ROBOTICS

Jean-Louis Boimond
University of Angers

Robotics can be defined as the set of techniques and studies tending to design mechanical, computer or mixed systems that can replace physical activity and decision making of human in the execution of a task.

Contents

Bibliographies

1) *Cours de robotique*, J. Gangloff, ENSPS 3A, 221 pages
2) *Robots. Principes et contrôle*, C. Vibet, Ellipses 1987, 207 pages
3) *Robotique. Aspects fondamentaux*, J.-P. Lallemand, S. Zeghloul, Masson 1994, 312 pages
4) *Introduction to Robotics Mechanics and Control,* 2th edition, J. J. Craig, Addison-Wesley Publishing Company, 1989, 450 pages
5) *Modeling, Identification & Control of Robots*, W. Khalil, E. Dombre, Hermes Penton Science 2002, 480 pages
6) *Robotics Modelling, Planning and Control,* B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Springer-Verlag 2009, 632 pages
7) *Robot Modeling and Control,* M. W. Spong, S. Hutchinson, M. Vidyasagar, John Wiley & Sons, 2005, 407 pages

1 GENERAL DEFINITIONS

A *robot manipulator* is characterized by: an *arm* to ensure mobility, a *wrist* to confer dexterity, and an *end-effector* to perform the required task of the robot. To design, simulate or control a robot manipulator, it is necessary to model its mechanism. Several levels of modelling are possible according to the specifications of the planned application (*geometrical*, *kinematic*, *static* or *dynamic models*). Obtaining these different models is not easy, the difficulty depends on the complexity of the mechanical structure (composed of a sequence of rigid bodies (*links*) interconnected by means of articulations (*joints*)). The number of *degrees of freedom*, the type of joints and the fact that the structure can be *serial* (simple open chain), *tree structured* or *closed* are all factors to take into account.

1.1 Definitions

A robot manipulator is commonly defined as an *automatic device capable of manipulating objects or executing operations according to a fixed or modifiable program*.

People's perception of a robot is usually vague. To merit the name of robot, a system must have a certain *versatility* (it must be able to perform a variety of tasks or the same task in different ways) and a certain *adaptability* (it must be able to adapt itself to a changing environment during the execution of its tasks).

A more thorough definition[1] is given below. A robot is *a mechanical manipulator controlled in position, reprogrammable, polyvalent (i.e., multiple use), with several degrees of freedom, capable of manipulating materials, parts, tools and specialized devices, during variable and programmed motions to perform a variety of tasks. It often has the appearance of one or more arms terminating in a wrist. Its control unit uses, among other things, a memory device and possibly perception and adaptation to the environment and circumstances. These multi-purpose machines are generally designed to perform the same function cyclically and can be adapted to other functions without permanent modification of the material.*

Historical timeline:

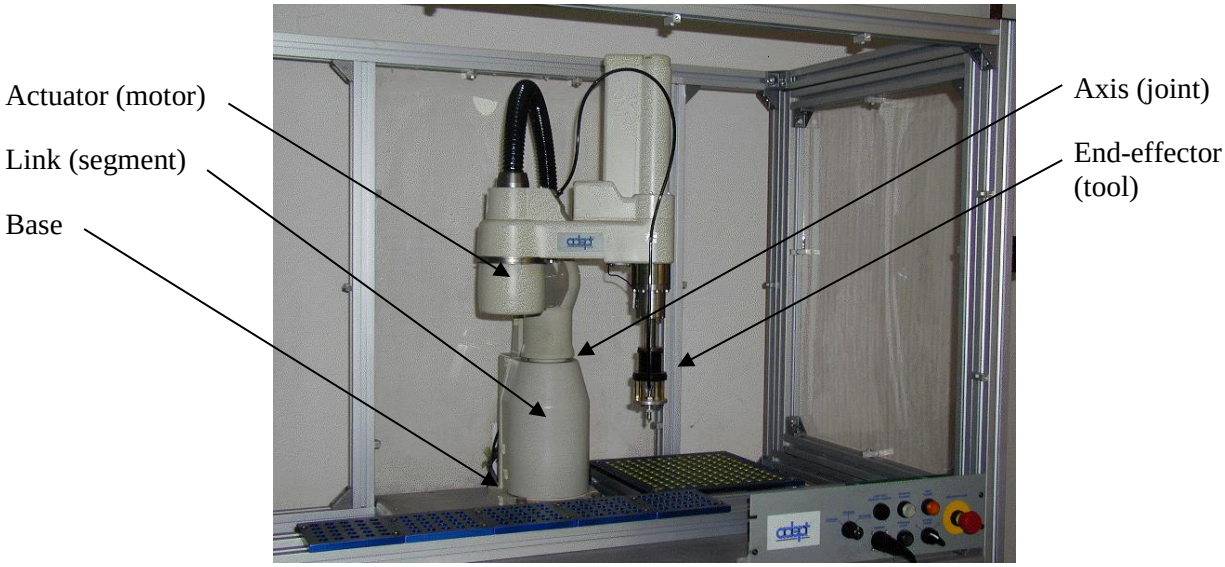
- 1947: First teleoperated electric manipulator,
- 1954: First programmable robot,
- 1961: First industrial robot from the company Unimation (USA) and used on a General Motors assembly line,
- 1961: First robot with force control,
- 1963: First time using vision to control a robot,
- 1973: First industrial robot with six electromechanically driven axes from the company Kuka (Germany).

See https://en.wikipedia.org/wiki/History_of_robots for a more complete history of robotics.

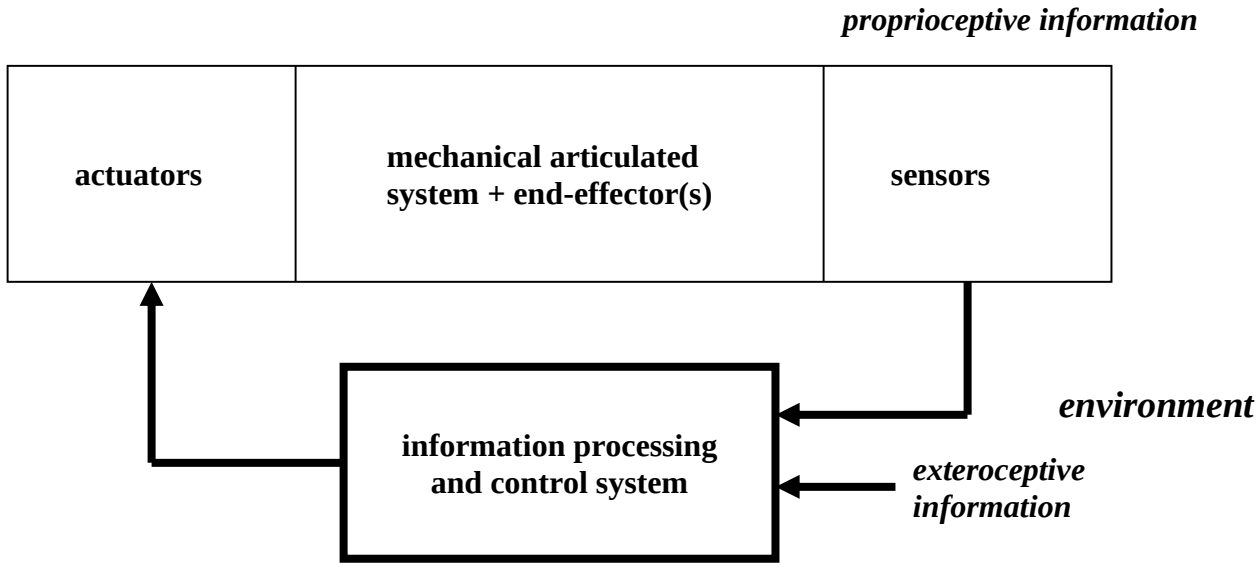
[1] definition from the "Association Française de Normalisation" (A.F.N.O.R.).

1.2 Robot components

Vocabulary:



There are classically four main parts in a robotic manipulator:



✓ The ***Mechanical Articulated System*** (M.A.S.) is a mechanism with a structure more or less similar to that of the human arm. It allows to replace or extend its action. Its role is to bring the terminal organ into a given *location* (position and orientation) according to given velocity and acceleration characteristics.

The terminal organ is named an ***end-effector*** (gripper or tool), it includes any device designed to manipulate objects (electric or pneumatic gripper, suction cup, etc.) or to process them (tool, welding torch, paint gun, etc.). It constitutes the interface allowing the free extremity of the M.A.S. to interact with its environment (the other extremity, called *base*, being fixed). An end-effector can be multi-functional in the sense that it can be equipped with several devices with different functionalities. It can also be mono-functional, but interchangeable. A robot may be multi-arm (tree structured chain), each arm carrying a different end-effector.

The architecture of a M.A.S. is a mechanical structure composed of *links*, generally rigid (or supposed as such), assembled by *joints*. Its motorization is usually achieved by electric or hydraulic actuators that transmit their motions to the joints through appropriate systems.

Let us precise the notion of ***joint***. A joint attaches two successive links by limiting the number of degrees of freedom (notion specified in §2.2) of one in relation to the other. Let m be the number of degrees of freedom of a joint, also called *mobility* of joint. The mobility of a joint is such that:

$$0 \leq m \leq 6.$$

The joint is said to be *simple* (*rotoid* or *prismatic*) when $m = 1$ which is commonly the case in industrial robotics.

- A ***rotoid*** joint is a pivot type joint, noted R, reducing the motion between two links to a rotation about a common axis. The relative situation between the two links is given by the angle about this axis (see the following figure).

Figure: Representation of a *rotoid* joint.

- A ***prismatic*** joint is a slide-type joint, noted P, reducing the motion between two links to a translation along a common axis. The relative situation between the two links is measured by the distance along this axis (see the following figure).

Figure: Representation of a *prismatic* joint.

The joint is said to be *complex* when $m \geq 2$. It can always be reduced to a combination of prismatic or rotoid joints. For example, a wrist is obtained with three rotoid joints whose axes are concurrent.

✓ In order to be driven, the M.A.S. comprises motors usually associated with transmissions (toothed belts, chains) which constitute the ***actuators***. Actuators frequently use permanent *electric* motors (generally the high velocity of the motor means that it is followed by a gearbox[2] in order to amplify the motor torque). More and more electronically commutated (brushless) motors (or stepper motors for small robots) are being used.
For robots that have to handle very heavy loads (e.g., an excavator), the actuators are usually *hydraulic*, acting in translation (hydraulic cylinder[3]) or rotation (hydraulic motor).

✓ Perception is used to manage the relationship between the robot and its environment. Perception devices are *proprioceptive*[4] ***sensors*** when they measure the internal state of the robot (joint positions and

[2] 'réducteur'.
[3] 'vérin'.
[4] proprioception: sensitivity specific to bones, muscles, tendons and joints, providing information on statics, balance, motion of the body in space, etc.

velocities) and *exteroceptive*[5] sensors when they collect information about the environment (presence detection, contact detection, distance measurement, artificial vision).

✓ The ***control*** part synthesizes the commands of the actuators, based on the perception function and the reference input of the user.

In addition to these parts:

- The man-machine interface through which the user programs the tasks that the robot is to perform,
- The workstation or the environment in which the robot operates.

> *Robotics is a multidisciplinary science that requires, in particular, knowledge of mechanics, automation, electronic, energetics, actuators, sensors, signal processing, communications and computer science knowing that improvements continue to be made in all these fields.*

1.3 Robot classification

There are basically three types of robots:

- The ***manipulators***:
  - The trajectories are not arbitrary in space,
  - The positions are discrete with two or three values per axis,
  - The control is sequential.
- ***Telemanipulators*** (remote handling devices as excavator, overhead crane[6]) appeared around 1945 in the USA:
  - Trajectories can be arbitrary in space,
  - Trajectories are defined instantaneously by the operator usually from a control panel (*joystick*).
- ***Robots***:
  - Trajectories can be arbitrary in space,
  - The execution is automatic,
  - Exteroceptive information can modify the robot's behaviour.

For the last category, we can distinguish:

1. ***Industrial robot manipulators*** for handling:
   Parts: Storage - destocking, palletizing - depalletizing, loading - unloading on machine tools, test tube handling, assembly of parts,
   Tools: Continuous or spot welding, painting, collage, deburring[7].

2. ***Training robots***: They are reduced versions of the previous robots, the technology is different. They have a training and teaching role, they can also be used to carry out feasibility tests of a robotic station.

3. ***Autonomous mobile robots***: The possibilities are wider due to their mobility. They can be used in dangerous areas (nuclear, fire, civil security, demining) or inaccessible areas (oceanography, space). Such robots use sophisticated sensors and software. We can distinguish two types of locomotion: the *walking robots* that imitate the human gait, and the *mobile robots* that look more like vehicles.

Only *robot manipulators* are tackled in the following.

---

[5] exteroceptive information: information coming from sensory receptors located on the surface of the body and stimulated by agents external to the body (heat, sting).
[6] 'pont roulant'.
[7] 'ébavurage'.

1.4 Characteristics of a robot

A robot must be chosen according to the planned application. Here are some parameters to consider:

- The *maximum transportable load* (from a few kilos to a few tons) to be determined under the most unfavourable conditions (in maximum elongation);
- The *architecture* of the M.A.S., the choice is guided by the task to realize (how rigid is the structure?);
- The *workspace*, defined as all the points that can be reached by the end-effector. Not all motions are possible at every point of the workspace. The *reachable workspace*, also called the maximum workspace, is the volume of space that the robot can reach *via* at least one orientation of the end-effector. The *dextrous workspace* is the volume of space that the robot can reach with all possible orientations of the end-effector, this workspace is a subset of the reachable workspace;
- The *position accuracy* corresponds to the error between the planned point (defined by its location in Cartesian space) and the point reached (computed *via* the inverse geometric model of the robot). This error is mainly due to the model used and the flexibility of the mechanical system. In general, the position accuracy is in the order of 1 *mm*;
- The *position repeatability* characterizes the ability of the robot to return to a given point (expressed by a location). It corresponds to the maximum positioning error on a predefined point in the case of repetitive paths. In general, position repeatability is in the order of 0.1 *mm*;
- *Resolution* corresponds to the smallest increment of motion that can be achieved by the joint or the end-effector;
- Motion velocity (maximum velocity in maximum elongation), acceleration;
- The mass of the robot;
- The cost of the robot and its maintenance.

1.5 Robot programming

Typically, two steps are used to make a robot know the task to execute.

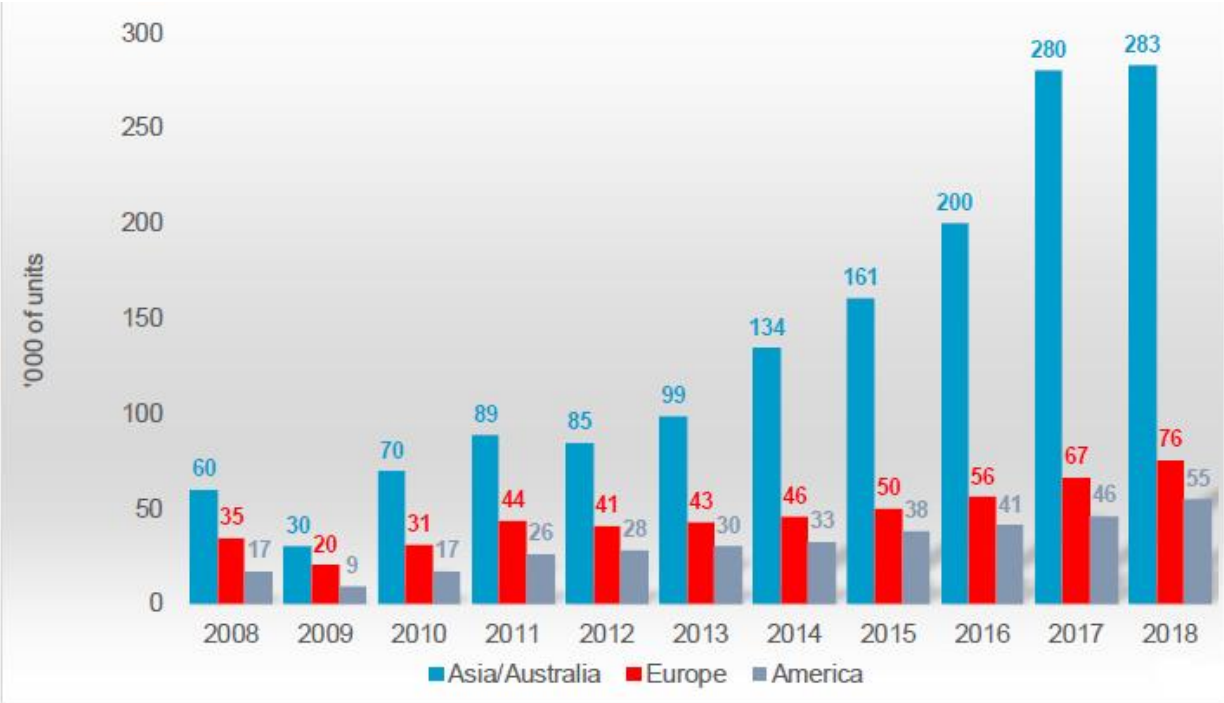
1. The *learning process* (under the control of a human operator) during which the trajectories to be executed are recorded in a memory. The use of a joystick or a *puppet*[8] (mechanical structure identical to that of the robot) to control the end-effector motion allows the storage of the "relevant" points.

2. The definition of the tasks (the operations) to be achieved along the trajectories. Based on the model of the robot and the trajectories to be achieved, a software program is used to elaborate the sequences of control for the actuators. We will use in the course the languages associated with the robots Stäubli (language VAL), Fanuc (language Karel), Kuka (Kuka Robot Language – KRL), Universal Robots (language URScript).

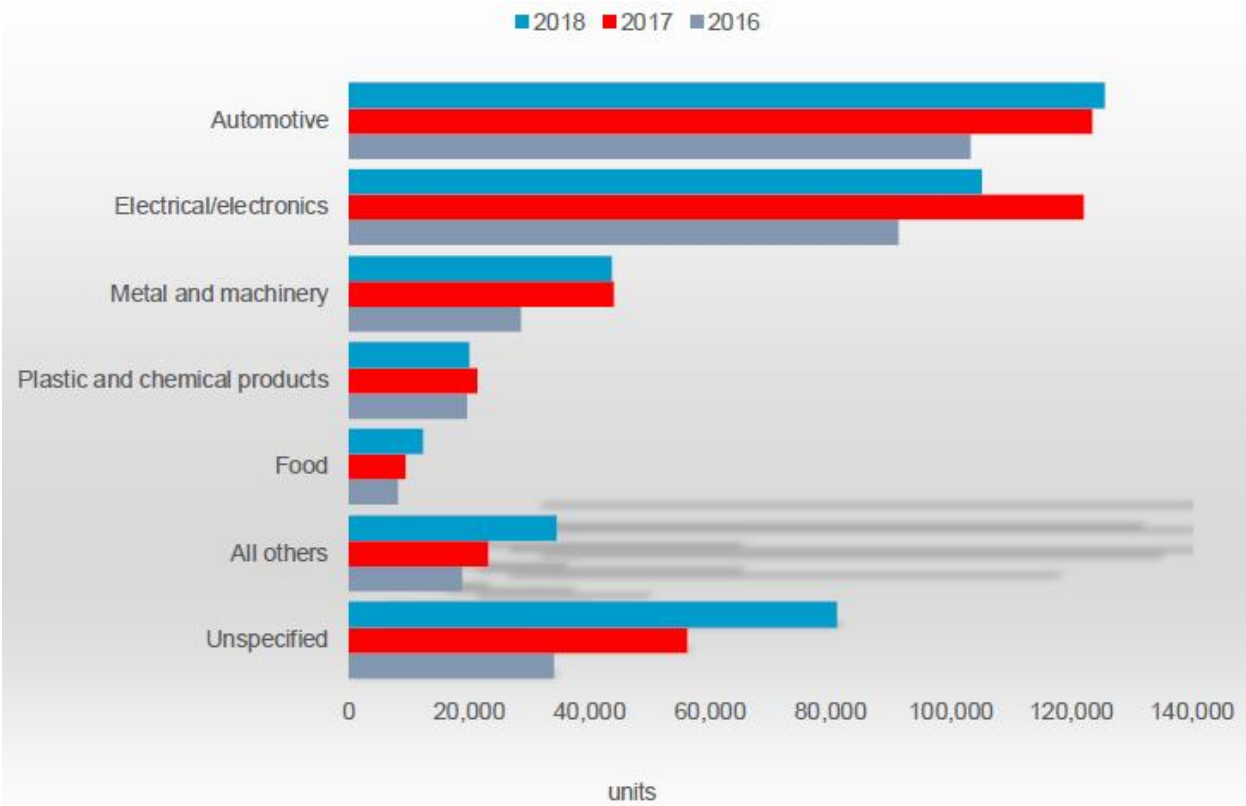
[8] 'pantin'.

1.6 Few stats

Annual installations of industrial robots by regions (more than 410 000 robots in 2018).



Annual installations of industrial robots by industries 2016-2018



World's Top 10 industrial robotics manufacturers are (in the order): ABB (Switzerland), Yaskawa (Japan), Kuka (Germany), Fanuc (Japan), Kawasaki (Japan), Epson (Japan), Stäubli (Switzerland), Nachi (Japan), Comau (Italy), Omron Adept (USA). Japan supplies about 50% of the world's industrial robots.

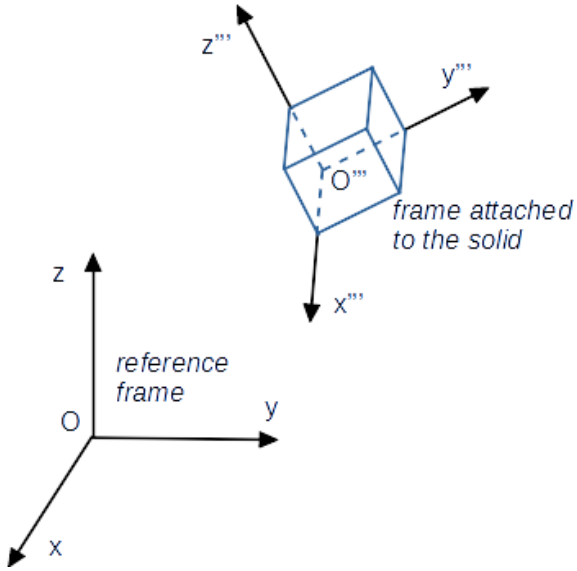
2 DEGREE OF FREEDOM - ARCHITECTURE

We suppose in the following that frames are orthonormal in the Euclidean space $\mathbb{R}^3$, that is, the three vectors defining a frame are unit and orthogonal to each other.

2.1 The location of a solid in space

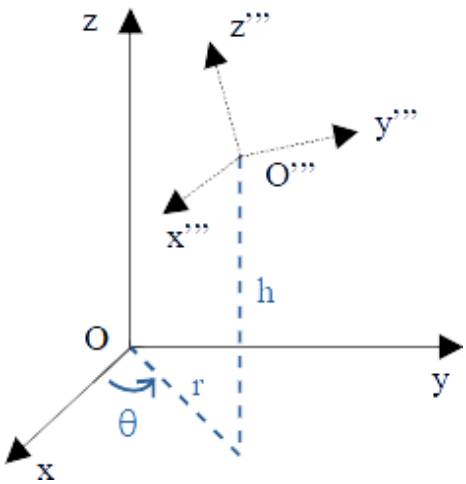
The ***location*** (position and orientation) in space of the frame $\left(O''', \overrightarrow{x'''}, \overrightarrow{y'''}, \overrightarrow{z'''}\right)$ attached to an arbitrary solid with respect to a (fixed) *reference frame* $(O, \vec{x}, \vec{y}, \vec{z})$, see the following figure, requires 6 independent parameters:

- 3 independent parameters to define the ***position*** of point O''' attached to the solid,
- 3 independent parameters to define the ***orientation*** of the frame $\left(O''', \overrightarrow{x'''}, \overrightarrow{y'''}, \overrightarrow{z'''}\right)$.



➢ **Position**

Cartesian (3 lengths), cylindrical (2 lengths and 1 angle) or spherical (1 length and 2 angles) coordinates can be used to define the *position*.
Example of cylindrical coordinates to position the point O''' with respect to the reference frame $(O, \vec{x}, \vec{y}, \vec{z})$:



In practise, industrial robots make use of the Cartesian coordinates for the position even though the cylindrical or spherical representations could be more adapted for some structures of robots.

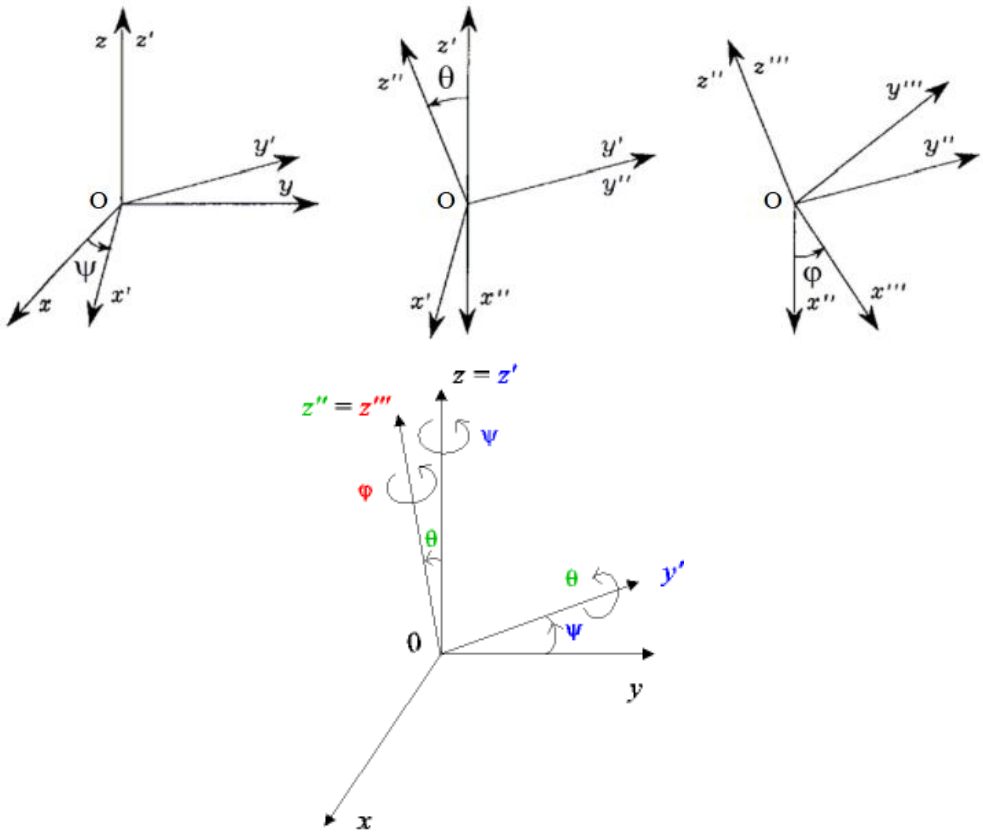
➢ Orientation

▪ *Euler angles*

The *orientation* of the frame $\left(O, \overrightarrow{x'''}, \overrightarrow{y'''}, \overrightarrow{z'''}\right)$ with respect to the reference frame $(O, \vec{x}, \vec{y}, \vec{z})$ is specified by 3 angles (ψ (psi), θ (theta) and φ (phi)) corresponding to a sequence of 3 rotations. Such angles are widely used in mechanics and allow a *minimal representation* of the orientation.

The angles ψ, θ, φ are defined in the following figure according to the convention (z, y, z) as:

- a first rotation of an angle ψ about the $\overrightarrow{Oz}$ axis (that is, the line generated by the $\overrightarrow{Oz}$ vector),
- a second rotation of an angle θ about the $\overrightarrow{Oy'}$ axis,
- a third rotation of an angle φ about the $\overrightarrow{Oz''}$ axis.



An animation on Euler angles ψ, θ, φ according to the convention (z, x, z) is available through this link.

Several choices are possible about the convention linked to Euler angles (ψ, θ, φ)[9]: (z, y, z) for the robot Stäubli RX 90, (x, y, z) for the robots Fanuc ARC or LR, (z, y, x) for the robot Kuka KR 3.

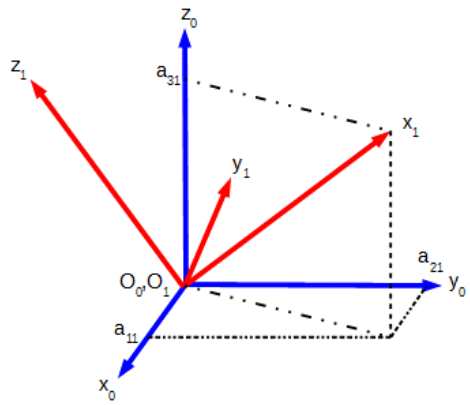
▪ *Direction cosines*

The composition of motions is difficult to apprehend by Euler angles which explains the use of *direction cosines* in robotics as we see later. Let us deal with the *orientation* of the frame $R_1 = (O_1, \overrightarrow{x_1}, \overrightarrow{y_1}, \overrightarrow{z_1})$ with respect to the reference frame $R_0 = (O_0, \overrightarrow{x_0}, \overrightarrow{y_0}, \overrightarrow{z_0})$. The *direction cosines* computing consists of considering the projections of unit vectors of frame R_1 on the unit vectors of frame R_0 which leads to 3×3 parameters. Indeed:

- 6 relationships are needed to indicate that the frame is orthonormal (3 to indicate unit norms + 3 to indicate frame orthogonality),
- and 3 parameters are needed to describe the orientation of the frame.

[9] 12 $(= 3 \times 2 \times 2)$ distinct sets of Euler angles are allowed out of all 27 $(= 3 \times 3 \times 3)$ possible combinations.

The unit vector $\overrightarrow{x_1}$ (of frame R_1) is expressed in the frame R_0 by the relation: $\overrightarrow{x_1} = a_{11}\overrightarrow{x_0} + a_{21}\overrightarrow{y_0} + a_{31}\overrightarrow{z_0}$, see the following figure.



In the same way, we have: $\overrightarrow{y_1} = a_{12}\overrightarrow{x_0} + a_{22}\overrightarrow{y_0} + a_{32}\overrightarrow{z_0}$ and $\overrightarrow{z_1} = a_{13}\overrightarrow{x_0} + a_{23}\overrightarrow{y_0} + a_{33}\overrightarrow{z_0}$.

The vector $(a_{11} \quad a_{21} \quad a_{31})^t$ represents the unit vector $\overrightarrow{x_1}$ along the $\overrightarrow{o_0x_0}$, $\overrightarrow{o_0y_0}$, $\overrightarrow{o_0z_0}$ axes (of frame R_0). In the same way, vectors $(a_{12} \quad a_{22} \quad a_{32})^t$ and $(a_{13} \quad a_{23} \quad a_{33})^t$ represent the unit vectors $\overrightarrow{y_1}$ and $\overrightarrow{z_1}$ respectively which leads to consider the following rotation matrix:

$$\begin{array}{c} \\ \overrightarrow{x_0} \\ \overrightarrow{y_0} \\ \overrightarrow{z_0} \end{array} \begin{array}{c} \begin{array}{ccc} \overrightarrow{x_1} & \overrightarrow{y_1} & \overrightarrow{z_1} \end{array} \\ \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \end{array}.$$

This rotation matrix satisfies 6 relationships among its 9 parameters (due to orthonormality of frame R_1), that is:

$$\|\overrightarrow{x_1}\| = \|\overrightarrow{y_1}\| = \|\overrightarrow{z_1}\| = 1, \quad \overrightarrow{x_1} \cdot \overrightarrow{y_1} = \overrightarrow{x_1} \cdot \overrightarrow{z_1} = \overrightarrow{y_1} \cdot \overrightarrow{z_1} = 0^{10}.$$

A solid situated in space has 6 *degrees of freedom* (*d.o.f.*). Conversely, 6 independent control variables are needed to place a solid in space in any *location* (position and orientation). So, in practice, the most common robots are equipped with 6 *d.o.f.* which allows to place an arbitrary object in space.
However, 5 *d.o.f.* are sufficient if the object manipulated by the robot exhibits a revolution symmetry in the sense that it is not necessary to specify the rotation about the revolution axis. In the same way, only 3 *d.o.f.* are needed to locate an object in a plane (2 *d.o.f.* for positioning a point in the plane and 1 *d.o.f.* to determine the orientation of the object to manipulate).

2.2 Joint

A *joint* between 2 successive rigid (or supposed as such) links limits the *d.o.f.* of one link in relation to the other. The *d.o.f. of the joint* is the number of independent parameters that is needed to define the location (position and orientation) of one link with respect to the other in any motion compatible with the joint.

Examples:

- A door in relation to the wall has 1 *d.o.f.*;
- A cube situated on a plane has 3 *d.o.f.*: 2 *d.o.f.* to fix the coordinates of a point of the cube in the plane, 1 *d.o.f.* to determine its orientation in the plane;
- A sphere situated on a plane has 5 *d.o.f.*: 2 *d.o.f.* to fix the coordinates of the point of the sphere in the plane, 3 *d.o.f.* to determine its orientation in the plane.

---

10 $\|\overrightarrow{x_1}\| = \sqrt{a_{11}^2 + a_{21}^2 + a_{31}^2}$ and $\overrightarrow{x_1} \cdot \overrightarrow{y_1} = a_{11}a_{12} + a_{21}a_{22} + a_{31}a_{32}$.

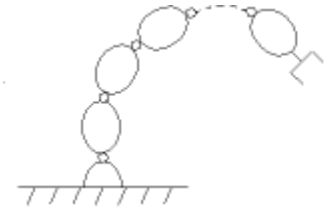
2.3 Mechanisms

A *mechanism* is a set of *links* connected two by two by *joints*. There are two types of *mechanisms*:

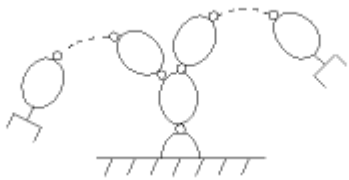
- *Simple open* (or *serial*) *chains* if one never passes on the same joint (or on the same link) twice when one goes through the mechanism. This is the most commonly used mechanism;
- *Complex chains* when they are not serial (at least one link has more than 2 joints). Such systems are subdivided into 2 groups: *chains structured as a tree*, and *closed chains* (whose advantage is to be *a priori* more rigid, more precise, able to handle heavy loads). As an example, the *pantograph*[11] is a *closed chain*.

Two methods are available to represent a mechanism:

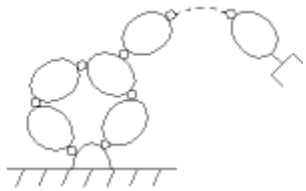
- The kinematic[12] diagram: The normalized representation of the links (in perspective or in projection) is used to represent the mechanism;
- The graph (not normalized): As examples, let us consider the following mechanisms.



simple open chain



tree-structured



closed chain

2.4 Morphology of manipulator robots

In the following section, we deal with simple open chains. Only two parameters are considered to enumerate the different possible architectures: the type of joint (rotoid (*R*) or prismatic (*P*)) and the angle between two successive joint axes (0° or 90°; except in very special cases, the consecutive axes of a robot are either parallel or orthogonal).

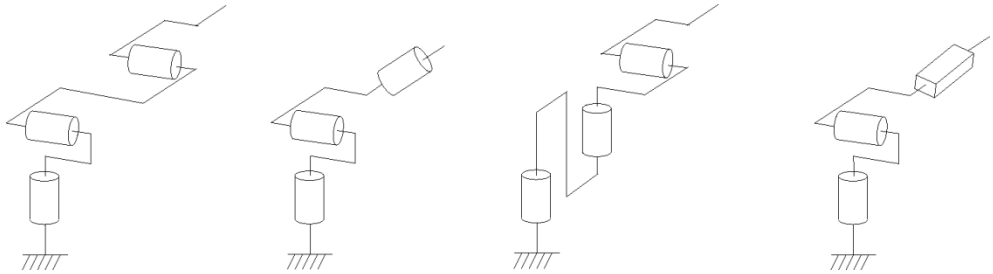
The first 3 *d.o.f.* form the ***shoulder*** of the robot. The remaining *d.o.f.* form the ***wrist*** which is characterized by much smaller dimensions and lower mass of links.

The 12 possible ***shoulder*** morphologies are schematized in the following figure (these morphologies are non-redundant in the sense that the structures limiting the motions of the shoulder to linear or planar displacements (e.g., 3 prismatic joints of parallel axes or 3 rotoid joints of parallel axes) are eliminated).

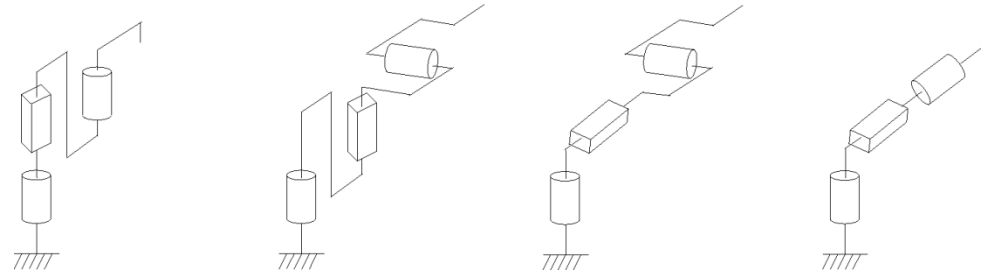
[11] A ***pantograph*** is an instrument consisting of 4 articulated bars used to mechanically reproduce a drawing, possibly at a different scale (see https://en.wikipedia.org/wiki/Pantograph).

[12] Relating to motion.

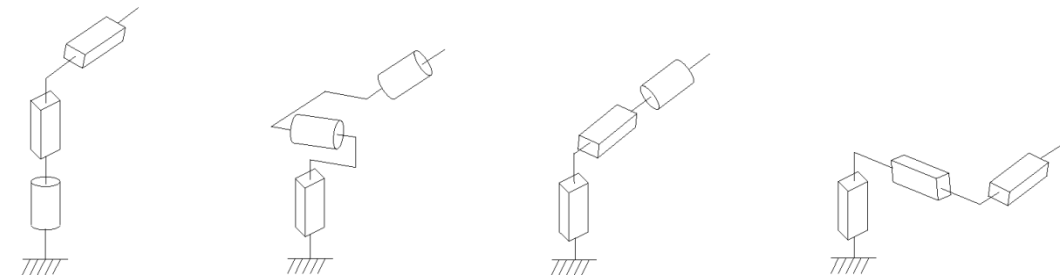
R R R **R R P**



R P R



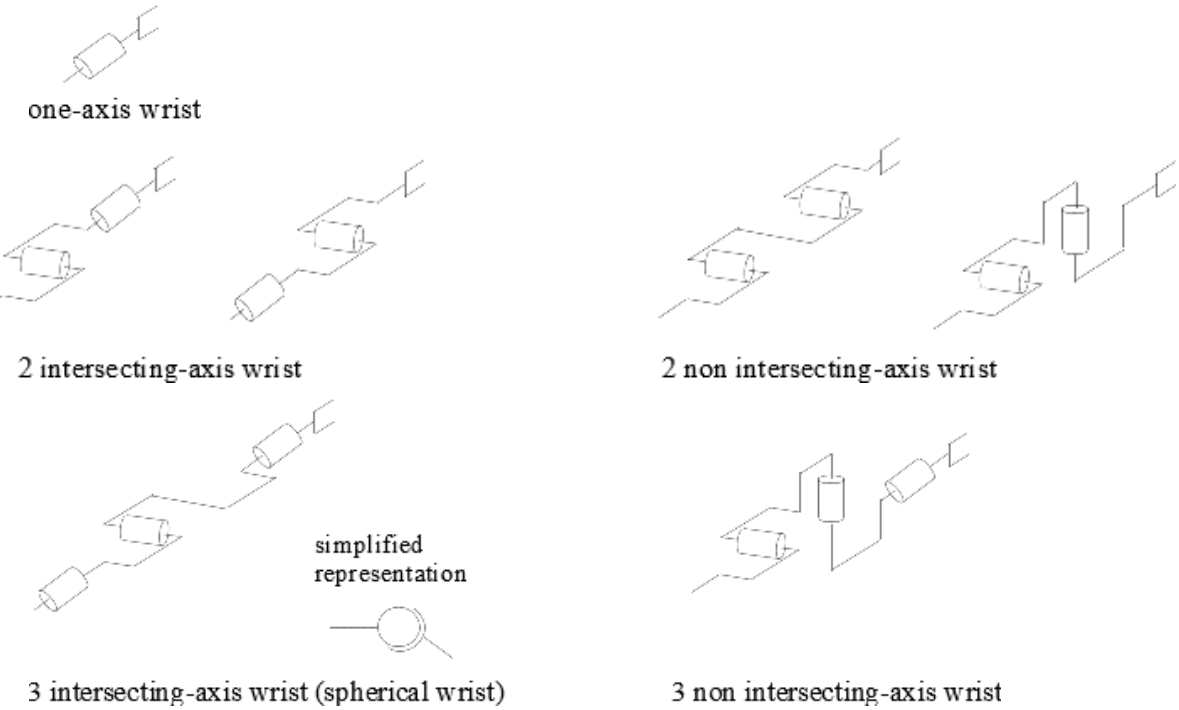
R P P **P R R** **P P R** **P P P**



In practice, the following five structures are manufactured:

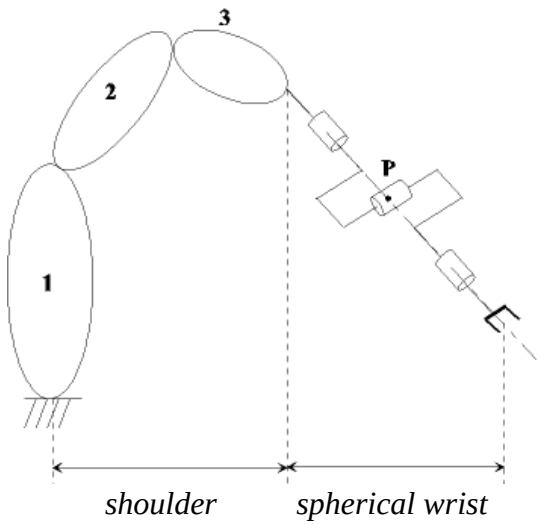
- *Anthropomorphic shoulder* (*RRR*), more precisely the first structure (see previous figure), such as robots FANUC (LR, ARC), STÄUBLI RX, ACMA (V80 et SR400), UNIMATION (PUMA), SCEMI (6P-01), AID (V5), CINCINNATI (T3-7XX), AKR 3000, ASEA (IRB6 et 60), KUKA (KR 3), AXEA (V08);
- *Spherical shoulder* (*RRP*) such as robots STANFORD, UNIMATION (1000, 2000, 4000), PSA (BARNABE);
- *Toric shoulder* (*RPR*), more precisely the first structure, such as robots ACMA (H80), robots of type SCARA;
- *Cylindrical shoulder* (*RPP*) such as robots ACMA (TH8), MANTEC (A, I, M), CINCINNATI (T3-363);
- *Cartesian shoulder* (*PPP*) such as robots ACMA (P80), IBM (7565), SORMEL (CADRATIC), OLIVETTI (SIGMA).

Wrists with 1, 2 or 3 axes are shown in the following figure. The *RRR* structure with the three intersecting-axis, called *spherical wrist*, forms the most common wrist.



The robot obtained by associating a *shoulder* (with 3 *d.o.f.*) and a *spherical wrist* (3 *d.o.f.*) is the most classical structure, as illustrated in the following figure. It allows a decoupling between the *position* and the *orientation* of the end-effector:

- The role of the *shoulder* is to fix the *position* of the point *P* situated at the intersection of the last 3 joint axes (centre of the wrist) knowing that this position (*P*) depends only on the values applied on joints 1, 2 and 3 (*i.e.*, of the shoulder),
- The *wrist* is used for the *end-effector orientation*.

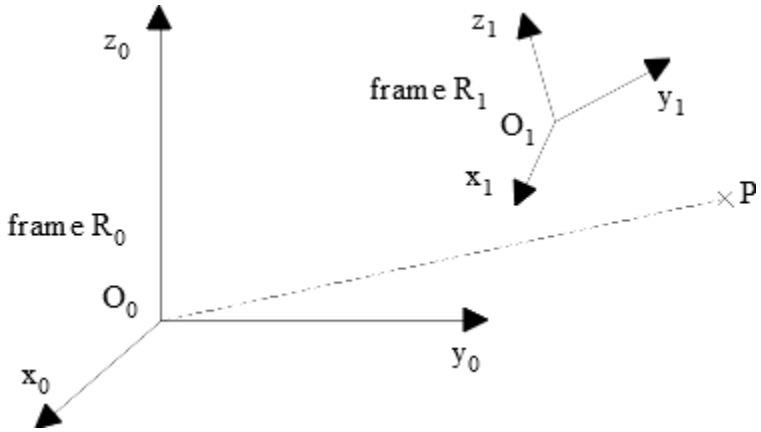


3 TRANSFORMATION MATRIX

A frame is assigned at each link of a robot thus the transformation of frames is an important step in the modelling of a robot.

3.1 Translation and rotation

It can be shown that the transformation of a frame R_1 into a (reference) frame R_0, or equivalently, the location of a frame R_1 with respect to a frame R_0, can be deduced by *a translation* and/or *a rotation*.



Let (X_1, Y_1, Z_1) be the Cartesian coordinates of an arbitrary point P with respect to frame R_1. Let us express the Cartesian coordinates of the point P with respect to frame R_0 knowing that (a, b, c) are the Cartesian coordinates of the origin (O_1) of frame R_1.

We have by definition: $\overrightarrow{O_1P}_{/1} = (X_1\ Y_1\ Z_1)^t$, or equivalently, $\overrightarrow{O_1P} = X_1\,\vec{x_1} + Y_1\,\vec{y_1} + Z_1\,\vec{z_1}$.

We deduced by using rule of Chasles that:

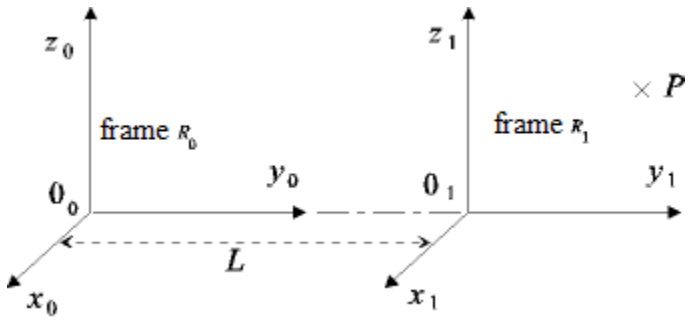
$$\begin{aligned}\overrightarrow{O_0P}_{/0} &= \overrightarrow{O_0O_1}_{/0} + \overrightarrow{O_1P}_{/0} \\ &= \overrightarrow{O_0O_1}_{/0} + R_{0,1} \times \overrightarrow{O_1P}_{/1} \\ &= \begin{pmatrix} a \\ b \\ c \end{pmatrix} + \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \times \begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \end{pmatrix}.\end{aligned}$$

The *rotation matrix* $\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$, denoted $R_{0,1}$, expresses the unit vectors of frame R_1 (that is, $\vec{x}_1, \vec{y}_1, \vec{z}_1$) with respect to frame R_0, *i.e.*, relative to unit vectors $\vec{x}_0, \vec{y}_0, \vec{z}_0$.

The *location* of frame R_1 with respect to frame R_0 can be deduced from:

- the translation vector $\overrightarrow{O_0O_1}_{/0} = (a\ b\ c)^t$,
- the rotation matrix $R_{0,1}$.

➢ *Case of a pure translation*



We deduce from the previous figure that the Cartesian coordinates of the point P with respect to frame R_0 are such that:

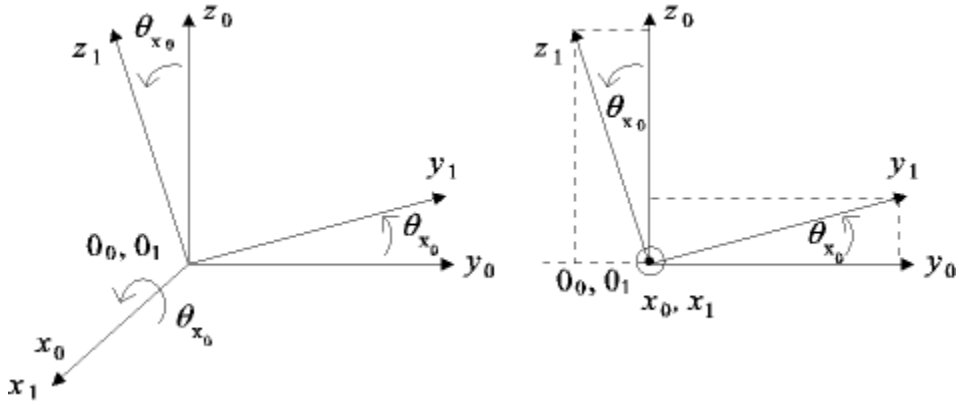
$$\overrightarrow{O_0P}_{/0} = \overrightarrow{O_0O_1}_{/0} + R_{0,1} \times \overrightarrow{O_1P}_{/1}$$
$$= \begin{pmatrix} 0 \\ L \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \end{pmatrix} = \begin{pmatrix} X_1 \\ L + Y_1 \\ Z_1 \end{pmatrix}.$$

The translation vector is along the $\overrightarrow{O_0y_0}$ axis. The rotation matrix (by null angle) is such that:

$$\overrightarrow{x_1} = \overrightarrow{x_0},\ \overrightarrow{y_1} = \overrightarrow{y_0},\ \overrightarrow{z_1} = \overrightarrow{z_0}.$$

➢ *Case of a rotation about the $\overrightarrow{O_0x_0}$ axis*

By example, let us consider the rotation about the $\overrightarrow{O_0x_0}$ axis by an angle θ_{x_0} as described below.



We deduce from the previous figure that: $R_{0,1}(\overrightarrow{x_0},\ \theta_{x_0}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_{x_0}) & -sin(\theta_{x_0}) \\ 0 & sin(\theta_{x_0}) & cos(\theta_{x_0}) \end{pmatrix}$,

or equivalently that:

$$\overrightarrow{x_1} = \overrightarrow{x_0},\ \ \overrightarrow{y_1} = cos(\theta_{x_0})\,\overrightarrow{y_0} + sin(\theta_{x_0})\,\overrightarrow{z_0},\ \ \overrightarrow{z_1} = -sin(\theta_{x_0})\,\overrightarrow{y_0} + cos(\theta_{x_0})\,\overrightarrow{z_0}.$$

Remark: We have $\|\overrightarrow{x_1}\| = \|\overrightarrow{y_1}\| = \|\overrightarrow{z_1}\| = 1,\ \ \overrightarrow{x_1} \cdot \overrightarrow{y_1} = \overrightarrow{x_1} \cdot \overrightarrow{z_1} = \overrightarrow{y_1} \cdot \overrightarrow{z_1} = 0$ due to orthonormality of frame R_1.

Exercise: Express the rotation matrices $R_{0,1}(\overrightarrow{y_0},\ \theta_{y_0})$ and $R_{0,1}(\overrightarrow{z_0},\ \theta_{z_0})$.

➢ *Example of a translation and a rotation about the $\overrightarrow{O_0x_0}$ axis by an angle θ_{x_0}*

Let (X_1, Y_1, Z_1) be the Cartesian coordinates of an arbitrary point P with respect to frame R_1 and (a, b, c) the ones of the origin (O_1) of frame R_1 with respect to frame R_0. Let us express the Cartesian coordinates of point P with respect to frame R_0.

We have:

$$\begin{aligned}\overrightarrow{O_0P}_{/0} &= \overrightarrow{O_0O_1}_{/0} + \overrightarrow{O_1P}_{/0} \\ &= \overrightarrow{O_0O_1}_{/0} + R_{0,1}(\overrightarrow{x_0}, \theta_{x_0}) \times \overrightarrow{O_1P}_{/1} \\ &= \begin{pmatrix} a \\ b \\ c \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_{x_0}) & -sin(\theta_{x_0}) \\ 0 & sin(\theta_{x_0}) & cos(\theta_{x_0}) \end{pmatrix} \times \begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \end{pmatrix} \\ &= \begin{pmatrix} a + X_1 \\ b + cos(\theta_{x_0})\ Y_1 - sin(\theta_{x_0})\ Z_1 \\ c + sin(\theta_{x_0})\ Y_1 + cos(\theta_{x_0})\ Z_1 \end{pmatrix}.\end{aligned}$$

Exercise: Express the matrix of direction cosines corresponding to the Euler angles ψ, θ, φ defined according to the convention (z, y, z), and vice versa.

3.2 Homogeneous transformation matrix

The simultaneous presence of products and sums in the vector equation $\overrightarrow{O_0P}_{/0} = \overrightarrow{O_0O_1}_{/0} + R_{0,1} \times \overrightarrow{O_1P}_{/1}$ is not suitable for performing computation, for example when successive changes of frames must be done which is often the case in robotics. A matrix representation of dimension 4 based on homogeneous coordinates is more efficient.

Representation of a point: Let (X, Y, Z) be the Cartesian coordinates of an arbitrary point *P* with respect to an orthonormal frame with origin *O*, the ***homogeneous coordinates of point P*** are expressed as quaternion (wX, wY, wZ, w) knowing that w is a scaling factor equal to 1 in robotics which leads to:

$$\overrightarrow{OP} = (X\ \ Y\ \ Z\ \ 1)^t.$$

Representation of a direction: Let (u_x, u_y, u_z) be the Cartesian coordinates of a unit vector $\vec{u}$ with respect to an orthonormal frame, the ***homogeneous coordinates of a direction*** indicated by vector $\vec{u}$ are expressed as quaternion:

$$\vec{u} = (u_x\ \ u_y\ \ u_z\ \ 0)^t.$$

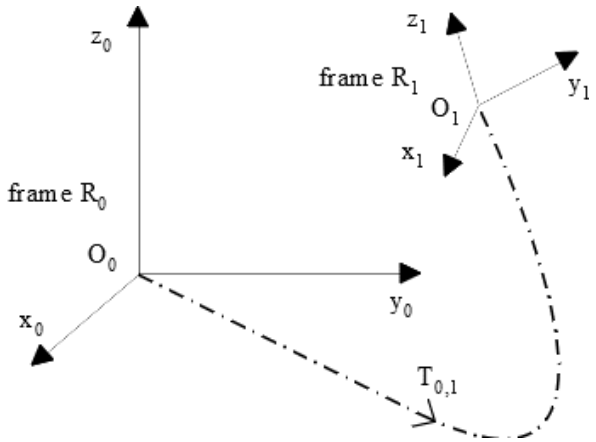
Transformation of frames: The transformation, through a translation and/or a rotation, of a frame R_0 into a frame R_1, see the following figure, is given by the (4×4) ***homogeneous transformation matrix*** $\boldsymbol{T_{0,1}}$ such that:

$$T_{0,1} = \begin{pmatrix} R_{0,1} & t_{0,1} \\ 0\ 0\ 0 & 1 \end{pmatrix},$$

where:

- $R_{0,1}$ is the (3×3) rotation matrix containing the components of the unit vectors $\overrightarrow{x_1}, \overrightarrow{y_1}, \overrightarrow{z_1}$ expressed in frame R_0,
- $t_{0,1}$ is the (3×1) translation vector containing the coordinates of the origin O_1 (of frame R_1) expressed in frame R_0.

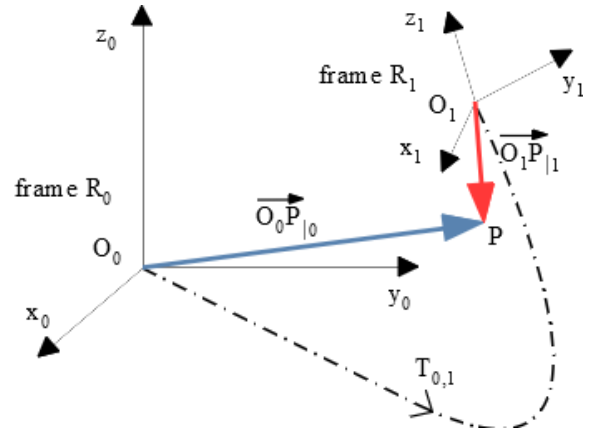
The matrix $T_{0,1}$ can also be interpreted as representing frame R_1 with respect to frame R_0, or equivalently, as defining frame R_1 relative to frame R_0.



Transformation of vectors: Let (X_1, Y_1, Z_1) be the Cartesian coordinates of an arbitrary point P with respect to frame R_1 then the ***homogeneous coordinates of point P*** with respect to frame R_0, that is, $(X_0, Y_0, Z_0, 1)$, can be obtained as:

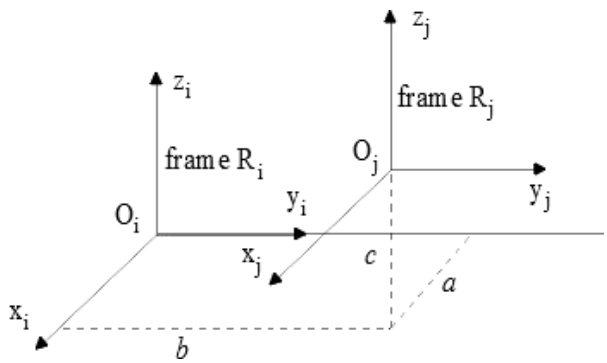
$$\begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{pmatrix} = T_{0,1} \begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{pmatrix}, \text{ or equivalently, } \overrightarrow{O_0P}_{|0} = T_{0,1}\, \overrightarrow{O_1P}_{|1},$$

see the following figure.



Examples of homogeneous transformations:

- **Case of a pure translation**
  Let $Trans(\vec{x}, a)$ be the homogeneous transformation matrix of translation along the $\vec{x}$ axis by a value a. Let us consider the following example:



We deduce from the previous figure that:

$$T_{i,j} = Trans(\vec{x}_i, a) \times Trans(\vec{y}_i, b) \times Trans(\vec{z}_i, c)$$
$$= \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

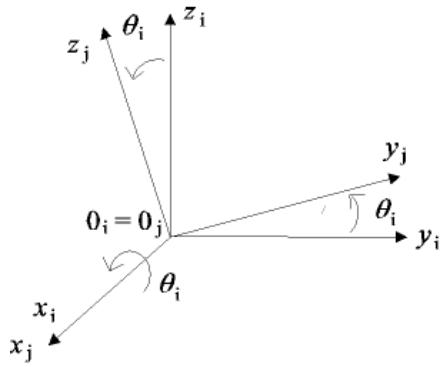
Let (X_j, Y_j, Z_j) be the Cartesian coordinates of an arbitrary point P with respect to frame R_j then the ***homogeneous coordinates of point P*** with respect to frame R_i, that is, $(X_i, Y_i, Z_i, 1)$, can be obtained as:

$$\begin{pmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{pmatrix} = T_{i,j} \times \begin{pmatrix} X_j \\ Y_j \\ Z_j \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} X_j \\ Y_j \\ Z_j \\ 1 \end{pmatrix} = \begin{pmatrix} X_j + a \\ Y_j + b \\ Z_j + c \\ 1 \end{pmatrix}.$$

Exercice: Compute the coordinates of the origin O_j with respect to frame R_i.

- **Case of a pure rotation**

Let $Rot(\vec{x}, \theta)$ be the homogeneous transformation matrix of rotation about the $\vec{x}$ axis by an angle θ. Let us consider the following example:



We deduce from the previous figure that:

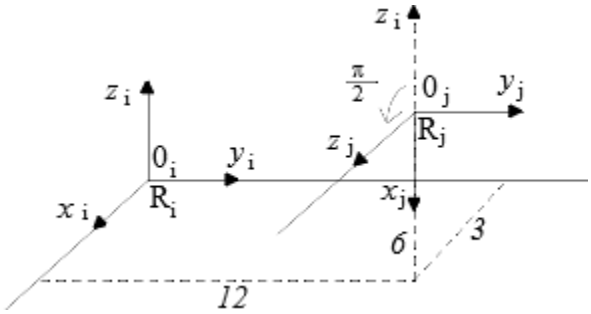
$$T_{i,j} = Rot(\overrightarrow{x_i}, \theta_i)$$
$$= \begin{pmatrix} \begin{pmatrix} & R_{i,j}(\overrightarrow{x_i}, \theta_i) & \end{pmatrix} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta_i) & -sin(\theta_i) & 0 \\ 0 & sin(\theta_i) & cos(\theta_i) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Let (X_j, Y_j, Z_j) be the Cartesian coordinates of an arbitrary point P with respect to frame R_j then the ***homogeneous coordinates of point P*** with respect to frame R_i, that is, $(X_i, Y_i, Z_i, 1)$, can be obtained as:

$$\begin{pmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{pmatrix} = T_{i,j} \times \begin{pmatrix} X_j \\ Y_j \\ Z_j \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta_i) & -sin(\theta_i) & 0 \\ 0 & sin(\theta_i) & cos(\theta_i) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} X_j \\ Y_j \\ Z_j \\ 1 \end{pmatrix} = \begin{pmatrix} X_j \\ cos(\theta_i)\ Y_j - sin(\theta_i)\ Z_j \\ sin(\theta_i)\ Y_j + cos(\theta_i)\ Z_j \\ 1 \end{pmatrix}.$$

Exercise:

- Compute the homogeneous transformation matrix $T_{i,j}$ corresponding to the following change of frames.



- Deduce the coordinates of the point O_j with respect to frame R_i.

Properties of homogeneous transformation matrices: Let us write a transformation matrix T as:

$$T = \begin{pmatrix} A & P \\ 0\ 0\ 0 & 1 \end{pmatrix},$$

where matrix (3×3) A represents the rotation whereas the vector (3×1) P represents the translation.

- The product of two homogeneous transformation matrices gives a homogeneous transformation matrix:

$$T_1 T_2 = \begin{pmatrix} A_1 & P_1 \\ 0\ 0\ 0 & 1 \end{pmatrix} \begin{pmatrix} A_2 & P_2 \\ 0\ 0\ 0 & 1 \end{pmatrix} = \begin{pmatrix} A_1 A_2 & A_1 P_2 + P_1 \\ 0\ 0\ 0 & 1 \end{pmatrix}.$$

- The rotation matrix A is *orthogonal* (*i.e.* its inverse is equal to its transpose: $A^{-1} = A^t$) and its determinant is equal to 1: $\det(A) = 1$. Such a matrix represents an orthonormal frame (its column vectors are normal to each other and have a norm equal to 1).

- The product of homogeneous transformation matrices of translation $Trans$ is commutative:

$$Trans(\vec{x}, a)\ Trans(\vec{y}, b) = Trans(\vec{y}, b)\ Trans(\vec{x}, a).$$

The inversion of homogeneous transformation matrix of translation is obtained by a simple change of sign:

$$(Trans(\vec{x}, a))^{-1} = Trans(\vec{x}, -a).$$

- The product of rotation matrices A_1, A_2 is not commutative: $A_1 \times A_2 \neq A_2 \times A_1$ except in particular cases (notably when rotation is about the same axis). So, the product of homogeneous transformation matrices is not commutative (except in particular cases).

- The inversion of homogeneous transformation matrix of rotation Rot about $\vec{u}$ axis is obtained by a simple change of sign:

$$(Rot(\vec{u}, \theta))^{-1} = Rot(\vec{u}, -\theta).$$

- The inverse of homogeneous transformation matrix T can be obtained as:

$$T^{-1} = \begin{pmatrix} A^t & -A^t P \\ 0\ 0\ 0 & 1 \end{pmatrix}.$$

- The product between a homogeneous transformation matrix of rotation about $\vec{u}$ axis and a homogeneous transformation matrix of translation along **the same axis** ($\vec{u}$) is commutative:

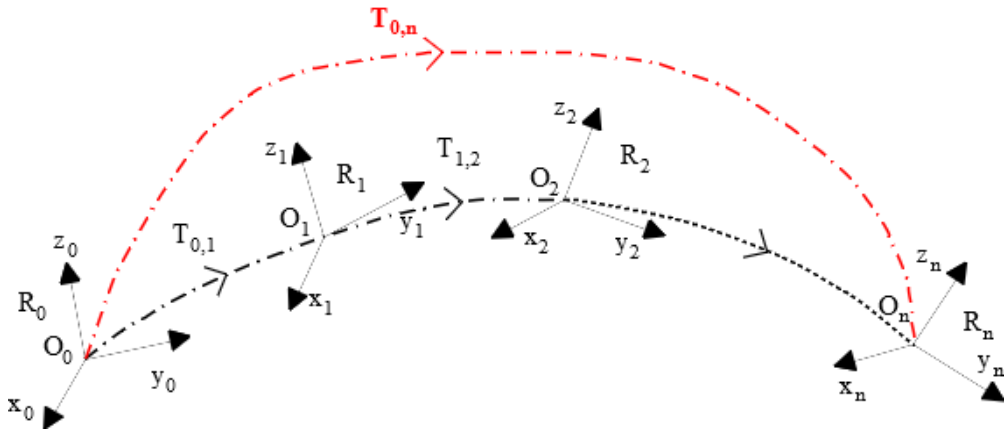
$$Rot(\vec{u}, \theta) \times Trans(\vec{u}, d) = Trans(\vec{u}, d) \times Rot(\vec{u}, \theta).$$

- A homogeneous transformation matrix can be split into two homogeneous transformation matrices (one matrix represents a pure translation, the other a pure rotation), that is:

$$T = \begin{pmatrix} A_{(3\times3)} & t_{(3\times1)} \\ 0\ 0\ 0 & 1 \end{pmatrix} = \begin{pmatrix} I & t_{(3\times1)} \\ 0\ 0\ 0 & 1 \end{pmatrix} \times \begin{pmatrix} & & & 0 \\ & A_{(3\times3)} & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Consecutive transformation: Consider that frame R_0 is subject to n consecutive transformation, see the following figure, then the homogeneous transformation $T_{0,n}$ can be obtained as:

$$T_{0,n} = T_{0,1} \times T_{1,2} \times \cdots \times T_{n-1,n}.$$



Exercise: Do the exercises of Tutorial 1.

4 GEOMETRIC AND KINEMATIC MODEL OF A SERIAL ROBOT

Let us define the joint space and the task space of a robot.

The ***joint space***, also called ***configuration space***, is the space in which the configuration/posture of the robot is defined. The *joint variables*, $q \in \mathbb{R}^N$, are the coordinates of this space, N is equal to the number of the independent (actuated) joints (joints are generally independent in an open chain robot contrary to closed chain structure where constraint relations occur between the joints variables) and corresponds to the number of degrees of freedom (*d.o.f.*) of the mechanical structure.

The ***task space***, also called ***operational space***, is the space in which the end-effector location is defined. An element of the task space is represented by a vector $X \in \mathbb{R}^M$ where M is equal to the maximum number of independent parameters (≤ 6) that are necessary to specify the end-effector location in space. Generally, the position and the orientation are specified respectively by Cartesian coordinates (in $\mathbb{R}^3$) and three independent angles.

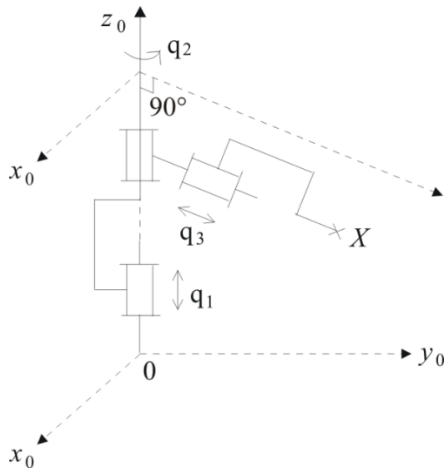
4.1 The necessity of a model

The design and control of robots require the use of models such as:

- The ***transformation model*** between the joint space and the task space. We can distinguish:
  - The ***direct geometric model*** to express the *location of the end-effector* as a function of the *joint variables* of the mechanism, the ***inverse geometric model*** to express the converse relationship,
  - The ***direct kinematic model*** which express the *velocity of the end-effector* as a function of the *joint velocities*, the ***inverse kinematic model*** to express the converse relationship,

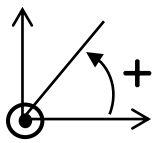
- The ***dynamic model*** defining the motion equations of the robot which establish the relationship between the input torques or forces of the actuators and the positions, velocities and accelerations of the joints.

Exercise: Consider the robot with 3 *d.o.f.*, that is q_1, q_2, q_3, described below.

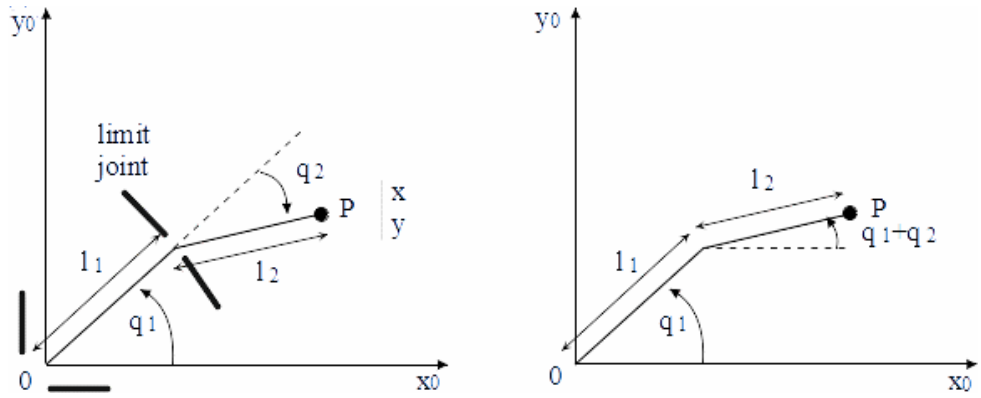


Let us consider the position of the point (X) located at the (free) extremity of the manipulator. Give the ***direct geometric model*** of the robot corresponding to the relation $X = f(q)$ with $X = (x\ y\ z)^t$, $q = (q_1\ q_2\ q_3)^t$ and where f is a static (time-independent) vectorial function.

N.B.: The convention, recalled in the figure below, gives the positive direction of an angle (the frame being orthonormal).



Exercise: Consider the RR planar robot described below.



1) Give the direct geometric model $(x, y)^t = f(q_1, q_2)^t$.

2) Propose a script (MatLab or Scilab) to represent the reachable workspace of the robot (*i.e.*, the volume of space that the robot can reach through at least one end-effector orientation) knowing that $l_1 = l_2 = 10\ cm$ and $0° \leq q_1 \leq 90°,\ -100° \leq q_2 \leq 90°$.

4.2 Direct geometric model of simple open chain – Modified Denavit-Hartenberg convention

Direct geometric model expresses the *location* (position and orientation) of the frame attached to the end-effector X (defined in *task space*) with respect to the reference frame R_0 (attached to the base of the robot) as a function of the *joint coordinates* q (defined in *joint space*) of the manipulator through the following equation:

$$X = f(q). \qquad (cf.\ §4.1)$$

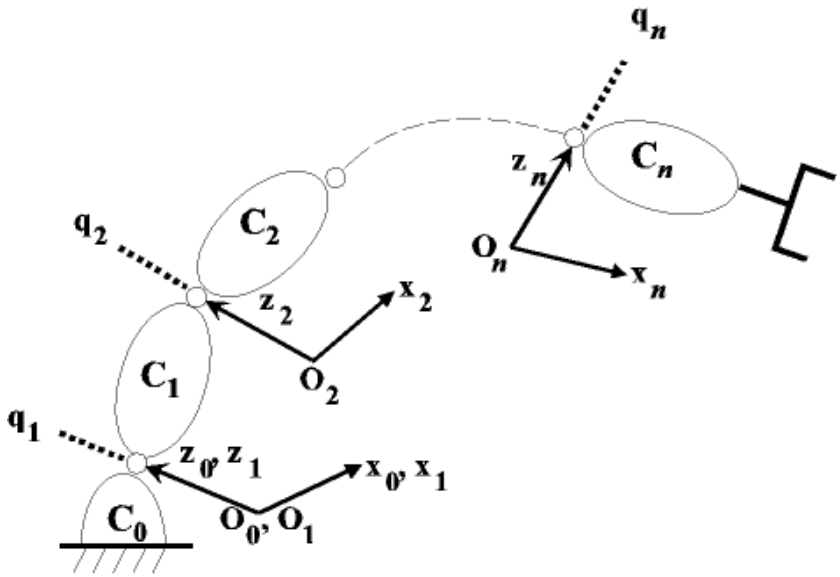
The method to obtain this model is based on homogeneous transformation matrices. A frame is attached to each link of the manipulator. The end-effector location with respect to the reference frame corresponds to the product between transformation matrices of frames attached to all the links of the manipulator. Notice that the expression of transformation matrices is not unique in the sense that there is an infinite number of possibilities to attach a frame to a link.

Denavit-Hartenberg convention allows the links of the manipulator to be parametrized so that the transformation matrices all have the same literal form, making their computations easier.

The following method applies when the robot corresponds to a simple open chain and its joints are rotoid or prismatic which is the general case. Fictitious links (of null length) are introduced then a joint has several *d.o.f.*. Moreover, the links constituting the chain are supposed to be perfectly rigid and connected by ideal joints (no backlash[13], no elasticity).

- **Notations**:

  The links are numbered in ascending order from the base to its free extremity. The robot is supposed to be composed of $n+1$ links, denoted $C_0, \cdots, C_n$, and n joints ($n \geq 1$). The link C_0 designates the base of the robot, the link C_n carries the end-effector.
  The orthonormal frame R_j is attached to link C_j of the robot.
  The variable of joint j (that is, the joint connecting link C_j to link C_{j-1}) is denoted q_j as shown in the following figure.



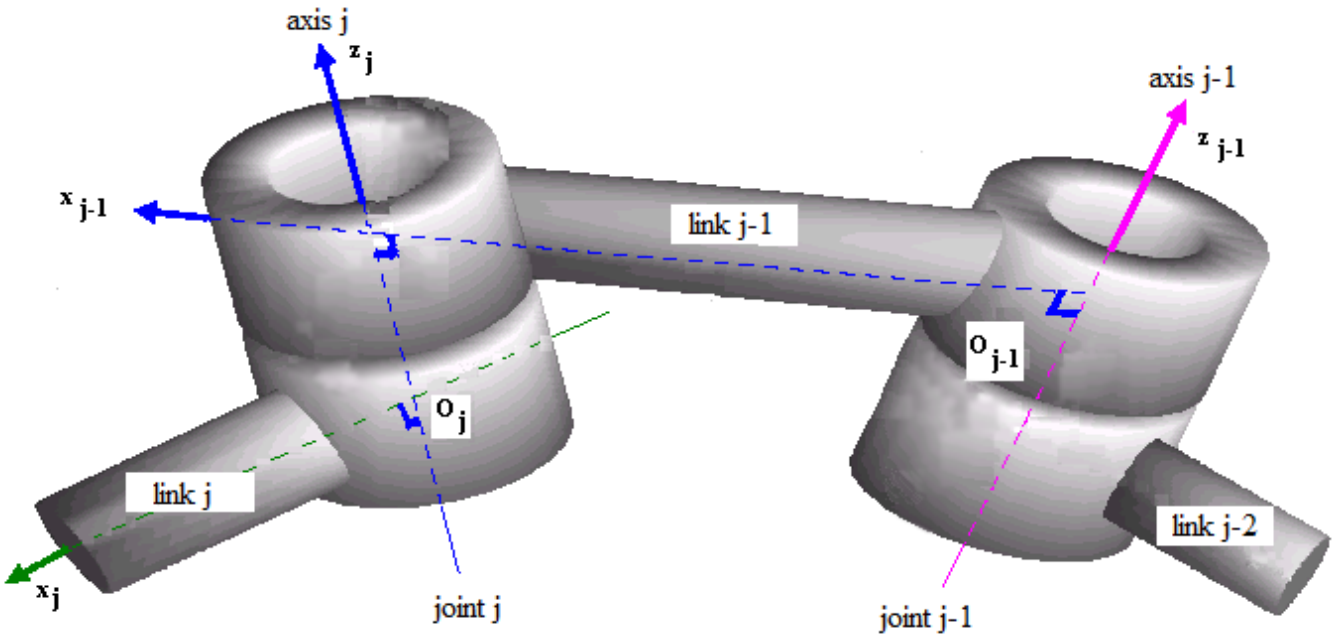
[13] 'jeu mécanique'.

➢ **Definition of frame R_j (attached to link C_j)**:

- The $\overrightarrow{o_j z_j}$ axis (that is, the line generated by the $\overrightarrow{o_j z_j}$ vector) is along the axis of (rotoid or prismatic) joint j. The Denavit-Hartenberg convention we use here is slightly modified from the original one (where the $\overrightarrow{o_j z_j}$ axis is merged with the axis of joint $j+1$) that enables to remove ambiguities when robot is closed or tree chains.

- The $\overrightarrow{o_j x_j}$ axis (that is, the line generated by the $\overrightarrow{o_j x_j}$ vector) is aligned with the common normal between $\overrightarrow{o_j z_j}$ and $\overrightarrow{o_{j+1} z_{j+1}}$ axes. The choice of $\overrightarrow{o_j x_j}$ axis is not unique if $\overrightarrow{o_j z_j}$ and $\overrightarrow{o_{j+1} z_{j+1}}$ axes are parallel, then $\overrightarrow{o_j x_j}$ axis is determined by considerations of symmetry or simplicity.

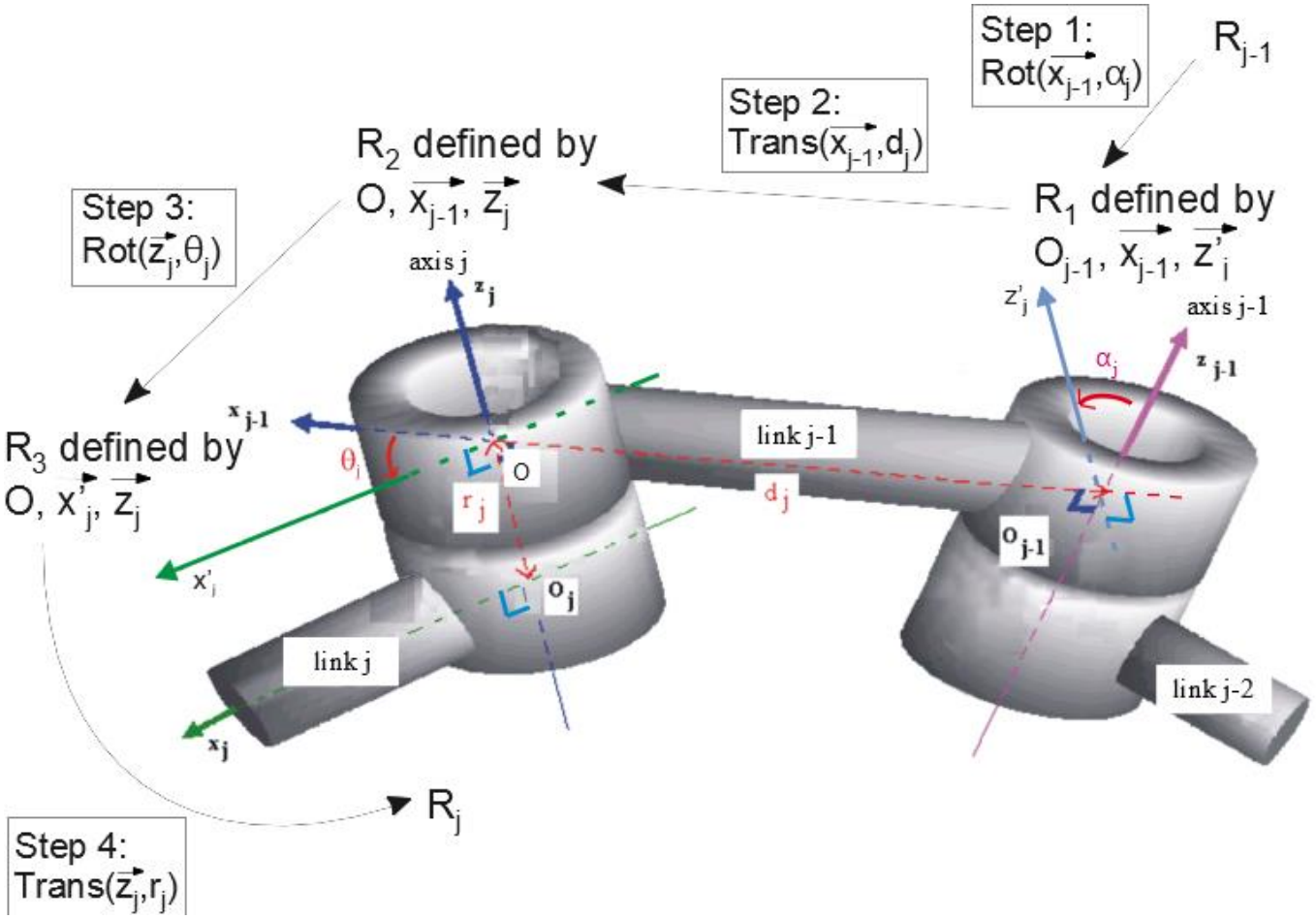
The intersection of $\overrightarrow{o_j x_j}$ and $\overrightarrow{o_j z_j}$ axes defines the origin O_j.



➢ **Transformation matrix from frame R_{j-1} into frame R_j, determination of parameters**:

The transformation matrix from frame R_{j-1} into frame R_j is expressed as a function of the following four geometric parameters (see figure below):

- the angle α_j between $\overrightarrow{o_{j-1}z_{j-1}}$ and $\overrightarrow{o_jz_j}$ axes about $\overrightarrow{o_{j-1}x_{j-1}}$ axis (transformation from R_{j-1} into R_1),
- the distance d_j between $\overrightarrow{o_{j-1}z_{j-1}}$ and $\overrightarrow{o_jz_j}$ axes along $\overrightarrow{o_{j-1}x_{j-1}}$ axis (transformation from R_1 into R_2),
- the angle θ_j between $\overrightarrow{o_{j-1}x_{j-1}}$ and $\overrightarrow{o_jx_j}$ axes about $\overrightarrow{o_jz_j}$ axis (transformation from R_2 into R_3),
- the distance r_j between $\overrightarrow{o_{j-1}x_{j-1}}$ and $\overrightarrow{o_jx_j}$ axes along $\overrightarrow{o_jz_j}$ axis (transformation from R_3 into R_j).



Let us note that the distances d_j, r_j are algebraic in the sense that their values can be positive or negative.

From these four successive changes of frames, we obtain the following transformation matrix $T_{j-1,\,j}$ defining frame R_j relative to frame R_{j-1}:

$$T_{j-1,\,j} = Rot(\overrightarrow{x_{j-1}}, \alpha_j) \times Trans(\overrightarrow{x_{j-1}}, d_j) \times Rot(\overrightarrow{z_j}, \theta_j) \times Trans(\overrightarrow{z_j}, r_j)$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha_j) & -sin(\alpha_j) & 0 \\ 0 & sin(\alpha_j) & cos(\alpha_j) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & d_j \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} cos(\theta_j) & -sin(\theta_j) & 0 & 0 \\ sin(\theta_j) & cos(\theta_j) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r_j \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} cos(\theta_j) & -sin(\theta_j) & 0 & d_j \\ cos(\alpha_j)\ sin(\theta_j) & cos(\alpha_j)\ cos(\theta_j) & -sin(\alpha_j) & -r_j\ sin(\alpha_j) \\ sin(\alpha_j)\ sin(\theta_j) & sin(\alpha_j)\ cos(\theta_j) & cos(\alpha_j) & r_j\ cos(\alpha_j) \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Remark: The variable q_j (of joint j) is either θ_j if the joint is rotoid or r_j if the joint is prismatic which leads to the relation:

$$q_j = (1 - \sigma_j)\ \theta_j + \sigma_j\ r_j$$

where $\sigma_j = 0$ if joint j is rotoid and $\sigma_j = 1$ if it is prismatic.

So, if joint is rotoid then $\begin{cases} q_j = \theta_j \text{ is variable} \\ \alpha_j, d_j, r_j \text{ are constants} \end{cases}$, if joint is prismatic then $\begin{cases} q_j = r_j \text{ is variable} \\ \alpha_j, d_j, \theta_j \text{ are constants} \end{cases}$.

Remarks:

- The simplest choice to define reference frame R_0 consists in merging frame R_0 with frame R_1 when $q_1 = 0$, as illustrated in the first figure of §4.2.
- If joint j is prismatic, the $\overrightarrow{o_j z_j}$ axis is parallel to the joint axis but can have any position in space. A simple solution consists in setting the $\overrightarrow{o_j z_j}$ axis such that $d_j = 0$ or $d_{j+1} = 0$.
- The two consecutive joint axes of a robot are generally orthogonal or parallel, the resulting angle α is equal to $0°$, $\pm 90°$ or $180°$.
  When $\overrightarrow{o_{j-1} z_{j-1}}$ and $\overrightarrow{o_j z_j}$ axes are parallel ($\alpha_j = 0°$ or $180°$), there are an infinite number of common normal between $\overrightarrow{o_{j-1} z_{j-1}}$ and $\overrightarrow{o_j z_j}$ axes, a simple solution consists in setting the $\overrightarrow{o_j x_j}$ axis such that $r_j = 0$ or $r_{j+1} = 0$.
  When $\overrightarrow{o_{j-1} z_{j-1}}$ and $\overrightarrow{o_j z_j}$ axes are orthogonal ($\alpha_j = \pm\ 90^0$), the point O_j is situated at the intersection of $\overrightarrow{o_{j-1} z_{j-1}}$ and $\overrightarrow{o_j z_j}$ axes which leads to $d_j = 0$.
- The inverse of the transformation matrix $T_{j-1,\ j}$ has the following expression:

$$T_{j,\ j-1} = Trans(\overrightarrow{z_j}, -r_j) \times Rot(\overrightarrow{z_j}, -\theta_j) \times Trans(\overrightarrow{x_{j-1}}, -d_j) \times Rot(\overrightarrow{x_{j-1}}, -\alpha_j),$$

or equivalently, $T_{j,\ j-1} = \begin{pmatrix} \begin{pmatrix} & A^t_{j-1,\ j} & \end{pmatrix} & \begin{matrix} -d_j\ cos(\theta_j) \\ d_j\ sin(\theta_j) \\ -r_j \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{pmatrix}$.

We have: $A_{j,\ j-1} = A^{-1}_{j-1,\ j} = A^t_{j-1,\ j}$, *cf. Properties of homogeneous transformation matrices* in §3.2.

Computation of direct geometric model

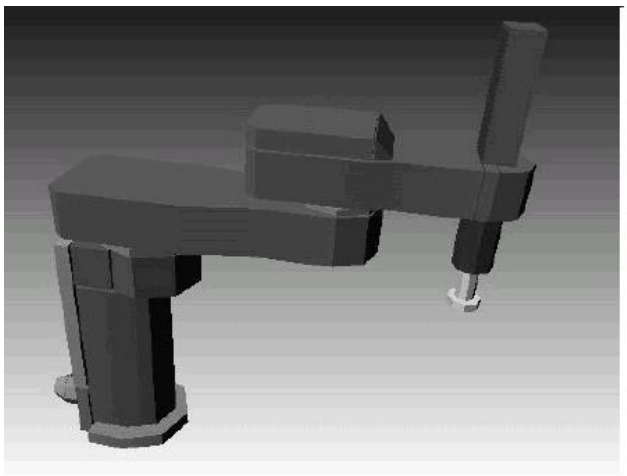
Let us consider a single open chain composed of n joints, $q_1, \cdots, q_n$. *Direct geometric model* expresses the *location* (position and orientation) of the frame R_n (attached to the end-effector) with respect to the reference frame R_0 (attached to the base of the robot) as a function of the *joint variables*. It is represented by the matrix $T_{0,\ n}(q_1, \cdots, q_n)$ corresponding to:

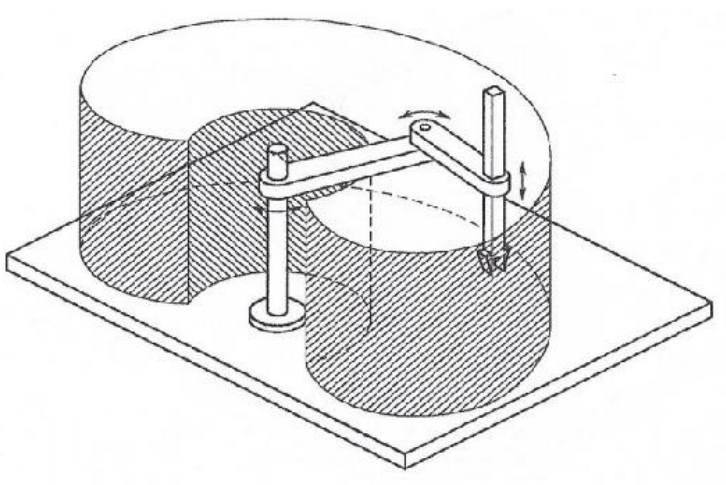
$$T_{0,\ n}(q_1, \cdots, q_n) = T_{0,\ 1}(q_1) \times T_{1,\ 2}(q_2) \times \cdots \times T_{n-1,\ n}(q_n).$$

4.3 Example

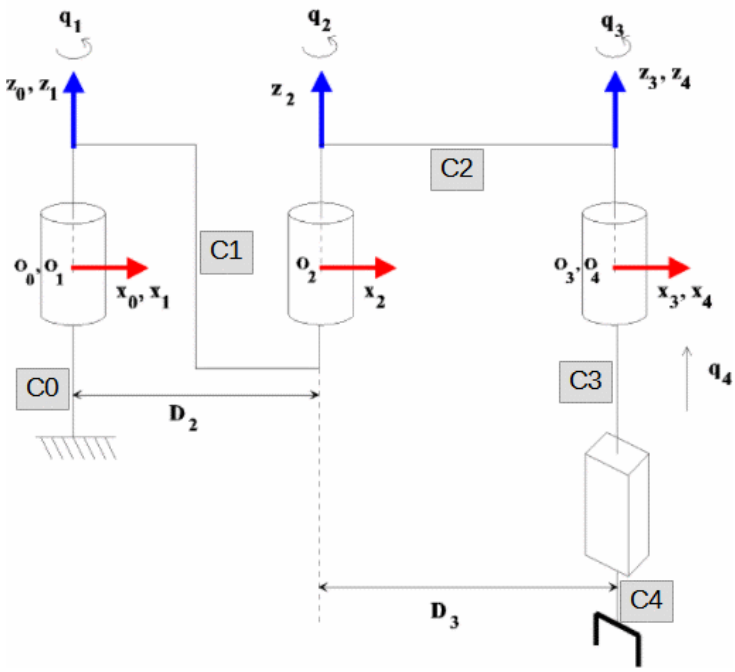
Let us provide the direct geometric model of the 4 *d.o.f.* robot SCARA[14] where its initial posture and its workspace are represented below:

[14] SCARA: Selective Compliance Articulated Robot for Assembly (compliance : conforme).





In the following figure a frame is attached to each link C_j $(j = 0, \cdots, 4)$ of the manipulator as specified in the modified Denavit-Hartenberg method.



Indeed, each frame R_j $(j = 0, \cdots, 4)$ is such that:

- The $\overrightarrow{o_j z_j}$ axis is along the axis of joint j,
- The $\overrightarrow{o_j x_j}$ axis is orthogonal to $\overrightarrow{o_j z_j}$, $\overrightarrow{o_{j+1} z_{j+1}}$ axes.

Once each frame R_j $(j = 0, \cdots, 4)$ has been set, we compute the four parameters of each line j $(j = 1, \cdots, 4)$ defining frame R_j relative to frame R_{j-1}.

1) Verify that the parameters listed below are correct.

j	σ_j	α_j	d_j	θ_j	r_j
1	0	0	0	q_1	0
2	0	0	D_2	q_2	0
3	0	0	D_3	q_3	0
4	1	0	0	0	q_4

2) Compute the transformation matrix $T_{0,4}$ when the robot is in its initial posture, *i.e.*, when $q_1 = q_2 = q_3 = 0\ rd$ and $q_4 = 0\ m$ (as illustrated in the previous figure).

3) Compute the location of the end-effector attached to frame R_4 with respect to reference frame R_0 when the robot is in its initial posture. Verify the obtained result by using the figure.
4) Compute the transformation matrix $T_{0,4}$ when the posture of the robot corresponds to:

$$q_1 = 0,\ q_2 = -\frac{\pi}{2},\ q_3 = \frac{\pi}{2},\ q_4 = 0.$$

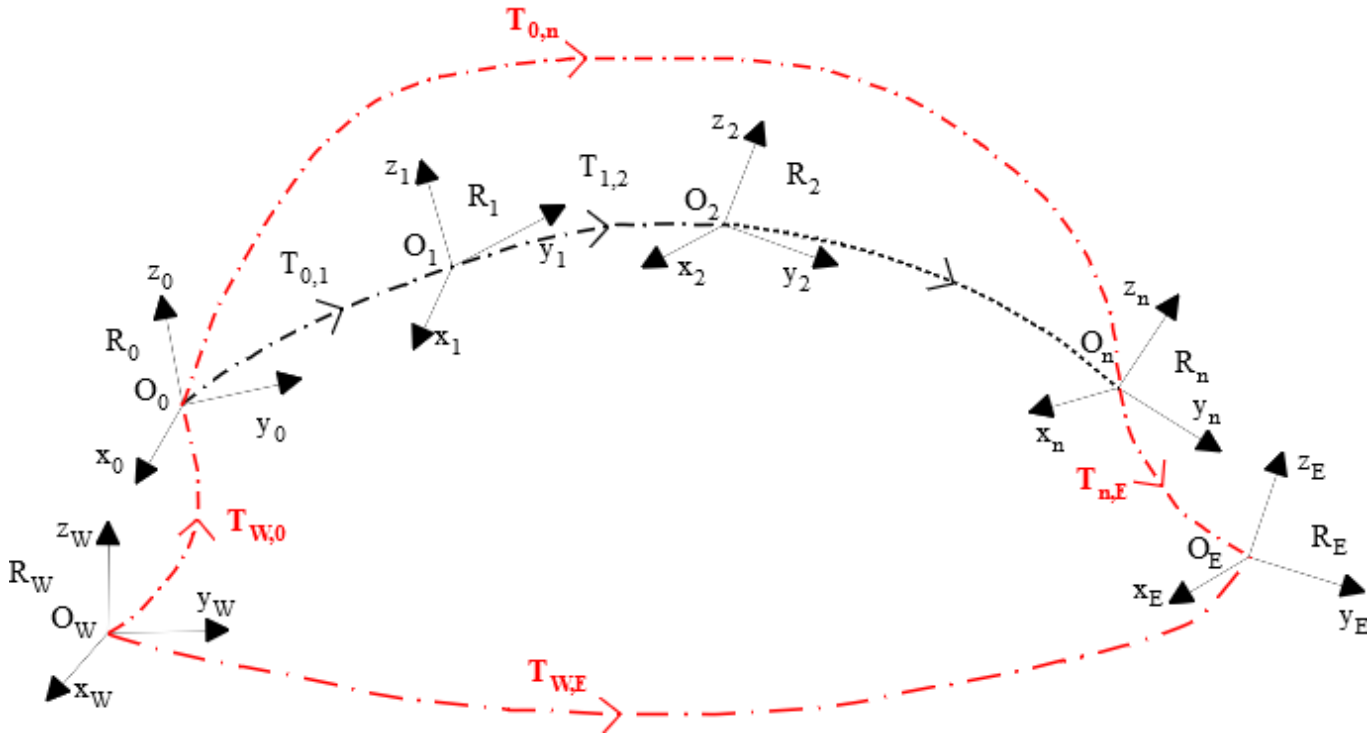
5) From the expression of the transformation matrix $T_{0,4}$ (in particular $\overrightarrow{O_0 O_{4|0}}$), what can you deduce on:
- the joints used to set the coordinates x, y of point O_4,
- the joints used to set the coordinate z of point O_4,
- the joints used to set the orientation of frame R_4?

4.4 Transformation matrix of the end-effector in the world frame

A robot is generally associated with other components as devices (production machine, conveyor belt), sensors (camera), other robots in a robotic work cell. So, it is practical to define a ***world frame*** R_W which may be different from the reference frame R_0 (attached to the base of the robot) to have a same reference frame for all the components. The transformation matrix defining frame R_0 relative to frame R_W is denoted $T_{W,0}$.
Moreover, a robot may have interchangeable different tools. So, it is also practical to define a frame, called ***tool frame*** R_E, for each of these tools. The transformation matrix defining frame R_E relative to frame R_n is denoted $T_{n,E}$, see the following figure.
Thus, the transformation matrix defining frame R_E relative to frame R_W, denoted $T_{W,E}$, is equal to:

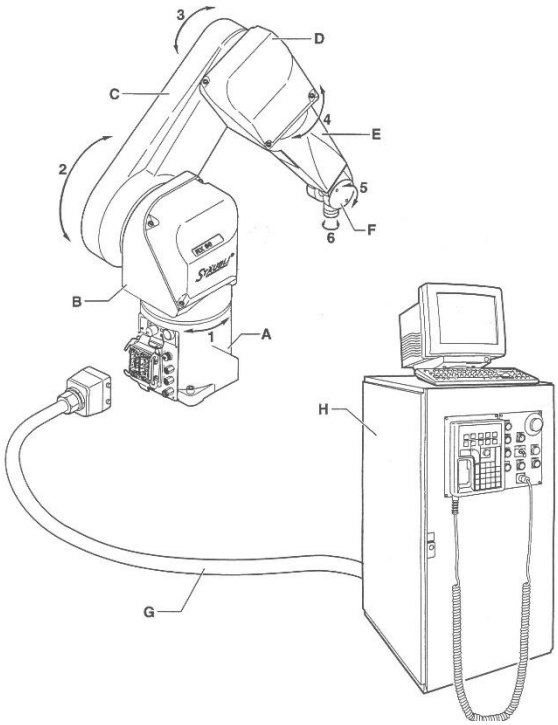
$$T_{W,E} = T_{W,0} T_{0,n} T_{n,E}.$$

Let us note that world frame and tool frame can be specified in most robot programming languages.



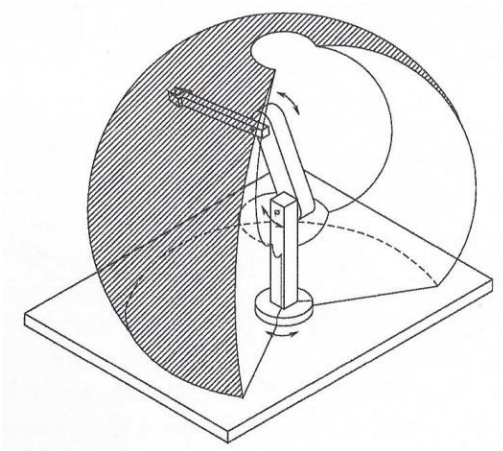
4.5 Exercise

A) Let us provide the direct geometric model of the robot Stäubli RX-90 depicted below:

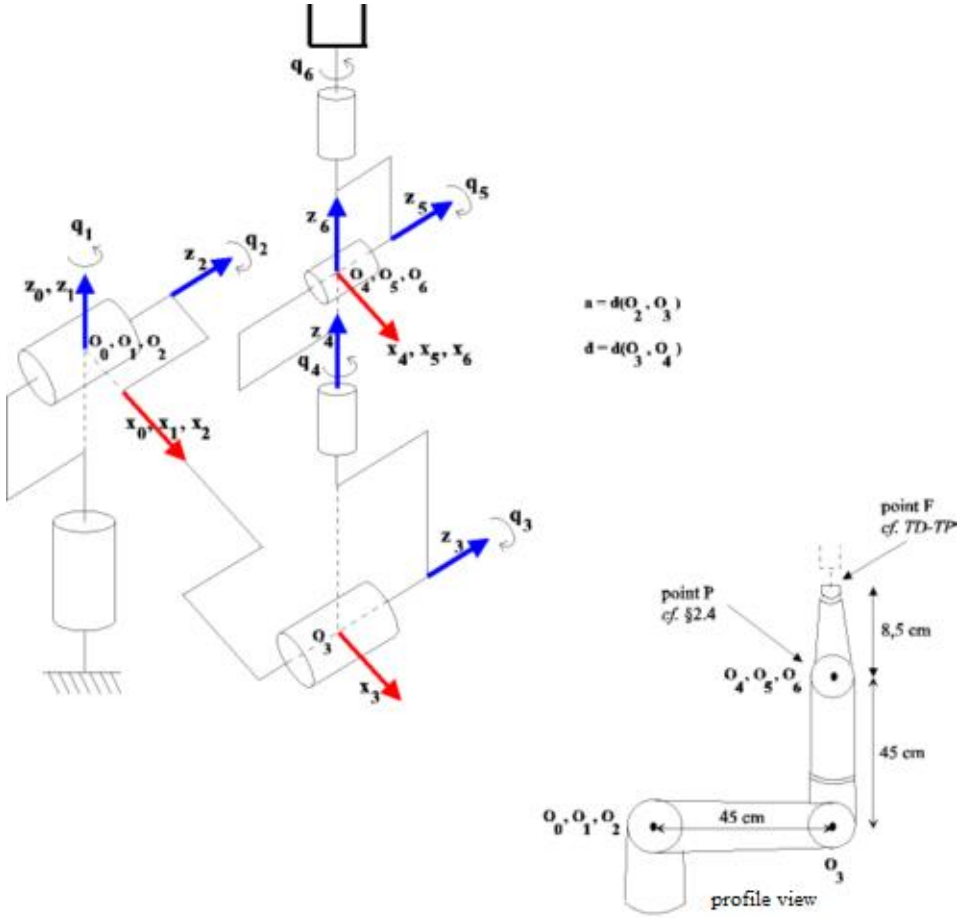


The *shoulder* of the robot (joints 1, 2, 3) is a RRR structure, joints 4, 5, 6 define a *spherical wrist* (their rotation axes are concurrent).

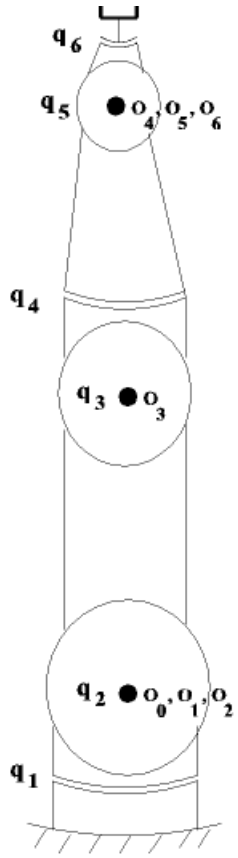
The workspace of this anthropomorphic robot is represented below:



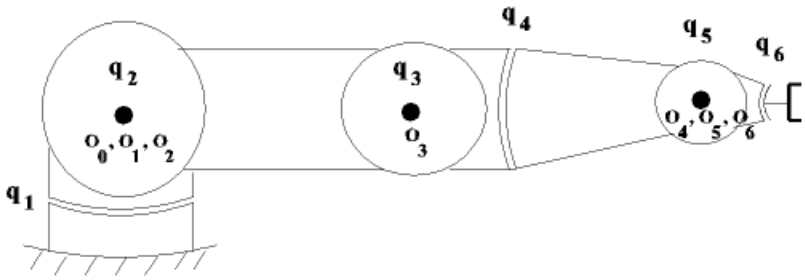
The initial posture of the robot and the frames attached to each of links are given in the following figure.



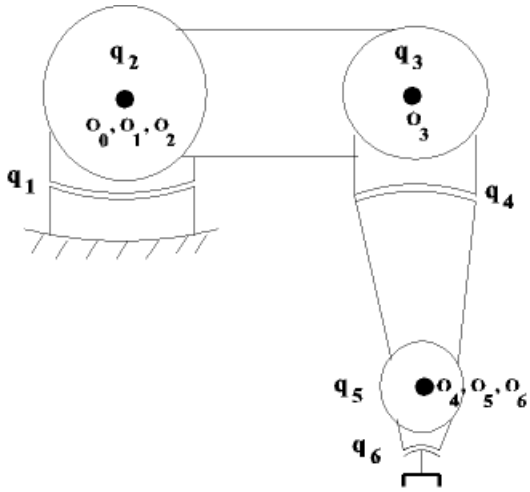
1) Give the modified Denavit-Hartenberg parameters of the robot.
2) Propose a script (MatLab or Scilab) that computes the transformation matrix $T_{0,6}$ as a function of given joint values $q_1, \dots, q_6$. From matrix $T_{0,6}$, give the frame R_6 location with respect to reference frame R_0 and verify the obtained result from the figure showing the corresponding posture when:
   - The arm is in its initial posture, *i.e.*, when $q_1 = q_2 = \cdots = q_6 = 0$ (see previous figure).
   - The arm is extended vertically along the $\overrightarrow{o_0 z_0}$ axis, see below its profile view:



- The arm extended horizontally along the $\overrightarrow{o_0x_0}$ axis, see below its profile view:



- The arm is bent with the gripper down, see below its profile view:

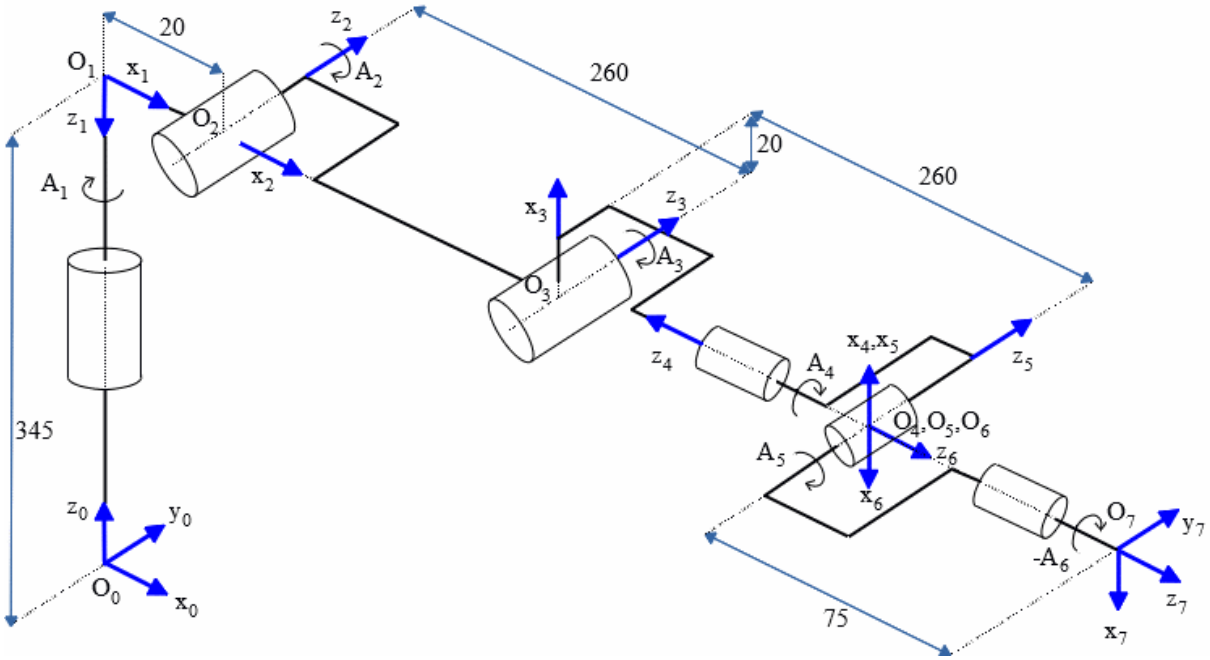


3) Let us consider an arbitrary posture, for example, when $q_1 = 10;\ q_2 = 15;\ q_3 = -30;\ q_4 = 50;\ q_5 = 0;\ q_6 = 0$. Verify that the corresponding matrix $T_{0,6}$ is given by:

$$T_{0,6} = \begin{pmatrix} R_{0,6} = \text{Direction\_Cosines\_from\_Euler(-170;15;-130)} & \begin{pmatrix} .3134 \\ .0553 \\ .3182 \end{pmatrix} \\ 0 \quad 0 \quad 0 & 1 \end{pmatrix}.$$

Recall that the Euler angles (ψ, θ, φ) convention is (z, y, z) for RX-90.

B) We look for the direct geometric model of the robot KUKA KR3 R540. The initial posture of the robot and the frames attached to each of links are given in the following figure. Give the modified Denavit-Hartenberg parameters of the robot.



4.6 Inverse geometric model – Method of Paul

Inverse geometric model expresses the *joint coordinates* q of the manipulator (defined in *joint space*) as a function of the *location* X (position and orientation) of the frame attached to the end-effector (defined in *task space*) with respect to the reference frame R_0.

There is no systematic method to compute the inverse geometric model. When it exists, it gives all possible solutions to the inverse problem (there is rarely uniqueness of the solution). Several methods exist to obtain the inverse geometric model, method of Paul[15] is one of them and is suitable for robots with simple geometry (which is the case of most industrial robots).

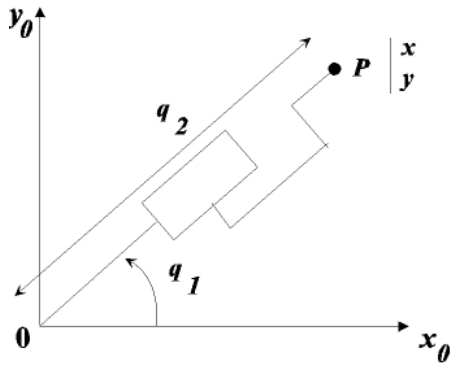
When the inverse geometric model does not exist, *i.e.*, there is no explicit form, a particular solution of the inverse problem can be numerically computed. The solution is only local in the sense that it depends on the initial conditions (see TD on the use of Newton's method). Note that such methods can be penalizing in terms of computation time.

Examples of simple planar robots

➢ **First example**

Consider the planar RP robot described below knowing that $q_2 > 0$.

15 Paul R.C.P., *Robot Manipulators: Mathematics, Programming and Control*, MIT Press, Cambridge, USA, 1981.



The direct geometric model is such that: $\begin{cases} x = q_2\, cos(\, q_1) \\ y = q_2\, sin(\, q_1) \end{cases}$.

A simple analytical approach enables to determine the inverse geometric model. We have:

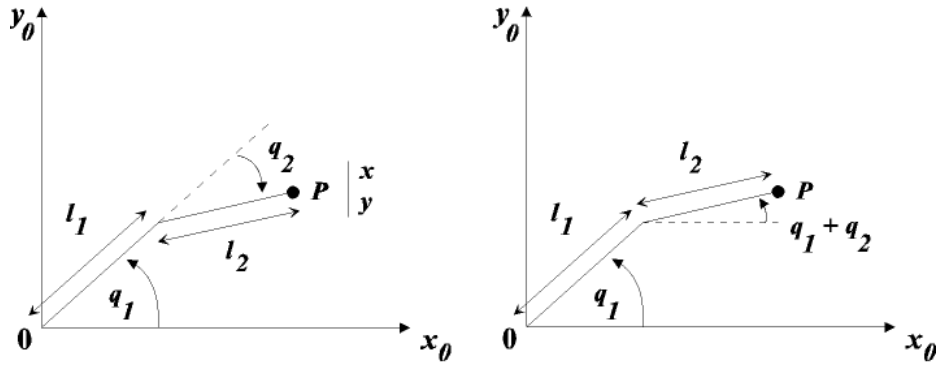
$$sin(\, q_1) = \frac{y}{q_2},\ cos(\, q_1) = \frac{x}{q_2} \text{ with } q_2 > 0 \Rightarrow q_1 = \text{atan2}(y, x),$$

and $x^2 + y^2 = q_2^2 \Rightarrow q_2 = \sqrt{x^2 + y^2}$.

NB: The mathematical function atan2 provides the arc tangent function from its two arguments (y, x) knowing that the sign of these arguments uniquely determines the angle q_1 in the interval $[-\pi, \pi]$ (as contrary to the function $\text{atan}(y/x)$ that determines the angle q_1 in the interval $[-\pi/2\,, \pi/2]$).

- **Second example**

Consider the planar RR robot described below.

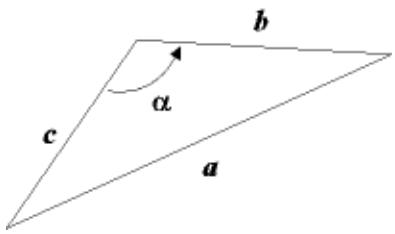


The direct geometric model is such that: $\begin{cases} x = l_1\, cos(\, q_1) + l_2\, cos(\, q_1 + q_2) \\ y = l_1\, sin(\, q_1) + l_2\, sin(\, q_1 + q_2) \end{cases}$,

that is, a 2 equations system with 2 unknowns.

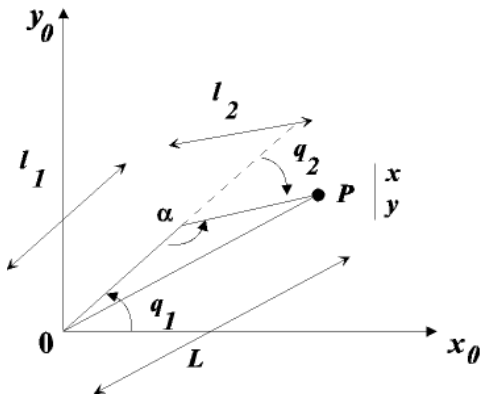
An analytical approach, proceeding by substitution, is used to obtain the inverse geometric model.

For this, let us recall the Al-Kashi theorem (also called the generalized Pythagorean theorem) by using the following triangle:



We have: $a^2 = b^2 + c^2 - 2\ b\ c\ \ cos(\, \alpha)$.

In order to apply this result, let us complete the previous figure:



We deduce from this figure and the generalized Pythagorean theorem that:

$L^2 = x^2 + y^2$ and $L^2 = l_1^2 + l_2^2 - 2\ l_1 l_2\ cos(\alpha)$ with $\alpha = \pi + q_2$,

knowing that $cos(\pi + a) = -cos(a)$, we obtain:

$$x^2 + y^2 = l_1^2 + l_2^2 + 2\ l_1 l_2\ cos(q_2) \qquad (1)$$

which leads to:

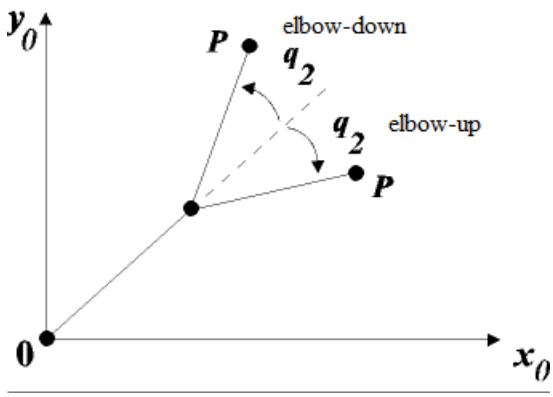
$$cos(q_2) = \frac{x^2+y^2-(l_1^2+l_2^2)}{2\ l_1 l_2}.$$

Since $cos(q) = a$ with $a \in [-1,1]$ implies that $q = \pm \text{acos}(a)$, we deduce that:

$$q_2 = \pm \text{acos}\left(\frac{x^2+y^2-(l_1^2+l_2^2)}{2\ l_1 l_2}\right) \text{ provided that } -1 \leq \frac{x^2+y^2-(l_1^2+l_2^2)}{2\ l_1 l_2} \leq 1.$$

Remarks:

- The condition $-1 \leq \frac{x^2+y^2-(l_1^2+l_2^2)}{2\ l_1 l_2} \leq 1$ means that the position of point *P* is in the workspace.
- The robot has an *elbow-down* posture, resp. *elbow-up* posture, when angle q_2 is positive, resp., negative, see the figure below.



Moreover, by developing the expressions $cos(q_1 + q_2)$ and $sin(q_1 + q_2)$[16] in the equations system corresponding to the direct geometric model, we obtain:

$$\begin{cases} (l_1 + l_2\ cos(q_2))\ cos(q_1) - l_2\ sin(q_2)\ sin(q_1) = x \\ l_2\ sin(q_2)\ cos(q_1) + (l_1 + l_2\ cos(q_2))\ sin(q_1) = y \end{cases}.$$

The computation of the determinant of this algebraic system of two equations linear in the two unknowns $cos(q_1)$ and $sin(q_1)$ yields to:

[16] $cos(a + b) = cos(a)\ cos(b) - sin(a)\ sin(b),\ \ sin(a + b) = sin(a)\ cos(b) + cos(a)\ sin(b).$

$$\begin{vmatrix} l_1 + l_2\,cos(q_2) & -l_2\,sin(q_2) \\ l_2\,sin(q_2) & l_1 + l_2\,cos(q_2) \end{vmatrix} = l_1^2 + l_2^2 + 2l_1 l_2\,cos(q_2) = x^2 + y^2$$, see Eq. (1),

which leads, by using Cramer's method, to:

$$cos(q_1) = \frac{\begin{vmatrix} x & -l_2\,sin(q_2) \\ y & l_1 + l_2\,cos(q_2) \end{vmatrix}}{x^2+y^2} \text{ and } sin(q_1) = \frac{\begin{vmatrix} l_1 + l_2\,cos(q_2) & x \\ l_2\,sin(q_2) & y \end{vmatrix}}{x^2+y^2},$$

that is:

$$cos(q_1) = \frac{1}{x^2+y^2}(x(l_1 + l_2\,cos(q_2)) + yl_2\,sin(q_2)),$$
$$sin(q_1) = \frac{1}{x^2+y^2}(y(l_1 + l_2\,cos(q_2)) - xl_2\,sin(q_2)).$$

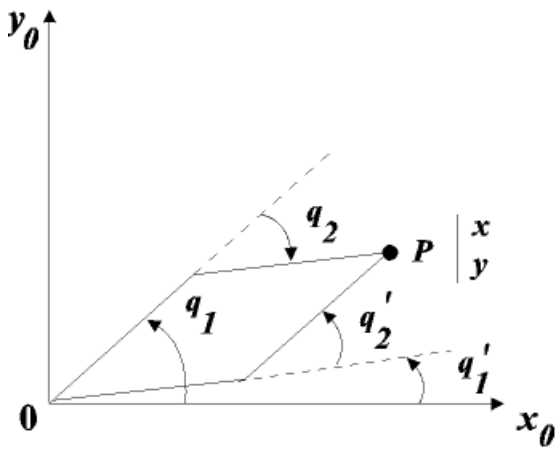
Finally, knowing that $x^2 + y^2 > 0$, we obtain:

$$q_1 = \text{atan2}(y(l_1 + l_2\,cos(q_2)) - xl_2\,sin(q_2), x(l_1 + l_2\,cos(q_2)) + yl_2\,sin(q_2)).$$

So, we end up with the following inverse geometric model:

$$q_1 = \text{atan2}(y(l_1 + l_2\,cos(q_2)) - xl_2\,sin(q_2), x(l_1 + l_2\,cos(q_2)) + yl_2\,sin(q_2)),$$
$$q_2 = \pm\,acos\left(\frac{x^2 + y^2 - (l_1^2 + l_2^2)}{2l_1 l_2}\right).$$

We note the existence of two solutions corresponding to the two possible postures of the manipulator: the elbow-down and the elbow-up postures, see figure below (assuming that there is no constraint (limit) on the joints).



Exercise: Provide the inverse geometric model of the manipulator SCARA described in §4.3.

Method of Paul

Method of Paul can be used to analytically provide the inverse geometric model of robots with simple geometry (for which most of the distances d_j, r_j are null and the angles θ_j, α_j are equal to 0 or $\pm \pi/2$).

- **Principle**

Consider the robot described by the following transformation matrix:

$$T_{0,n}(q_1, q_2, \cdots, q_n) = T_{0,1}(q_1) \times T_{1,2}(q_2) \times \cdots \times T_{n-1,n}(q_n).$$

Let U_0 be the ***planned location*** of the frame R_n attached to the end-effector defined as:

$$U_0 = \begin{bmatrix} S_x & N_x & A_x & P_x \\ S_y & N_y & A_y & P_y \\ S_z & N_z & A_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Let us recall (*cf.* §2) that:

- $\overrightarrow{x_n} = S_x\,\overrightarrow{x_0} + S_y\,\overrightarrow{y_0} + S_z\,\overrightarrow{z_0}$, $\overrightarrow{y_n} = N_x\,\overrightarrow{x_0} + N_y\,\overrightarrow{y_0} + N_z\,\overrightarrow{z_0}$, $\overrightarrow{z_n} = A_x\,\overrightarrow{x_0} + A_y\,\overrightarrow{y_0} + A_z\,\overrightarrow{z_0}$ knowing that only three (independent) parameters are needed to define the orientation of the frame R_n with respect to the reference frame R_0,
- $\overrightarrow{O_0O_n} = P_x\,\overrightarrow{x_0} + P_y\,\overrightarrow{y_0} + P_z\,\overrightarrow{z_0}$.

The inverse geometric model is obtained from the following matrix equation:

$$U_0 = T_{0,1}(q_1) \times T_{1,2}(q_2) \times \cdots \times T_{n-1,n}(q_n). \quad (2)$$

The method of Paul begins to compute q_1, then q_2 and so on until the last joint q_n. The method is to move each joint variable (one after the other) to the left side of Eq. 2 by successively premultiplying the equation by $T_{j,j-1}$ (for $j = 1, \cdots, n$).

Let us apply the method on a 6 *d.o.f.* robot ($n = 6$):

- Premultiply Eq. (2) by $T_{1,0}$ which leads to:

$$T_{1,0}(q_1) \times U_0 = T_{1,2}(q_2) \times \cdots \times T_{5,6}(q_6). \quad (3)$$

  By construction, the elements of the left side are constants or functions of q_1 whereas the elements of the right side are constants or functions of $q_2, \cdots, q_6$. Deduce q_1 from matrix Eq. (3).
- Premultiply Eq. (3) by $T_{2,1}$ which leads to:

$$T_{2,1}(q_2) \times T_{1,0}(q_1) \times U_0 = T_{2,3}(q_3) \times \cdots \times T_{5,6}(q_6).$$

  Deduce q_2 from this new matrix equation.
- Continue the process until the determination of all the joint variables, that is, $q_3, \cdots, q_6$.

In summary, the joint variables are deduced from the following matrix equations:

$$\begin{aligned} U_0 &= T_{0,1}(q_1) \times T_{1,2}(q_2) \times T_{2,3}(q_3) \times T_{3,4}(q_4) \times T_{4,5}(q_5) \times T_{5,6}(q_6) \\ T_{1,0}(q_1) \times U_0 &= T_{1,2}(q_2) \times T_{2,3}(q_3) \times T_{3,4}(q_4) \times T_{4,5}(q_5) \times T_{5,6}(q_6) \\ T_{2,1}(q_2) \times U_1 &= T_{2,3}(q_3) \times T_{3,4}(q_4) \times T_{4,5}(q_5) \times T_{5,6}(q_6) \\ T_{3,2}(q_3) \times U_2 &= T_{3,4}(q_4) \times T_{4,5}(q_5) \times T_{5,6}(q_6) \\ T_{4,3}(q_4) \times U_3 &= T_{4,5}(q_5) \times T_{5,6}(q_6) \\ T_{5,4}(q_5) \times U_4 &= T_{5,6}(q_6) \end{aligned}$$

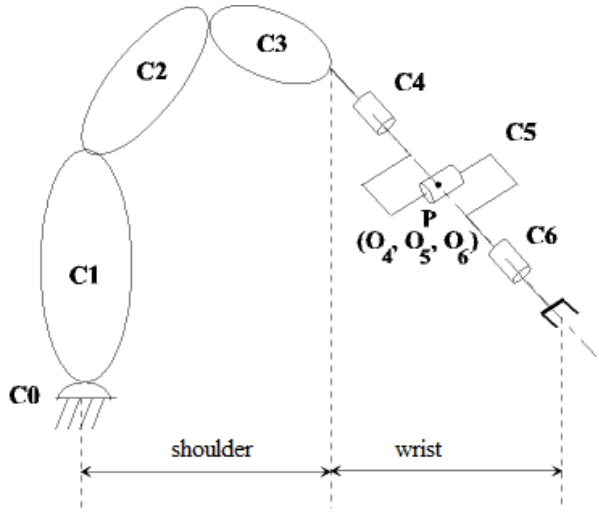
with $U_j = T_{j,6} = T_{j,j-1} \times U_{j-1}$ for $j = 1, 2, 3, 4$.

The resolution of these equations is not systematic, however (often) only few types of equations (for which the analytical solution is known) are involved, for example:

- $Xr_i = Y$,
- $X\,sin(\theta_i) + Y\,cos(\theta_i) = Z$,
- $\begin{cases} X1\,sin(\theta_i) + Y1\,cos(\theta_i) = Z1 \\ X2\,sin(\theta_i) + Y2\,cos(\theta_i) = Z2 \end{cases}$.

➢ **Case of a 6 *d.o.f.* robot with spherical wrist**

Let us apply method of Paul on the Stäubli RX 90 robot. Let P be the intersection point between the last three concurrent joint axes, see the following figure.



Such a robot is characterized by the following Denavit-Hartenberg parameter values:

$$\begin{cases} d_5 = r_5 = d_6 = 0, \\ \sigma_4 = \sigma_5 = \sigma_6 = 0, \\ sin(\alpha_5) \neq 0,\ sin(\alpha_6) \neq 0, \end{cases}$$

the last conditions are satisfied when the robot is not redundant which is the case for a spherical wrist.

Since the position of point P (centre of the wrist) is only function of joint variables q_1, q_2, q_3, such a robot structure enables to split in two problems the computation of the 6 joint variables:

- The first problem, named *position problem*, depends on joint variables q_1, q_2, q_3 and so enables to compute these variables (such that position of P coincides with the desired one),
- The second problem, named *orientation problem*, depends on joint variables q_4, q_5, q_6 (knowing that q_1, q_2, q_3 are previously computed) and so enables to compute these variables.

Equation of position

Since $O_4 = O_5 = O_6 = P$, we have:

$$\overrightarrow{O_0P}_{|0} = T_{0,4}(q_1, q_2, q_3, q_4) \times \overrightarrow{O_4P}_{|4},$$

that is:

$$\begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = T_{0,1}(q_1) \times T_{1,2}(q_2) \times T_{2,3}(q_3) \times T_{3,4}(q_4) \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$ (Eq. corresponding to the position part of Eq. (2)),

which leads to a system of 4 equations knowing that the last one is not relevant.

Let us note that these equations depend on q_1, q_2, q_3, but not on q_4 in the sense that $T_{3,4}(q_4) \times \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} =$

$$\begin{pmatrix} \cos(q_4) & -\sin(q_4) & 0 & 0 \\ 0 & 0 & -1 & -d \\ \sin(q_4) & \cos(q_4) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -d \\ 0 \\ 1 \end{pmatrix}$$ does not depend on q_4.

We obtain the values of $q_1,\ q_2,\ q_3$ by successively premultiplying this equation by $T_{j,j-1}$, for $j = 1,2,3$, in order to compute one after the other these joint variables.

Equation of orientation

The equation corresponding to the orientation part of Eq. (2) is:

$$[S \quad N \quad A] = A_{0,6}(q_1, \cdots, q_6),$$

where $A_{0,6}$ is the orientation matrix (3×3) (included in $T_{0,6}$) that enables to orientate the frame R_6 with respect to the reference frame R_0 which leads to:

$$A_{3,0}(q_1, q_2, q_3) \times [S \quad N \quad A] = A_{3,6}(q_4, q_5, q_6).$$

Let $[F \quad G \quad H]$ be equal to matrix $A_{3,0}(q_1, q_2, q_3) \times [S \quad N \quad A]$ to simplify the previous matrix equation which leads to the following system of (3×3) equations:

$$[F \quad G \quad H] = A_{3,6}(q_4, q_5, q_6).$$

The values of matrix $[F \quad G \quad H]$ are known thanks to the previous computation of variables q_1, q_2, q_3. Thus, the values of q_4, q_5, q_6 are obtained by successively premultiplying the previous equation by $A_{4,3}$, then by $A_{5,4}$.

Application of Paul's method to the Stäubli RX 90 robot

Equation of position

We have:

$$\begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = T_{0,1}(q_1) \times T_{1,2}(q_2) \times T_{2,3}(q_3) \times T_{3,4}(q_4) \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

$$= \begin{bmatrix} cos(q_1)\ [d\ sin(q_2 + q_3) + a\ cos(q_2)] \\ sin(q_1)\ [d\ sin(q_2 + q_3) + a\ cos(q_2)] \\ d\ cos(q_2 + q_3) - a\ sin(q_2) \\ 1 \end{bmatrix}.$$

- By premultiplying this equation by the matrix $T_{1,0}$, we obtain:

$$T_{1,0} \times \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = T_{1,2} \times T_{2,3} \times T_{3,4} \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Let us recall that:

$$T_{0,1} = \begin{bmatrix} cos(q_1) & -sin(q_1) & 0 & 0 \\ sin(q_1) & cos(q_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ then } T_{1,0} = \begin{bmatrix} cos(q_1) & sin(q_1) & 0 & 0 \\ -sin(q_1) & cos(q_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ (}cf.\text{ §4.2),}$$

and

$$T_{1,2} \times T_{2,3} \times T_{3,4} \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

$$= \begin{bmatrix} cos(q_2) & -sin(q_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -sin(q_2) & -cos(q_2) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} cos(q_3) & -sin(q_3) & 0 & a \\ sin(q_3) & cos(q_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} cos(q_4) & -sin(q_4) & 0 & 0 \\ 0 & 0 & -1 & -d \\ sin(q_4) & cos(q_4) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

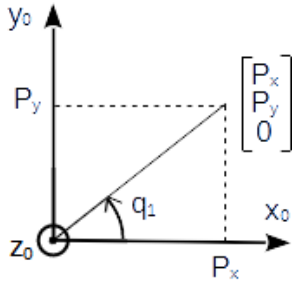
$$= \begin{bmatrix} cos(q_2+q_3) & -sin(q_2+q_3) & 0 & a\,cos(q_2) \\ 0 & 0 & 1 & 0 \\ -sin(q_2+q_3) & -cos(q_2+q_3) & 0 & -a\,sin(q_2) \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ -d \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} d\,sin(q_2+q_3) + a\,cos(q_2) \\ 0 \\ d\,cos(q_2+q_3) - a\,sin(q_2) \\ 1 \end{bmatrix}.$$

We obtain the following equations:

$$\begin{cases} cos(q_1)P_x + sin(q_1)P_y = d\,sin(q_2+q_3) + a\,cos(q_2) \\ -sin(q_1)P_x + cos(q_1)P_y = 0 \\ P_z = d\,cos(q_2+q_3) - a\,sin(q_2) \end{cases}.$$

The second equation enables to compute the value q_1, that is:

$$\frac{sin(q_1)}{cos(q_1)} = \frac{P_y}{P_x} \rightarrow \begin{vmatrix} q_1 = \text{atan}(P_y/P_x) \\ q_1' = q_1 + \pi \end{vmatrix}.$$



- By premultiplying $T_{1,0} \times \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = T_{1,2} \times T_{2,3} \times T_{3,4} \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ by the matrix $T_{2,1}$, we obtain:

$$T_{2,1} \times T_{1,0} \times \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = T_{2,3} \times T_{3,4} \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Let us recall that:

$$T_{1,2} = \begin{bmatrix} cos(q_2) & -sin(q_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -sin(q_2) & -cos(q_2) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ which leads to } T_{2,1} = \begin{bmatrix} cos(q_2) & 0 & -sin(q_2) & 0 \\ -sin(q_2) & 0 & -cos(q_2) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

then we obtain:

$$T_{2,1} \times T_{1,0} \times \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} cos(q_2) & 0 & -sin(q_2) & 0 \\ -sin(q_2) & 0 & -cos(q_2) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} cos(q_1) & sin(q_1) & 0 & 0 \\ -sin(q_1) & cos(q_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix},$$

$$= \begin{bmatrix} cos(q_2) & 0 & -sin(q_2) & 0 \\ -sin(q_2) & 0 & -cos(q_2) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} cos(q_1)P_x + sin(q_1)P_y \\ -sin(q_1)P_x + cos(q_1)P_y \\ P_z \\ 1 \end{bmatrix},$$

$$= \begin{bmatrix} cos(q_2)\ (cos(q_1)P_x + sin(q_1)P_y) - sin(q_2)P_z \\ -sin(q_2)\ (cos(q_1)P_x + sin(q_1)P_y) - cos(q_2)P_z \\ -sin(q_1)P_x + cos(q_1)P_y \\ 1 \end{bmatrix}.$$

Knowing that: $T_{2,3} \times T_{3,4} \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} cos(q_3) & -sin(q_3) & 0 & a \\ sin(q_3) & cos(q_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ -d \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} d\ sin(q_3) + a \\ -d\ cos(q_3) \\ 0 \\ 1 \end{bmatrix},$

we deduce the following system of equations:

$$\begin{bmatrix} cos(q_2)\ (cos(q_1)P_x + sin(q_1)P_y) - sin(q_2)P_z \\ -sin(q_2)\ (cos(q_1)P_x + sin(q_1)P_y) - cos(q_2)P_z \\ -sin(q_1)P_x + cos(q_1)P_y \\ 1 \end{bmatrix} = \begin{bmatrix} d\ sin(q_3) + a \\ -d\ cos(q_3) \\ 0 \\ 1 \end{bmatrix}.$$

Let $b_1 = cos(q_1)P_x + sin(q_1)P_y$, we deduce from the previous system of equations that:

$$\begin{cases} d\ sin(q_3) = cos(q_2)b_1 - sin(q_2)P_z - a \\ d\ cos(q_3) = sin(q_2)b_1 + cos(q_2)P_z \end{cases}. \quad (4)$$

By adding these 2 equations, previously squared, we obtain the following equation:

$d^2 = b_1^2 + P_z^2 + a^2 - 2ab_1\ cos(q_2) + 2aP_z\ sin(q_2)$,

that is,

$2aP_z\ sin(q_2) - 2ab_1\ cos(q_2) = -b_1^2 - P_z^2$ (knowing that $a = d$),

or equivalently:

$X\ sin(q_2) + Y\ cos(q_2) = Z$ with $X = 2aP_z; Y = -2ab_1; Z = -b_1^2 - P_z^2$.

We deduce from this equation the variable q_2, that is:

$$\begin{cases} sin(q_2) = \frac{XZ + \varepsilon Y\sqrt{X^2+Y^2-Z^2}}{X^2+Y^2} \\ cos(q_2) = \frac{YZ - \varepsilon X\sqrt{X^2+Y^2-Z^2}}{X^2+Y^2} \end{cases} \text{ with } \varepsilon = \pm 1,$$

to finally obtain knowing that $X^2 + Y^2 > 0$:

$q_2 = \text{atan2}(XZ + \varepsilon Y\sqrt{X^2+Y^2-Z^2}, YZ - \varepsilon X\sqrt{X^2+Y^2-Z^2})$ with $\varepsilon = \pm 1$.

From the system of Eq. (4) and knowing the value of variable q_2, we also deduce the value of variable q_3, that is:

$q_3 = \text{atan2}(b_1\ cos(q_2) - P_z\ sin(q_2) - a, b_1\ sin(q_2) + P_z\ cos(q_2))$.

Equation of orientation

Once the variables q_1, q_2, q_3 are obtained, we compute the variables q_4, q_5, q_6 from the following relation:

$$[F \quad G \quad H] = A_{3,6}(q_4, q_5, q_6) \tag{5}$$

with

$$[F \quad G \quad H] = A_{3,0}(q_1, q_2, q_3) \times [S \quad N \quad A].$$

Computation of matrix $A_{3,0}$:

$$A_{3,0} = A_{3,2} \times A_{2,1} \times A_{1,0} = \begin{pmatrix} cos(q_3) & sin(q_3) & 0 \\ -sin(q_3) & cos(q_3) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cos(q_2) & 0 & -sin(q_2) \\ -sin(q_2) & 0 & -cos(q_2) \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} cos(q_1) & sin(q_1) & 0 \\ -sin(q_1) & cos(q_1) & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

$$= \begin{pmatrix} cos(q_3)cos(q_2)cos(q_1) - sin(q_3)sin(q_2)cos(q_1) & cos(q_3)cos(q_2)sin(q_1) - sin(q_3)sin(q_2)sin(q_1) & -cos(q_3)sin(q_2) - sin(q_3)cos(q_2) \\ -sin(q_3)cos(q_2)cos(q_1) - cos(q_3)sin(q_2)cos(q\theta_1) & -sin(q_3)cos(q_2)sin(q_1) - cos(q_3)sin(q_2)sin(q_1) & sin(q_3)sin(q_2) - cos(q_3)cos(q_2) \\ -sin(q_1) & cos(q_1) & 0 \end{pmatrix},$$

$$= \begin{pmatrix} cos(q_2+q_3)cos(q_1) & cos(q_2+q_3)sin(q_1) & -sin(q_2+q_3) \\ -sin(q_2+q_3)cos(q_1) & -sin(q_2+q_3)sin(q_1) & -cos(q_2+q_3) \\ -sin(q_1) & cos(q_1) & 0 \end{pmatrix}.$$

Computation of matrix $[F \quad G \quad H] = A_{3,0} \times [S \quad N \quad A]$:

We have:

$$F = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = A_{3,0} \times \begin{bmatrix} S_x \\ S_y \\ S_z \end{bmatrix} = \begin{pmatrix} cos(q_2+q_3)(cos(q_1)S_x + sin(q_1)S_y) - sin(q_2+q_3)S_z \\ -sin(q_2+q_3)(cos(q_1)S_x + sin(q_1)S_y) - cos(q_2+q_3)S_z \\ -sin(q_1)S_x + cos(q_1)S_y \end{pmatrix},$$

$$G = \begin{bmatrix} G_x \\ G_y \\ G_z \end{bmatrix} = A_{3,0} \times \begin{bmatrix} N_x \\ N_y \\ N_z \end{bmatrix} = \begin{pmatrix} cos(q_2+q_3)(cos(q_1)N_x + sin(q_1)N_y) - sin(q_2+q_3)N_z \\ -sin(q_2+q_3)(cos(q_1)N_x + sin(q_1)N_y) - cos(q_2+q_3)N_z \\ -sin(q_1)N_x + cos(q_1)N_y \end{pmatrix},$$

$$H = \begin{bmatrix} H_x \\ H_y \\ H_z \end{bmatrix} = A_{3,0} \times \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} = \begin{pmatrix} cos(q_2+q_3)(cos(q_1)A_x + sin(q_1)A_y) - sin(q_2+q_3)A_z \\ -sin(q_2+q_3)(cos(q_1)A_x + sin(q_1)A_y) - cos(q_2+q_3)A_z \\ -sin(q_1)A_x + cos(q_1)A_y \end{pmatrix}.$$

- By premultiplying Eq. (5) by matrix $A_{4,3}$, we obtain:

$$A_{4,3}\ [F \quad G \quad H] = A_{4,3}\ A_{3,6}(q_4, q_5, q_6),$$

with

$$A_{4,3} = \begin{pmatrix} cos(q_4) & 0 & sin(q_4) \\ -sin(q_4) & 0 & cos(q_4) \\ 0 & -1 & 0 \end{pmatrix}.$$

The resulting system is the following:

$$\begin{pmatrix} cos(q_4)F_x + sin(q_4)F_z & cos(q_4)G_x + sin(q_4)G_z & cos(q_4)H_x + sin(q_4)H_z \\ -sin(q_4)F_x + cos(q_4)F_z & -sin(q_4)G_x + cos(q_4)G_z & -sin(q_4)H_x + cos(q_4)H_z \\ -F_y & -G_y & -H_y \end{pmatrix} = \begin{pmatrix} cos(q_5)cos(q_6) & -cos(q_5)sin(q_6) & sin(q_5) \\ sin(q_6) & cos(q_6) & 0 \\ -sin(q_5)cos(q_6) & sin(q_5)sin(q_6) & cos(q_5) \end{pmatrix}.$$

The element (2, 3), that is $(-sin(q_4)H_x + cos(q_4)H_z = 0)$, enables to compute the variable q_4. Indeed, we have:

$$\frac{sin(q_4)}{cos(q_4)} = \frac{H_z}{H_x} \rightarrow \begin{vmatrix} q_4 = \text{atan}(H_z/H_x) \\ q_4' = q_4 + \pi \end{vmatrix}.$$

Moreover, the elements (1,3) and (3,3) enable to compute the variable q_5, we have:

$$\begin{cases} cos(q_4)H_x + sin(q_4)H_z = sin(q_5) \\ -H_y = cos(q_5) \end{cases},$$

which leads to:

$q_5 = \text{atan2}(cos(q_4)H_x + sin(q_4)H_z, -H_y)$.

Finally, the elements (2,1) and (2,2) enable to compute the variable q_6, we have:

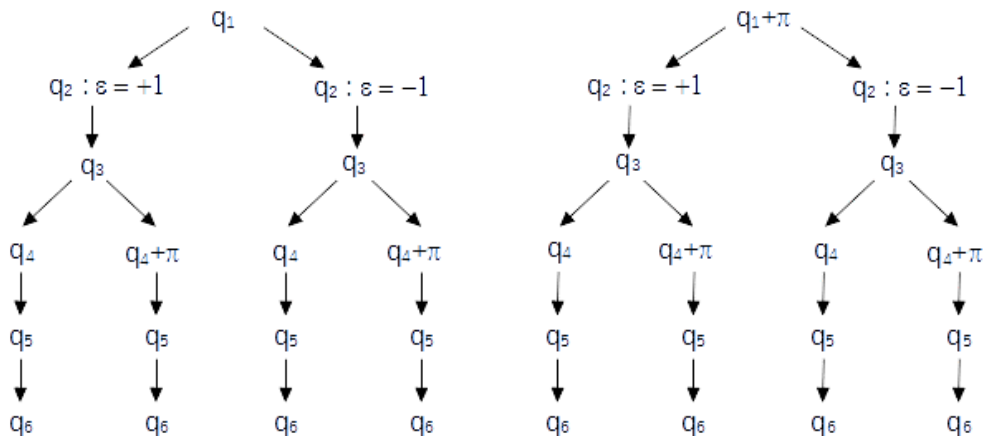
$$\begin{cases} -sin(q_4)F_x + cos(q_4)F_z = sin(q_6) \\ -sin(q_4)G_x + cos(q_4)G_z = cos(q_6) \end{cases}$$

which leads to:

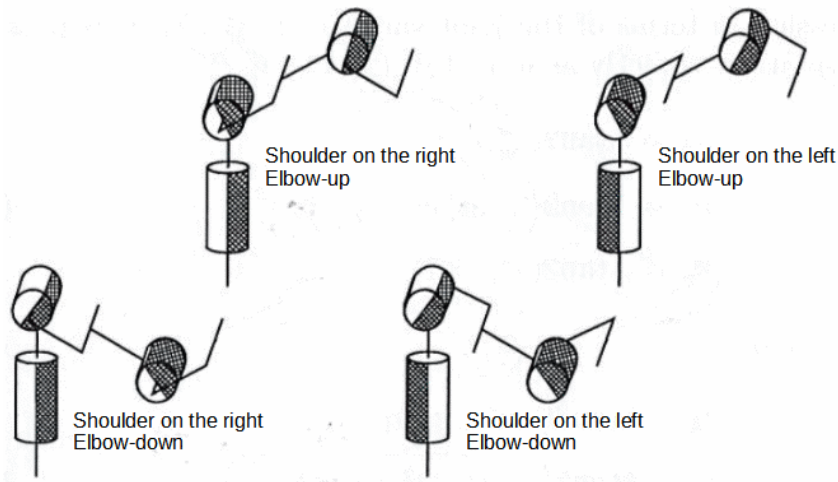
$q_6 = \text{atan2}(-sin(q_4)F_x + cos(q_4)F_z, -sin(q_4)G_x + cos(q_4)G_z)$.

Number of solutions to the inverse geometric model:

The inverse geometric model of the Stäubli RX-90 robot provides 8 solutions. For particular positions, named *singular positions* (described in §4.9), an infinite number of solutions exists. For example when $q_1 = \cdots = q_6 = 0$ (corresponding to the initial posture of the robot, see §4.5), the arguments of the function $atan2$ used to compute the variable q_4 are null ($H_x = H_z = 0$) which leads to an indeterminate variable. The choice of this variable being arbitrary, the value of q_4 is frequently set to its current value (which sets the value of variable q_6).



The following figure describes four possible postures of the robot shoulder (function of q_1, q_2, q_3) leading to a same position of point O_4 depending on whether the shoulder is on the right or on the left (function of q_1) and whether the elbow is up or down (function of q_2, q_3).



Note that perhaps some postures cannot be reached due to constraints (limits) on the joint variables.

The change of robot posture during the next motion can be specified in most robot programming languages. For example in V+ language, instructions LEFTY, resp. RIGHTY, request a change so that the first two links of the robot look like a human's left, resp. right, arm; instructions ABOVE, resp. BELOW, request a change so that the elbow is above, resp. below, the line from the shoulder to the wrist.

Exercise:

1) Propose a script (MatLab or Scilab) to compute the four sets of variables q_1, q_2, q_3 as a function of planned position of point P (centre of wrist), that is P_X, P_Y, P_Z. To verify your script, consider the initial posture of robot (see §4.5) and find (again) the set of variables $q_1 = q_2 = q_3 = 0$ in the four possible ones (note that $a = d = 0{,}45\ m$).

2) The control panel of Staübli RX 90 robot gives several information including:

- (in World mode) the end-effector location, that is, the point F location situated at 8,5 *cm* from the point P $(= O_6)$ along the $\overrightarrow{o_6 z_6}$ axis, through six data:
  - the position of point F through its coordinates X, Y, Z in *mm*;
  - the orientation of the frame R_F attached to the end-effector (which is the same as that of the frame R_6) by using the angles y, p, r (*yaw*, *pitch*, *roll*[17]), in degree, corresponding respectively to the three Euler angles $\psi,\ \theta,\ \varphi$ according to the convention (z, y, z), described in §2.1.
- (in Joint mode) the joint values noted $J_1, \cdots, J_6$ (corresponding to $q_1, \cdots, q_6$).

Consider for example the following values corresponding to an arbitrary end-effector location:

X (*mm*)	Y (*mm*)	Z (*mm*)	y (°)	p (°)	r (°)
598,629	-372,697	518,632	-23,395	93,034	47,881
J_1 (°)	J_2 (°)	J_3 (°)	J_4 (°)	J_5 (°)	J_6 (°)
-33,064	-65,607	141,025	29,283	20,053	19,586

Deduce the values of variables q_1, q_2, q_3 from the information giving the location of the point F, that is X, Y, Z, y, p, r. To do this, complete the previous script by deducing the coordinates of point P, that is (P_X, P_Y, P_Z), from the coordinates of point F, that is (X, Y, Z) (given in the control panel).

3) Suppose that you dispose of the whole part of the inverse geometric model (that is the eight sets of variables $q_1, q_2, \cdots, q_6$ for a given location of point P). Express the value of the SNA matrix (which indicates the orientation of the frame attached to point P) corresponding to point F (defined by X, Y, Z, y, p, r).

Exercise: Do the exercises of Tutorial 2.

4.7 Direct kinematic model

The ***direct kinematic model*** of a robot expresses the *linear and angular velocities*[18] $(\dot{X})$ *of the end-effector* as a function of the *joint velocities* $(\dot{q})$ through the following equation:

$$\dot{X} = J(q)\ \dot{q}$$

where $J(q)$ denotes the *Jacobian matrix*.

The Jacobian matrix enables among other things:

[17] *yaw*: 'lacet', *pitch*: 'tangage', *roll*: 'roulis'.
[18] *Linear and angular velocities* are sometimes also called *velocities of translation and rotation* respectively.

- To compute, through its inversion, the velocities to be applied to the joints in order to obtain a planned velocity (linear and angular) of the end-effector,
- To compute *numerically* (*i.e.*, without using a mathematical model), through its inversion, a solution for the inverse geometric model (see TD described in file *TD_MGI_Newton.pdf*),
- To compute, through its transpose, the static model of the robot in order to have the joint forces (for prismatic joints) and joint torques (for rotoid joints) to be applied at the joints necessary to exert specified forces or moments by the end-effector on the environment,
- To determine the singularities and to analyse the workspace of the robot.

1) Computation of the Jacobian matrix from the direct geometric model

The Jacobian matrix J can be obtained by differentiating the direct geometric model of the robot ($X = f(q)$). Let m be the dimension of X and n the dimension of q with $m = n = 6$ in standard case of industrial robots (for example the Stäubli RX90 robot).

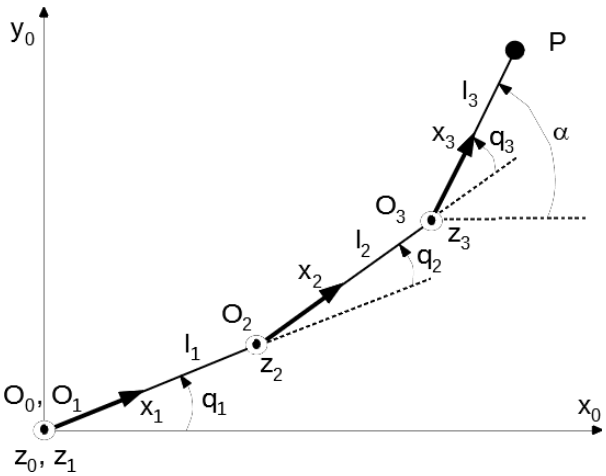
We have: $\dot{X}(t) = \dot{f}(q(t)) = \frac{d}{dt} f(q(t)) = \frac{\partial f}{\partial q}\ \dot{q}(t) = J(q)\ \dot{q}(t)$,

with

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial q_1} & \cdots & \frac{\partial f_1}{\partial q_n} \\ \vdots & \ddots & \ddots \\ \frac{\partial f_m}{\partial q_1} & \cdots & \frac{\partial f_m}{\partial q_n} \end{pmatrix}.$$

This approach is convenient for robots with few *d.o.f.* as the one described in the following example. A direct method, described below, is recommended for more complex robots.

Example: Let us consider the three rotoid joints (q_1, q_2, q_3) planar robot where l_1, l_2, l_3 denote the lengths of its three links, see figure below.



Let (P_x, P_y) be the coordinates of point P in frame $(O_0, \overrightarrow{x_0}, \overrightarrow{y_0})$ and α be the angle between $\overrightarrow{o_0x_0}$ and $\overrightarrow{o_3x_3}$ axes. Let us note that the linear velocity along the $\overrightarrow{o_0z_0}$ axis and the angular velocities about $\overrightarrow{o_0x_0}, \overrightarrow{o_0y_0}$ axes are null. The direct geometric model is such that:

$$\begin{cases} P_x = l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) + l_3 \cos(q_1 + q_2 + q_3) \\ P_y = l_1 \sin(q_1) + l_2 \sin(q_1 + q_2) + l_3 \sin(q_1 + q_2 + q_3) \\ \alpha = q_1 + q_2 + q_3 \end{cases}.$$

A differentiation of this model leads to the following relation:

$$\begin{pmatrix} \dot{P}_x \\ \dot{P}_y \\ \dot{\alpha} \end{pmatrix} = J_P \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{pmatrix}$$

where J_P is the Jacobian at point P with respect to reference frame R_0 equal to:

$$\begin{pmatrix} -l_1 \sin(q_1) - l_2 \sin(q_1+q_2) - l_3 \sin(q_1+q_2+q_3) & -l_2 \sin(q_1+q_2) - l_3 \sin(q_1+q_2+q_3) & -l_3 \sin(q_1+q_2+q_3) \\ l_1 \cos(q_1) + l_2 \cos(q_1+q_2) + l_3 \cos(q_1+q_2+q_3) & l_2 \cos(q_1+q_2) + l_3 \cos(q_1+q_2+q_3) & l_3 \cos(q_1+q_2+q_3) \\ 1 & 1 & 1 \end{pmatrix}.$$

$\dot{P}_x, \dot{P}_y$ represent the two components of linear velocity along $\overrightarrow{o_0x_0}, \overrightarrow{o_0y_0}$ axes respectively; $\dot{\alpha}$ represents the component of angular velocity about the $\overrightarrow{o_0z_0}$ axis.

2) Direct method to compute the Jacobian matrix

The method we describe below enables the computing of the Jacobian matrix, denoted J_{O_n}, at point O_n (origin of frame R_n) with respect to reference frame R_0 without differentiating the direct geometric model. The obtained matrix expresses the relationship between the joint velocities (noted $\dot{q}$) and the linear and angular velocities (noted V_n, ω_n resp.) of the frame R_n (attached to the end-effector). These linear and angular velocities are expressed by a kinematic screw[19] (noted $\begin{bmatrix} V_n \\ \omega_n \end{bmatrix}$) which leads to the following equation:

$$\begin{bmatrix} V_n \\ \omega_n \end{bmatrix} = J_{O_n} \dot{q}.$$

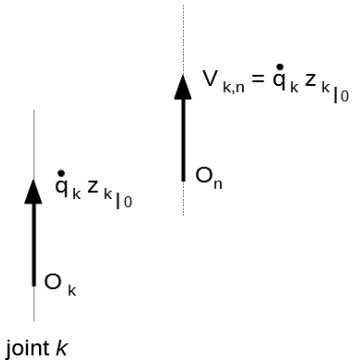
Let us consider the joint *k* (along $\overrightarrow{o_kz_k}$ axis if joint is *prismatic* or about $\overrightarrow{o_kz_k}$ axis if joint is *rotoid*) by assuming that the other joints are fixed. A velocity $\dot{q}_k$ of joint *k* produces:

- a linear velocity $V_{k,n}$ of point O_n (origin of frame R_n),
- and an angular velocity $\omega_{k,n}$ of frame R_n,

with respect to reference frame R_0,

which leads to the kinematic screw: $\begin{bmatrix} V_{k,n} \\ \omega_{k,n} \end{bmatrix}$.

The case of a **prismatic joint** *k* along $\overrightarrow{o_kz_k}$ axis is depicted in the following figure.

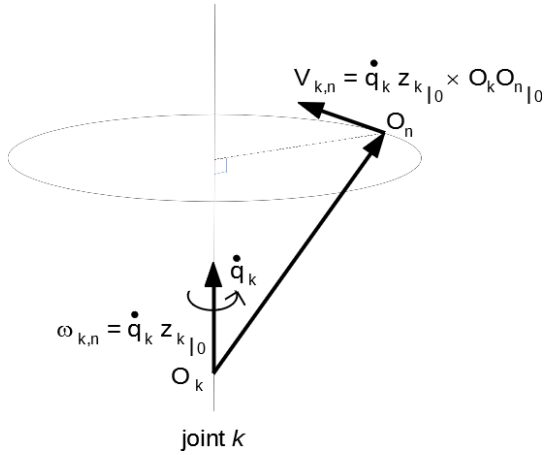


Thus, we obtain:

$$\begin{matrix} V_{k,n} = \dot{q}_k \, z_{k|0} \\ \omega_{k,n} = 0 \end{matrix}.$$

The case of a **rotoid joint** *k* about $\overrightarrow{o_kz_k}$ axis is depicted in the following figure.

[19] 'torseur cinématique'.



Thus, we obtain:

$$\begin{aligned} V_{k,n} &= \dot{q}_k \, z_{k|0} \times O_k O_{n|0} \\ \omega_{k,n} &= \dot{q}_k \, z_{k|0} \end{aligned},$$

knowing that: $O_k O_{n|0} = R_{0,k} \, O_k O_{n|k}$ or $O_k O_{n|0} = O_0 O_{n|0} - O_0 O_{k|0}$.

N.B.: The symbol × corresponds to the *vector product*. Let $x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ and $y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$ be two vectors in the Euclidean space, we have: $x \times y = \begin{pmatrix} x_2 y_3 - x_3 y_2 \\ x_3 y_1 - x_1 y_3 \\ x_1 y_2 - x_2 y_1 \end{pmatrix}$.

For more details, see *https://fr.wikipedia.org/wiki/Vecteur_vitesse_angulaire*
or *https://fr.wikipedia.org/wiki/Torseur_cinématique*

By using the binary parameter σ_k (1 for *prismatic* joint, 0 for *rotoid* joint), the two previous relations can be aggregated for any joint *k* (prismatic or rotoid) in the following way:

$$\begin{aligned} V_{k,n} &= \left(\sigma_k \, z_{k|0} + \bar{\sigma}_k \, (z_{k|0} \times O_k O_{n|0})\right) \dot{q}_k \\ \omega_{k,n} &= \left(\bar{\sigma}_k \, z_{k|0}\right) \dot{q}_k \end{aligned}.$$

Thus, we obtain the relation below by considering all the joints $(q_1, \cdots, q_n)$ of the robot:

$$\begin{bmatrix} V_n \\ \omega_n \end{bmatrix} = J_{O_n} \dot{q}$$

with the following $6 \times n$ Jacobian matrix:

$$J_{O_n} = \begin{pmatrix} \sigma_1 \, z_{1|0} + \bar{\sigma}_1 \, (z_{1|0} \times O_1 O_{n|0}) & \cdots & \sigma_n \, z_{n|0} + \bar{\sigma}_n \, (z_{n|0} \times O_n O_{n|0}) \\ \bar{\sigma}_1 \, z_{1|0} & \cdots & \bar{\sigma}_n \, z_{n|0} \end{pmatrix}.$$

Example: Let us apply the method on the three rotoid joints ($\sigma_1 = \sigma_2 = \sigma_3 = 0$) planar robot described in the previous example. We have:

$$\begin{bmatrix} V_3 \\ \omega_3 \end{bmatrix} = J_{O_3} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{pmatrix}$$

with

$$J_{O_3} = \begin{pmatrix} z_{1|0} \times O_1 O_{3|0} & z_{2|0} \times O_2 O_{3|0} & z_{3|0} \times O_3 O_{3|0} \\ z_{1|0} & z_{2|0} & z_{3|0} \end{pmatrix},$$

or equivalently:

$$J_{O_3} = \begin{pmatrix} z_{1|0} \times (O_0O_{3|0} - O_0O_{1|0}) & z_{2|0} \times (O_0O_{3|0} - O_0O_{2|0}) & z_{3|0} \times (O_0O_{3|0} - O_0O_{3|0}) \\ z_{1|0} & z_{2|0} & z_{3|0} \end{pmatrix}.$$

Knowing that: $O_0O_{1|0} = 0$, $O_0O_{2|0} = \begin{pmatrix} l_1\cos(q_1) \\ l_1\sin(q_1) \\ 0 \end{pmatrix}$, $O_0O_{3|0} = \begin{pmatrix} l_1\cos(q_1) + l_2\cos(q_1+q_2) \\ l_1\sin(q_1) + l_2\sin(q_1+q_2) \\ 0 \end{pmatrix}$,

we have:

$$z_{1|0} \times \left(O_0O_{3|0} - O_0O_{1|0}\right) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times \begin{pmatrix} l_1\cos(q_1) + l_2\cos(q_1+q_2) \\ l_1\sin(q_1) + l_2\sin(q_1+q_2) \\ 0 \end{pmatrix},$$
$$= \begin{pmatrix} -l_1\sin(q_1) - l_2\sin(q_1+q_2) \\ l_1\cos(q_1) + l_2\cos(q_1+q_2) \\ 0 \end{pmatrix},$$

$$z_{2|0} \times \left(O_0O_{3|0} - O_0O_{2|0}\right) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times \begin{pmatrix} l_2\cos(q_1+q_2) \\ l_2\sin(q_1+q_2) \\ 0 \end{pmatrix},$$
$$= \begin{pmatrix} -l_2\sin(q_1+q_2) \\ l_2\cos(q_1+q_2) \\ 0 \end{pmatrix},$$

$$z_{3|0} \times \left(O_0O_{3|0} - O_0O_{3|0}\right) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},$$

which leads to:

$$J_{O_3} = \begin{pmatrix} -l_1\sin(q_1) - l_2\sin(q_1+q_2) & -l_2\sin(q_1+q_2) & 0 \\ l_1\cos(q_1) + l_2\cos(q_1+q_2) & l_2\cos(q_1+q_2) & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

Only the three non-null rows of matrix J_{O_3} are relevant (the rank of the matrix being at most equal to 3); the three null rows are due to the fact that the linear velocity along the $\overrightarrow{o_0z_0}$ axis and the angular ones about $\overrightarrow{o_0x_0}, \overrightarrow{o_0y_0}$ axes are null.

Let us compute the Jacobian matrix noted J_P in the previous example in order to express the kinematic screw attached to point P, noted $\begin{bmatrix} V_P \\ \omega_P \end{bmatrix}$, as a function of the joint velocities. We use the relation linking V_P, ω_P to V_3, ω_3, that is:

$$\begin{matrix} V_P = V_3 + \omega_3 \times O_3P_{|0} \\ \omega_P = \omega_3 \end{matrix}.$$

Let us transform the *vector product* used in the previous relation into a *matrix product*, that is:

$$V_P = V_3 - \widehat{O_3P}_{|0}\, \omega_3.$$

Indeed, let u, v be two vectors of the Euclidean space, we have: $u \times v = -v \times u = -\hat{v}\, u$ where $\hat{v}$ is the *skew-symmetric* matrix[20] equal to:

[20] A matrix A is *skew-symmetric* ('antisymétrique') if $A^t = -A$.

$$\begin{pmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{pmatrix}.$$

It follows the equivalent matrix relation:

$$\begin{bmatrix} V_P \\ \omega_P \end{bmatrix} = \begin{bmatrix} I_3 & -\widehat{O_3P}_{|0} \\ 0_3 & I_3 \end{bmatrix} \begin{bmatrix} V_3 \\ \omega_3 \end{bmatrix},$$

where $I_3, 0_3$ are the (3×3) identity and null matrices respectively,

which corresponds to the following Jacobian matrix J_P:

$$J_P = \begin{bmatrix} I_3 & -\widehat{O_3P}_{|0} \\ 0_3 & I_3 \end{bmatrix} J_{O_3},$$

with: $\widehat{O_3P}_{|0} = \begin{pmatrix} 0 & 0 & l_3 \sin(q_1+q_2+q_3) \\ 0 & 0 & -l_3 \cos(q_1+q_2+q_3) \\ -l_3 \sin(q_1+q_2+q_3) & l_3 \cos(q_1+q_2+q_3) & 0 \end{pmatrix}.$

We finally obtain the following Jacobian matrix:

$$J_P = \begin{pmatrix} -l_1 \sin(q_1) - l_2 \sin(q_1+q_2) - l_3 \sin(q_1+q_2+q_3) & -l_2 \sin(q_1+q_2) - l_3 \sin(q_1+q_2+q_3) & -l_3 \sin(q_1+q_2+q_3) \\ l_1 \cos(q_1) + l_2 \cos(q_1+q_2) + l_3 \cos(q_1+q_2+q_3) & l_2 \cos(q_1+q_2) + l_3 \cos(q_1+q_2+q_3) & l_3 \cos(q_1+q_2+q_3) \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

As for matrix J_{O_3}, only three non-null rows of the matrix are relevant which leads to the expression obtained in the previous example, that is:

$$J_P = \begin{pmatrix} -l_1 \sin(q_1) - l_2 \sin(q_1+q_2) - l_3 \sin(q_1+q_2+q_3) & -l_2 \sin(q_1+q_2) - l_3 \sin(q_1+q_2+q_3) & -l_3 \sin(q_1+q_2+q_3) \\ l_1 \cos(q_1) + l_2 \cos(q_1+q_2) + l_3 \cos(q_1+q_2+q_3) & l_2 \cos(q_1+q_2) + l_3 \cos(q_1+q_2+q_3) & l_3 \cos(q_1+q_2+q_3) \\ 1 & 1 & 1 \end{pmatrix}.$$

Remark: A more efficient method to obtain the Jacobian matrix noted J_P in the previous example (expressing the kinematic screw attached to point P, noted $\begin{bmatrix} V_P \\ \omega_P \end{bmatrix}$, as a function of the joint velocities) simply consists in replacing the coordinates O_n (of frame R_n) by the ones of point P in the expression of the Jacobian matrix J_{O_n}, that is:

$$J_P = \begin{pmatrix} \sigma_1 \, z_{1|0} + \bar{\sigma}_1 \, (z_{1|0} \times O_1P_{|0}) & \cdots & \sigma_n \, z_{n|0} + \bar{\sigma}_n \, (z_{n|0} \times O_nP_{|0}) \\ \bar{\sigma}_1 \, z_{1|0} & \cdots & \bar{\sigma}_n \, z_{n|0} \end{pmatrix}.$$

In the previous example on the 3R planar robot, this leads to compute:

$$J_P = \begin{pmatrix} z_{1|0} \times (O_0P_{|0} - O_0O_{1|0}) & z_{2|0} \times (O_0P_{|0} - O_0O_{2|0}) & z_{3|0} \times (O_0P_{|0} - O_0O_{3|0}) \\ z_{1|0} & z_{2|0} & z_{3|0} \end{pmatrix}.$$

Exercise: Let us consider the RX90 Stäubli robot described in §4.5, in particular the figure describing its initial posture. The objective is to compute the Jacobian matrix corresponding to the shoulder part of the robot (where the three first joints q_1, q_2, q_3 involve). More precisely, we look for the Jacobian matrix noted J_P expressing the kinematic screw attached to the point P (corresponding to points O_4, O_5, O_6, situated at the intersection of the three concurrent axes of the robot) as a function of the joint velocities $\dot{q}_1, \dot{q}_2, \dot{q}_3$.

Let us use the two methods described previously to compute the Jacobian matrix J_P, that is:

- by differentiating the direct geometric model,
- by using the 'direct' method. As for the previous example, the computation of J_P is possible:

- by pre-calculating the Jacobian matrix J_{O_3} (that is the matrix enabling to obtain the linear velocity (noted V_3) of point O_3 and the angular velocity (noted ω_3) of frame R_3 (represented in the figure describing the initial posture of the robot)),
- or directly by considering the point P, as mentioned in the previous remark.

4.8 Dimension of the task space

At a given joint posture/configuration q, the rank r (definition of rank is recalled below) of the $(m \times n)$ Jacobian matrix J (able to express the kinematic screw attached to the frame associated with the end-effector of a robot) corresponds to the number of *d.o.f. of the end-effector*. This rank defines the dimension of the accessible task space at the joint configuration q. The number, denoted M, of the *d.o.f. of the task space* of a robot is equal to the maximum rank, denoted r_{max}, that the Jacobian matrix can have by considering all possible joint configurations. Let N be the number of *d.o.f. of a robot* (equal to the number of joints of a serial robot, see §2.3), three cases are considered:

- if $N = M$, the robot is *non redundant* and has just the number of joints required to provide M *d.o.f.* to the end-effector;
- if $N > M$, the robot is *redundant* of order $N - M$. The robot has more joints than required to provide M *d.o.f.* to the end-effector;
- at a given joint configuration q, if $r < M$, the rank of the Jacobian matrix is deficient. The robot is at a *singular configuration* of order $M - r$. At this configuration, the robot cannot generate an end-effector velocity along some directions, called *degenerate directions*, of the task space. When the Jacobian matrix J is square, the *singularities* are obtained by calculating the roots of $\det(J) = 0$, see §4.9 for more details. They correspond to the roots of $\det(J\,J^t) = 0$ for redundant robots.

Definitions (minor, rank):

- The *minors* of a matrix are the determinants of its square sub-matrices. Let A be a matrix of order (*i.e.*, dimension) $(m \times n)$, a *minor* of order k is the determinant of a $(k \times k)$ sub-matrix obtained by removing $m - k$ rows and $n - k$ columns of the matrix A, which can be denoted $\det(A_{I,J})$ where I, resp. J, is a set of k elements of $\{1, \cdots, m\ \}$, resp. $\{1, \cdots, n\ \}$.
- The *rank* of a matrix is the *largest* order of any non-null *minor* in the matrix. In other words, it exists a non-null *minor* of order r of the matrix and all *minors* of order strictly greater than r are null.

4.9 Multiple solutions – Workspace – Aspects

Let us consider the direct geometric model (defined from the joint space (Q) to the task space (X)) of a robot through the following application f:

$$X = f(q),$$

or equivalently:

$$X_i = f_i(q_1, q_2, \ldots, q_n), \quad i = 1, \cdots, n.$$

The application f is not *bijective*, in the sense that to each element of Q space corresponds only one image in X space whereas an element of X can be the image of several elements of Q.
Indeed, most of the robots have the possibility to reach a target (in task space) through several configurations, also called postures (for example with the posture ‘elbow-up’ and ‘elbow-down’). For example, the Staübli RX 90 robot has 8 joint solutions/configurations/postures for the *inverse geometric model* which means that 8 joint solutions/configurations are possible to reach a *point* in the task space. A change of posture may be necessary to avoid a joint limit on an axis (as illustrated in the next example on the planar RR robot) or a collision with an obstacle.

Some trajectories require the use of the inverse geometric model to be generated because they must be controlled *at all times* (e.g., by having to be straight) in the presence of obstacles close to the manipulator. Also, the robot may have to reconfigure itself (change its configurations) during a trajectory which can be dangerous in the presence of nearby obstacles.
An alternative to prevent "unintentional" reconfiguration of the manipulator is to subdivide the joint space Q into a set of *m* disjointed domains, noted $(A_i),\ i = 1, \cdots,\ m$, called *aspects*[21] in which the application f is *bijective*. So, *in each of these domains*, it is guaranteed that each corresponding point in the task space (that is $f((A_i))$) can be reached through one configuration only.

The computation of these domains is based on the fact that the application f is *bijective* in a restricted domain of Q if in this domain the determinant of the Jacobian matrix J (equal to:

$$\begin{pmatrix} \frac{\partial f_1}{\partial q_1} & \cdots & \frac{\partial f_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial q_1} & \cdots & \frac{\partial f_n}{\partial q_n} \end{pmatrix},$$

keeps a constant sign, which enables an exhaustive description of the task space of the robot[22].

Workspace – Aspects

The joint domain Q is commonly an hyperparallelepiped defined by:

$$q_{i_{min}} \leq q_i \leq q_{i_{max}}, \quad i = 1, \cdots, n,$$

where $q_{i_{min}}$ and $q_{i_{max}}$ are the limits of joint i.

The equation $\det(J\,(q)) = 0$ defines a partition of the joint space Q into m disjointed domains $(A_i),\ i = 1, \cdots,\ m$, called ***aspects***. Thus, an *aspect* is defined by the relation:

$$\forall q \in (A_i), \quad \text{Sign of } (\det(J(q)) = \text{constant}$$

with the following properties:

$$(A_i) \cap (A_j) = \emptyset \quad \forall i, j = 1, \ldots, m,$$
$$(Q) = \bigcup_{i=1}^{m} (A_i)\,.$$

Remark: The configurations for which $\det(J\,(q)) = 0$ are called *singular configurations* or *singularities* of the mechanism, see §4.10 for a description of such configurations.

Let us define the sub-domain, noted (X_i), of the task space (X) of points x corresponding to all joints q in aspect (A_i), that is:

$$(X_i) = Im(A_i).$$

In general, domains (X_i) are not disjointed (*i.e.*, $\exists\ i, j\ (i \neq j) \mid (X_i) \cap (X_j) \neq \emptyset$) in contrast to domains (A_i).
And the task space (X) corresponds to the union of the image of the *m aspects*, that is:

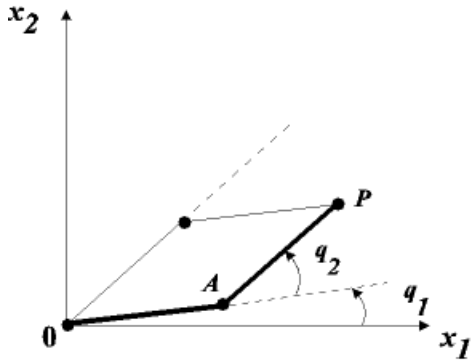
$$(X) = \cup_{i=1}^{m} Im(A_i).$$

[21] Borrel P., *Contribution à la modélisation géométrique des robots manipulateurs ; application à la conception assistée par ordinateur*, Thèse d'Etat, USTL Montpellier, juillet 1986.
[22] This result applies to most industrial robots, more precisely to *non-cuspidal* robots, see *Modeling, Identification & Control of Robots*, W. Khalil, E. Dombre, Hermes Penton Science 2002 for more details.

The manipulator is said to be in the *configuration*, or *posture*, number i for a joint $q \in (A_i)$. By example, the point $x \in (X_i) \cap (X_j) \cap (X_k)$ is the image of a joint of (A_i), a joint of (A_j) and a joint of (A_k) which means that the point x can be reached by postures i, j and k.

This description enables to solve the problem of the end-effector motion in a given posture of the manipulator from a point x^1 to a point x^2. In particular, it may be interesting to know whether these two points can be joined by a given trajectory, e.g., a straight line.

Example: Let us consider the planar RR robot described in §4.6 with $l_1 = l_2 = L$.



- **Computation of *aspects***

Consider the corresponding direct geometric model:

$$\begin{cases} x_1 = L\ cos(q_1) + L\ cos(q_1 + q_2) = f_1(q_1, q_2) \\ x_2 = L\ sin(q_1) + L\ sin(q_1 + q_2) = f_2(q_1, q_2) \end{cases},$$

with joint limits:

$$0° \leq q_1 \leq 90°,\ -100° \leq q_2 \leq 90°.$$

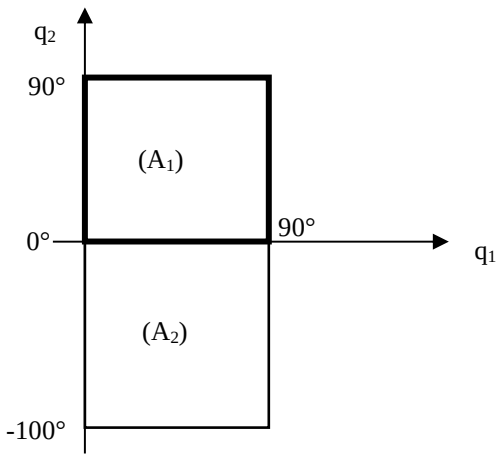
We have the following *Jacobian* matrix:

$$J = \frac{\partial f}{\partial q} = \begin{pmatrix} -L\ sin(q_1) - L\ sin(q_1 + q_2) & -L\ sin(q_1 + q_2) \\ L\ cos(q_1) + L\ cos(q_1 + q_2) & L\ cos(q_1 + q_2) \end{pmatrix},$$

which leads to:

$$\det(J) = L^2\ sin(q_2).$$

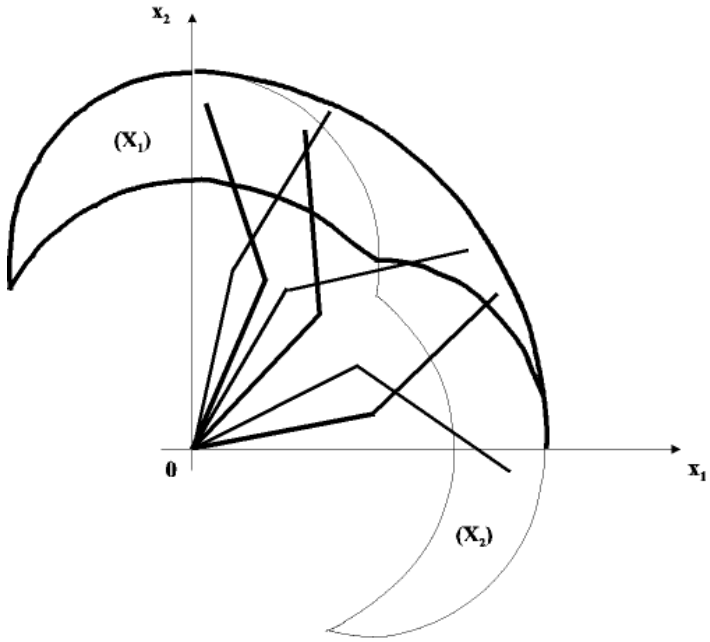
Knowing that $\det(J) = 0$ implies $q_2 = 0$, we deduce that the joint domain (Q) is split in two *aspects* (A_1) and (A_2) according to the following figure:



The *aspect* (A_1) is the domain represented in bold line and corresponds to $\det(J) > 0$, that is $q_2 > 0$. The corresponding posture of the robot is said to be *elbow-down* (domain in bold line in the figure below representing the workspace of the robot).
The *aspect* (A_2) is the domain represented in thin line and corresponds to $\det(J) < 0$, that is $q_2 < 0$. The corresponding posture is said to be *elbow-up* (domain in thin line in the figure below representing the workspace of the robot).

- **Workspace generation**

A Monte-Carlo method can be used to depict domains (X_1) and (X_2), images of aspects (A_1) and (A_2) respectively. The behaviour of the endpoint *P* of the manipulator is in the domain $(X) = (X_1) \cup (X_2)$, see the following figure.



Three cases are to be considered to reach a point *P* when the use of the inverse geometric model is required (due to constrained planned trajectory):

- point $P \in (X) - (X_2)$ is reachable through the posture *elbow-down*, the inverse geometric model has one solution in (A_1).
- point $P \in (X) - (X_1)$ is reachable through the posture *elbow-up*, the inverse geometric model has one solution in (A_2).
- point $P \in (X_1) \cap (X_2)$ is reachable through two postures: *elbow-down* or *elbow-up*, the inverse geometric model has two solutions: one in (A_1), the other in (A_2).

- **Realization of trajectories between two points *I* and *F***

Let us define points $I_1 \in (X) - (X_1)$, $F_1 \in (X) - (X_2)$, I_2 and $F_2 \in (X_1) \cap (X_2)$. Let us consider the trajectories:

- I_1F_1, it is not possible to move from I_1 to F_1 without reconfiguration. The trajectory necessarily passes through a point of the common border, it is in particular impossible to achieve a straight trajectory between the points I_1 et F_1.
- I_2F_2, a straight trajectory between the points I_2 et F_2 is possible by using the *elbow-down* posture or the *elbow-up* one.
- I_2F_1, a straight trajectory between the points I_2 et F_1 is possible if the initial posture (at point I_2) is *elbow-down*. Otherwise, a reconfiguration is required.

4.10 Singular configuration

Let us consider the simple case where the *Jacobian* matrix J is square. *Singular configurations*, or more simply *singularities*, of the mechanism are the configurations for which $\det(J(q)) = 0$. These singularities occur when there is no unique solution during the use of the inverse geometric model to transform Cartesian coordinates (expressed in task space) into joint variables (expressed in joint space).

These singularities are important to detect for the following reasons:

- Singularities represent configurations in which mobility of the mechanism is reduced, in the sense that it is not possible to impose an arbitrary end-effector motion;
- When the structure is at a singularity, infinite solutions to the inverse geometric model problem may exist;
- In the neighbourhood of a singularity, small velocities in the task space may cause large velocities in the joint space. Indeed, we deduce from the expression of the inverse kinematic model: $\dot{q} = J^{-1}(q)\,\dot{X}$ that the joint velocities tend towards infinity for any (non-null) value $\dot{X}$ when $\det(J(q))$ tends towards 0.

Two types of singularities exist:

- *Boundary singularities* that is when the mechanism is out-stretched or retracted. These singularities can be avoided on condition that the mechanism is not driven to the boundaries of its *reachable workspace*;
- *Internal singularities* that occur inside the reachable workspace. They are generally caused by the alignment of two or more joint axes, or else by the attainment of particular end-effector configuration. Unlike the above, these singularities are more problematic to take into account (they are less predictable) because they can occur anywhere in the workspace for a planned path in the task space.

Let us consider the standard case of anthropomorphic robots (a 3R shoulder with a spherical wrist), e.g., robots Stäubli RX 90, Kuka KR 3, Fanuc ARC. Like the inversion problem of direct geometric models that can be decomposed into two sub-problems, the computation of the singularities of these robots can be split into two parts:

- the computation of *shoulder singularities* from the motion of the first three joints;
- the computation of *wrist singularities* from the motion of the wrist joints (that is the last three joints).

Indeed, for such robots, the *Jacobian* matrix can be partitioned into four blocks of dimension (3×3) as follows:

$$J = \begin{pmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{pmatrix},$$

with:

- $J_{11} = (z_1 \times O_1P \quad z_2 \times O_2P \quad z_3 \times O_3P)$ where point P corresponds to points O_4, O_5, O_6 (located at the intersection of the last three axes), see Remark at the end of §4.7.2;
- $J_{12} = (0 \quad 0 \quad 0)$ knowing that $J_{12} = (z_4 \times O_4O_6 \quad z_5 \times O_5O_6 \quad z_6 \times O_6O_6)$ with $O_4 = O_5 = O_6$;
- $J_{22} = (z_4 \quad z_5 \quad z_6)$.

It follows that:

$$\det(J) = \det(J_{11})\ \det(J_{22}),$$

which allows the decoupling of the computation of singularities in two parts:

- the condition $\det(J_{11}) = 0$ to determine the *shoulder singularities*;

- the condition $\det(J_{22}) = 0$ to determine the *wrist singularities*.

Let us consider for example the robot Stäubli RX90 described in §4.5 in particular the figure of its initial posture.

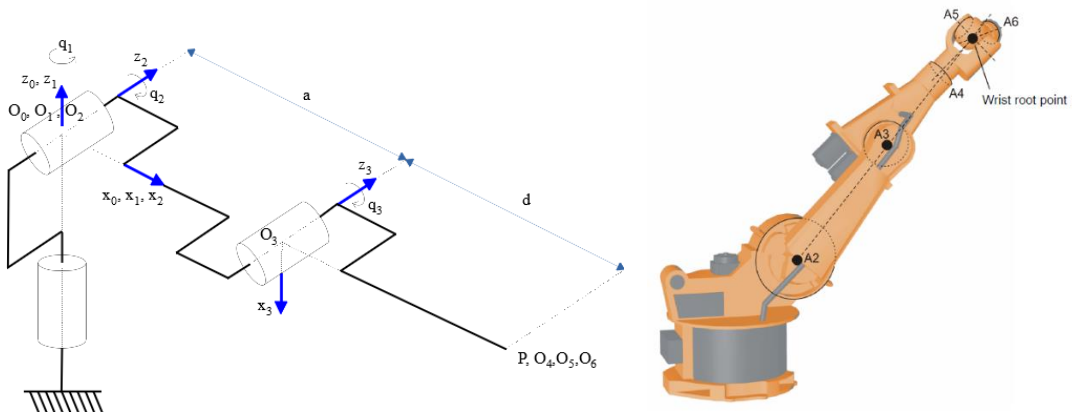
1) *Shoulder singularities*

We have: $\det(J_{11}) = -ad\ \cos(q_3)\ (a\cos(q_2) + d\sin(q_2 + q_3))$.

The determinant is independent of joint q_1 and is zeroed when:

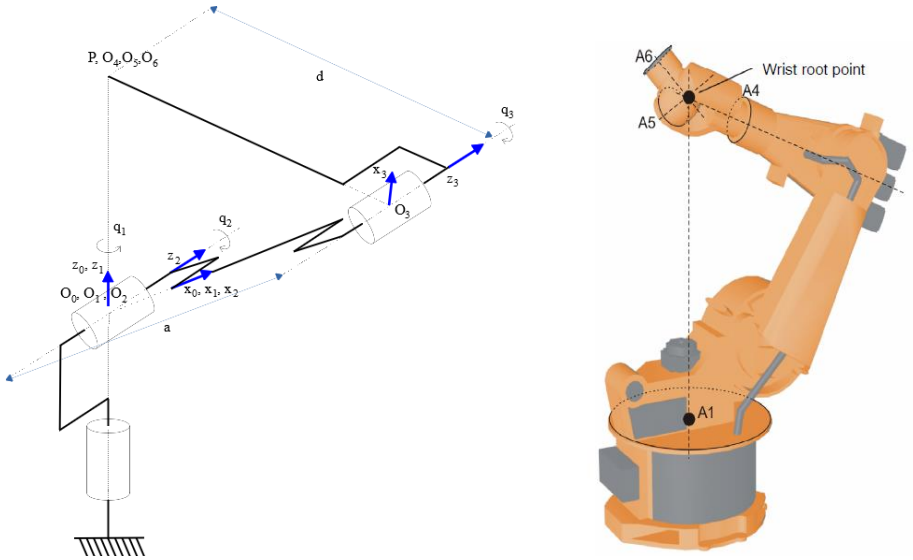
$$\cos(q_3) = 0 \text{ and/or } a\cos(q_2) + d\ \sin(q_2 + q_3) = 0.$$

Thus, a first singularity occurs when $\cos(q_3) = 0$, *i.e.*, when $q_3 = \frac{\pi}{2}$ or $q_3 = -\frac{\pi}{2}$, the point P belongs to the straight line O_2O_3 which corresponds respectively to the case when the shoulder is *out-stretched* (at the boundary of its workspace, see the following figure) or *retracted* (which is not a possible posture in practice).



Note that this singularity is similar to the one of the 2R planar robot.

A second singularity, sometimes called *overhead singularity*, occurs when $a\cos(q_2) + d\ \sin(q_2 + q_3) = 0$, *i.e.*, when the point P belongs to $\overrightarrow{o_0z_0}(= \overrightarrow{o_1z_1})$ axis, see the following figure.



Let us note that the $\overrightarrow{o_1z_1}$ axis represents a *continuum* of singular configurations.

A rotation about the $\overrightarrow{o_1 z_1}$ axis does not modify the point P coordinates which means that the angle value of q_1 can take any value. Thus, the inverse geometric model proposes an infinite number of solutions for angle q_1. By example, the controller of a robot Kuka KR 3 offers two options (chosen through a Kuka variable) when the end point of a motion (defined in task space) belongs to the $\overrightarrow{o_1 z_1}$ axis: axis 1 is moved to 0° during the motion or the angle value for axis 1 is the same for both the start point and the end point.

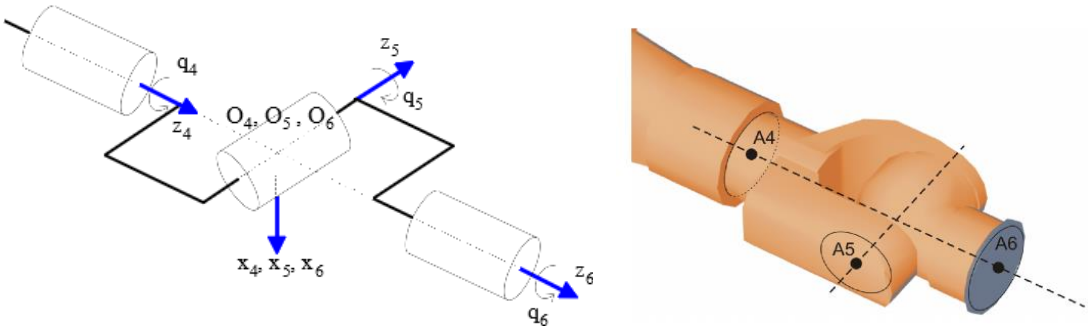
Unlike the wrist singularities described below, the shoulder singularities are well identified in task space, allowing them to be taken into account when programming trajectories.

Exercise: Provide the reasoning leading to the expression of $\det(J_{11})$ and check that the point P is effectively such that $P_x = P_y = 0$ when $a\cos(q_2) + d\sin(q_2 + q_3) = 0$.

2) *Wrist singularities*

We have: $\det(J_{22}) = \det(z_4 \quad z_5 \quad z_6)$.

The determinant is zeroed when vectors z_4, z_5, z_6 are linearly dependent. In other words, a singularity occurs when vectors z_4 and z_6 are aligned, *i.e.*, when angle $q_5 = 0$ (see the following figure) or when $q_5 = \pi$ (which is not a possible posture in practice).



The presence of the singularity is such that the rotations of equal magnitude but of opposite directions on joints q_4 and q_6 do not modify the end-effector rotation (in other words, the sum of these two angles remains constant). It follows that the value of axes 4 and 6 can be arbitrary and thus cannot be determined by the inverse geometric model. By example, the controller of a robot Kuka KR 3 offers two options (chosen through a Kuka variable) when the end point of a motion (defined in task space) is close to this singularity ($-0{,}01812° \leq q_5 \leq 0{,}01812°$): axis 4 is moved to 0° during the motion or the angle value for axis 4 is the same for both the start point and the end point.

Such a singularity can occur anywhere inside the reachable workspace of the robot but is easily detected in the joint space which enables to avoid it when programming trajectories.

5 TRAJECTORY GENERATION

A motion is specified by defining a *path* along which the robot must move. A *path* is a sequence of *points* (which can be more or less distant) defined either in joint space (through their joint coordinates) or in task space (through their end-effector location coordinates).

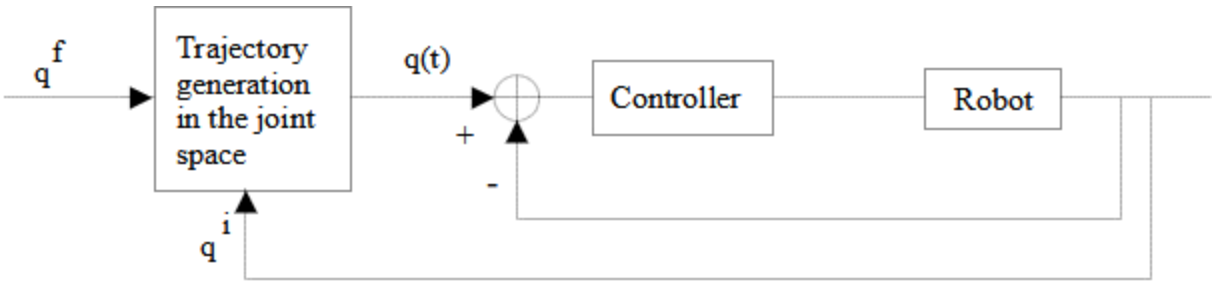
These *points* can be obtained by learning or be extracted from a database of a Computer Aided Design (CAD) system.

The issue of *trajectory generation* is to compute for the controller (in charge to implement the position/speed/acceleration control of the joints $q(t)$) the planned reference joint or end-effector variables

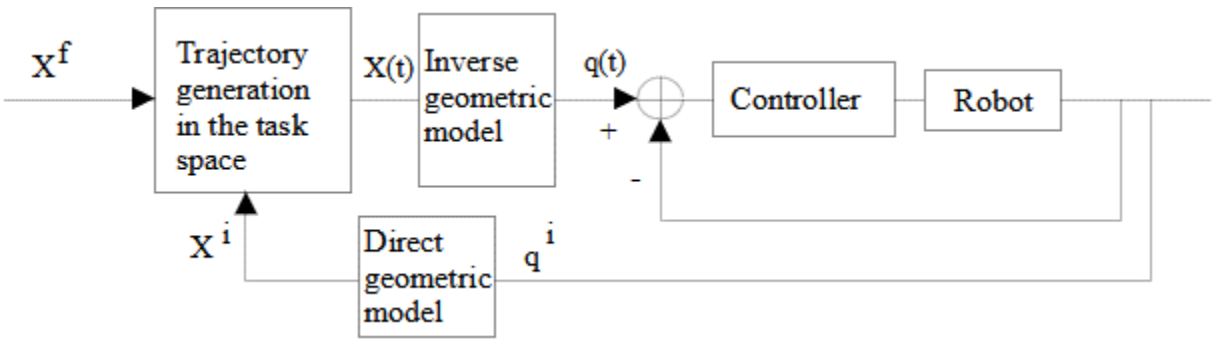
as functions of time such that the robot tracks the *planned path*. Thus, a *trajectory* is characterized by a *path* and a *time history* along the path.

The trajectories of a robot can be classified as follows:
- trajectory between 2 points with free motion between them,
- trajectory between 2 points *via* a sequence of planned intermediate points (to avoid obstacles) with free motion between the intermediate points,
- constrained trajectory between 2 points (straight lines segment for example),
- trajectory between 2 points *via* a sequence of planned intermediate points, the trajectories being constrained between the intermediate points.

In the first two cases, the trajectory is generally directly generated in the *joint space* which leads to compute a sequence of joint vectors $q(t)$ (corresponding to the controller inputs) from the initial one q^i and the final one q^f, see the following figure.



In the last two cases, the trajectory is generated in the *task space* which leads to a sequence of homogeneous transformation matrices $X(t)$ from the initial one X^i and the final one X^f. Let us note that X^i is computed from the initial joint vector q^i by using the direct geometric model. Then the sequence $X(t)$ is transformed into a sequence of joint vectors $q(t)$ by using the inverse geometric model, see the following figure.



The choice of either the joint space or the task space for the trajectory generation depends on the context in which the trajectory is realized knowing that each approach has its advantages and drawbacks.

➢ Trajectory generation in the *joint space* presents several advantages:
- motion is minimal at each joint,
- it requires less on-line computation (inverse and direct geometric model are not required),
- motion is not affected by crossing singular configurations,
- joint limits, maximum velocities and torques (determined from physical limits of actuators) are directly usable.

The drawback is that the end-effector trajectory (in the task space) cannot be imposed (although it is repetitive) which can cause collisions when the robot moves in a cluttered environment.

In conclusion, the trajectory generation in the *joint space* is appropriate to achieve fast motions in a free space.

- Trajectory generation in the *task space* enables to control all along the time the behaviour of the trajectory (for example by imposing a straight motion of the end-effector). But it has some drawbacks:
  - it may fail when the computed trajectory crosses a singular configuration (due to the use of inverse geometric model to transform task coordinates in joint coordinates),
  - it fails when the generated points are out of the joint limits or when the robot is forced to reconfigure itself (due to a change of its current aspect),
  - velocities and accelerations of the task coordinates depend on the robot configuration. Lower bounds are generally used such that joint velocity and torque limits are satisfied. Consequently, the robot may work under its nominal performance.

In the following we briefly deal with the trajectory generation in the *joint space*. See §13.4, §13.5, *Modeling, Identification & Control of Robots, W. Khalil, E. Dombre, Hermes Penton Science 2002* for the trajectory generation in the *task space*.

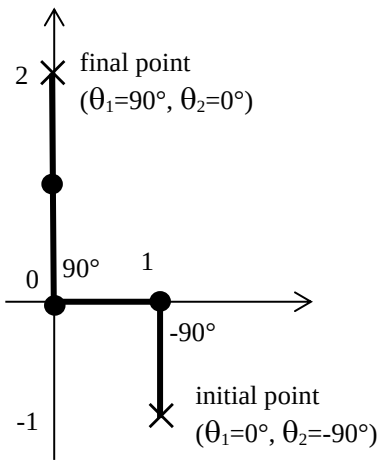
5.1 Trajectory between 2 points in the joint space

The trajectory generation between 2 points in the *joint space* is for example used through the motion instructions **MOVE** for Stäubli, **JOINT** for FANUC or **PTP** for Kuka.
This trajectory generation uses an interpolation method to compute the intermediate points located between the *initial point* (corresponding to the joint values just before the execution of the motion instruction) and the *final point* (information associated with the motion instruction corresponding to the joint values to be reached by the manipulator) knowing that the robot takes a time equal to t_f to reach the final point.

Let us consider for example the 2R planar robot described in the last exercise of §4.1 with $l_1 = l_2 = 1$ and no joint limits, its direct geometric model is:

$$\begin{cases} x = \cos(\theta_1) + \cos(\theta_1 + \theta_2) \\ y = \sin(\theta_1) + \sin(\theta_1 + \theta_2) \end{cases},$$

with the motion described below and a value t_f equal to 10 *sec*:

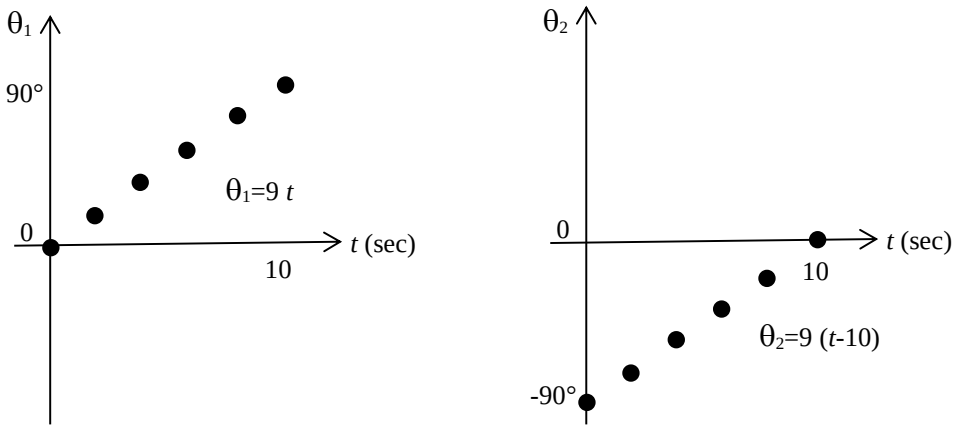


Let us compute for each joint variable θ_j $(j = 1, 2)$ the trajectory (all along the time) between its initial value ($\theta_j^{initial}$) and the final value (θ_j^{final}). We use for this a *linear interpolation* (*via* a 1-order polynomial) such that: $\theta_j(0) = \theta_j^{initial}$, $\theta_j(t_f) = \theta_j^{final}$ that is:

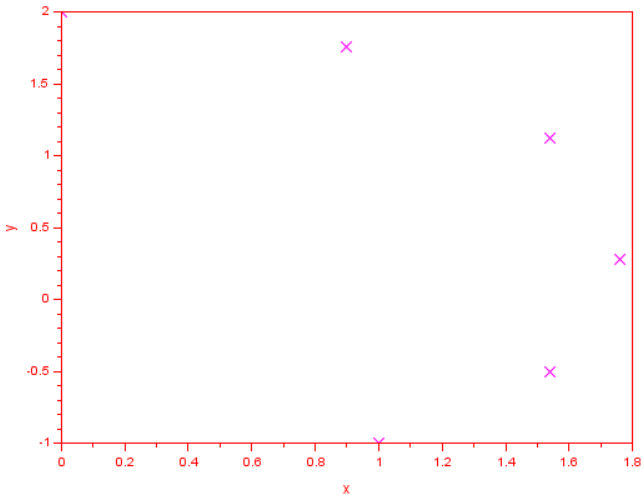
$$\theta_j(t) = \theta_j^{initial} + \frac{\theta_j^{final} - \theta_j^{initial}}{t_f} \times t,$$

i.e., $\theta_1(t) = 9t$ and $\theta_2(t) = 9(t - 10)$.

To simplify only 4 intermediate points are represented in the following figure, *i.e.*, 6 points in all with the initial and final points:



The following figure represents the 6 corresponding points in the task space:



The position is **continuous** but not the velocity at time 0 and t_f:

$$\dot{\theta}_j(t) = \frac{\theta_j^{final} - \theta_j^{initial}}{t_f} \quad \text{for } 0 \leq t \leq t_f \quad \text{and} \quad \dot{\theta}_j(0\_) = \dot{\theta}_j\left(t_f^+\right) = 0,$$

which can cause jolts[23] on the manipulator.

To ensure the **continuity of position and velocity** with initial and final velocities equal to zero (for example), the interpolation must be done from a polynomial with an order *at least* equal to 3, that is:

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3.$$

It follows a parabolic velocity profile ($\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2$) and a linear acceleration profile ($\ddot{\theta}(t) = 2a_2 + 6a_3 t$).

Exercise: Compute the coefficients of the previous 3-order polynomial such that:

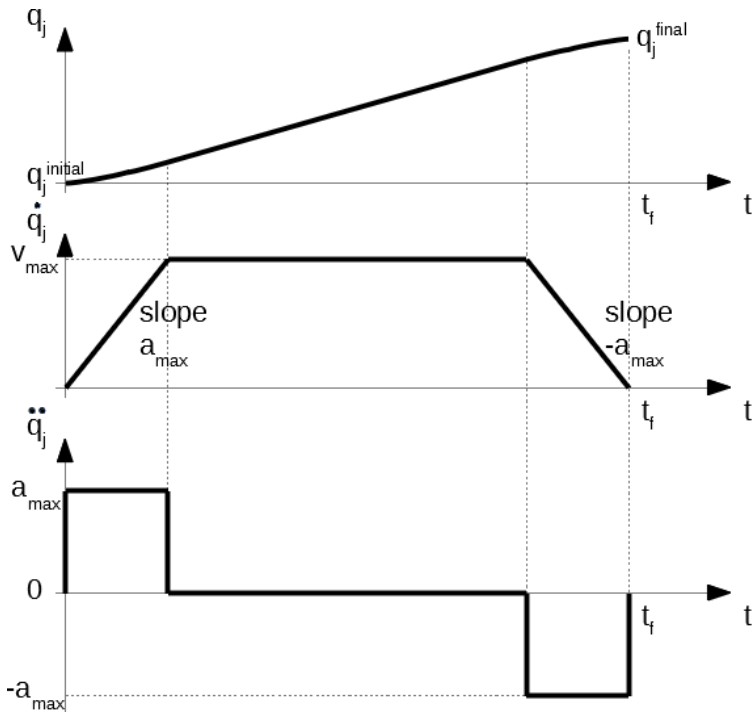
$$\theta(0) = \theta^{initial}, \theta\left(t_f\right) = \theta^{final}, \dot{\theta}(0) = \dot{\theta}\left(t_f\right) = 0.$$

[23] 'à-coups'.

The use of a polynomial of order *at least* equal to 5 is necessary to also ensure a **continuity of acceleration** with, for example, initial and final accelerations equal to zero.

Generally, the duration t_f of a trajectory is not specified. The goal is to minimize the time to travel from the initial configuration q^i to the final one q^f while satisfying velocity and acceleration constraints. The approach is to compute the minimum time separately for each joint, and then to synchronize all the joints at a common time. The interpolation of joint variables frequently adopted in industrial practise is based on a *trapezoidal velocity profile.*

Let us apply this interpolation method on joint q_j. The minimum traveling time t_f of joint q_j occurs if either the velocity ($\dot{q}_j$) or the acceleration ($\ddot{q}_j$) is saturated during the trajectory. Then the acceleration profile consists of a maximum constant acceleration, noted a_{max}, in the start phase, followed by a cruise velocity during which velocity is maximum (noted v_{max}) and acceleration is null, ended by a maximum constant deceleration ($-a_{max}$) in the arrival phase. The initial and final velocities are null. See the resulting trapezoidal velocity profile in the following figure.



Remark: The previous method can be improved in order to have a continuous trajectory $\ddot{q}_j$ by replacing the acceleration and deceleration phases either by a trapezoidal profile or by a second order polynomial.

Generally, the trajectories are synchronized when there are several joints. Indeed, all the joints are moved at the same time with different velocities in order to arrive at the same time at the final destination. The controller computes the cruise velocity v_{max} of each axis knowing that (at least) one axis moves at its maximum velocity. Adjusting all the axes to the slowest axis reduces the solicitation level on the actuators, without reducing the travel time of the end-effector from one point to the next one. Some methods provide equal times during the acceleration and deceleration phases for all joints to enable the synthesis of more precise control laws.

5.2 Trajectory between several points in the joint space

As described previously, the execution of *only one* motion instruction (as **MOVE**) produces a *point to point* trajectory, that is, a trajectory from the current point (corresponding to joint values before the execution of the motion instruction) to the destination point (indicated in the motion instruction) through an acceleration phase until a planned cruising speed is reached and then, in the proximity of the destination point, a deceleration phase before stopping at destination point.

When a *sequence* of instructions **MOVE** between several points (**loc_1**, **loc_2**, …, **loc_n**) are executed, the resulting trajectory starts from the current point to the final one **loc_n** (the point indicated in the last instruction **MOVE**) by smoothly passing near intermediate points **loc_1**, …, **loc_n-1** without stopping to reduce the traveling time. By example, the execution the following sequence of 2 motion instructions:

MOVE loc_1

MOVE loc_2

leads to a trajectory from the current point toward the point **loc_1** through an acceleration phase until a planned cruising speed is reached. However, the manipulator does not decelerate near this point (for a stop) but turns around near the point **loc_1** to move towards the point **loc_2**. The robot ends its trajectory with a deceleration phase near the destination point **loc_2**, before a stop at this point.

Such a trajectory smoothly made through the passage near several points, without stopping at these points (except the last point) is sometimes called a *continuous-path operation*.

Exercise: Do the exercises of Tutorial 3.