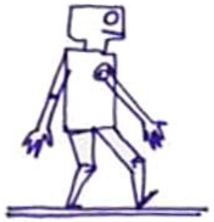
#1 Inverse geometry

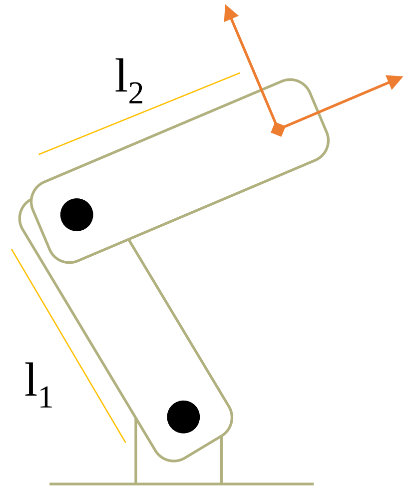
Nicolas Mansard (slide author)

Stéphane Caron, Justin Carpentier, Olivier Roussel, Guilhem Saurel



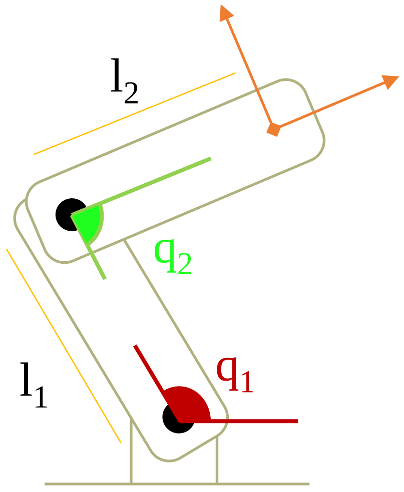
Geometry model

Robot configuration q



Geometry model

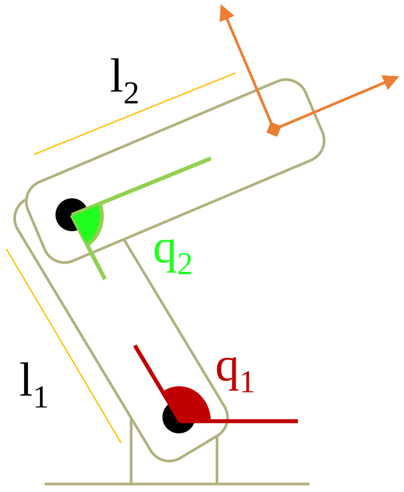
Robot configuration q



Geometry model

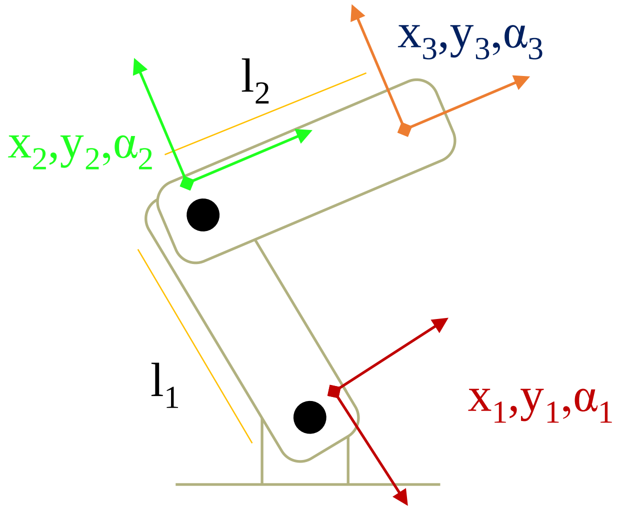
Robot configuration q

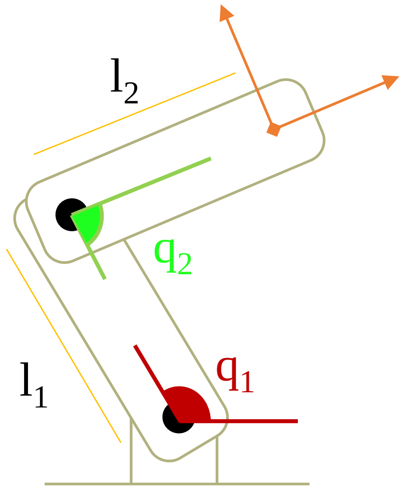
$$\begin{bmatrix} l_1 \cos(q_1) + l_2 \cos(q_1+q_2) \\ l_1 \sin(q_1) + l_2 \sin(q_1+q_2) \end{bmatrix}$$



Geometry model

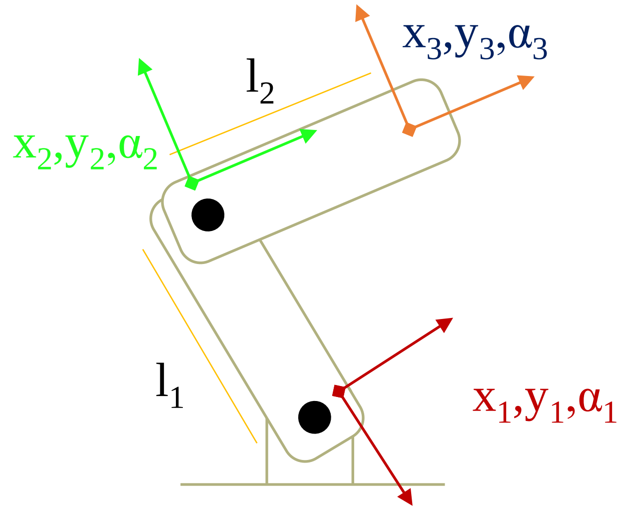
Robot configuration q

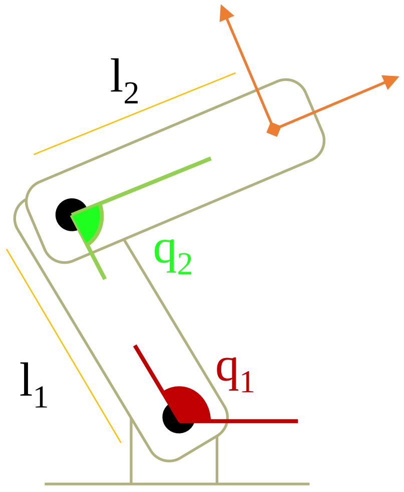




Geometry model

Robot configuration q



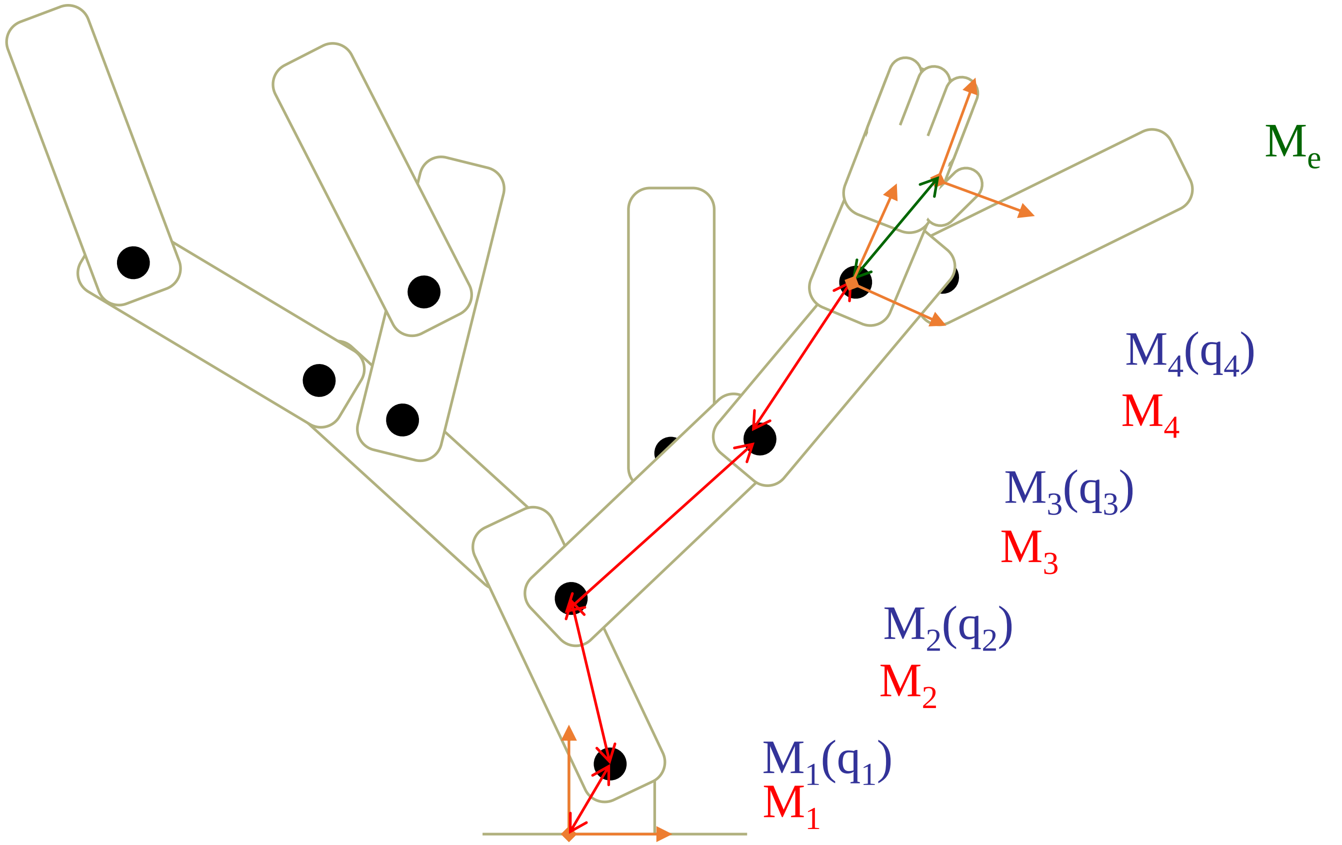


… such that

$$(x_1-x_2)^2+(y_1-y_2)^2 = cst$$
$$(x_2-x_3)^2+(y_2-y_3)^2 = cst$$
… etc …

Direct geometry

The geometric model is a tree of joints and bodies



$$M(q) = M_1 \oplus M_1(q_1) \oplus M_2 \oplus \ \dots \oplus M_4 \oplus M_4(q_4) \oplus M_e$$

About representation of motion

The geometric model is a tree of joints and bodies

$$M = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}$$

Homogeneous matrix
… represents SE(3)

$$\dot{R} = \omega \times R$$

Canonical definition
of angular velocity

What is $\mathrm{M} \in \mathrm{SE}(3)$?

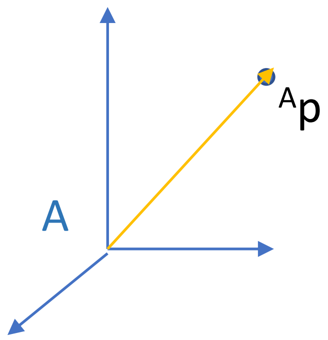
What is $\dot{M}$ (and $\dot{R}$)?

Links with the differential geometry?

$$\mathrm{M(q)} = \mathrm{M_1} \oplus \mathrm{M_1(q_1)} \oplus \mathrm{M_2} \oplus \ \ldots \oplus \mathrm{M_4} \oplus \mathrm{M_4(q_4)} \oplus \mathrm{M_e}$$

Representation!

This is a point



This is not a point
This is the representation of a point

Rotation

- Rotation matrices

$$R = \begin{pmatrix} r00 & r01 & r02 \\ r10 & r11 & r12 \\ r20 & r21 & r22 \end{pmatrix}$$

- Derivation of a matrix

$$\dot{R} = \cdots$$

Angular velocity / Angle vector

- Formal definition

$$\dot{R} = \omega \times R$$

- From rotation to velocity
  - $R \rightarrow \omega$ $\quad R = \exp(\omega \times)$
- From velocity to rotation?
  - $\omega \rightarrow R$ … integrate $\quad \omega \times = \log(R)$

- Meaning of ω : angular velocity
- Angle axis representation $(\omega = u\theta,$ with u the axis, θ the angle)
- Quaternions… see Joan Sola ☺

Quaternions

- Start from complex

  $1, i, -1, -i, 1 \ldots$

  $X, Y, -X, -Y, X \ldots$

Complexes can map the 2D plan , and the 2D rotation

- Hamilton (again!) says: let's do it more complex

  j so that $j^2 = -1$ *and* $ij = -ji$

  $ij = k$, $jk = i \ldots$

  x G $y = z$, y G $z = x \ldots$

- Unit quaternions map 3D rotations

  $q = [\ w, x, y, z\ ] = cos(\alpha/2), sin(\alpha/2)[a,b,c]$

Joint models

- Maps
  - From configuration space
  - To SE3 space

$$h(q) = {}^{k}M_{k+1}(q) \quad \in SE(3)$$

- For example, Revolute-Z is:

$$h(q) = \begin{bmatrix} \cos q & \sin q & 0 & 0 \\ -\sin q & \cos q & 0 & 0 \\ 0 & 0 & 1 & 0 \\ & 0 & & 1 \end{bmatrix}$$

Kinematic model parametrization

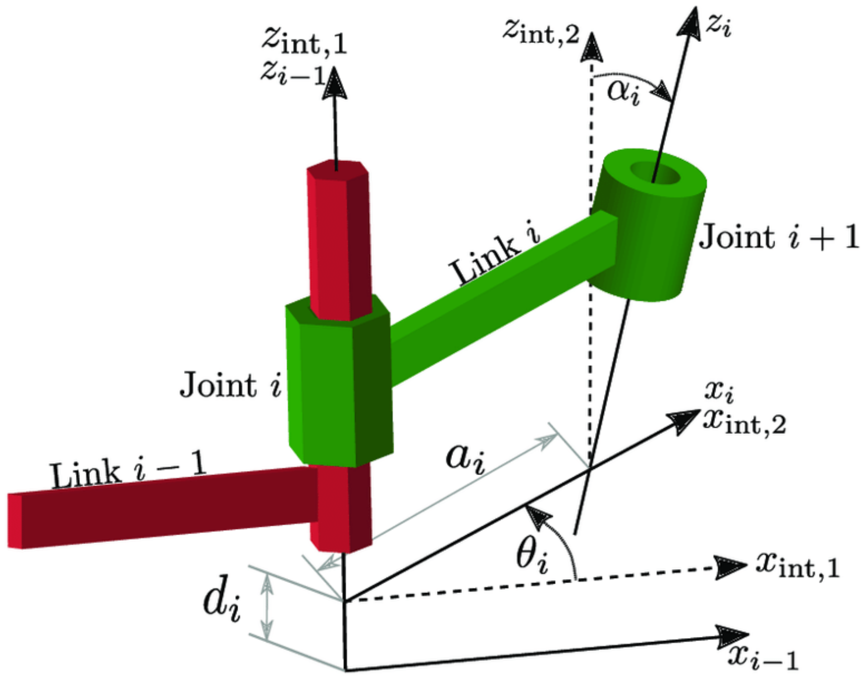
- Parent-to-child joint transformation

- Modern solution:
  - URDF model
  - with your favorite SE3 representation

- Good-old days:

  with Denavit-Hartenberg

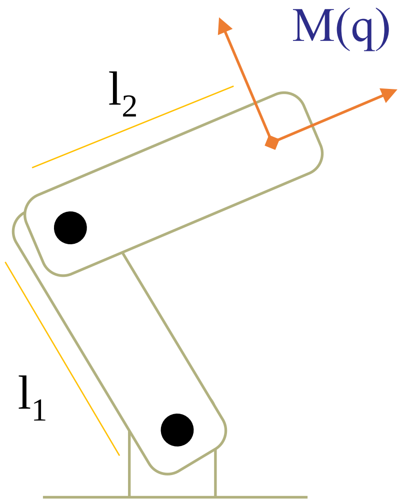
  minimal parameters



Recap

`pin.Model`	Kinematic tree, frame placement
`pin.Data`	Buffers for algorithms computations
`    data.oMi`	Placement of the joints wrt world
`    data.oMf`	Placement of frames wrt world
`pin.framesForwardKinematics(model,data,q,v)`	
`M = pin.SE3(R,p)`	Placement (rotation+translation) matrix
`    M.rotation,M.translation`	
`pin.log6(M).vector`	SE3 log, convert placement to motion
`pin.exp3(np.array([1,2,3]))`	SO3 exponential, "integrate" velocity to rotation matrix

Inverse geometry



Being given a M^* …

what is q such that $M(q) = M^*$

$$M^{-1}: M^* \rightarrow q = M^{-1}(M^*)$$

Numerical inversion of the geometry

- Computing analytically h^{-1} is difficult and tedious
- We can compute it numerically!

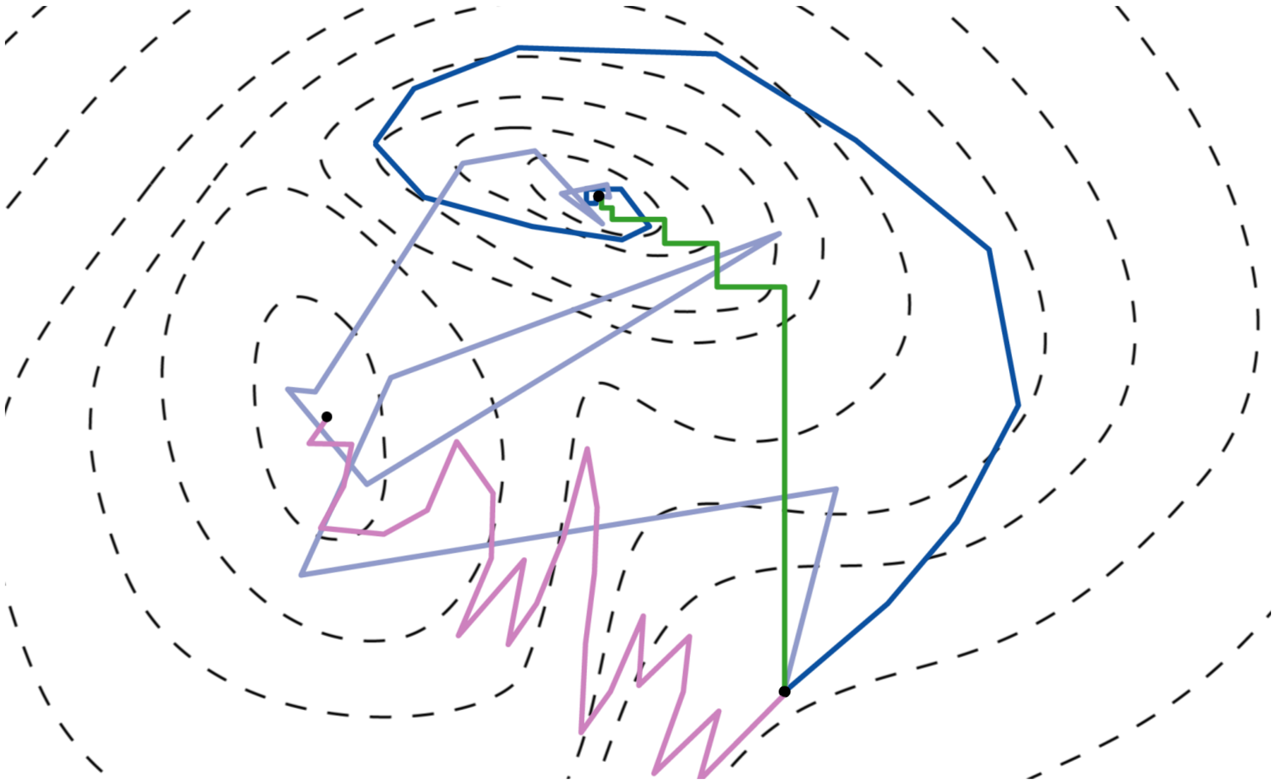
- Problem definition

$$search\ f(x) - f^{*} = 0$$

$$min\ \| f(x) - f^{*} \|^2$$

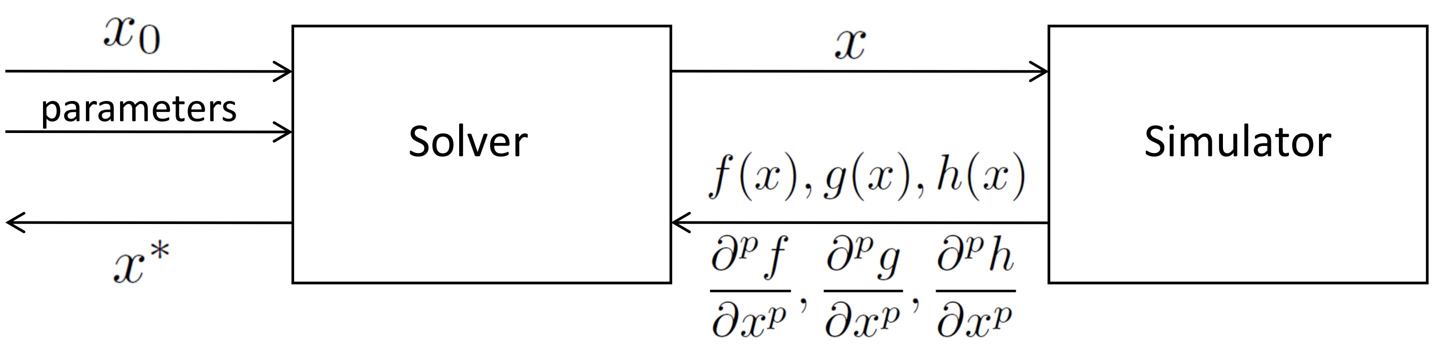
Follow the slope

- Decreasing sequence: $f(x_{k+1}) < f(x)$



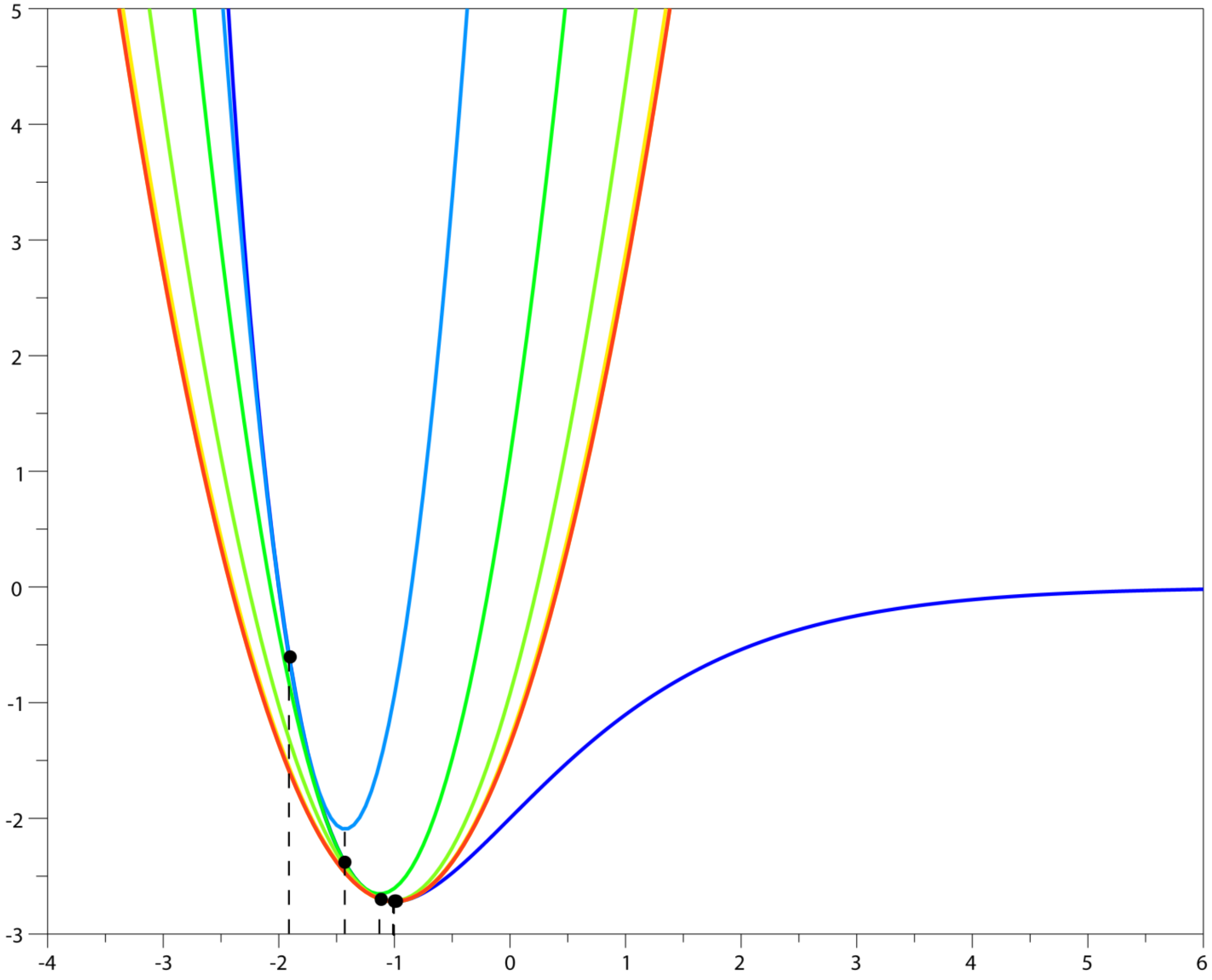
Problem specifications

- Problem specification
  - Computing $f(x)$ is easy
  - We can derivate $f: x \rightarrow f(x)$
  - We know the distance to the reference value



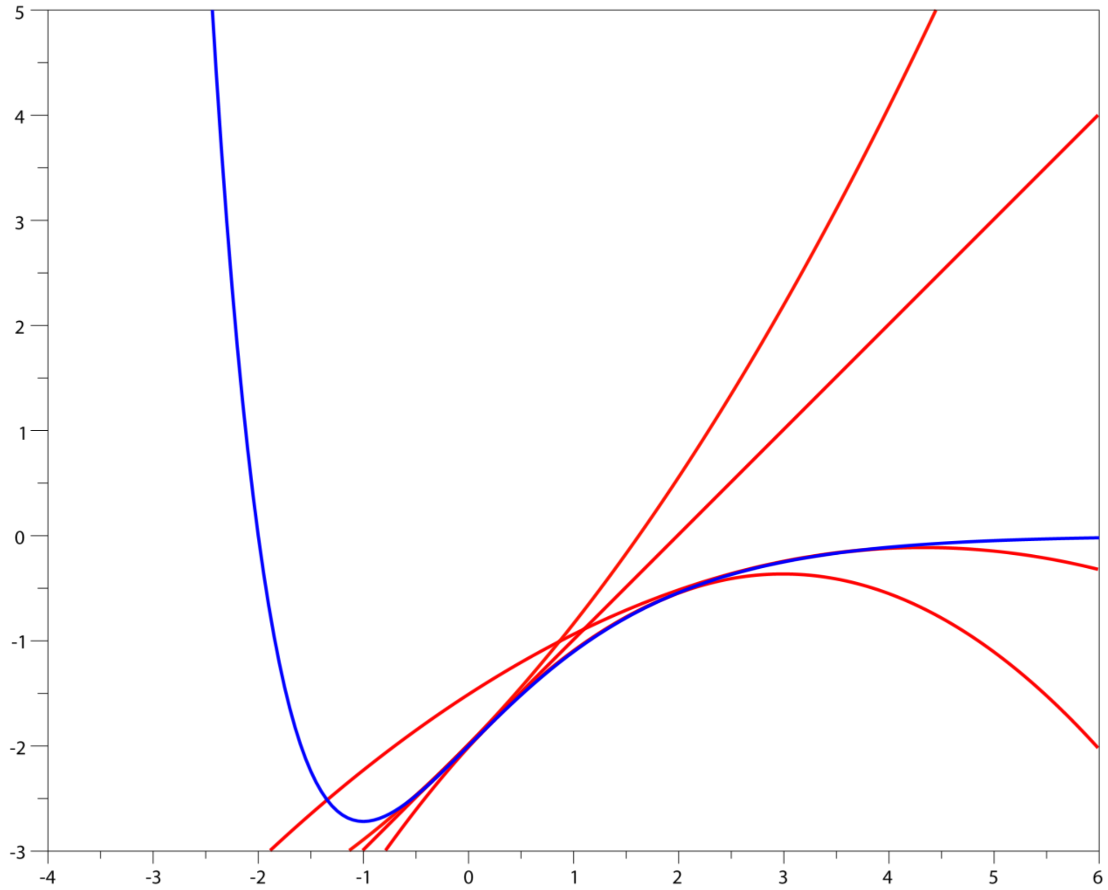
Newton method (unconstrained)

x0=-1.9
x1=-1.4263158
x2=-1.1274228
x3=-1.0144015
x4=-1.0002045
x5=-1.00000004
x6=-1.



Newton method (unconstrained)

- Ill-conditionned hessian
- Non positive hessian



Inverse geometry

- Decide: the robot configuration q

- Minimizing:

$$\|p(q) - p^*\|^2 \quad \text{(3d objective)}$$

$$\left\|\log\left({}^{o}M_i(q)^{-1}\, {}^{o}M_*\right)\right\|^2 \quad \text{(6d objective)}$$