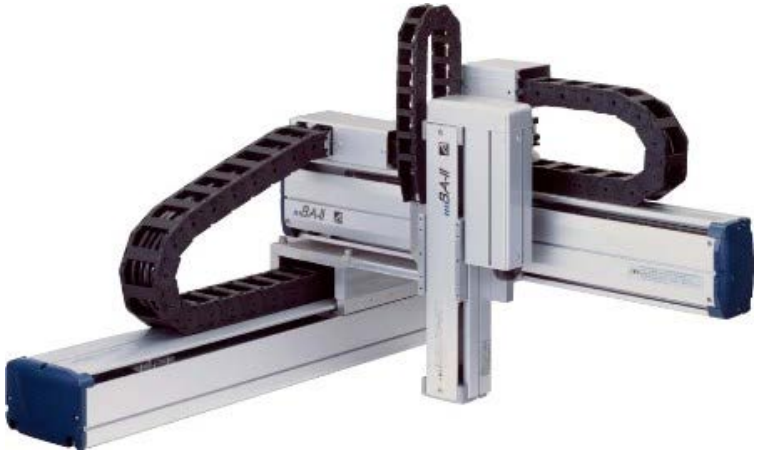
- All joints with vertical axes
- Very rigid to vertical loads, compliant to horizontal loads

Common robot configurations (2/2)

Delta

- Parallel kinematic structure
- Very fast and accurate
- Limited workspace

Cartesian



- All joints give linear motion
- Very rigid mechanically

New robots: dual arm robots

EPSON dual-arm

ABB YuMi

KAWADA HIRO

KAWASAKI DUARO

Automation in industry

Rigid automation

- The sequence of operations is fixed
- Production process composed of a sequence of simple operations
- Large production with very small variations

Pneumatic/electrical actuation

Programmable automation

- The sequence of operations can be changed
- Medium-low production batches
- Between batches the production plant has to be reconfigured

PLC

Flexible automation

- Production can be varied without idle times for conversion
- Machine characterized by high flexibility and configurability (FMS: Flexible Manufacturing Systems)

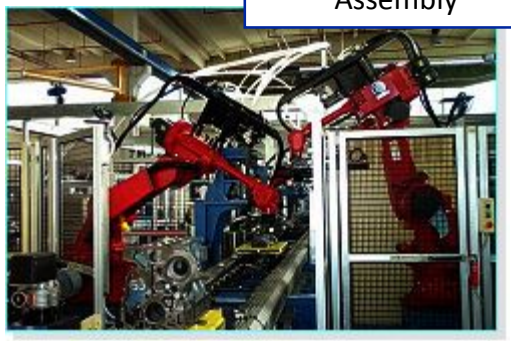
Robot

Typical applications in industrial robotics

Spot welding

Arc welding

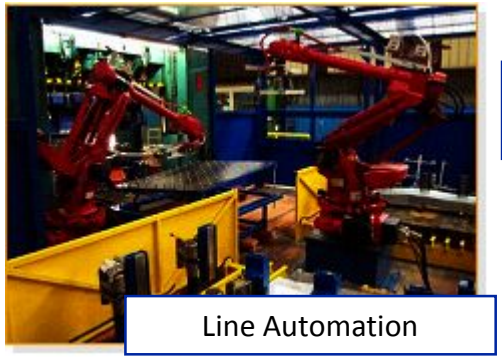
Assembly



Loading – Unloading
Palletizing

Manipulation

Gluing - Sealing



Line Automation

Machining

Source: Comau Robotics

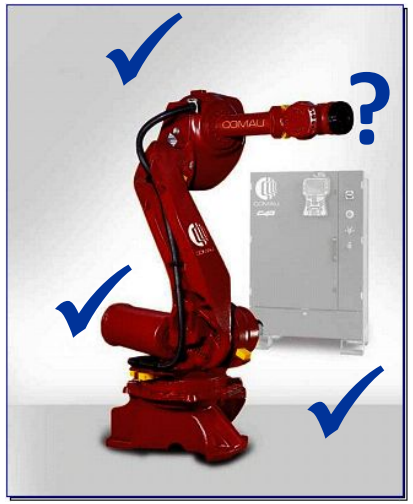
Agenda

- Introduction
- **Robot kinematics**
- Robot dynamics
- Motion planning
- Motion control

Direct and inverse kinematics

Robot kinematics is the study of the motion of the robot.
Two problems are of interest:

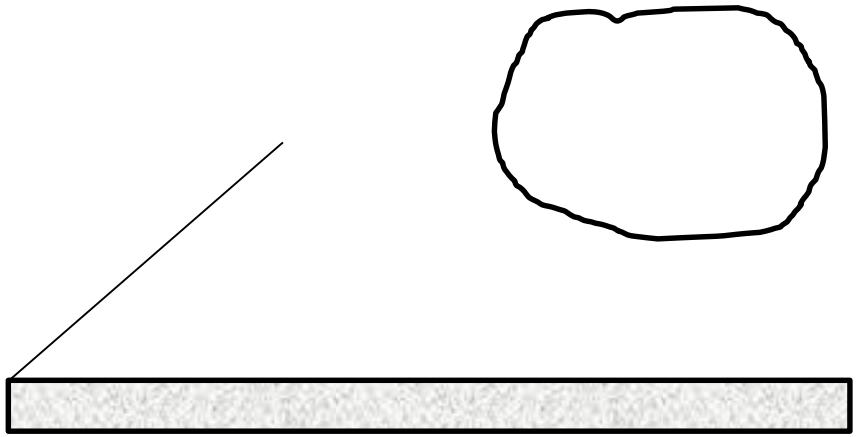
Direct kinematics



Inverse kinematics

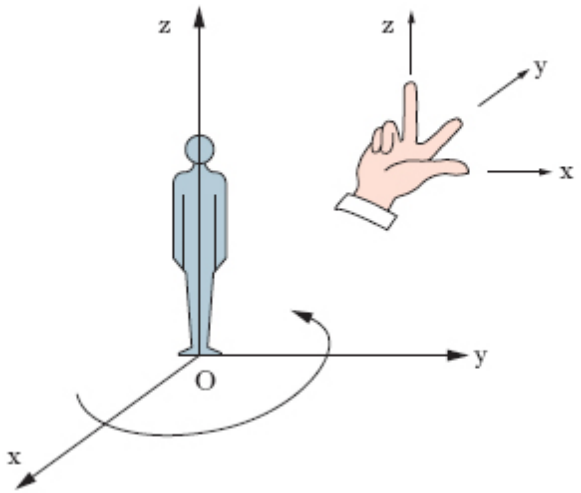
Position and orientation of a body in space

Let us consider a rigid body in space:



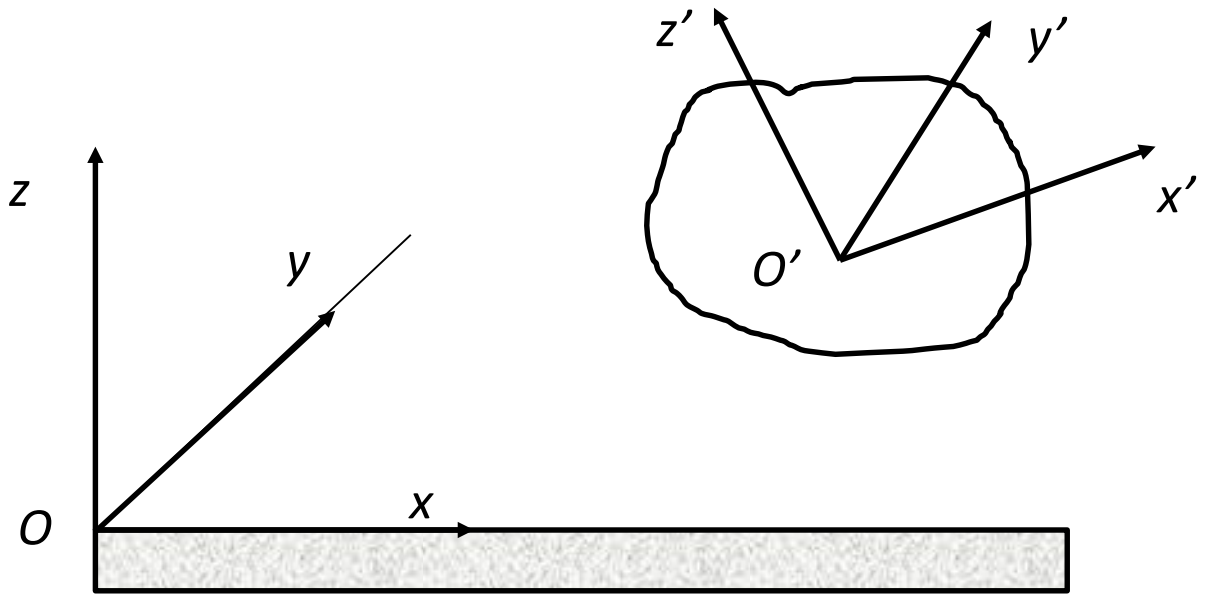
How can we characterize the position and orientation of the body in space?

The study of kinematics of mechanical bodies is facilitated if Cartesian frames are introduced. Each point in space has 3 coordinates (*x*, *y*, *z*) in the Cartesian frame.



Position and orientation of a body in space

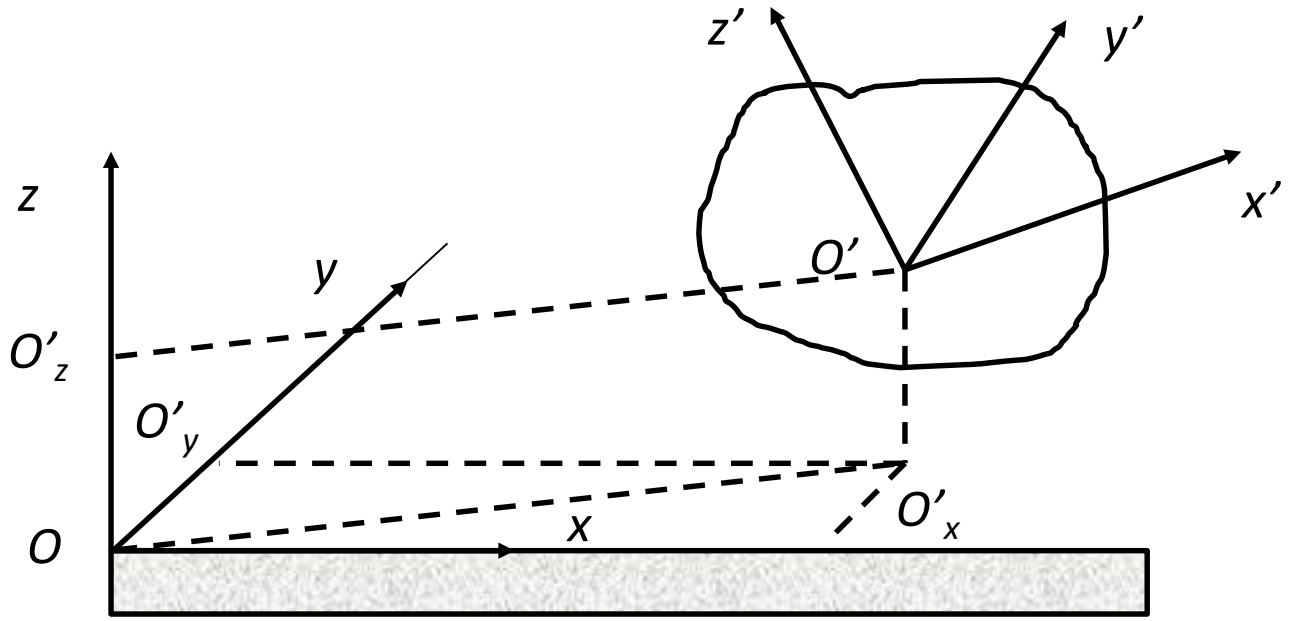
The best thing to do is to consider a reference frame and to attach a second frame to the body.



The problem is now how to characterize the position and orientation of a frame with respect to another one.

Position and orientation of a body in space

The representation of the position is just made with the components of the origin of the body-attached frame with respect to the reference frame:

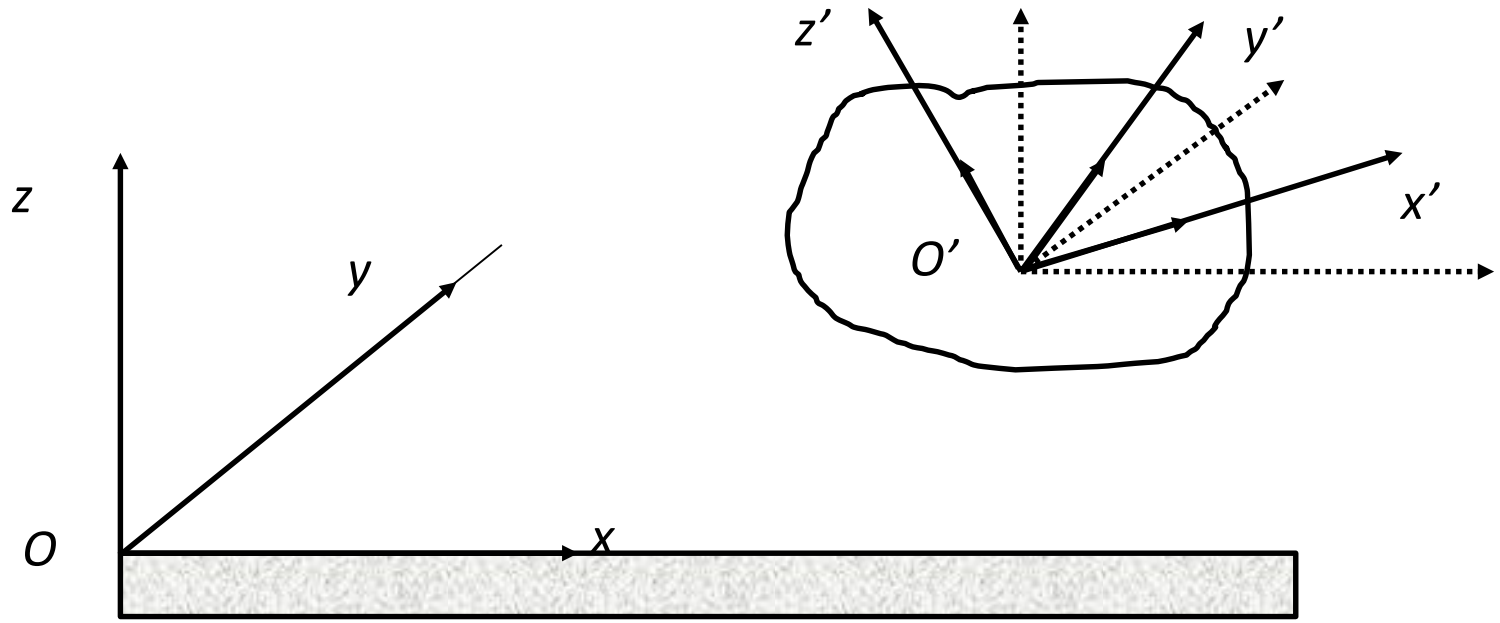


The three components can be conveniently gathered in a **vector**:

$$\boldsymbol{O}' = \begin{bmatrix} o'_x \\ o'_y \\ o'_z \end{bmatrix}$$

Position and orientation of a body in space

The representation of the orientation can be made considering unit length vectors along the axes of the rotated frame and evaluating their components in the reference frame:



We obtain three vectors:

$$\boldsymbol{x}' = \begin{bmatrix} x'_x \\ x'_y \\ x'_z \end{bmatrix}, \quad \boldsymbol{y}' = \begin{bmatrix} y'_x \\ y'_y \\ y'_z \end{bmatrix}, \quad \boldsymbol{z}' = \begin{bmatrix} z'_x \\ z'_y \\ z'_z \end{bmatrix}$$

Rotation matrix

We can gather the elements of $\boldsymbol{x}'$, $\boldsymbol{y}'$, $\boldsymbol{z}'$ in a matrix:

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{x}' & \boldsymbol{y}' & \boldsymbol{z}' \end{bmatrix} = \begin{bmatrix} x'_x & y'_x & z'_x \\ x'_y & y'_y & z'_y \\ x'_z & y'_z & z'_z \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}'^T\boldsymbol{x} & \boldsymbol{y}'^T\boldsymbol{x} & \boldsymbol{z}'^T\boldsymbol{x} \\ \boldsymbol{x}'^T\boldsymbol{y} & \boldsymbol{y}'^T\boldsymbol{y} & \boldsymbol{z}'^T\boldsymbol{y} \\ \boldsymbol{x}'^T\boldsymbol{z} & \boldsymbol{y}'^T\boldsymbol{z} & \boldsymbol{z}'^T\boldsymbol{z} \end{bmatrix}$$

This matrix is called **rotation matrix** of the frame ($\boldsymbol{x}'$, $\boldsymbol{y}'$, $\boldsymbol{z}'$) with respect to the frame ($\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$) .
Since the following relations hold:

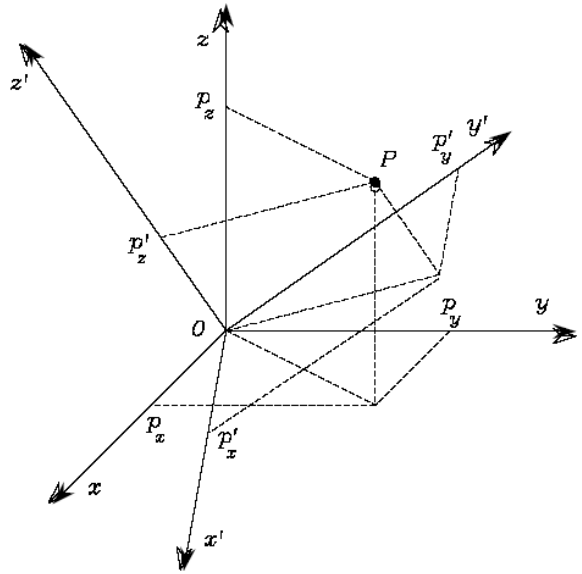
$$\boldsymbol{x}'^T\boldsymbol{x}' = 1, \quad \boldsymbol{y}'^T\boldsymbol{y}' = 1, \quad \boldsymbol{z}'^T\boldsymbol{z}' = 1$$

$$\boldsymbol{x}'^T\boldsymbol{y}' = 0, \quad \boldsymbol{y}'^T\boldsymbol{z}' = 0, \quad \boldsymbol{z}'^T\boldsymbol{x}' = 0$$

we have: $\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I} \quad \left(\boldsymbol{R}^T = \boldsymbol{R}^{-1}\right)$ orthogonal matrix

Representation of a vector

Consider now a point *P* whose coordinates are expressed in two reference frames:



The coordinates of the same point in the two frames are:

$$\boldsymbol{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}, \quad \boldsymbol{p}' = \begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix}$$

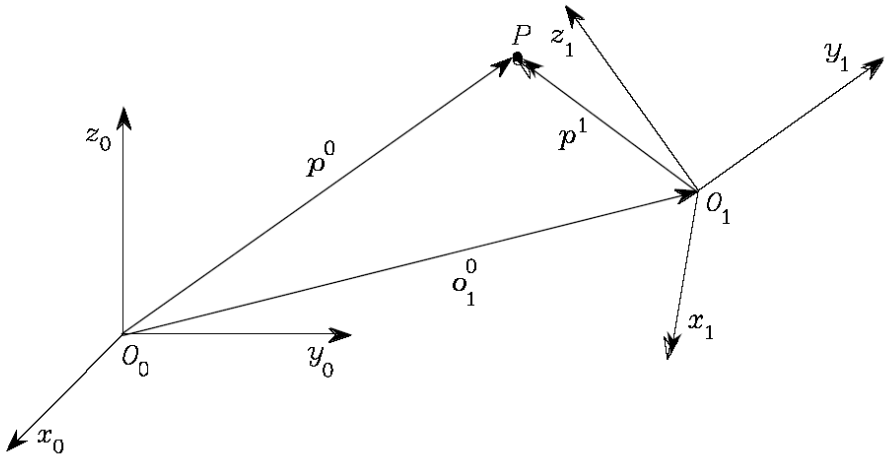
Therefore:

$$\boldsymbol{p} = p'_x \boldsymbol{x}' + p'_y \boldsymbol{y}' + p'_z \boldsymbol{z}' = \begin{bmatrix} \boldsymbol{x}' & \boldsymbol{y}' & \boldsymbol{z}' \end{bmatrix} \boldsymbol{p}' = \boldsymbol{R}\boldsymbol{p}'$$

The rotation matrix thus contains the transformation which maps the coordinates expressed in the frame (***x***', ***y*** ', ***z*** ') into the coordinates expressed in frame (***x***, ***y***, ***z***).

Inverse transformation: $\boldsymbol{p}' = \boldsymbol{R}^T \boldsymbol{p}$

Homogeneous representation



How can we express coordinates of point P in frame 0, based on its coordinates in frame 1?

$$\boldsymbol{p}^0 = \boldsymbol{o}_1^0 + \boldsymbol{R}_1^0 \boldsymbol{p}^1$$

Rotation matrix of frame 1 w.r.t. frame 0

Inverse transform:

$$\boldsymbol{p}^1 = -\boldsymbol{R}_0^1 \boldsymbol{o}_1^0 + \boldsymbol{R}_0^1 \boldsymbol{p}^0$$

In order to represent in a compact form these transformations, it is advisable to introduce a 4-dim vector:

$$\tilde{\boldsymbol{p}} = \begin{bmatrix} w\boldsymbol{p} \\ w \end{bmatrix}$$

Homogeneous representation

w is a scale factor which is always set to 1 in robotics (it is used in computer graphics)

Homogeneous transformations

Let us introduce the homogeneous transformation matrix (dimensions 4×4): $\boldsymbol{A}_1^0 = \begin{bmatrix} \boldsymbol{R}_1^0 & \boldsymbol{o}_1^0 \\ \boldsymbol{0}^T & 1 \end{bmatrix}$

The relationship:

$$\boldsymbol{p}^0 = \boldsymbol{o}_1^0 + \boldsymbol{R}_1^0 \boldsymbol{p}^1$$

can be expressed, in terms of homogeneous coordinates, as :

$$\tilde{\boldsymbol{p}}^0 = \boldsymbol{A}_1^0 \tilde{\boldsymbol{p}}^1$$

$\boldsymbol{A}_1{}^0$ relates the description (position/orientation) of a point on frame 1 with the description in frame 0.

The inverse transformation is:

$$\tilde{\boldsymbol{p}}^1 = \boldsymbol{A}_0^1 \tilde{\boldsymbol{p}}^0 = (\boldsymbol{A}_1^0)^{-1} \tilde{\boldsymbol{p}}^0 \qquad \boldsymbol{A}_0^1 = \begin{bmatrix} \boldsymbol{R}_0^1 & -\boldsymbol{R}_0^1 \boldsymbol{o}_1^0 \\ \boldsymbol{0}^T & 1 \end{bmatrix}$$

N.B. $\boldsymbol{A}$ is not orthogonal

Composing several transformations: $\tilde{\boldsymbol{p}}^0 = \boldsymbol{A}_1^0 \boldsymbol{A}_2^1 \ldots \boldsymbol{A}_n^{n-1} \tilde{\boldsymbol{p}}^n$

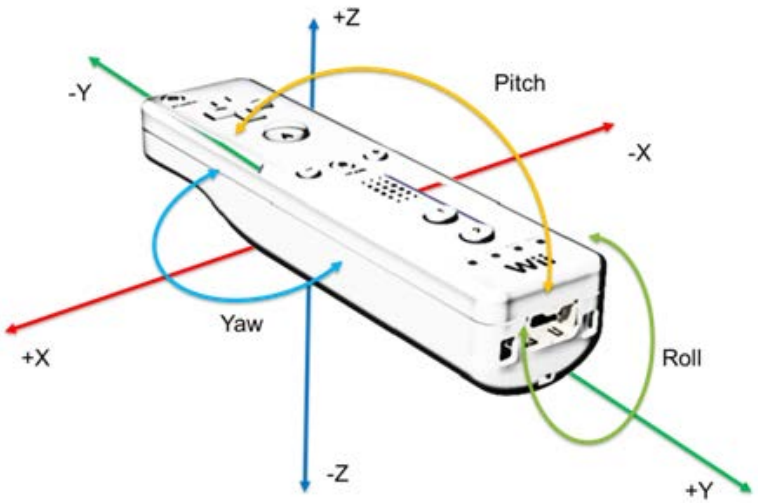
Minimal representation of the orientation

A rotation matrix represents the orientation of a frame with respect to another one by means of 9 parameters, among which 6 constraints exist.

In a minimal representation the orientation is described by means of 3 independent parameters.

Possible representations are:

- Euler angles (3 parameters)
- roll-pitch-yaw angles (3 parameters)
- axis/angle (4 parameters)
- quaternions (4 parameters)



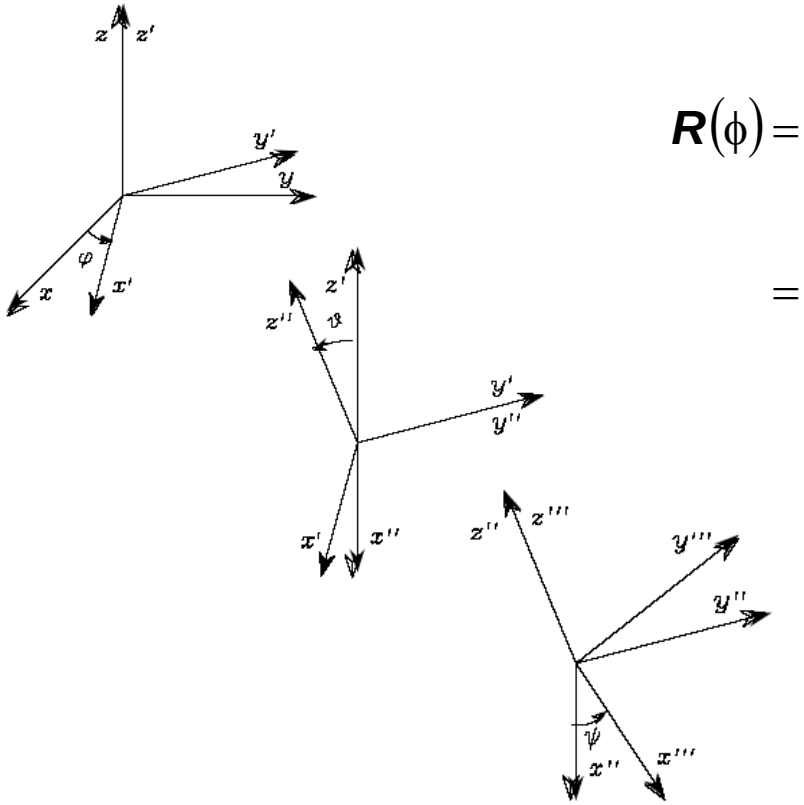
ZYZ Euler angles

With ZYZ Euler angles the sequence is composed as:

I) Rotation around Z (angle φ)

II) Rotation around Y' (angle ϑ)

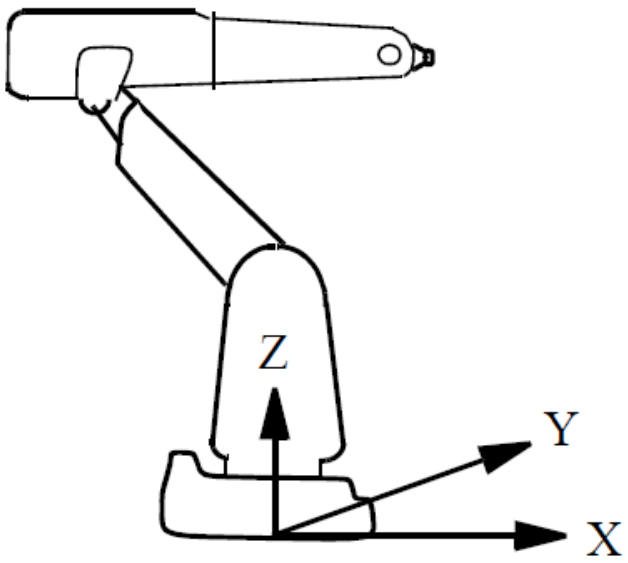
III) Rotation around Z'' (angle ψ)



$$\boldsymbol{R}(\phi) = \boldsymbol{R}_z(\varphi)\boldsymbol{R}_{y'}(\vartheta)\boldsymbol{R}_{z''}(\psi) =$$

$$= \begin{bmatrix} c_\varphi c_\vartheta c_\psi - s_\varphi s_\psi & -c_\varphi c_\vartheta s_\psi - s_\varphi c_\psi & c_\varphi s_\vartheta \\ s_\varphi c_\vartheta c_\psi + c_\varphi s_\psi & -s_\varphi c_\vartheta s_\psi + c_\varphi c_\psi & s_\varphi s_\vartheta \\ -s_\vartheta c_\psi & s_\vartheta s_\psi & c_\vartheta \end{bmatrix}$$

The base coordinate system



Source: ABB

The **base coordinate system** is located on the base of the robot:

- *The origin* is situated at the intersection of axis 1 (axis of the first joint) and the base mounting surface.
- *The xy plane* is the same as the base mounting surface.
- *The x-axis* points forwards.
- *The y-axis* points to the left (from the perspective of the robot).
- *The z-axis* points upwards.

The TCP (Tool Center Point)

The task the robot has to accomplish is usually expressed in terms of the TCP (**Tool Center Point**).

This point is normally defined as being "somewhere" on the tool.

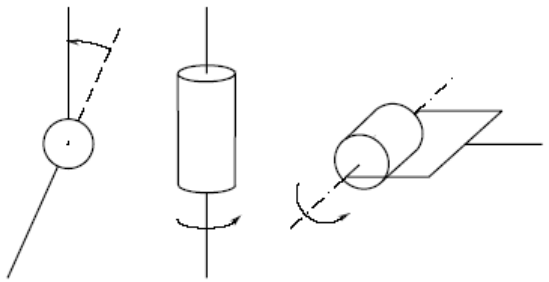
The TCP can be expressed in different coordinate systems (Cartesian frames).

The joints

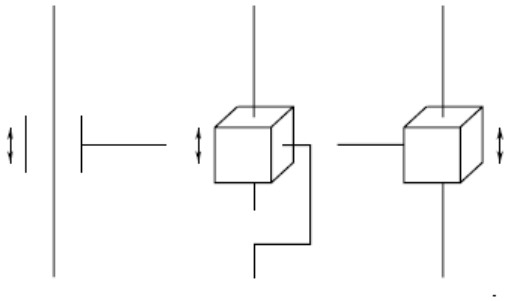
Each joint allows for one (and only one) *degree of freedom* between two links. We call joint variable the coordinate associated to such degree of freedom, and then we introduce the vector of joint variables:

$$\boldsymbol{q} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix}$$

Schematic draws of the joints:



ROTATIONAL JOINTS



PRISMATIC JOINTS

Joint space and operational space

The **joint space** is defined by the vector of joint variables:

$$\boldsymbol{q} = \begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix}$$

q_i = angle (rotating joint)

q_i = displacement (prismatic joint)

The **operational space** is the space where the task that the manipulator has to accomplish is specified. It is defined by the posture $\boldsymbol{x}$:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p} \\ \phi \end{bmatrix}$$

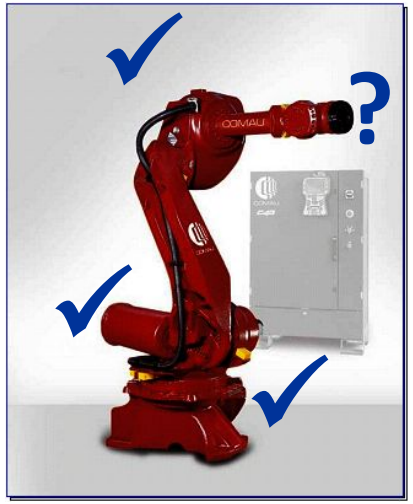
$\boldsymbol{p}$ (position)

ϕ (minimal representation of the orientation)

m components

Direct kinematic relation: $\boldsymbol{x} = \boldsymbol{k}(\boldsymbol{q})$

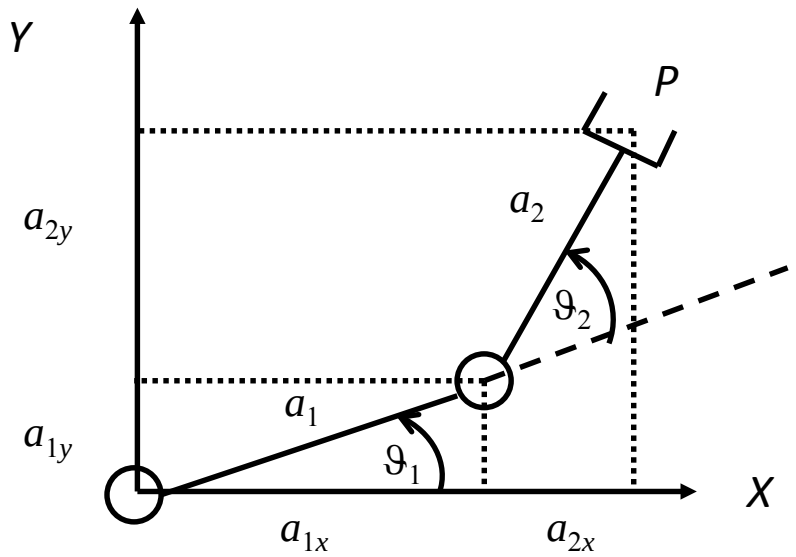
Direct kinematics



The direct kinematics problem is:

find position and orientation of the TCP frame w.r.t. the base frame, as a function of the joint variables.

For a two-link planar robot:



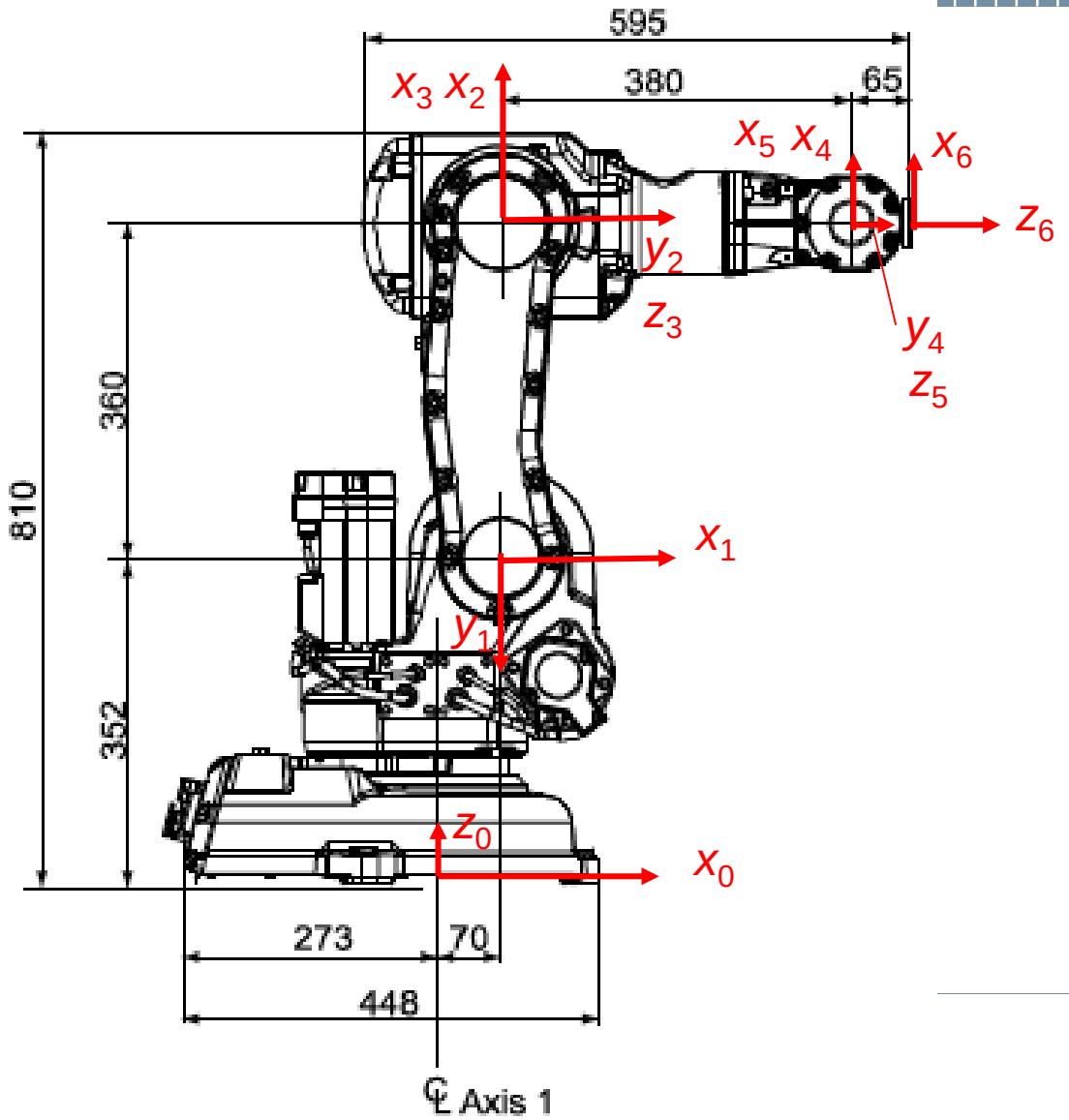
$$p_x = a_{1x} + a_{2x} = a_1 \cos(\vartheta_1) + a_2 \cos(\vartheta_1 + \vartheta_2)$$
$$p_y = a_{1y} + a_{2y} = a_1 \sin(\vartheta_1) + a_2 \sin(\vartheta_1 + \vartheta_2)$$

Direct kinematics

In the general case the direct kinematics computation is complicated.

We need to introduce several intermediate coordinate systems.

But there are **systematic ways** to solve it (Denavit-Hartenberg method).



Inverse kinematics problem

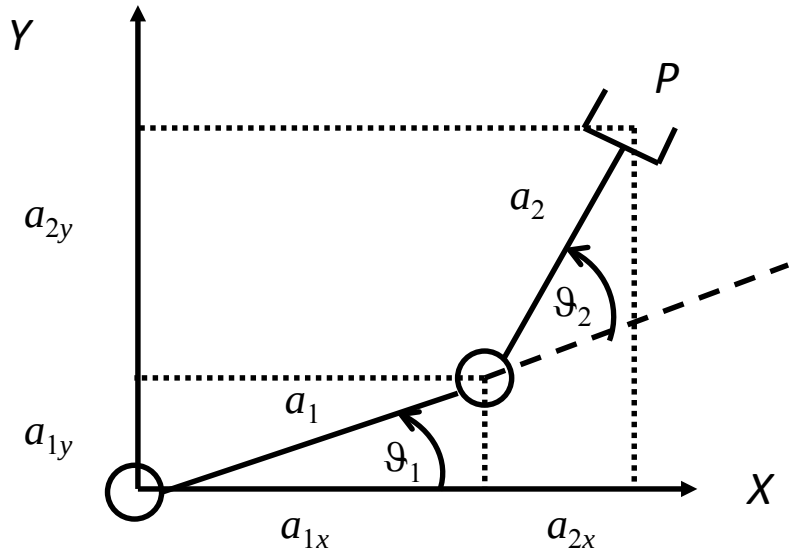
The inverse kinematics problem is:

find joint variables given position and orientation of the tool frame w.r.t. the base frame.

- The problem may admit no solutions (if position and orientation do not belong to the workspace of the manipulator)
- The analytical solution (in closed form) may not exist. In this case numerical techniques are used
- Multiple or an infinite number of solutions might exist

In general the solution is found without a systematic procedure, rather relying on intuition in manipulating the equations.

Inverse kinematics for a 2 d.o.f. planar manipulator



$$p_x = a_{1x} + a_{2x} = a_1 \cos(\vartheta_1) + a_2 \cos(\vartheta_1 + \vartheta_2)$$
$$p_y = a_{1y} + a_{2y} = a_1 \sin(\vartheta_1) + a_2 \sin(\vartheta_1 + \vartheta_2)$$

Squaring and summing:

$$c_2 = \frac{p_x^2 + p_y^2 - a_1^2 - a_2^2}{2a_1a_2} \quad \Rightarrow \quad \vartheta_2 = \mathrm{Atan2}(s_2, c_2)$$

$$s_2 = \pm\sqrt{1 - c_2^2}$$

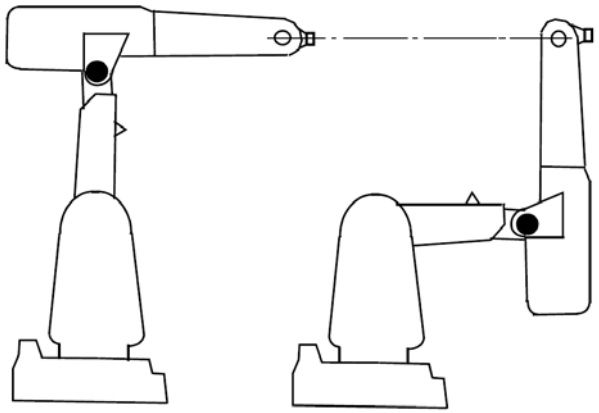
2 solutions

$$c_1 = \frac{(a_1 + a_2c_2)p_x + a_2s_2p_y}{p_x^2 + p_y^2}$$
$$s_1 = \frac{(a_1 + a_2c_2)p_y - a_2s_2p_x}{p_x^2 + p_y^2} \quad \Rightarrow \quad \vartheta_1 = \mathrm{Atan2}(s_1, c_1)$$

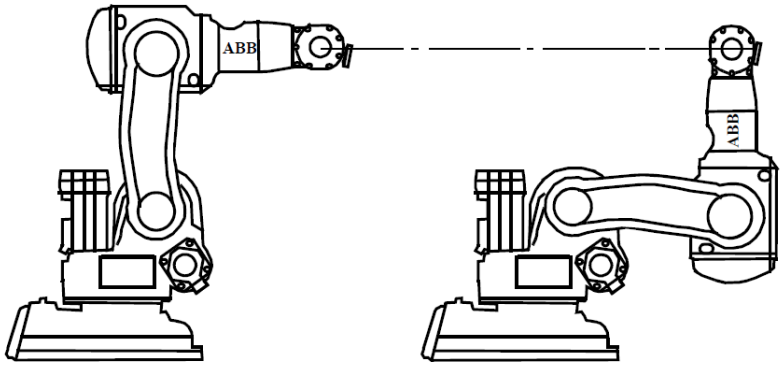
COMPLICATED!

Anthropomorphic manipulator

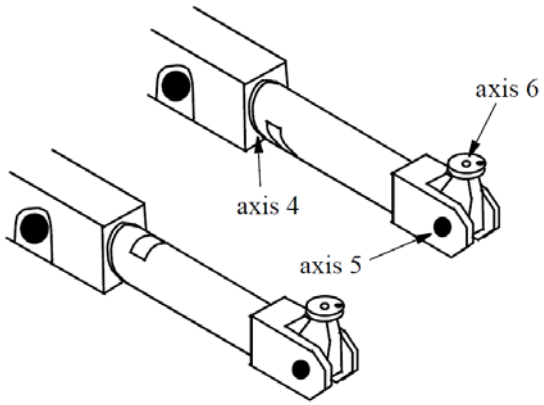
Eight admissible configurations exist:



right/left shoulder



up/down elbow



up/down wrist

Differential kinematics

Let's introduce now the linear velocity and the angular velocity of the tool frame (attached to the tool): $\dot{\boldsymbol{p}}$ and $\boldsymbol{\omega}$.

The goal of **differential kinematics** is to express these velocities in terms of the joint velocities.

$$\dot{\boldsymbol{p}} = \boldsymbol{J}_P(\boldsymbol{q})\dot{\boldsymbol{q}}$$
$$\boldsymbol{\omega} = \boldsymbol{J}_O(\boldsymbol{q})\dot{\boldsymbol{q}}$$

In a compact form: $$\boldsymbol{v} = \begin{bmatrix} \dot{\boldsymbol{p}} \\ \boldsymbol{\omega} \end{bmatrix} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$$

The $(6\times n)$ matrix: $$\boldsymbol{J}(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{J}_P(\boldsymbol{q}) \\ \boldsymbol{J}_O(\boldsymbol{q}) \end{bmatrix}$$

is called **geometrical Jacobian** of the manipulator.
Systematic methods exist to compute the Jacobian.

Analytical Jacobian

Let's go back to the direct kinematic equation of a manipulator:

$$\boldsymbol{x} = \boldsymbol{k}(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{p}(\boldsymbol{q}) \\ \boldsymbol{\phi}(\boldsymbol{q}) \end{bmatrix}$$

where $\boldsymbol{\phi}$ is a minimal representation of the orientation.
Differentiating w.r.t. time we obtain:

$$\dot{\boldsymbol{x}} = \frac{\partial \boldsymbol{k}(\boldsymbol{q})}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} = \boldsymbol{J}_A(\boldsymbol{q}) \dot{\boldsymbol{q}}$$

On the other hand:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{\phi}} \end{bmatrix} = \begin{bmatrix} (\partial \boldsymbol{p}(\boldsymbol{q}) / \partial \boldsymbol{q}) \dot{\boldsymbol{q}} \\ (\partial \boldsymbol{\phi}(\boldsymbol{q}) / \partial \boldsymbol{q}) \dot{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_P(\boldsymbol{q}) \\ \boldsymbol{J}_\phi(\boldsymbol{q}) \end{bmatrix} \dot{\boldsymbol{q}}$$

Matrix: $\boldsymbol{J}_A(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{J}_P(\boldsymbol{q}) \\ \boldsymbol{J}_\phi(\boldsymbol{q}) \end{bmatrix}$ is called **analytical Jacobian** of the manipulator.

Geometrical vs. analytical Jacobians

The link between the angular velocity $\boldsymbol{\omega}$ and the derivative of vector $\boldsymbol{\phi}$ expressing the orientation is:

$$\boldsymbol{\omega} = \boldsymbol{T}(\boldsymbol{\phi}) \dot{\boldsymbol{\phi}}$$

where $\boldsymbol{T}$ is a matrix that depends on the representation of the orientation.
Then:

$$\boldsymbol{v} = \begin{bmatrix} \dot{\boldsymbol{p}} \\ \boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{T}(\boldsymbol{\phi}) \end{bmatrix} \dot{\boldsymbol{x}} = \boldsymbol{T}_A(\boldsymbol{\phi}) \dot{\boldsymbol{x}} = \boldsymbol{T}_A(\boldsymbol{\phi}) \boldsymbol{J}_A \dot{\boldsymbol{q}}$$

$$\boldsymbol{J} = \boldsymbol{T}_A(\boldsymbol{\phi}) \boldsymbol{J}_A$$

Kinematic singularities

The equation defining the geometrical Jacobian is:

$$\boldsymbol{v} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$$

The values of $\boldsymbol{q}$ for which matrix $\boldsymbol{J}$ is rank-deficient are called **kinematic singularities**. At a kinematic singularity we have:

1. Loss of mobility (it is not possible to impose arbitrary motion laws)
2. Possibility of infinite solutions to the kinematic inversion problem
3. High velocities in joint space (around the singularity)

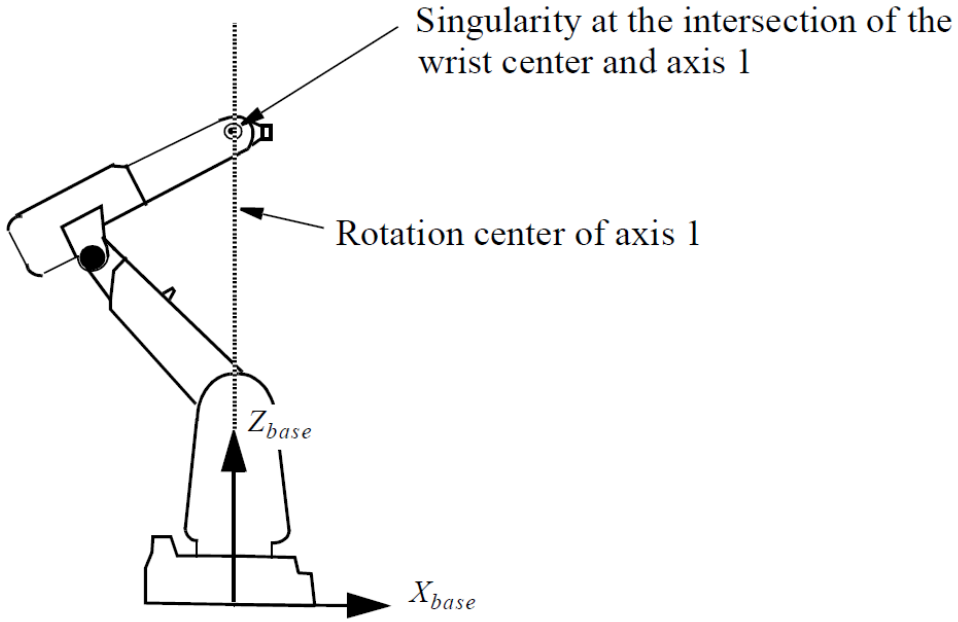
The singularities may happen:

1. At the borders of the manipulator work-space
2. Inside the manipulator work-space

The latter are more problematic, since they can be incurred with trajectories planned in the operational space.

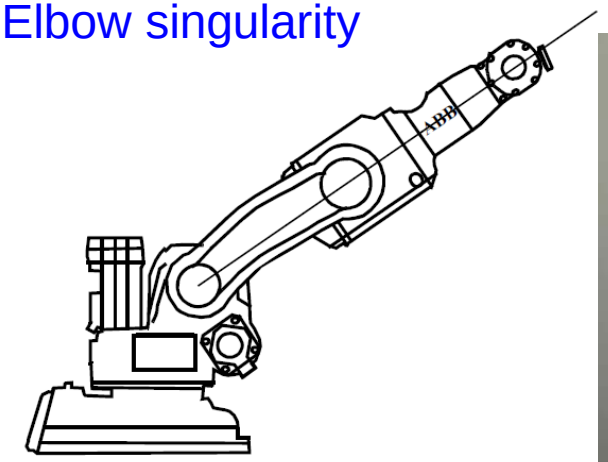
Kinematic singularities of a complete manipulator

Arm singularity

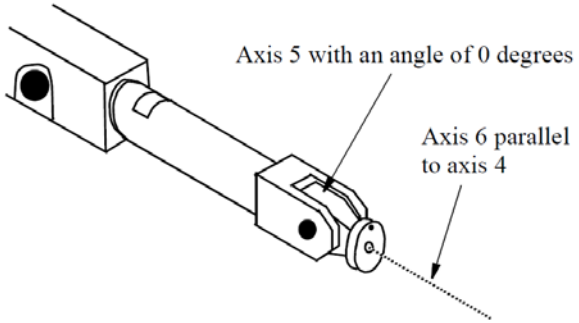


Source: ABB

Elbow singularity



Wrist singularity

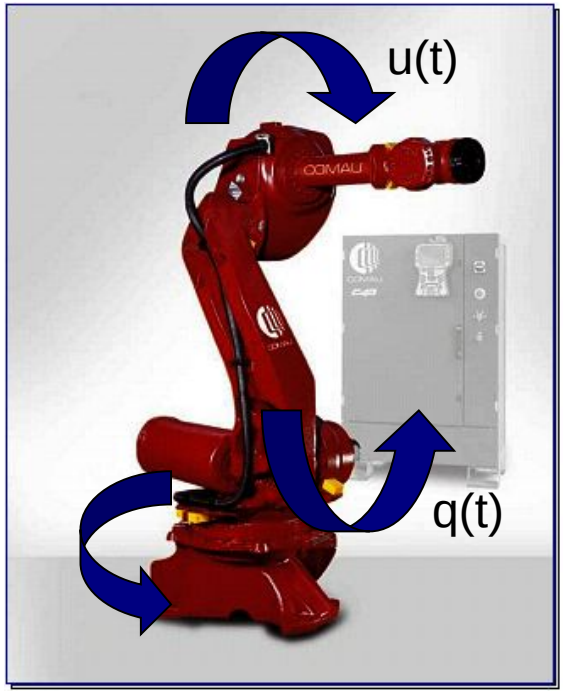


Agenda

- Introduction
- Robot kinematics
- **Robot dynamics**
- Motion planning
- Motion control

Dynamic model

The dynamic model accounts for the relation between the sources of motion (forces and moments) and the resulting motion (positions and velocities)



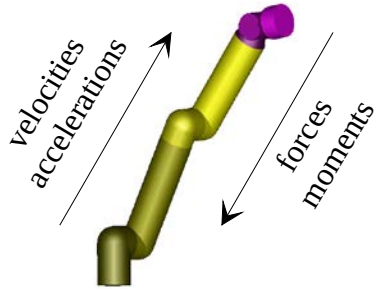
Systematic methods exist to derive the dynamic model of the manipulators:

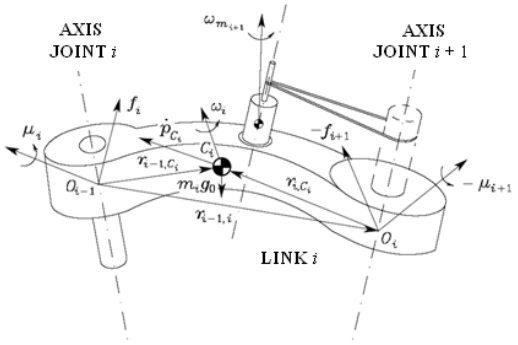
- **Euler-Lagrange method**

$$L(\boldsymbol{q},\dot{\boldsymbol{q}}) = T(\boldsymbol{q},\dot{\boldsymbol{q}}) - U(\boldsymbol{q})$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \xi_i, \quad i = 1,\ldots,n$$

- **Newton-Euler method**





Dynamic model

$$\boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}$$

gravitational terms

Coriolis and centrifugal terms

inertial terms

$$T = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij}(\boldsymbol{q})\dot{q}_i\dot{q}_j = \frac{1}{2}\dot{\boldsymbol{q}}^T\boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{q}}$$

Kinetic energy

B symmetric and positive definite

$$\begin{cases} b_{ii} : \text{inertia moment as “seen” from the axis of joint } i \\ b_{ij} : \text{effect of the acceleration of joint } j \text{ on joint } i \end{cases}$$

$$c_{ij} = \sum_{k=1}^{n}\frac{1}{2}\left(\frac{\partial b_{ij}}{\partial q_k} + \frac{\partial b_{ik}}{\partial q_j} - \frac{\partial b_{jk}}{\partial q_i}\right)\dot{q}_k$$

Dynamic model

Some properties of the dynamic model:

- Matrix:

$$\boldsymbol{N}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \dot{\boldsymbol{B}}(\boldsymbol{q}) - 2\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$$

is skew-symmetric:

$$\boldsymbol{w}^T \boldsymbol{N}(\boldsymbol{q},\dot{\boldsymbol{q}})\boldsymbol{w} = 0, \quad \forall \boldsymbol{w}$$

→ Used in proofs of stability of some control laws

- The dynamic model is linear with respect to a set of parameters

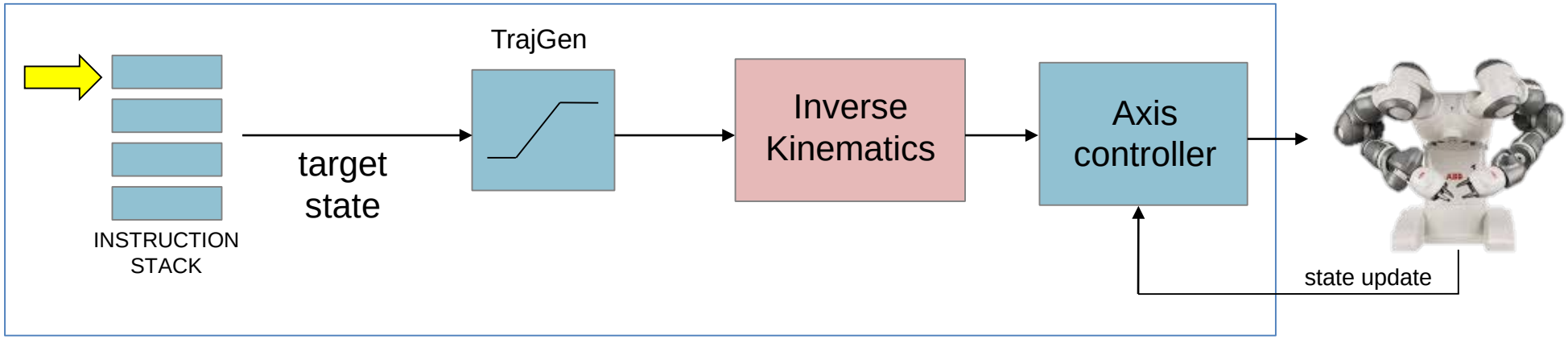
$$\tau = \boldsymbol{Y}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})\pi$$

→ Used in identification and adaptation of the parameters of the model

Agenda

- Introduction
- Robot kinematics
- Robot dynamics
- **Motion planning**
- Motion control

Motion planning and control

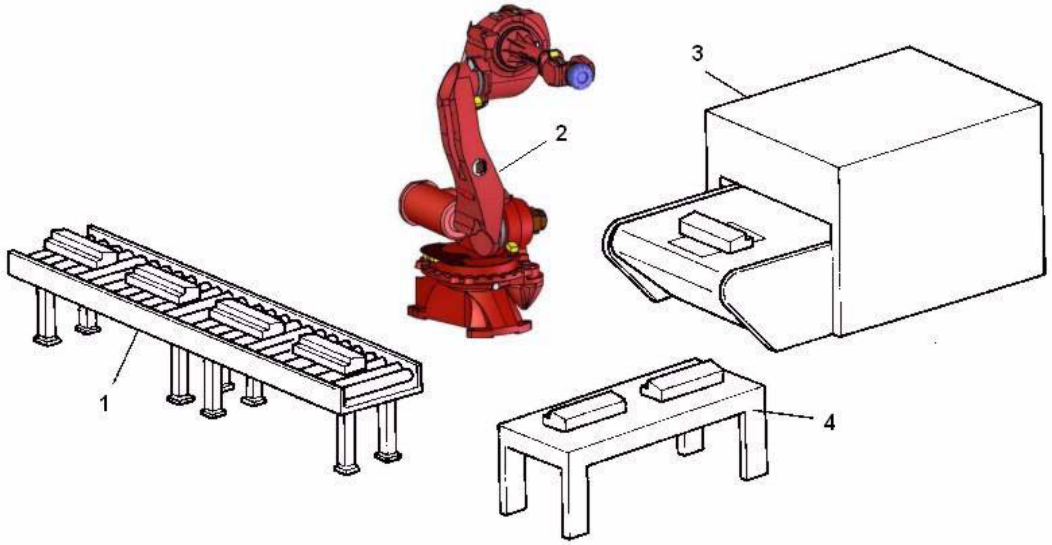


- **Instruction stack**: list of instructions to be executed, specified using the proprietary programming language
- **Trajectory generation**: converts an instruction into a trajectory to be executed
- **Inverse kinematics**: maps the trajectory from the Cartesian space to the joint space
- **Axis controllers & drives**: closes the control loop ensuring tracking performance

Robot programming

The program moves pieces from a feeder to a table or to a discard bin, depending on digital input signals:

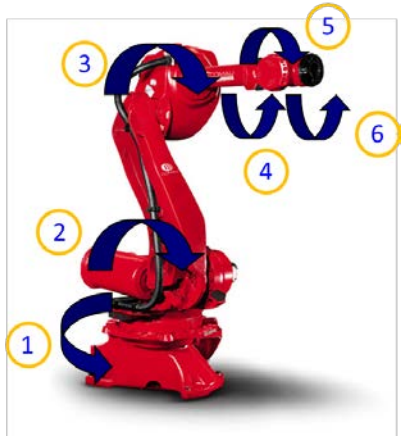
```
PROGRAM pack
VAR
   home, feeder, table, discard : POSITION
BEGIN CYCLE
   MOVE TO home
   OPEN HAND 1
   WAIT FOR $DIN[1] = ON
-- signals feeder ready
   MOVE TO feeder
   CLOSE HAND 1
   IF $DIN[2] = OFF THEN
-- determines if good part
      MOVE TO table
   ELSE
      MOVE TO discard
   ENDIF
   OPEN HAND 1
-- drop part on table or in bin
END pack
```



1. Feeder
2. Robot
3. Discard Bin
4. Table

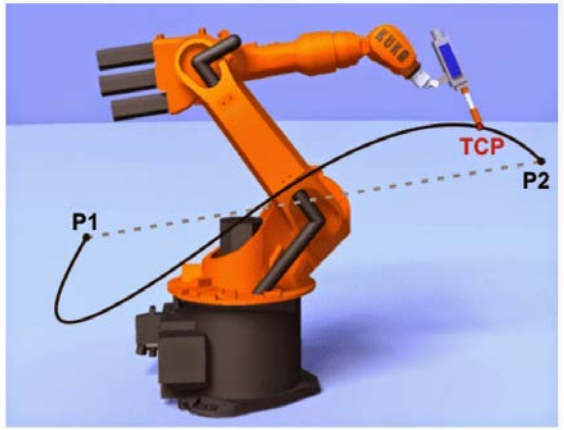
Joint space and operational space trajectories

Trajectories in joint space: the desired joint positions are directly specified



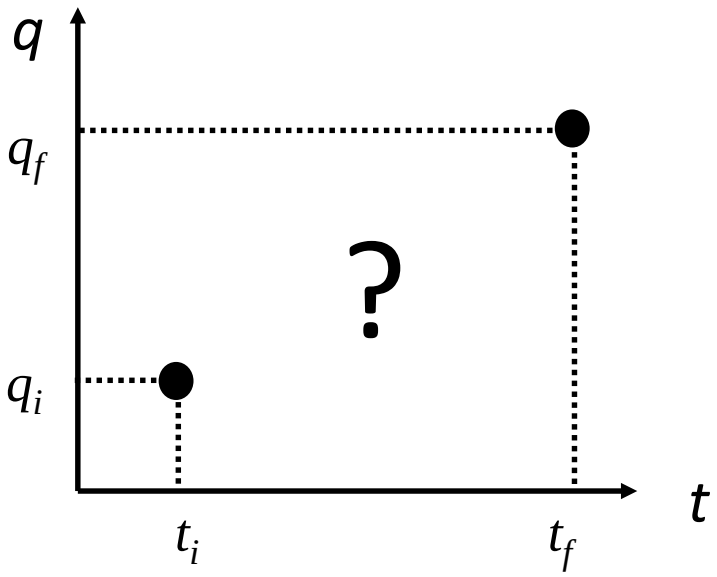
- problems related to kinematic singularities and redundant degrees of freedom are solved directly
- it is a mode of interest when we just want that the axes move from an initial to a final pose (and we are not interested in the resulting motion of the end effector)
- online kinematic inversion is not needed

Trajectories in the operational space: the path (position and orientation) of the robot end effector is specified in the common Cartesian space



- task description is natural
- constraints on the path can be accounted for
- singular points or redundant degrees of freedom generate problems
- online kinematic inversion is needed

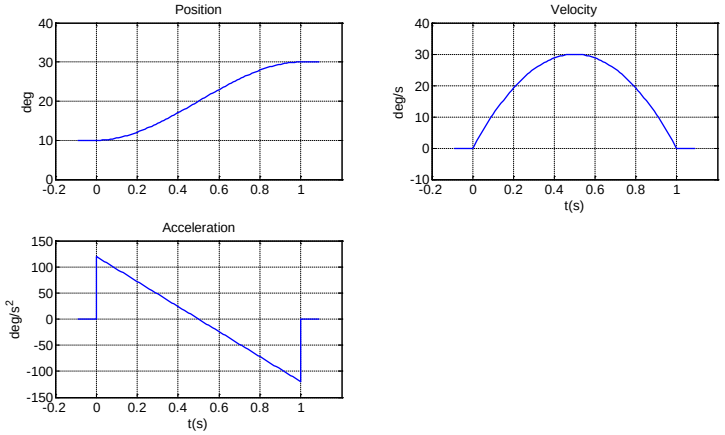
Point-to-point motion



For each joint variable we define how it evolves from an initial value to a final one in a finite time. Several methods exist…

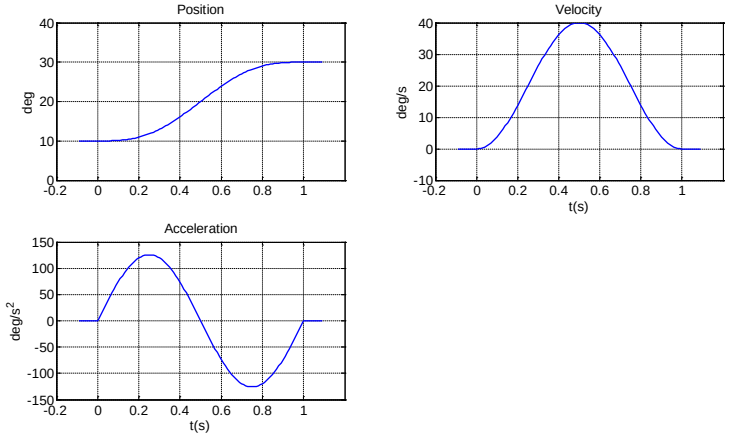
Cubic polynomial

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$



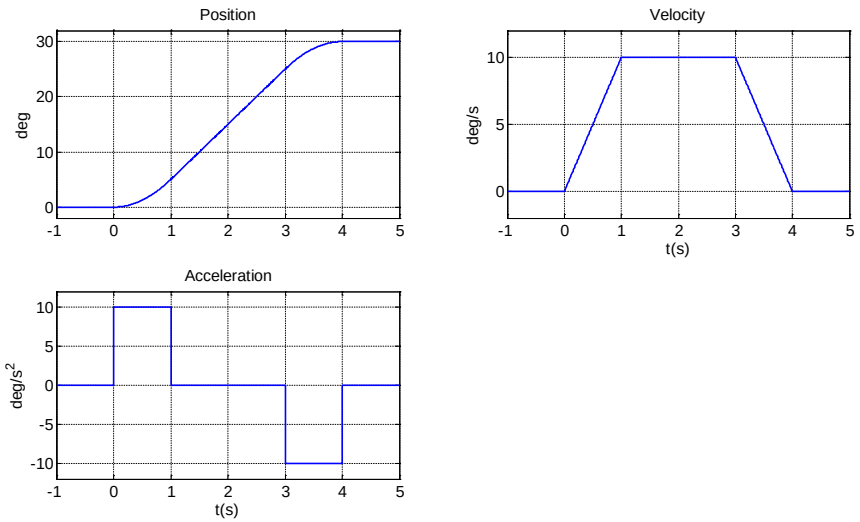
Cycloidal

$$q(t) = (q_f - q_i)\left(\frac{t - t_i}{t_f - t_i} - \frac{1}{2\pi}\sin\left(\frac{2\pi(t - t_i)}{t_f - t_i}\right)\right) + q_i$$



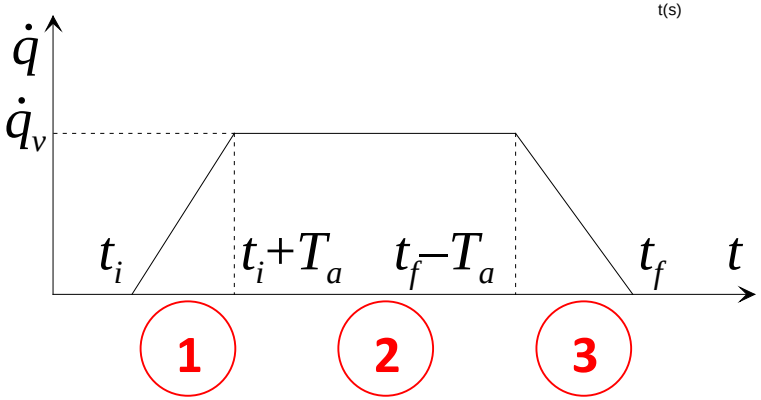
Trapezoidal velocity profile

A quite common industrial practice to generate the trajectory consists in planning a linear position profile adjusted at the beginning and at the end of the trajectory with parabolic blendings. The resulting velocity profile has the typical trapezoidal shape.



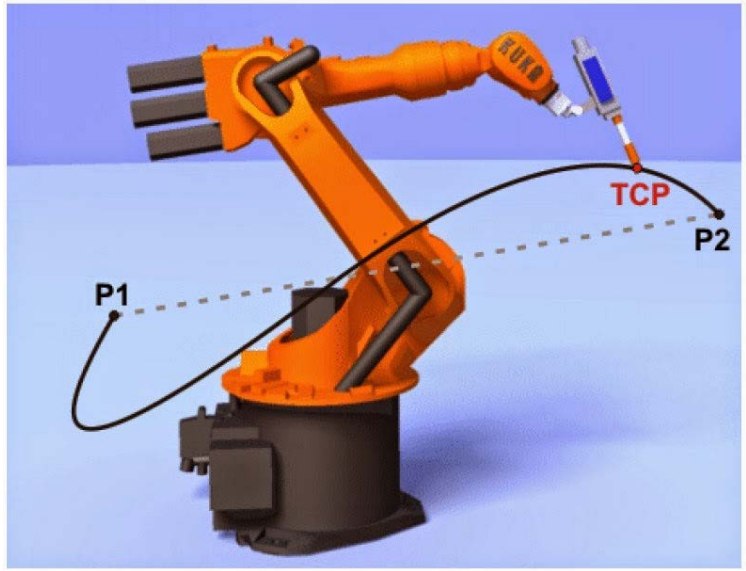
The trajectory is then composed of three parts:

1. Constant acceleration, linear velocity, parabolic position;
2. Zero acceleration, constant velocity, linear position;
3. Constant deceleration, linear velocity, parabolic position.

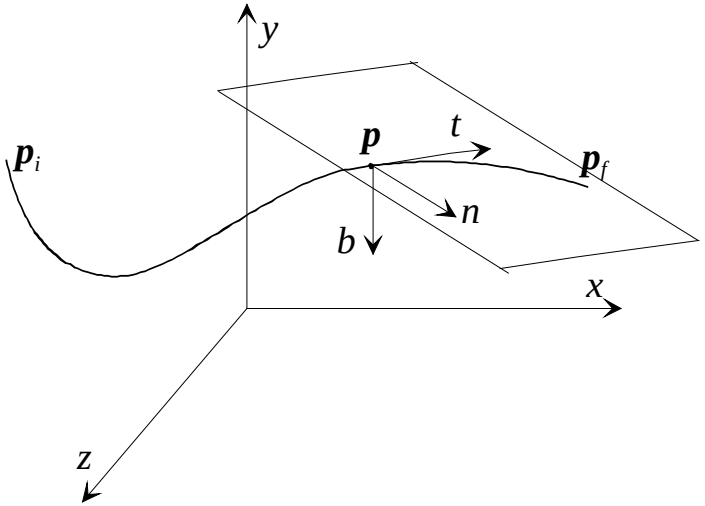


Trajectories in the operational space

Trajectory planning in the joint space yields unpredictable motions of the end-effector. When we want the motion to evolve along a predefined path in the operational space, it is necessary to plan the trajectory directly in this space.



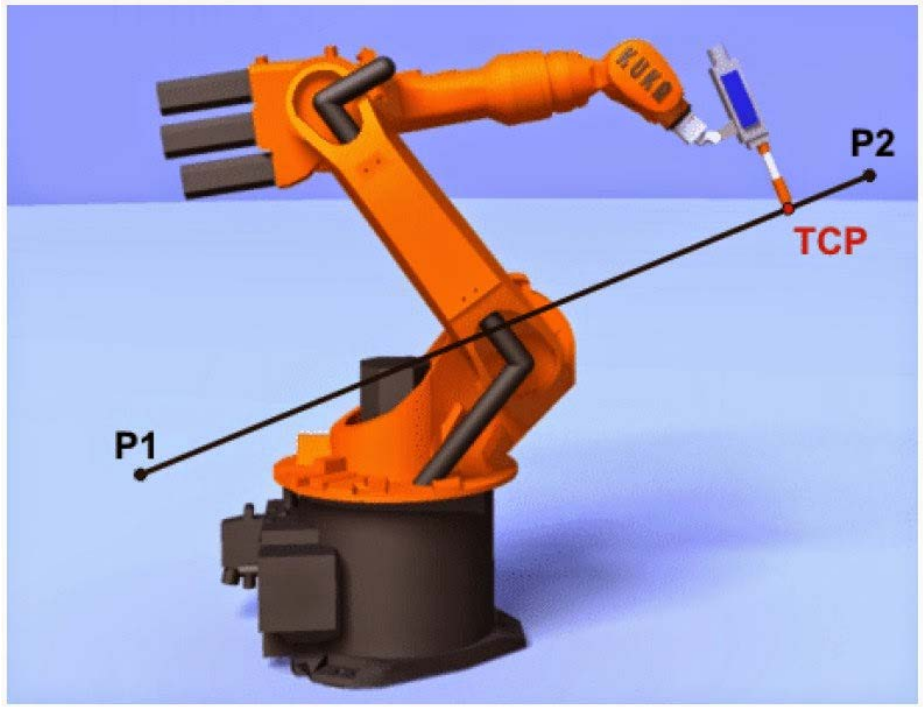
Trajectory planning in the operational space entails both a path planning problem and a timing law planning problem.



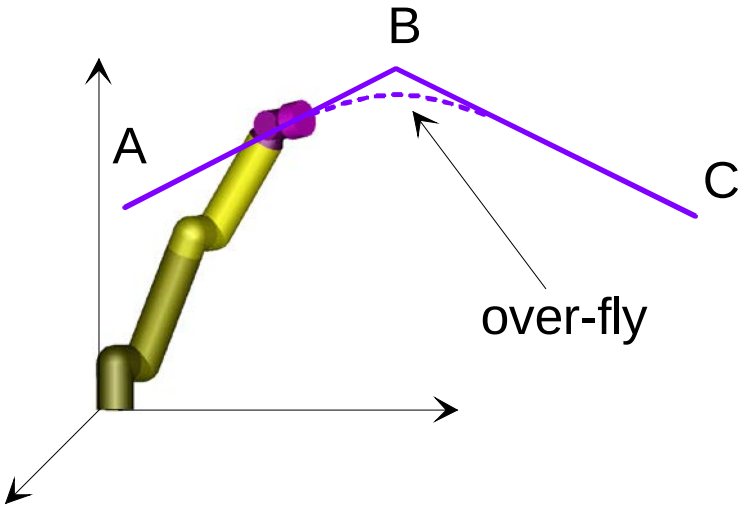
The parameterization of the path can be performed with respect to the natural coordinate (length of the arc of trajectory) : $\boldsymbol{p} = \boldsymbol{p}(s)$

Linear path

A linear path is completely characterized once two points in Cartesian space are given



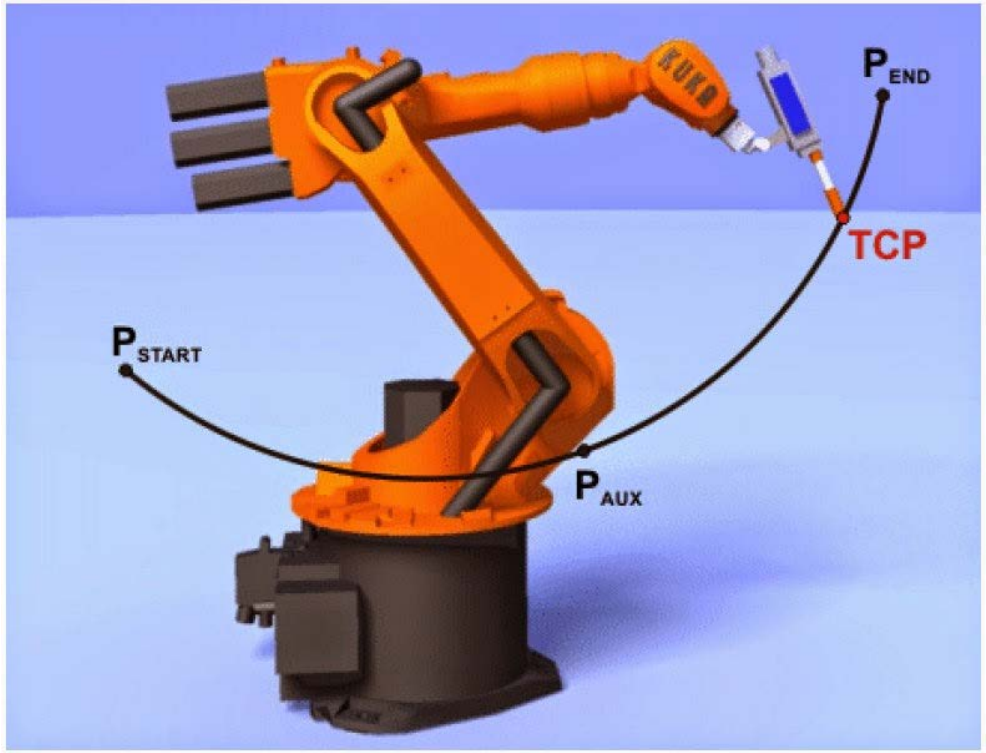
$$\boldsymbol{p}(s) = \boldsymbol{p}_1 + \frac{s}{\|\boldsymbol{p}_f - \boldsymbol{p}_i\|}(\boldsymbol{p}_2 - \boldsymbol{p}_1)$$



Linear paths can be concatenated through via points

Circular path

A circular path can be defined assigning three points in space belonging to the same plane:



A parametric representation of a circumference of radius ρ laying in a plane $x'\,y'$ and having the centre in the origin of such plane is:

$$\boldsymbol{p}'(s) = \begin{bmatrix} \rho\cos(s/\rho) \\ \rho\sin(s/\rho) \\ 0 \end{bmatrix}$$

Defining:

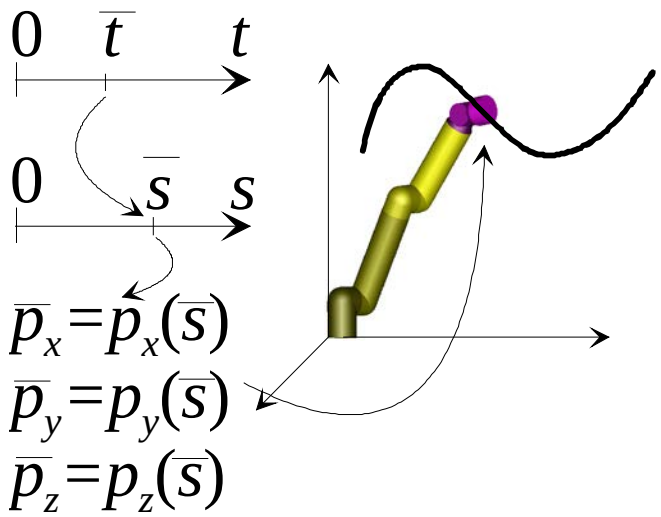
- $\boldsymbol{c}$ the vector that identifies the centre of the circumference in the base frame
- $\boldsymbol{R}$ the rotation matrix from base frame to frame $x'\,y'\,z'$

the general parametric representation of a circumference in space is:

$$\boldsymbol{p}(s) = \boldsymbol{c} + \boldsymbol{R}\boldsymbol{p}'(s)$$

Time law

Taking into account the parameterization of the path with respect to the natural coordinate $\boldsymbol{p} = \boldsymbol{p}(s)$, we will assign the time law through the function $s(t)$.



In order to determine function $s(t)$ we can use any of the time laws (cubic, cycloidal, trapezoidal velocity profile …)
Also we notice that:

$$\dot{\boldsymbol{p}} = \dot{s}\frac{d\boldsymbol{p}}{ds} = \dot{s}\boldsymbol{t}$$

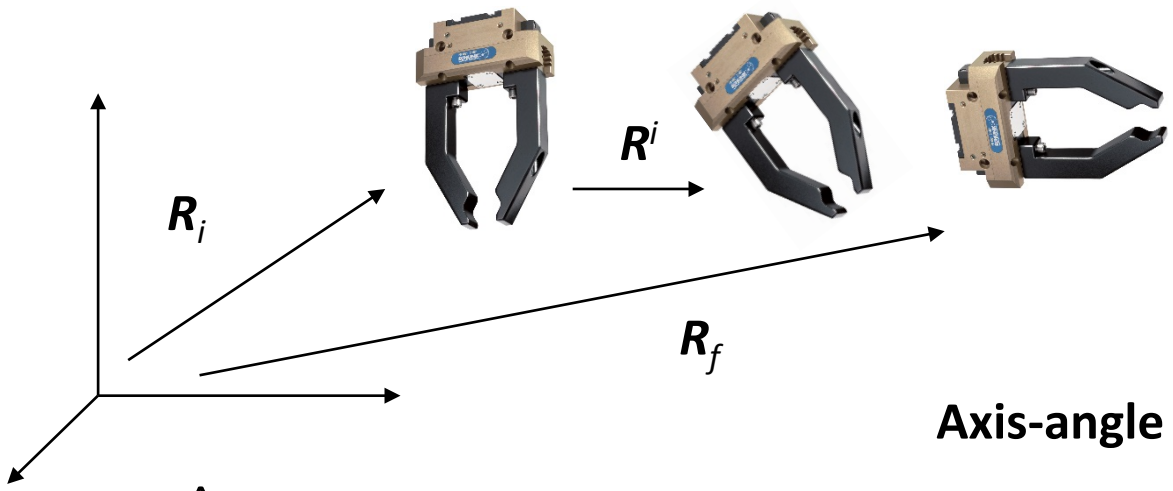
$|\dot{s}|$ is then the norm of the velocity ➡ the time law $s(t)$ takes then an immediate meaning!

For the segment: $$\dot{\boldsymbol{p}} = \frac{\dot{s}}{\|\boldsymbol{p}_f - \boldsymbol{p}_i\|}(\boldsymbol{p}_f - \boldsymbol{p}_i) = \dot{s}\boldsymbol{t}$$

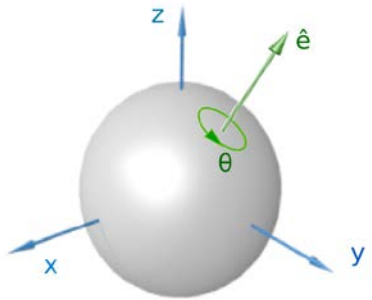
Orientation trajectories

Define with $\boldsymbol{R}^i(t)$ the matrix that describes the transition from $\boldsymbol{R}_i$ to $\boldsymbol{R}_f$.
Then:

$$\boldsymbol{R}^i(0) = \boldsymbol{I}, \quad \boldsymbol{R}^i(t_f) = \boldsymbol{R}_f^i$$



Axis-angle representation



Matrix $\boldsymbol{R}^i(t)$ can be interpreted as $\boldsymbol{R}^i(\vartheta(t), \hat{\boldsymbol{e}})$, where:

- $\hat{\boldsymbol{e}}$ is constant and can be computed from the elements of $\boldsymbol{R}^i_f$
- $\vartheta(t)$ can be made variable with time, through a timing law, with $\vartheta(0)=0$, $\vartheta(t_f)=\vartheta_f$

Inversion of the differential kinematics

We have already noticed that the inverse kinematics is difficult to compute, in general. On the other hand, the differential kinematics is linear for a certain value of $\boldsymbol{q}$:

$$\dot{\boldsymbol{x}} = \boldsymbol{J}_A(\boldsymbol{q})\dot{\boldsymbol{q}}$$

Given a velocity $\dot{\boldsymbol{x}}$ in the operational space and an initial condition on $\boldsymbol{q}$ we might solve the kinematic inversion problem by inverting the differential kinematics and then integrating. If the Jacobian is square and non singular:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_A^{-1}(\boldsymbol{q})\dot{\boldsymbol{x}} \quad \Rightarrow \quad \boldsymbol{q}(t) = \int_0^t \dot{\boldsymbol{q}}(\sigma)d\sigma + \boldsymbol{q}(0)$$

However, using this expression directly, drifts of the solution may occur.

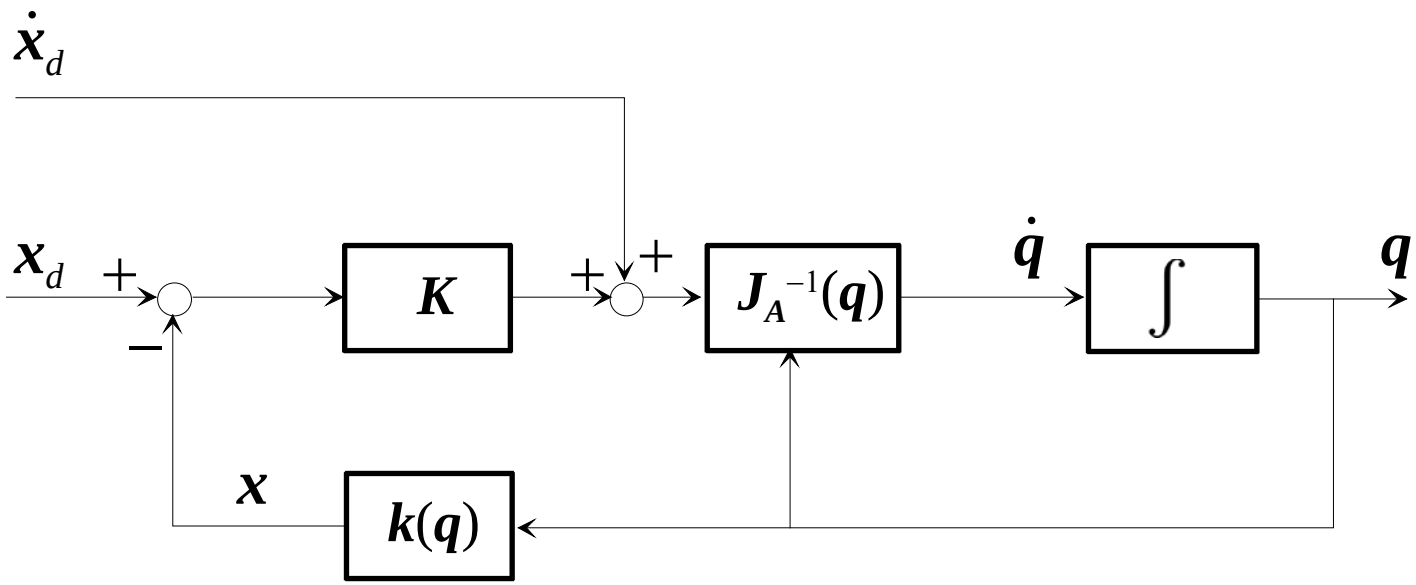
The error in the operational space is then introduced: $\boldsymbol{e} = \boldsymbol{x}_d - \boldsymbol{x}$

Inversion of the differential kinematics

If we adopt the following dependence of $\dot{\boldsymbol{q}}$ from $\boldsymbol{e}$:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_A^{-1}(\boldsymbol{q})(\dot{\boldsymbol{x}}_d + \boldsymbol{K}\boldsymbol{e})$$

we obtain this block diagram:



Mathematically, this corresponds to solve the inverse kinematics problem through a Gauss-Newton iterative method.
Proof of convergence is trivial as:

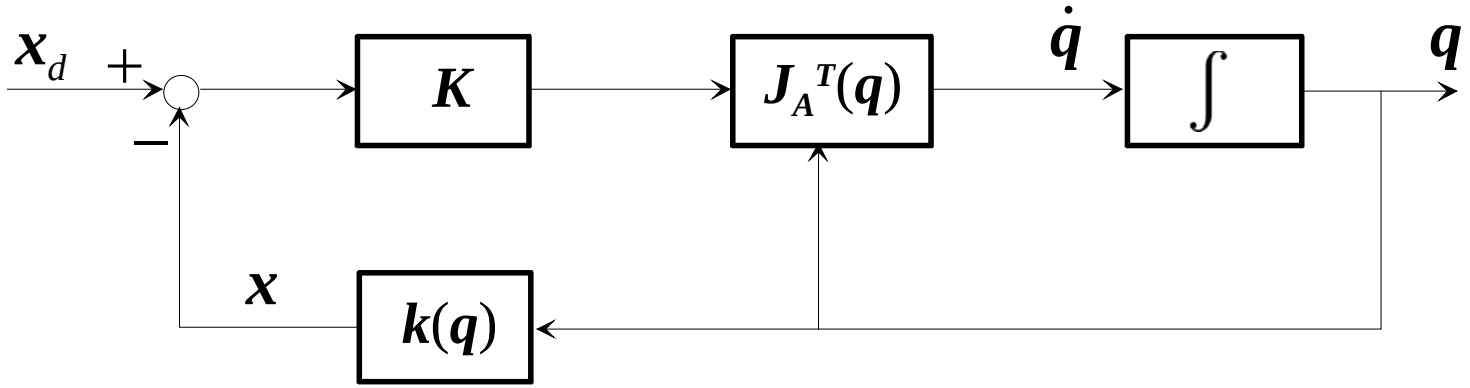
$$\dot{\boldsymbol{e}} + \boldsymbol{K}\boldsymbol{e} = \boldsymbol{0}$$

Inversion of the differential kinematics

If we adopt the following (simpler) dependence:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_A^T(\boldsymbol{q})\boldsymbol{K}\boldsymbol{e}$$

we obtain the diagram:



Mathematically, this corresponds to solve the inverse kinematics problem through a gradient descent iterative method.
Proof of convergence can be obtained through a Lyapunov argument.

Kinematic redundancy

Direct kinematics of the manipulator:

$$\boldsymbol{x} = \boldsymbol{k}(\boldsymbol{q})$$

$$\mathbb{R}^n \rightarrow \mathbb{R}^m$$

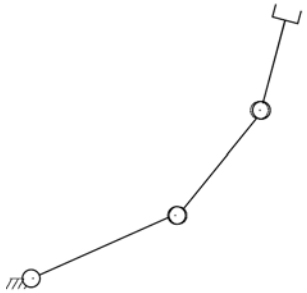
Joint space

Task space

A robot is kinematically redundant if: $\boxed{n > m}$

i.e. if it has more degrees of freedom than those strictly necessary to perform a task.

Redundancy is a relative concept: the number of task variables may be less than the dimension of the operational space.



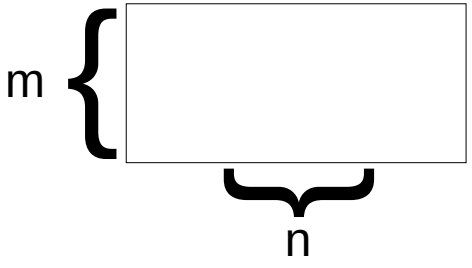
A 3 d.o.f. planar manipulator is *functionally* redundant if the task is specified with respect to the end-effector position only.

Redundancy and kinematic inversion

The inversion of the differential kinematics equation:

$$\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \qquad (*)$$

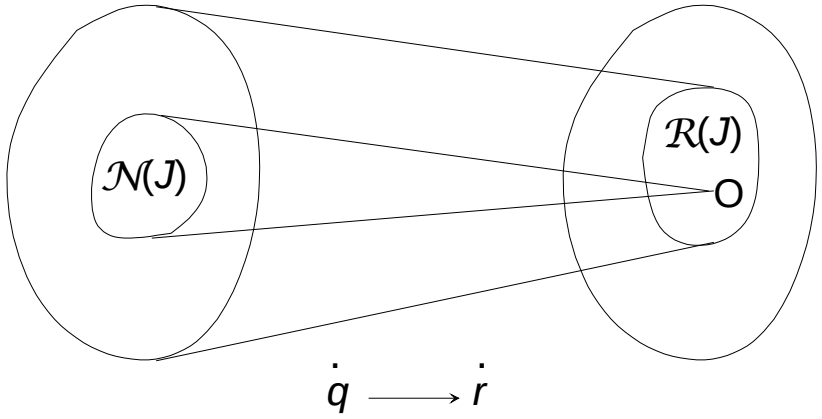
is not trivial since the Jacobian is a lower rectangular matrix



(*) we simplify here the notation, dropping the "A" for the analytical Jacobian

For a given configuration $\boldsymbol{q}$ the equation establishes a linear mapping from the space of joint velocities to the space of task velocities.

This mapping can be characterized in terms of range and null spaces:



$$\dim(\mathcal{R}(\boldsymbol{J})) = m$$

if the Jacobian is full rank: rank($\boldsymbol{J}$) = m

$$\dim(\mathcal{N}(\boldsymbol{J})) = n - m$$

The null space exists only if the robot is redundant ($n > m$).
It is the space of joint velocities that do not produce task velocities.

Null space methods

A general solution of the inverse kinematics:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\dot{\boldsymbol{x}} + \left(\boldsymbol{I}_n - \boldsymbol{J}^{\#}\boldsymbol{J}\right)\dot{\boldsymbol{q}}_0$$

particular solution

homogeneous solution: self-motions of the manipulator

$\boldsymbol{J}^{\#} = \boldsymbol{J}^T\left(\boldsymbol{J}\boldsymbol{J}^T\right)^{-1}$ is the right (Moore Penrose) pseudo inverse of the Jacobian matrix $\boldsymbol{J}$, since $\boldsymbol{J}\boldsymbol{J}^{\#} = \boldsymbol{I}$.

$\boldsymbol{I}_n - \boldsymbol{J}^{\#}\boldsymbol{J}$ is a projection matrix: it projects the velocities $\dot{\boldsymbol{q}}_0$ into the null space of the Jacobian

How to specify vector $\dot{\boldsymbol{q}}_0$?

One possible choice is the **projected gradient** method:

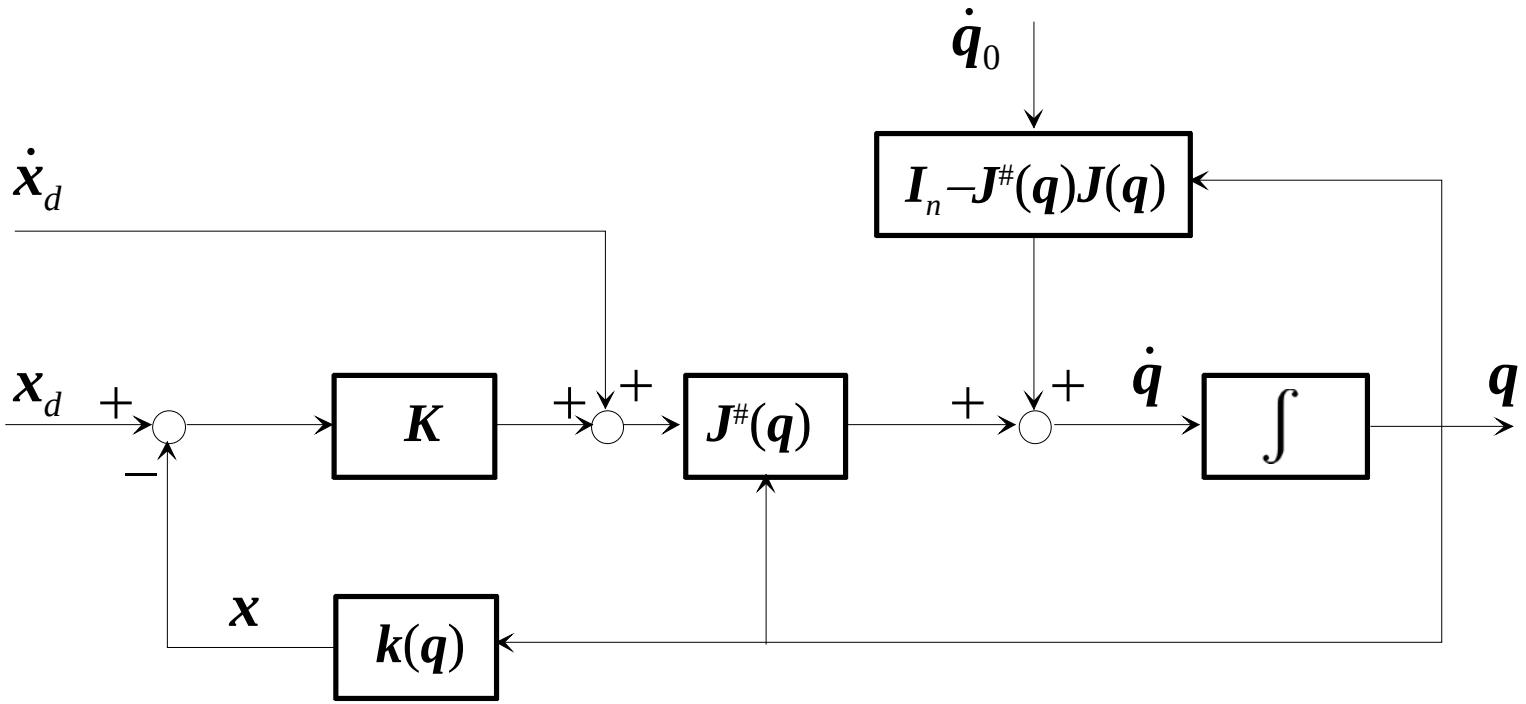
$$\dot{\boldsymbol{q}}_0 = k\left(\frac{\partial U(\boldsymbol{q})}{\partial \boldsymbol{q}}\right)^T$$

where $U(\boldsymbol{q})$ is a differentiable objective function and $k > 0$.

We try to locally maximize $U(\boldsymbol{q})$ while executing the time varying task $\boldsymbol{x}(t)$.

Kinematic control

The closed loop inverse kinematics method for a redundant robot is sketched in this block diagram:



Agenda

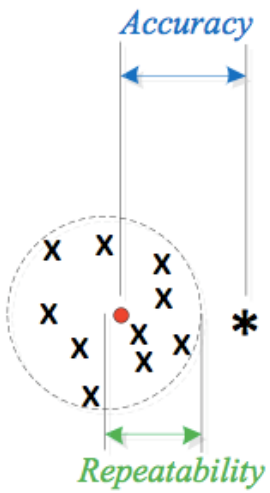
- Introduction
- Robot kinematics
- Robot dynamics
- Motion planning
- **Motion control**

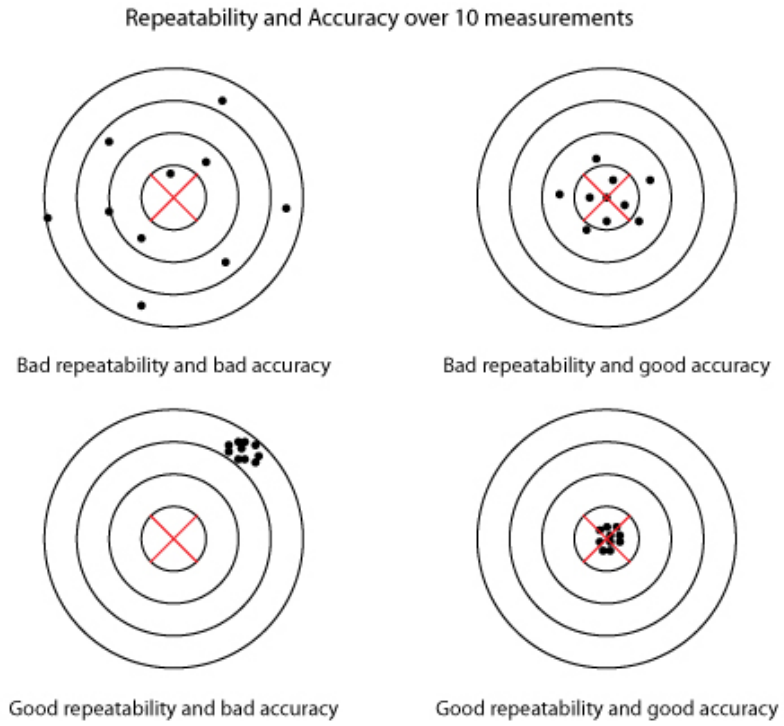
Evaluation of control performance

What are possible criteria to evaluate the performance of a motion control system?

- Quality of motion in nominal conditions
  - accuracy/repeatability
  - speed of task execution
  - energy saving

- Robustness of motion in perturbed conditions
  - adaptation to the environment
  - high repeatability in spite of uncertainties in modelling errors

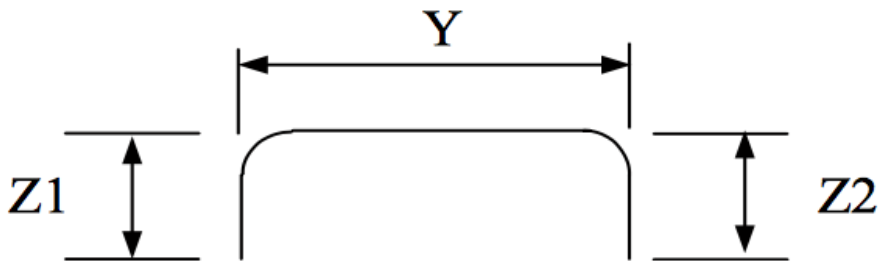




∗ : Desired position
x : Obtained positions
● : Barycenter of the obtained positions

Evaluation of control performance

Cycle time is another good key performance indicator for a pick and place task. It is the (minimum) time required by the robot to carry a specified payload along a bridge-like path.



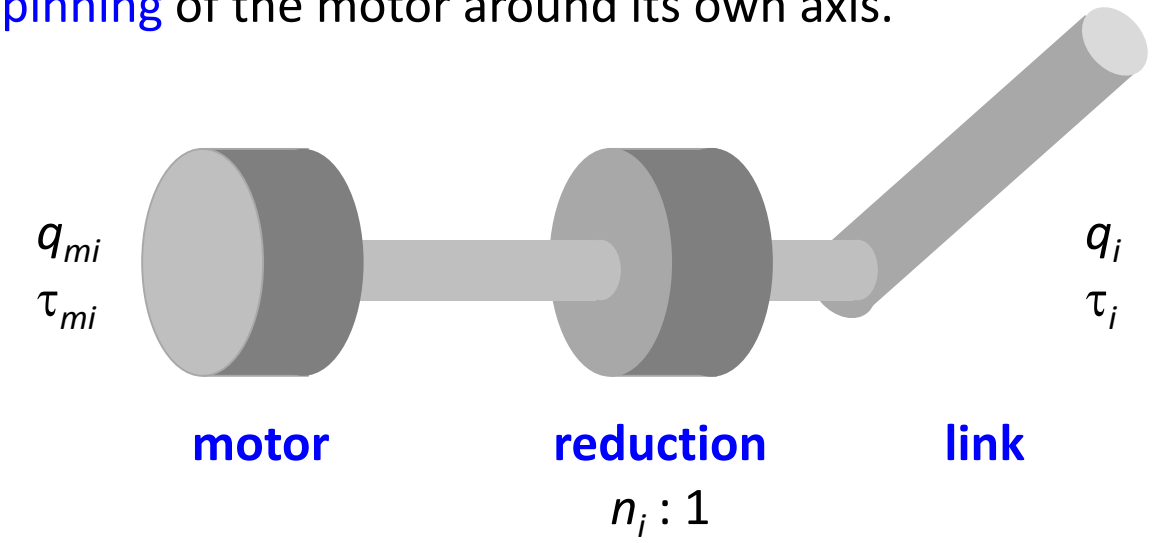
Decentralized model

Consider the dynamic model of the manipulator:

$$\boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}$$

Assume that a motor acts on each joint.

A simplified way to account for the dynamics of such motors is to consider just the effect related to the spinning of the motor around its own axis.



It is convenient to decompose the inertia matrix into the sum of a diagonal and constant term ("average" inertias) and a residual term:

$$\boldsymbol{B}(\boldsymbol{q}) = \overline{\boldsymbol{B}} + \Delta\boldsymbol{B}(\boldsymbol{q})$$

Furthermore:

$$\boldsymbol{J}_m = diag\{J_{mi}\}$$

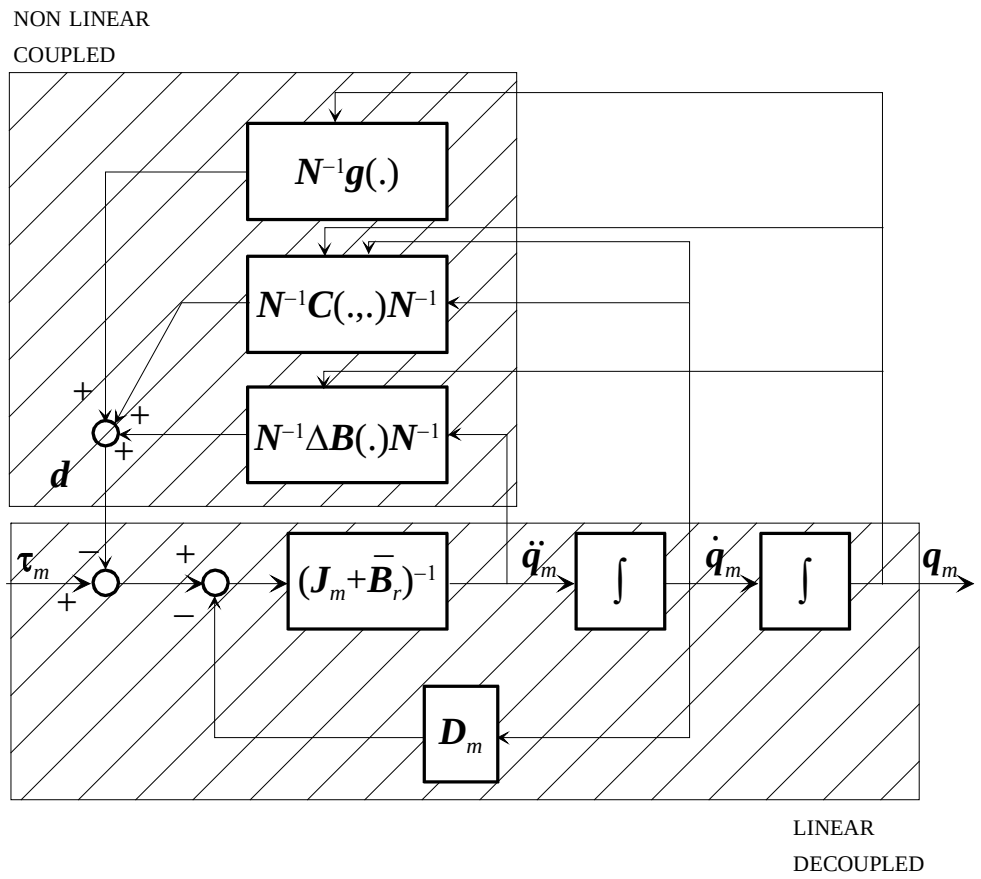
$$\boldsymbol{D}_m = diag\{D_{mi}\}$$

$$\boldsymbol{N} = diag\{n_i\}$$

$$\overline{\boldsymbol{B}}_r = \boldsymbol{N}^{-1}\overline{\boldsymbol{B}}\boldsymbol{N}^{-1}$$

"average" inertias scaled by the squares of the transmission ratios: $\overline{b}_{rii} = \dfrac{\overline{b}_{ii}}{n_i^2}$

Decentralized model

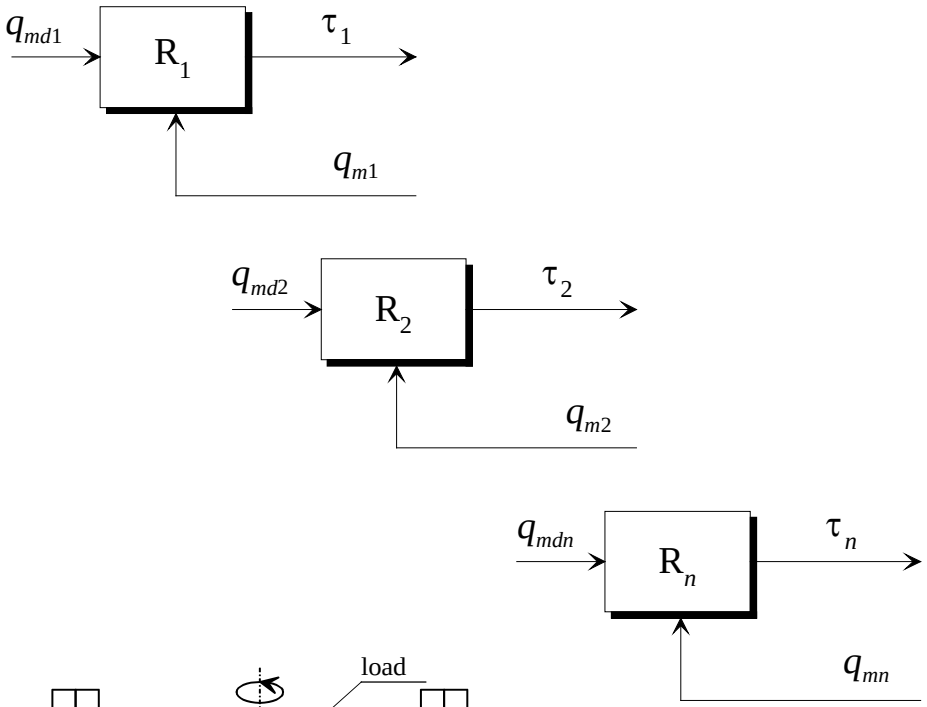


The resulting equations can be interpreted as those of a linear and completely decoupled system, subjected to a disturbance deriving from the nonlinear and coupled terms of the dynamic model.

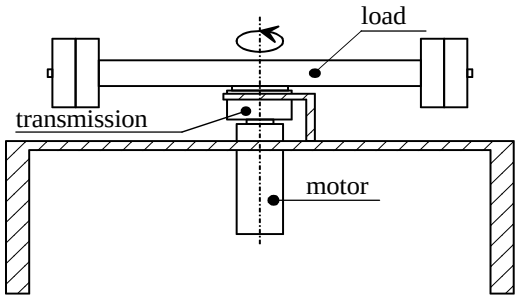
The larger the reduction ratios n_i, the less relevant is the disturbance term.

Independent joint control

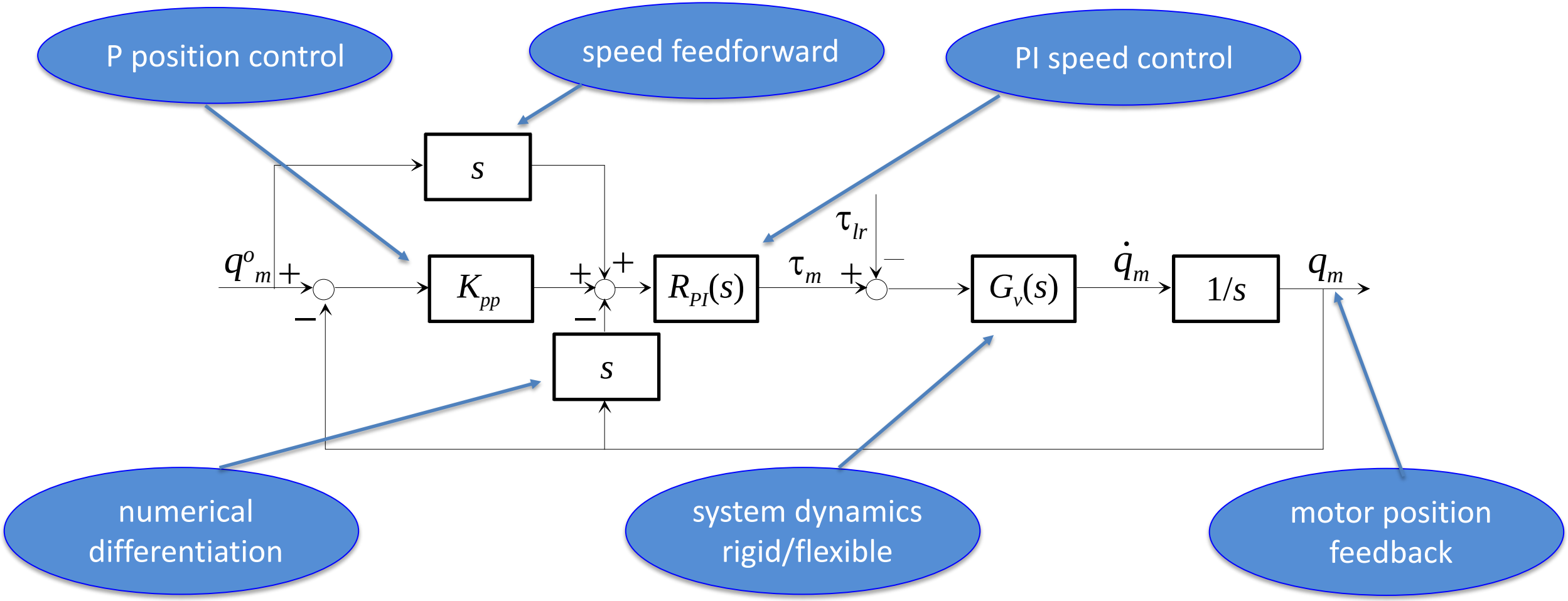
- The decentralized model of the robot dynamics is used in the independent joint control, a widely used approach in the industrial robot controllers
- The control system is articulated in *n* SISO control loops, ignoring the dynamic coupling effects which are dealt with as disturbances
- The method heavily relies on the large values of the reduction ratios adopted in robotics.



- The single control problems are assimilated to the control of a servomechanism

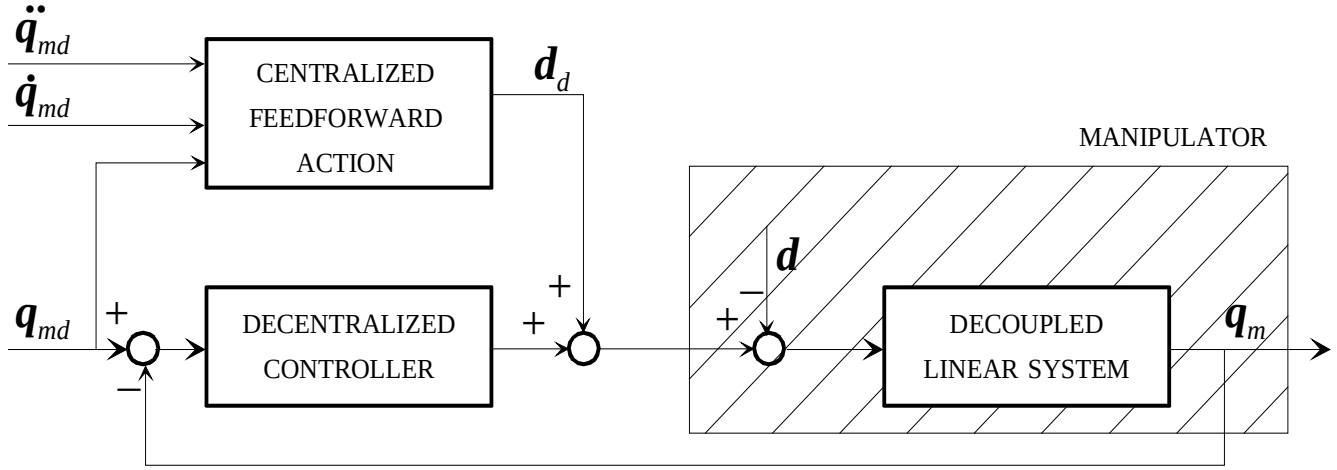


Independent joint control



Computed torque control

In this scheme, the decentralized controller is complemented by a controller that operates in open loop, computing the disturbance torques based on the mathematical model, fed by the position reference signal and its derivatives.

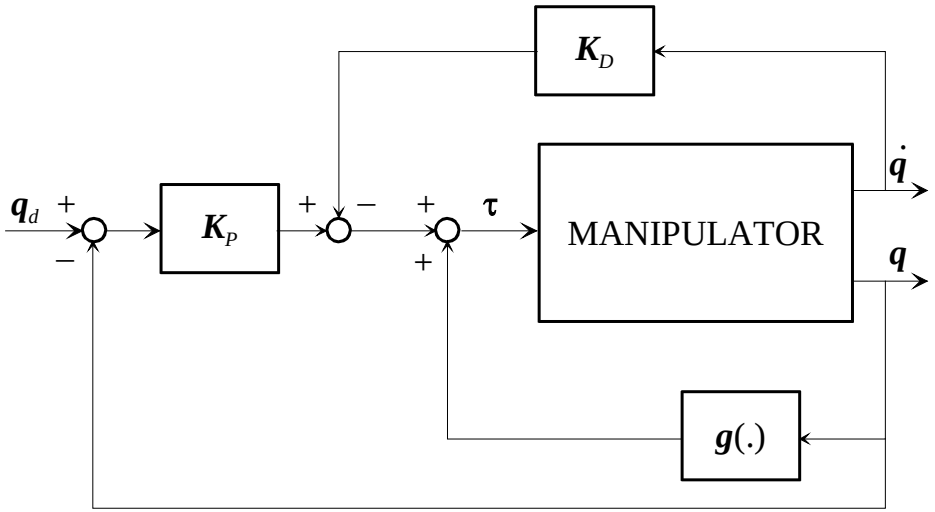


$$d_d = N^{-1}\Delta B(q_d)N^{-1}\ddot{q}_{md} + N^{-1}C(q_d,\dot{q}_d)N^{-1}\dot{q}_{md} + N^{-1}g(q_d)$$

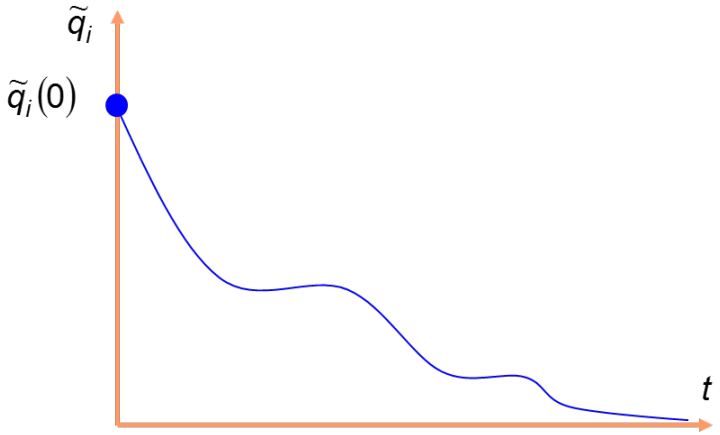
- Compensation of nonlinearities can be only partial (for instance just the gravitational terms and the diagonal terms of the inertia matrix)
- Compensation terms can be computed offline, in case of trajectories repeated several times

PD plus gravity compensation

This control law is formed by a position and speed feedback in joint space plus a term that compensates for gravitational effects.

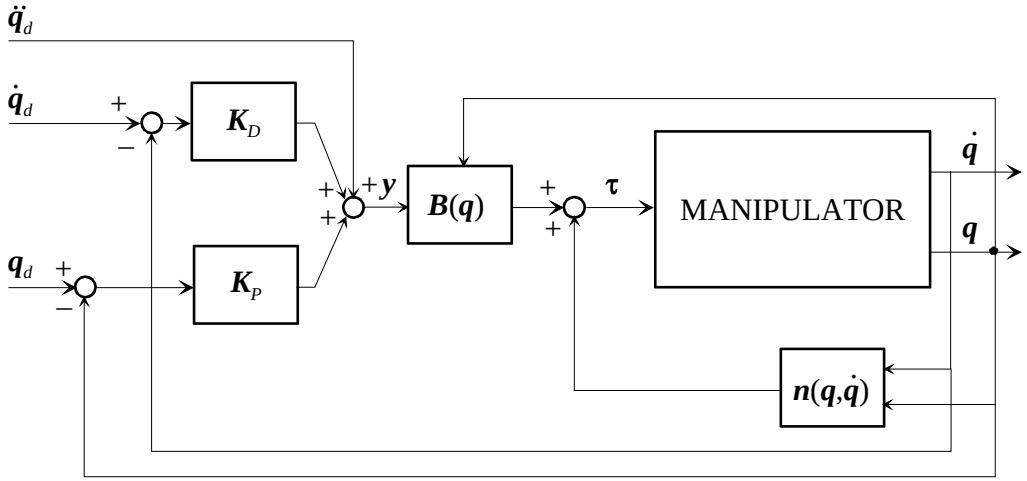


- The proportional gains can be seen as virtual springs while the derivative gains as virtual dampers
- Global asymptotic stability of the equilibrium state for constant inputs can be proven (as long as perfect gravity compensation is achieved)

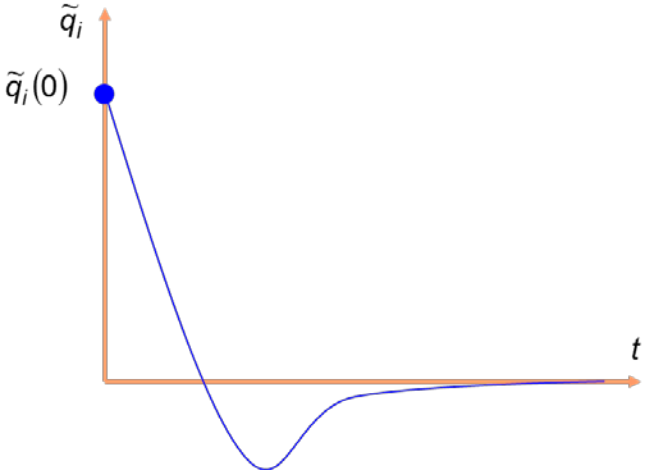


Inverse dynamics control

Here we compensate for all nonlinear and coupling terms of the dynamic model, then stabilizing the resulting linear dynamics with PD controllers.



- The method requires perfect cancellation of the terms of the dynamic model
- Robust and adaptive variants have been proposed
- Complete control of the error dynamics is nominally achieved



POLITECNICO
MILANO 1863