Multibody simulation

Multibody systems (position)

Dimitar Dimitrov, Henrik Andreasson

Örebro University

September 15, 2016

Main points covered

- multibody systems
- constraints and degrees of freedom (DoF)
- forward geometric model
- inverse geometric model

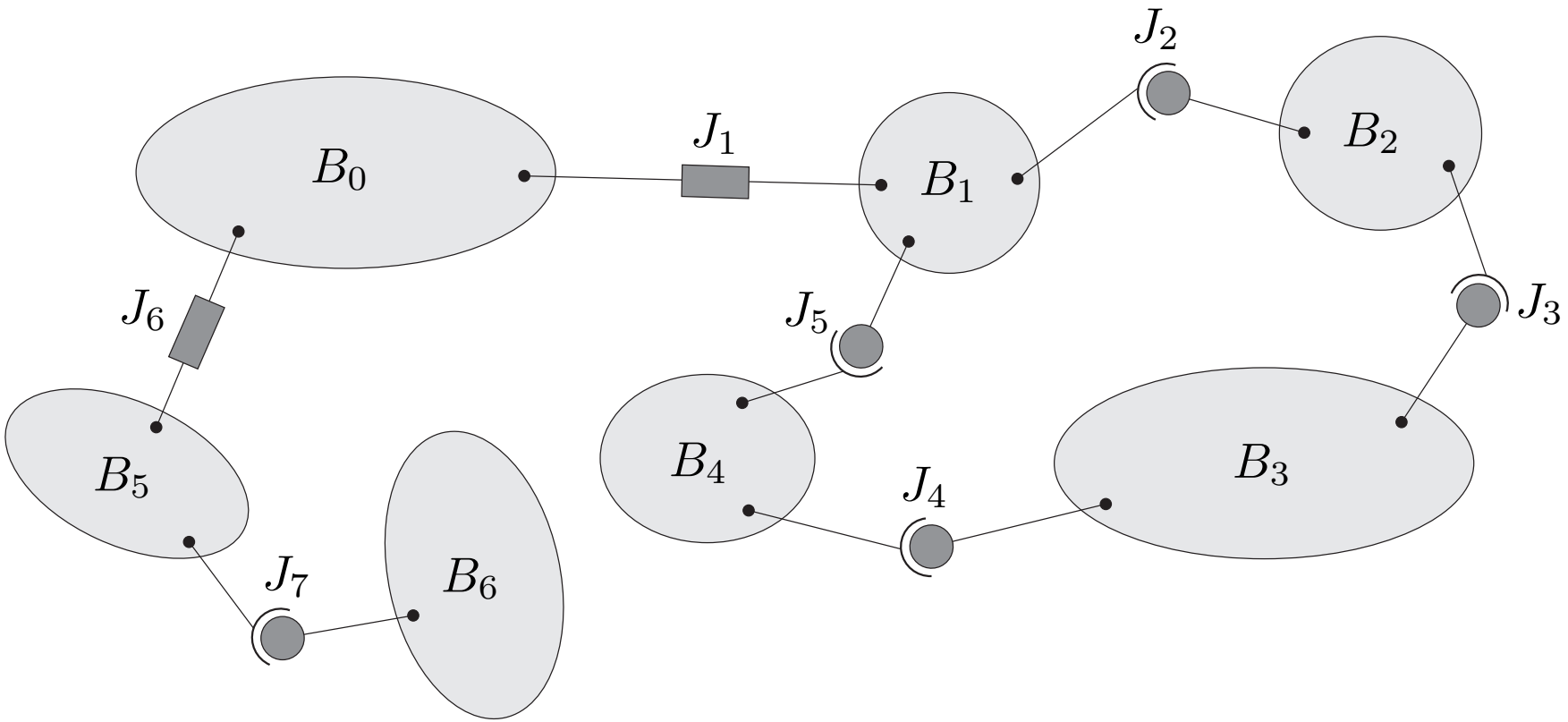
Particle $\rightarrow$ rigid body $\rightarrow$ multibody system

- first, we started by analyzing the motion of a **particle** (mass point)
- next, we considered a **rigid body** as a collection of particles
- next, we will discuss systems whose *building blocks* are rigid bodies - we call then **multibody systems**

Question to answer in this lecture

How do we specify the position of a multibody system?

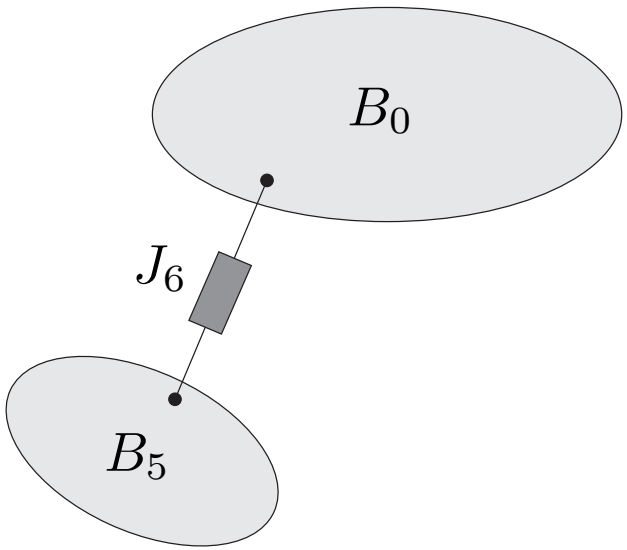
A multibody system



A multibody system Σ (B_i and J_i stand for rigid body i and joint i)

The system Σ consists of seven rigid bodies, interconnected through a set of *prismatic* and *revolute* **joints** (hinges). A valid parametrization for the position (also referred to as configuration) of Σ, has to uniquely specify the posture of each component of Σ.

Constraints



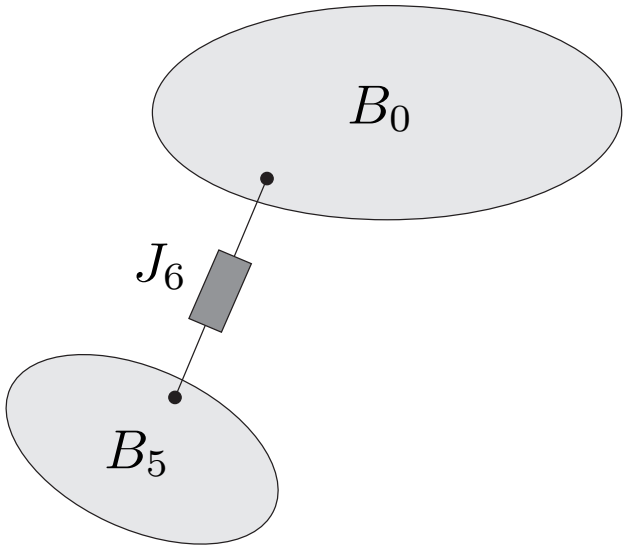
- body B_5 is connected to B_0 through a **prismatic joint** J_6
- we can think of J_6 as a constraint that limits the motion of B_5 relative to B_0 to be only along one fixed axis
- the position of B_5 relative to a frame fixed in B_0 could be specified using only one parameter

We will think of constraints as limitations (restrictions) to the motion of the rigid bodies in Σ. Constraints and rigid bodies are the *building blocks* of a multibody system.

DoF

We define the number of **independent** variables required to determine the system configuration as the degrees of freedom (**DoF**) of the system.

Example (constraints)

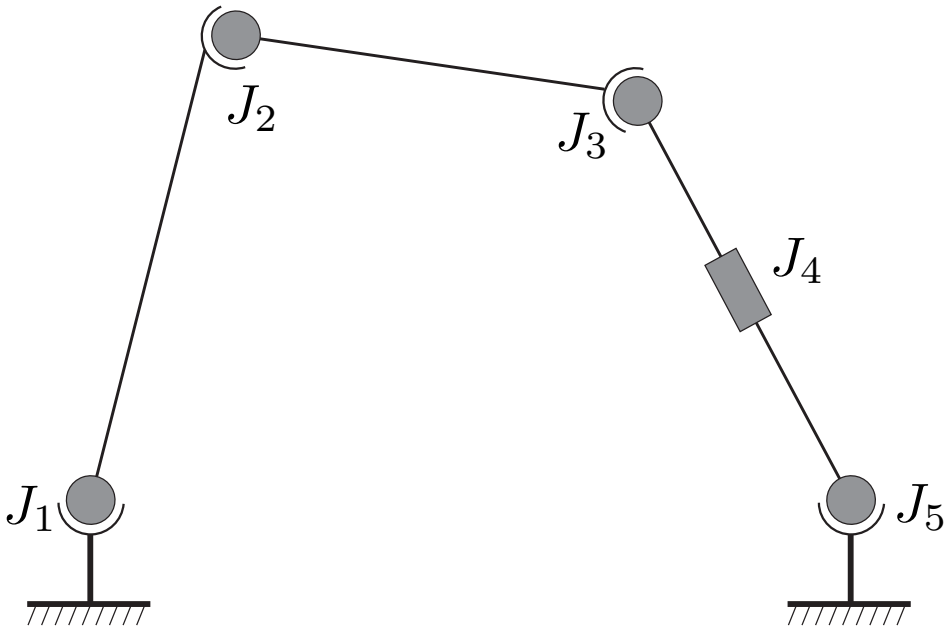


B_5 has only one DoF relative to B_0. Hence, we could say that the **prismatic joint** J_6 imposed five constraints on the motion of B_5 relative to B_0.

Rigid body - DoF

In general, we needed at least six parameters (three for linear position and three for orientation) in order to specify the posture of a rigid body with respect to a given coordinate frame. Hence, an *unconstrained* rigid body has 6 DoF.

Counting DoF



The figure depicts a planar system with

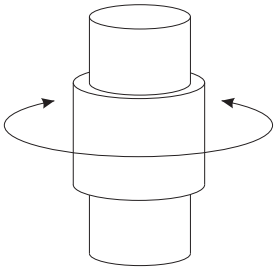
- four links
- four revolute joints
- one prismatic joint

In the plain each body has three DoF (two translations and one rotation). Each joint imposes two constraints on the relative motion of two bodies. We can compute the DoF of the system as follows

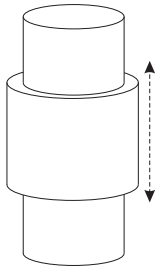
1. If there were no joints, the four bodies could move freely in the plane, hence, $4 \times 3 = 12$ DoF.
2. The motion of each link is constrained by one joint (the last link, by two). Hence, $4 \times (3 - 2) - 2 = 2$ DoF.

The above reasoning could be applied to 3D systems as well.

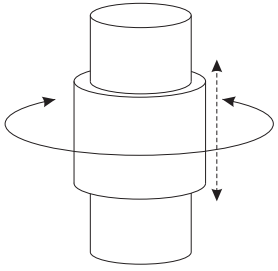
Example (different types of joints)



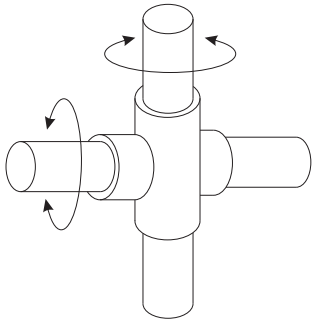
Revolute
1 DoF



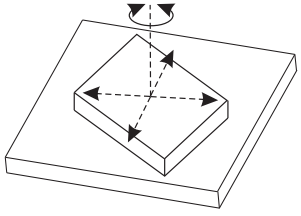
Prismatic
1 DoF



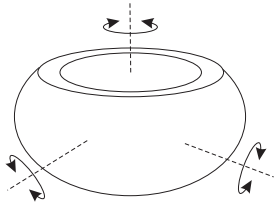
Cylindrical
2 DoF



Universal
2 DoF



Planar
3 DoF



Spherical
3 DoF

most complex joint can be modelled as a combination of revolute and prismatic joints

- a **universal joint** is a combination of two revolute joints
- a **cylindrical joint** is a combination of a revolute and a prismatic joint
- a **planar joint** is a combination of two prismatic and one revolute joint
- a **spherical joint** is a combination of three revolute joints

Configuration of a multibody system

The space of all configurations that a physical system may attain is called **configuration space** (we will denote it by Σ_c). We can think of a single configuration as a point in the configuration space.

Alternative definition of DoF

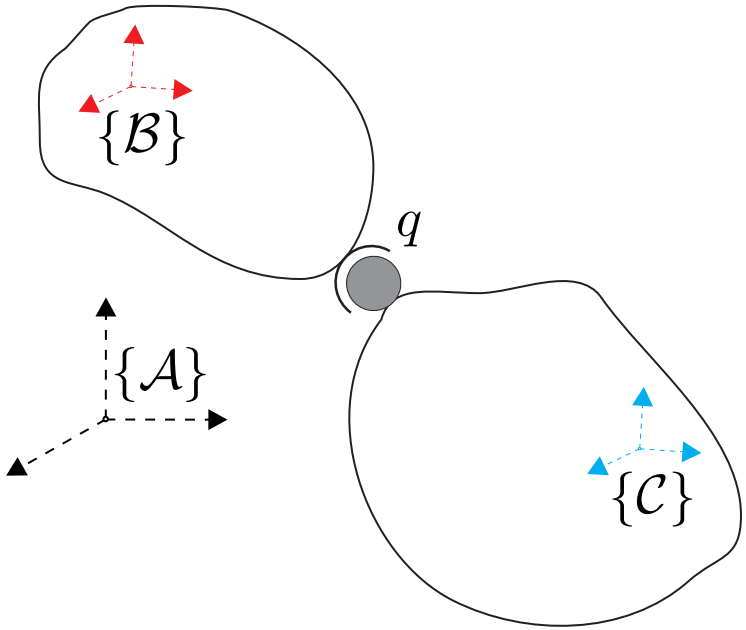
The DoF of a system is equal to the dimension of its configuration space.

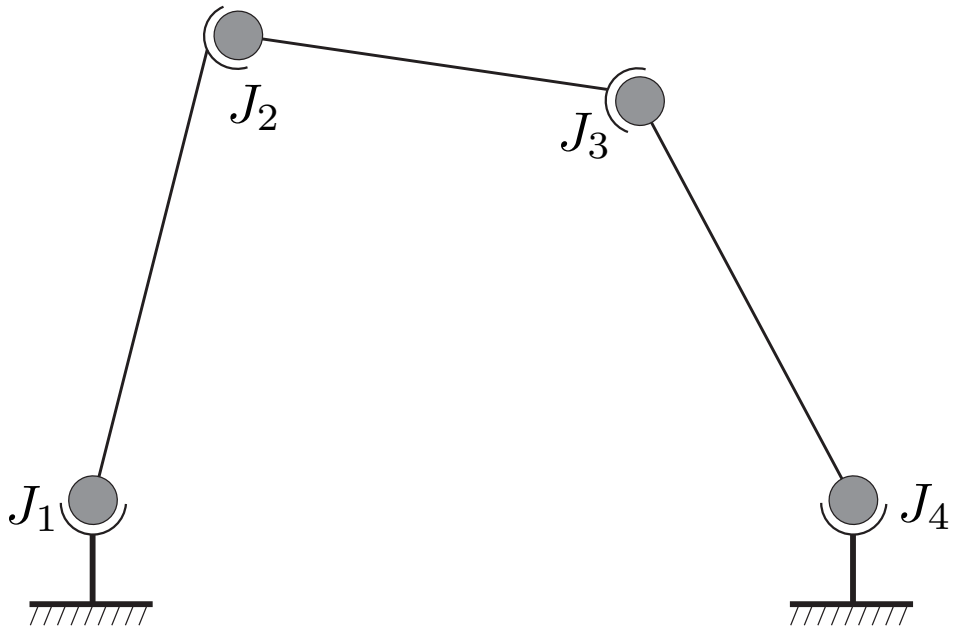
Two common ways to parametrize Σ_c are

1. Treat each rigid body as a separate system, and "make sure" that the constraints imposed by the joints are satisfied. For a system of n_b rigid bodies we can use n_b pairs $(\boldsymbol{r},\boldsymbol{R})$ and m_c constraints.

2. Exploit the structure of the multibody system, and define n independent **generalized positions** (where $n \leq n_b$ is equal to the DoF of the system). In such a way we obtain a parametrization for Σ_c with minimal number of variables and do not need to explicitly deal with constraints.

Exercise



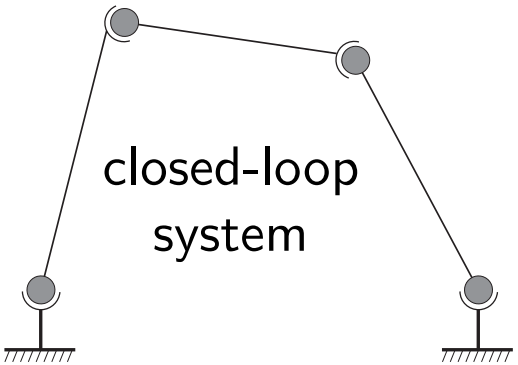


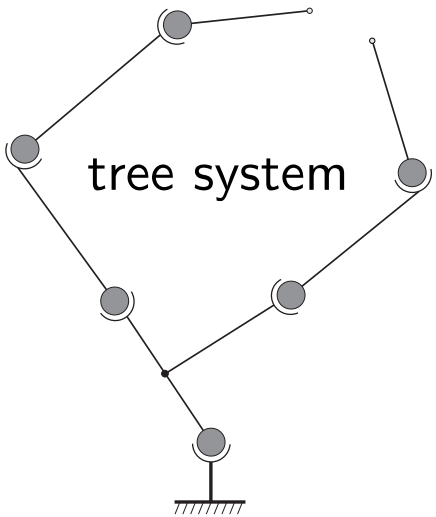
Problems

The system on the left is in 3D, while the system on the right is in 2D

- find the number of DoF for each system
- find a parametrization of Σ_c for each system

Open-loop $\Leftrightarrow$ closed-loop systems





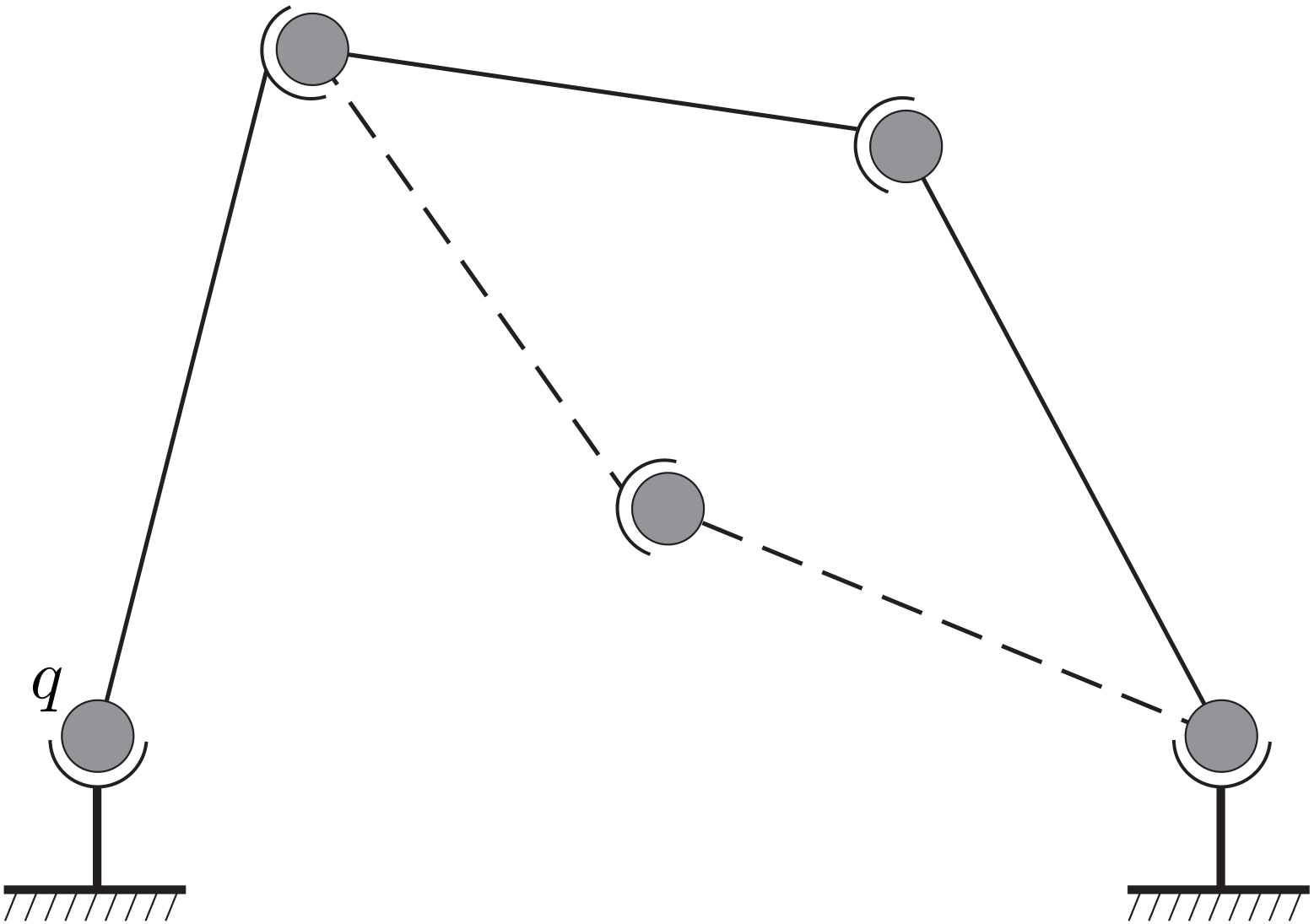
In this course we will work with serial manipulators (also called tree systems, or open-loop chains).

The configuration of a serial manipulator (with a fixed base) can be described uniquely using the manipulator joint angles q_i, $i = 1, \ldots, n$. All joint angles could be set **independently** to generate a unique point in the configuration space Σ_c.

The configuration of closed-loop systems could also be described using the joints angles, however, we can **not** set all of them independently.

When a closed-loop system has p DoF, only p joint angles could be set independently. The remaining joint angles are determined by the loop closure constraints (and some *additional considerations*).

. . . additional considerations



Configuration space $\Leftrightarrow$ task space

- It is often the case that we are more interested in the motion of one particular part of a manipulator rather than the motion of the whole structure.
- Usually there is an attached tool and we want to control its position, velocity, force, etc., in order to complete a given task.
- In many cases, specifying the position and/or orientation of the tool is of great importance. We can think of this as imposing m additional (task related) constraints to our n joint manipulator.

We will call the space of m task related constraints, **task space**, and we will denote it by Σ_t. In general, $m \leq n$ will be assumed.

Example (configuration space and task space)

Configuration space (Σ_c)

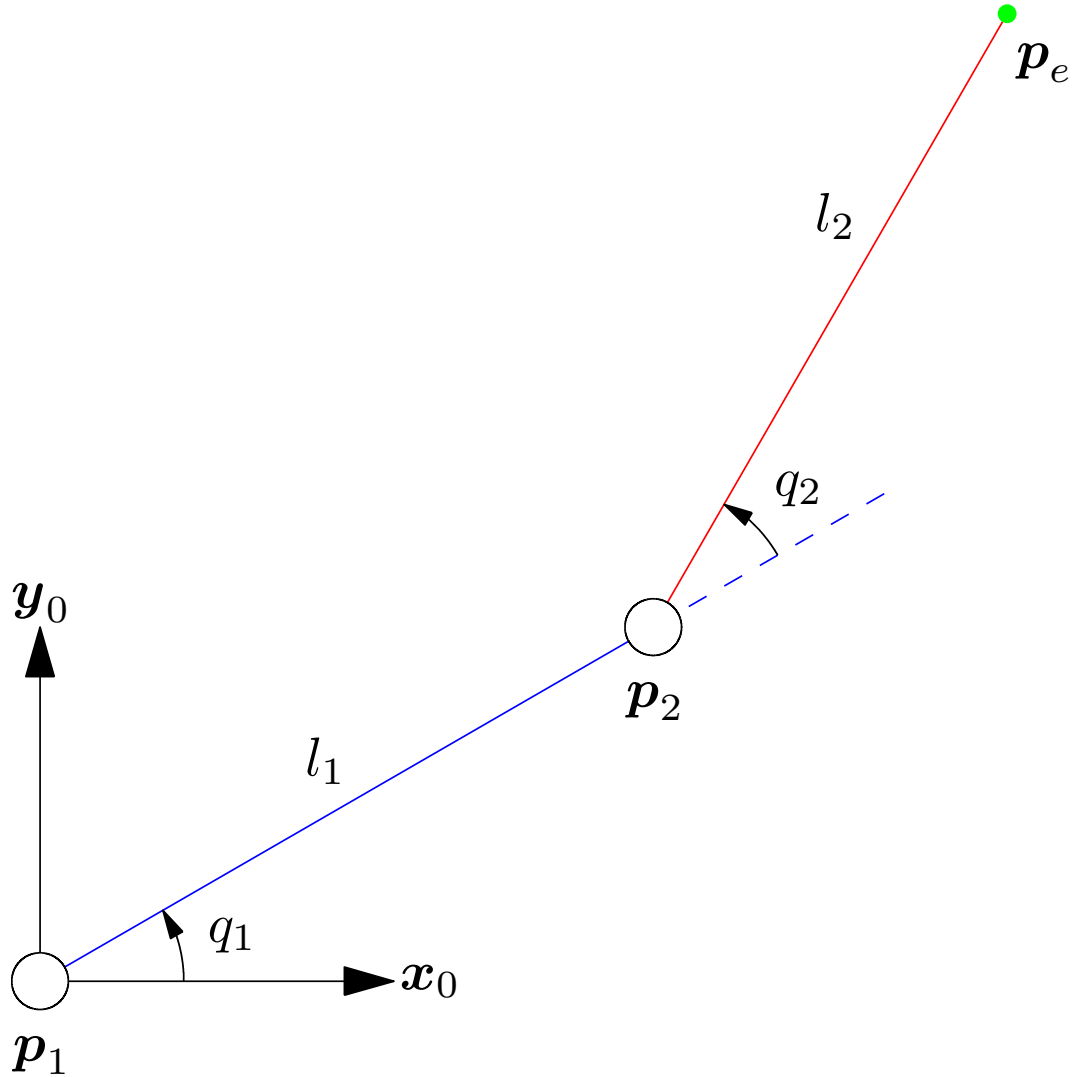
the span of vectors

$$\begin{bmatrix} q_1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ q_2 \end{bmatrix}$$

Task space (Σ_t)

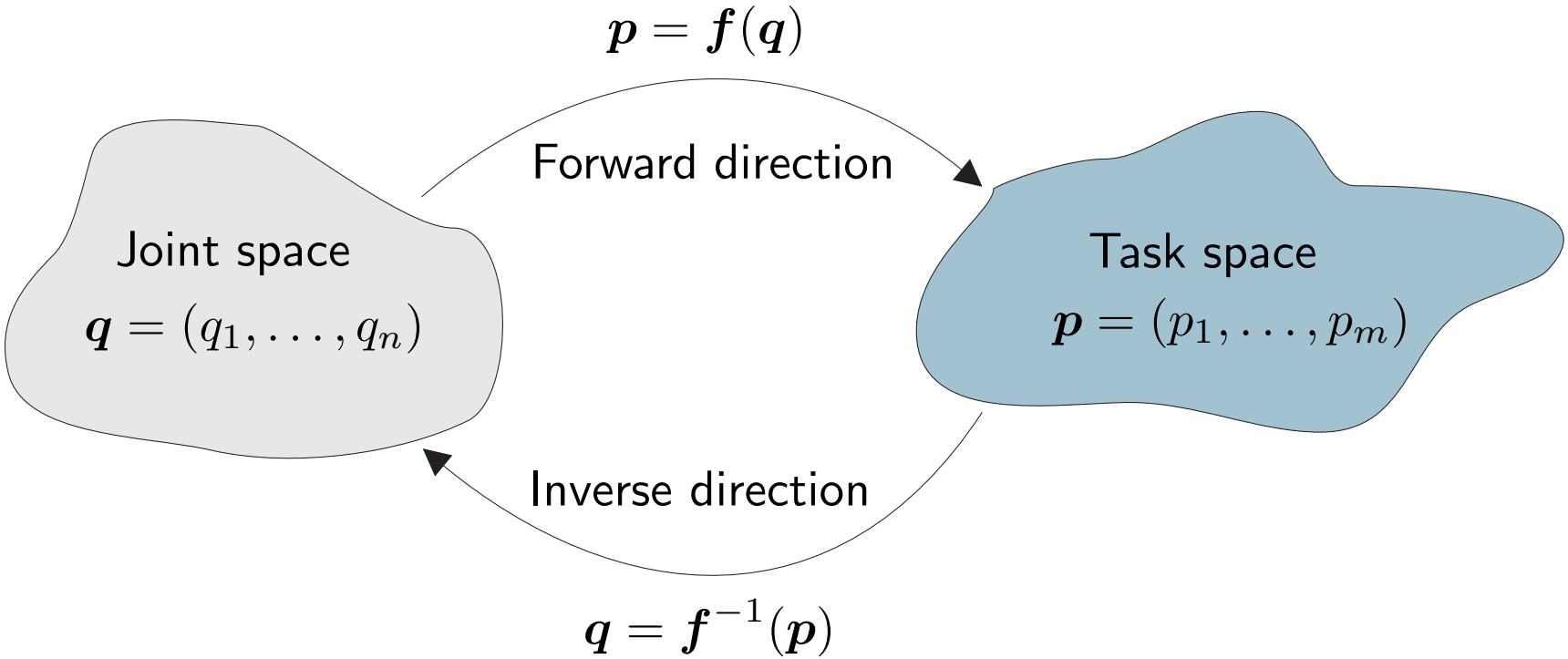
the span of vectors $\boldsymbol{x}_0, \boldsymbol{y}_0$

in this example both Σ_c and Σ_t are 2D ($m = n$)



In general, we are interested in what is the correspondence (the map) between (q_1, q_2) and the position of the end-effector in $\{\boldsymbol{x}_0, \boldsymbol{y}_0\}$.

Forward $\Leftrightarrow$ inverse geometric models



- Given a configuration $\boldsymbol{q} = (q_1, \ldots, q_n)$ of a manipulator system, find the corresponding position and orientation of its end-effector in the $\{\boldsymbol{x}_0, \boldsymbol{y}_0, \boldsymbol{z}_0\}$ frame. This is called (evaluation of the) **forward geometric model** (FGM) or **forward kinematics**.
- Given the position and orientation of the end-effector, find a corresponding $\boldsymbol{q}$. This is called (evaluation of the) **inverse geometric model** (IGM) or **inverse kinematics**.

Forward geometric model (FGM)

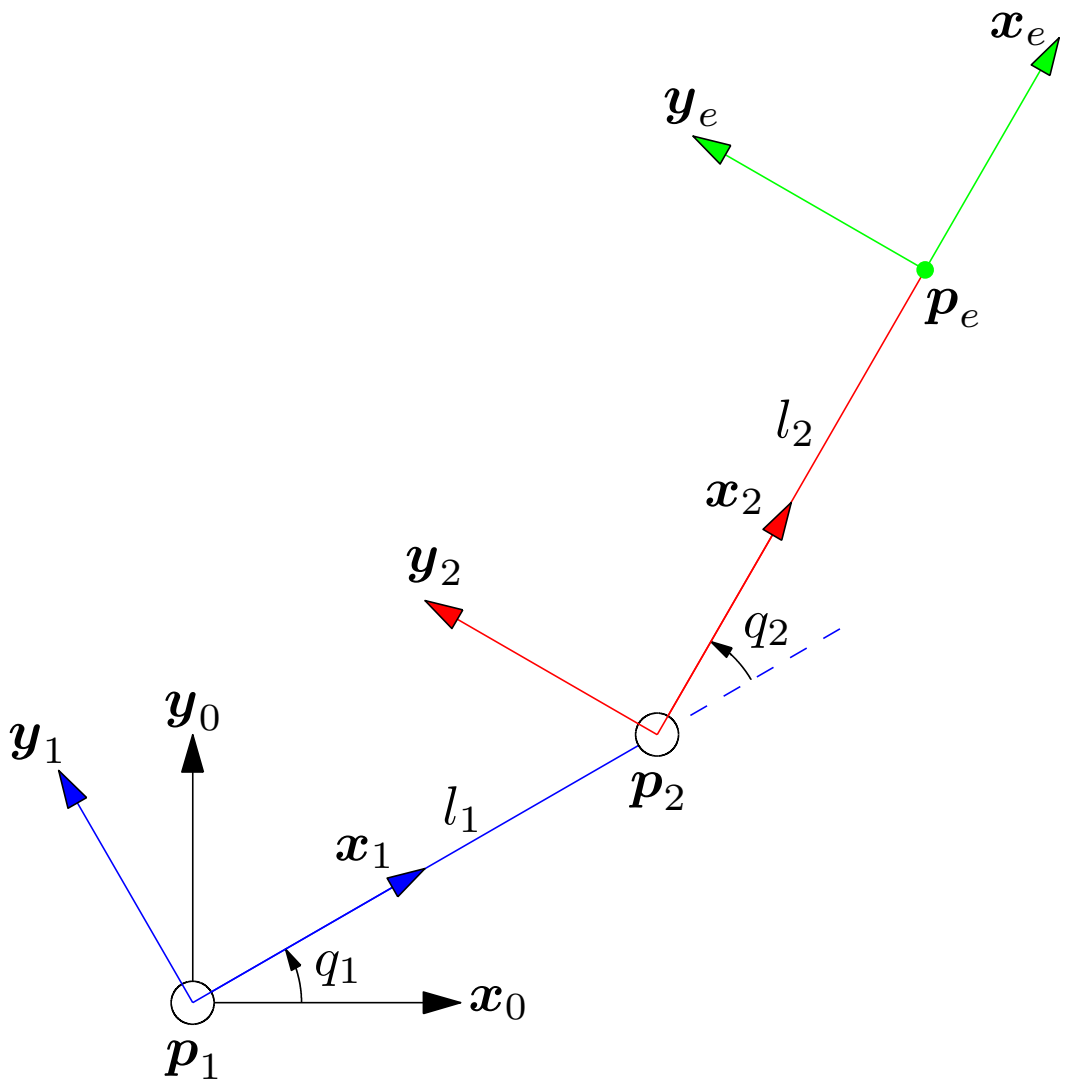
Problem

Find ${}^0\boldsymbol{p}_e$ and ${}^0_e\boldsymbol{R} \equiv {}^0_2\boldsymbol{R}$

Solution

- ${}^0\boldsymbol{p}_1 = (0, 0, 0)$
- ${}^1\boldsymbol{p}_2 = (l_1, 0, 0)$
- ${}^2\boldsymbol{p}_e = (l_2, 0, 0)$
- ${}^0_1\boldsymbol{R} = \boldsymbol{R}_z(q_1)$
- ${}^1_2\boldsymbol{R} = \boldsymbol{R}_z(q_2)$
- ${}^2_e\boldsymbol{R} = \boldsymbol{R}_z(0)$

$${}^0\boldsymbol{p}_e = {}^0_1\boldsymbol{R}\left({}^1\boldsymbol{p}_2 + {}^1_2\boldsymbol{R}\,{}^2\boldsymbol{p}_e\right)$$
$${}^0_2\boldsymbol{R} = {}^0_1\boldsymbol{R}\,{}^1_2\boldsymbol{R}$$



The FGM has a unique solution and, in general, it is easy to compute.

FGM using homogeneous matrices

Position and orientation separately

- ${}^0\boldsymbol{p}_1 = (0,0,0)$
- ${}^1\boldsymbol{p}_2 = (l_1,0,0)$
- ${}^2\boldsymbol{p}_e = (l_2,0,0)$
- ${}^0_1\boldsymbol{R} = \boldsymbol{R}_z(q_1)$
- ${}^1_2\boldsymbol{R} = \boldsymbol{R}_z(q_2)$
- ${}^2_e\boldsymbol{R} = \boldsymbol{R}_z(0)$

$$
\begin{aligned}
{}^0\boldsymbol{p}_e &= {}^0\boldsymbol{p}_1 + {}^0_1\boldsymbol{R}\,{}^1\boldsymbol{p}_2 + {}^0_1\boldsymbol{R}\,{}^1_2\boldsymbol{R}\,{}^2\boldsymbol{p}_e \\
&= \begin{bmatrix} l_1\cos q_1 + l_2\cos(q_1+q_2) \\ l_1\sin q_1 + l_2\sin(q_1+q_2) \end{bmatrix}
\end{aligned}
$$

Using homogeneous matrices

$$
{}^0_1\boldsymbol{T} = \begin{bmatrix} {}^0_1\boldsymbol{R} & {}^0\boldsymbol{p}_1 \\ \boldsymbol{0}^T & 1 \end{bmatrix}
$$

$$
{}^1_2\boldsymbol{T} = \begin{bmatrix} {}^1_2\boldsymbol{R} & {}^1\boldsymbol{p}_2 \\ \boldsymbol{0}^T & 1 \end{bmatrix}
$$

$$
{}^2_e\boldsymbol{T} = \begin{bmatrix} {}^2_e\boldsymbol{R} & {}^2\boldsymbol{p}_e \\ \boldsymbol{0}^T & 1 \end{bmatrix}
$$

$$
{}^0_e\boldsymbol{T} = {}^0_1\boldsymbol{T}\,{}^1_2\boldsymbol{T}\,{}^2_e\boldsymbol{T}
$$

Evaluating the FKM for a n-link manipulator amounts to forming and multiplying n 4×4 homogeneous matrices

$$
{}^0_n\boldsymbol{T} = {}^0_1\boldsymbol{T}\;{}^1_2\boldsymbol{T}\;{}^2_3\boldsymbol{T}\;\ldots\;{}^{n-1}_n\boldsymbol{T}.
$$

The product of homogeneous matrices is a homogeneous matrix.

Choosing body-fixed frames

The form of each homogeneous matrix (in the above example) depends on where (and with what orientation) are the body-fixed frames placed.

There are standard conventions, that provide a systematic procedure for performing the geometric analysis. A commonly used one is the **Denavit-Hartenberg** convention.

I will not adopt a specific convention in this course. Each of you will have to develop their own! In order to do so, among other things, you would have to

- associate a coordinate frame with each rigid body and/or joint
- assign a type to each joint ($e.g.$, prismatic, revolute, ...)
- represent the connectivity information of the system

Connectivity of serial manipulators

One possible solution

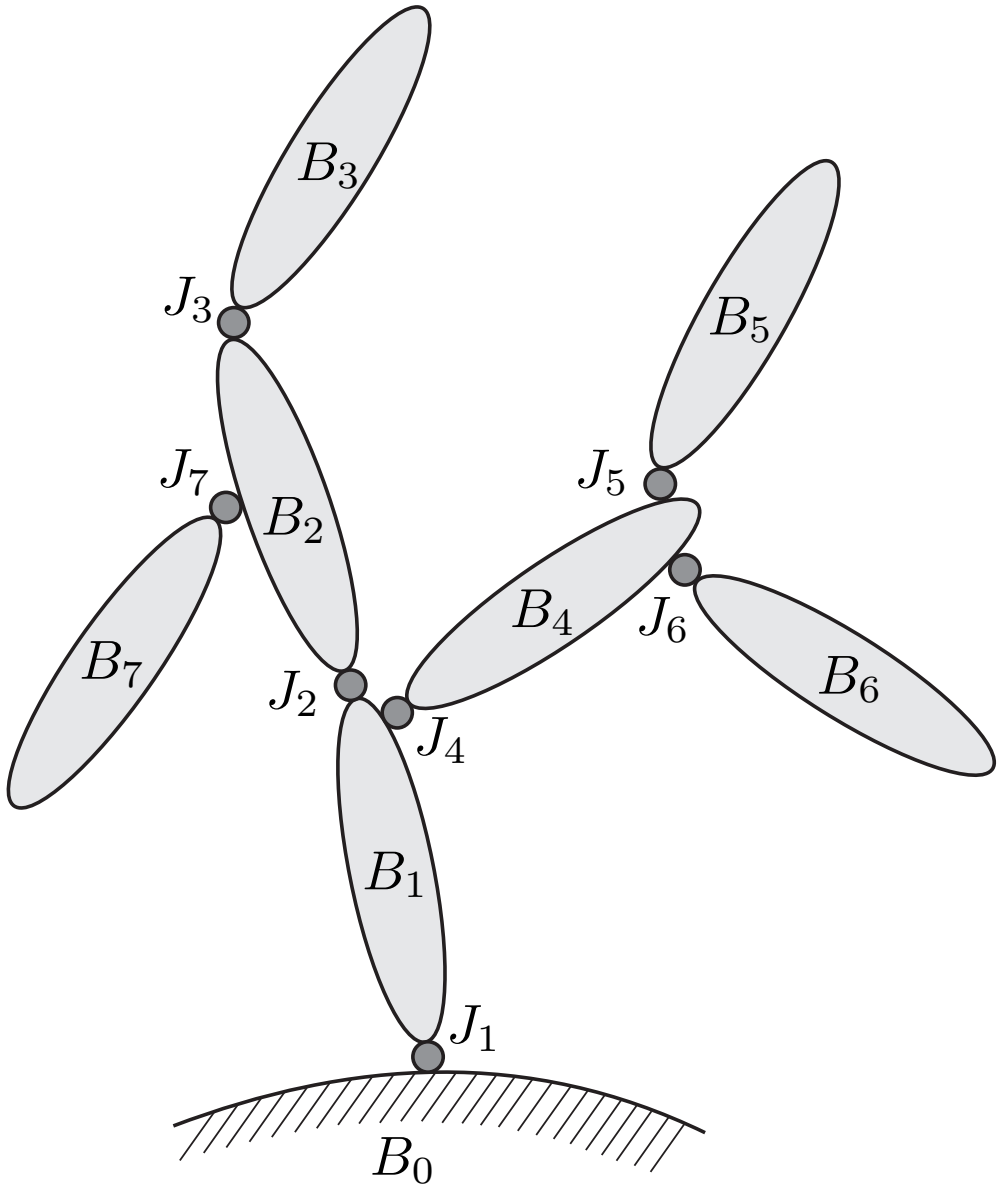
$$\mathbb{P}(B_i) = \mathcal{C}_i, \quad i = 1, \ldots, 7$$

$$\mathcal{C} = \{0, 1, 2, 1, 4, 4, 2\}$$

The connectivity of the system in the figure is encoded in $\mathcal{C}$.

$\mathbb{P}(B_i)$ stands for "the parent body of body i".

Joint i is the parent joint of body i.



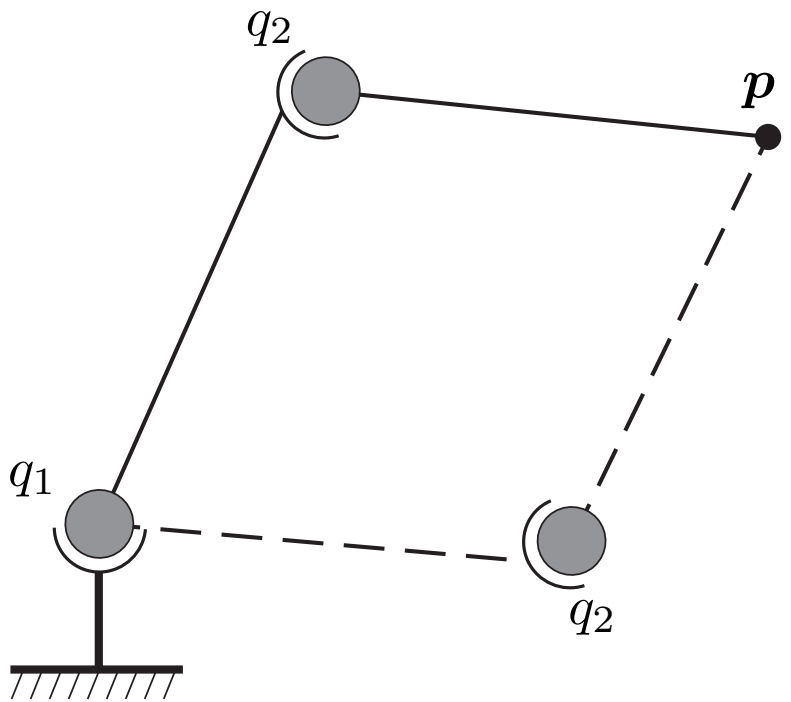
Inverse geometric model (IGM)

In general, the problem of evaluating the IGM could have no solutions, one solution or multiple solutions (even when $m = n$).

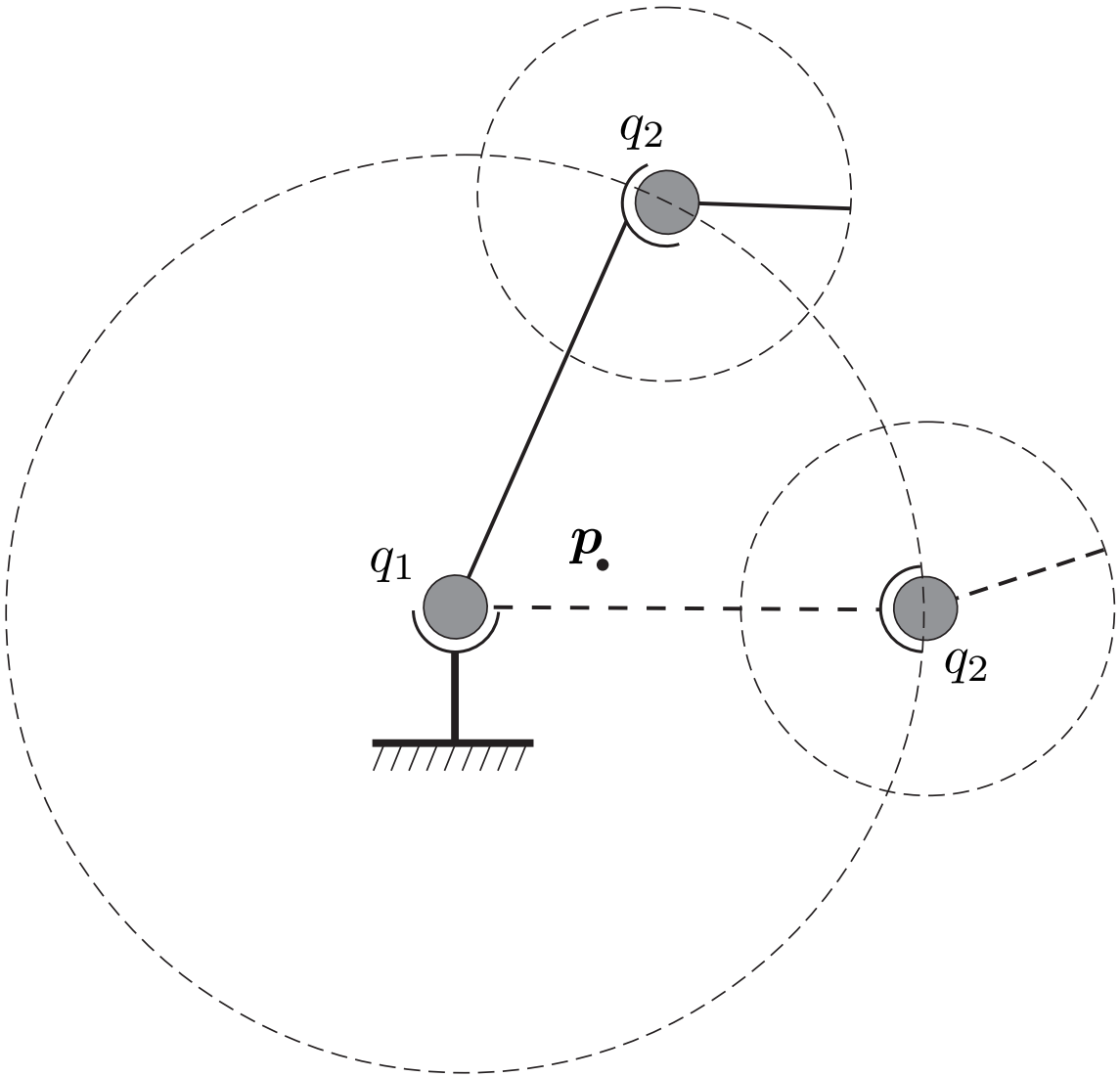
- when $m > n$, there is no solution (if the m constraints are independent)
- $m > n$ implies that the constraints imposed by the task are more than the DoF of the system

In the figure, the end-effector of the manipulator can reach the point $\boldsymbol{p}$ with two different values of q_1 and q_2.

If we want to reach $\boldsymbol{p}$ with a specific orientation of the end-link, then in most cases there will be no solution.



Example (IGM with no solution $m = n$)



The manipulator can not reach point $\boldsymbol{p}$ with its end-effector.

Solution methods

Analytical solution

- analyze the geometric structure of a given manipulator (usually in an ad-hoc way), and develop specialized analytical solutions
- usually applied to systems with simple or special structure, $e.g.$,
  - 3 consecutive rotational joint axes intersect
  - 3 consecutive rotational joint axes are parallel
- fast computation

Numerical solution (more details later on ...)

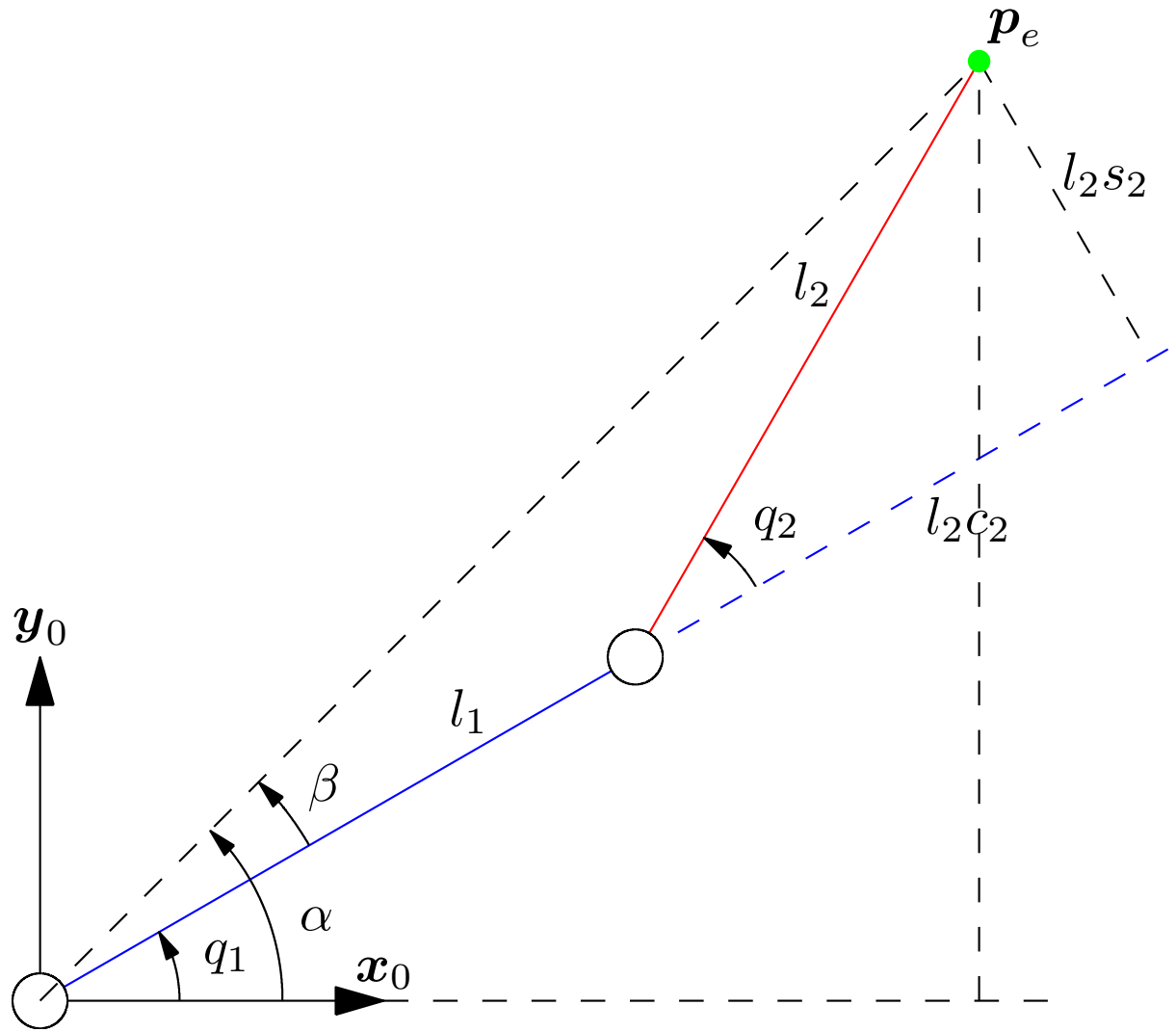
- applicable to any type of system
- very useful in the presence of redundancy with respect to the task
- can account for additional constraints
- can have convergence problems, and produce unexpected results
- depend on an initial guess for $\boldsymbol{q}$
- slower compared to a analytical solution, however, sufficiently fast for many practical applications

IGM for a 2 DoF planar manipulator (analytical solution)

$$c_1 = \cos q_1,\ s_1 = \sin q_1$$
$$c_2 = \cos q_2,\ s_2 = \sin q_2$$
$$c_{12} = \cos(q_1 + q_2)$$
$$s_{12} = \sin(q_1 + q_2)$$
$$p_{e_x} = l_1 c_1 + l_2 c_{12}$$
$$p_{e_y} = l_1 s_1 + l_2 s_{12}$$

Forming $\|\boldsymbol{p}_e\|^2$ leads to

$$c_2 = \frac{p_{x_e}^2 + p_{e_y}^2 - l_1^2 - l_2^2}{2 l_1 l_2}$$
$$s_2 = \pm\sqrt{1 - c_2^2}$$
$$\alpha = \text{atan2}(p_{e_y}, p_{e_x})$$
$$\beta = \text{atan2}(l_2 s_2, l_1 + l_2 c_2)$$
$$q_1 = \alpha - \beta$$
$$q_2 = \text{atan2}(s_2, c_2)$$



Some comments (about the IGM example)

- when forming c_2 we squared and summed the x and y parts of the forward geometric model

$$p_{e_x} = l_1 c_1 + l_2 c_{12}$$
$$p_{e_y} = l_1 s_1 + l_2 s_{12}$$

to obtain (note that q_1 and q_2 are the unknowns, while $\boldsymbol{p}_e$ is given)

$$p_{e_x}^2 + p_{e_y}^2 - l_1^2 - l_2^2 = 2 l_1 l_2 \underbrace{(c_1 c_{12} + s_1 s_{12})}_{c_2}$$

- there are two solutions
  - $s_2 = -\sqrt{1 - c_2^2}$ corresponds to "elbow-up" configuration
  - $s_2 = \sqrt{1 - c_2^2}$ corresponds to "elbow-down" configuration
- if $c_2 \notin [-1, 1]$, $\boldsymbol{p}_e$ is outside the robot workspace
- $l_1 \neq 0$ and $l_2 \neq 0$ is assumed

Constraints (again)

Holonomic constraints

The constraints that we discussed so far are called **holonomic**. They can be expressed as functions of the system's position coordinates $\boldsymbol{f}(\boldsymbol{q}) = \mathbf{0}$.

- Each holonomic constraint decreases the dimension of the configuration space, hence removes one DoF.

Constraints imposed by prismatic and revolute joints are holonomic.

Nonholonomic constraints

Constraints that can not be expressed as $\boldsymbol{f}(\boldsymbol{q}) = \mathbf{0}$ are called **nonholonomic**.

- Consider a car moving in the plane. It can not move instantaneously sideways (this is a constraint on the velocities of the system) however, it has 3 DoF as it can reach any configuration in the plane.

Constraints of the form $\boldsymbol{f}(\boldsymbol{q}) > \mathbf{0}$, are called **unilateral**. We can think of then as *one-sided* constraints (a ball can not penetrate the ground, but it can bounce up).