Inverse Kinematics: Manipulators

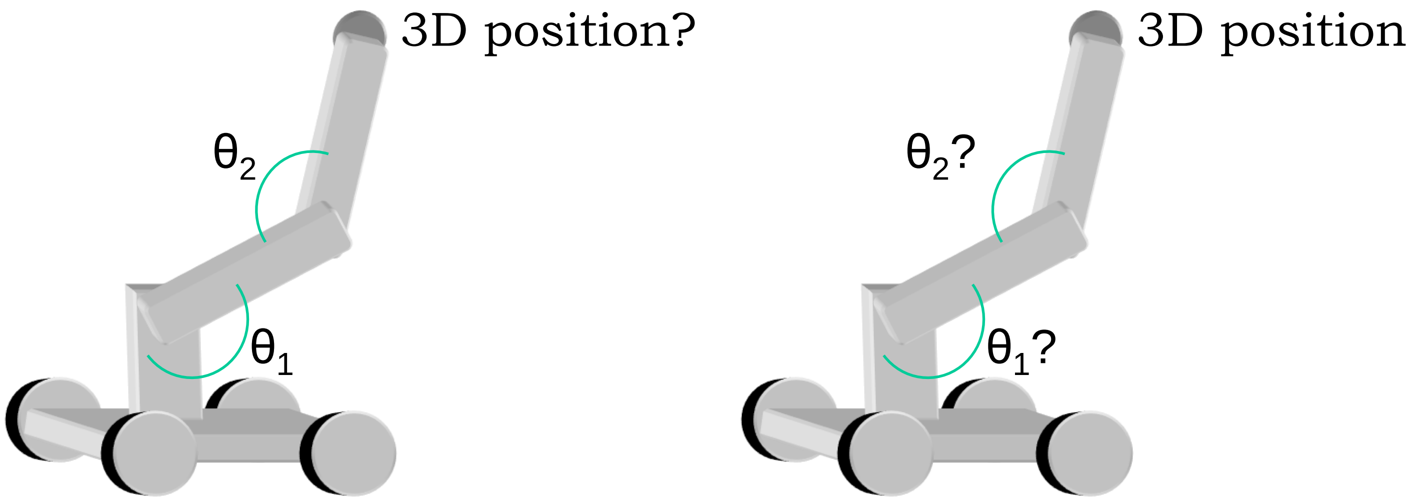
ZA-2203 Robotic Systems

Topics

- Approaches in inverse kinematics (IK)
- Analytical IK for 2R planar robot: geometry
- Analytical IK for 2R planar robot: algebra
- Analytical IK for 6R Puma robot
- Numerical IK: Newton-Raphson method for numerical IK

Forward & Inverse Kinematics

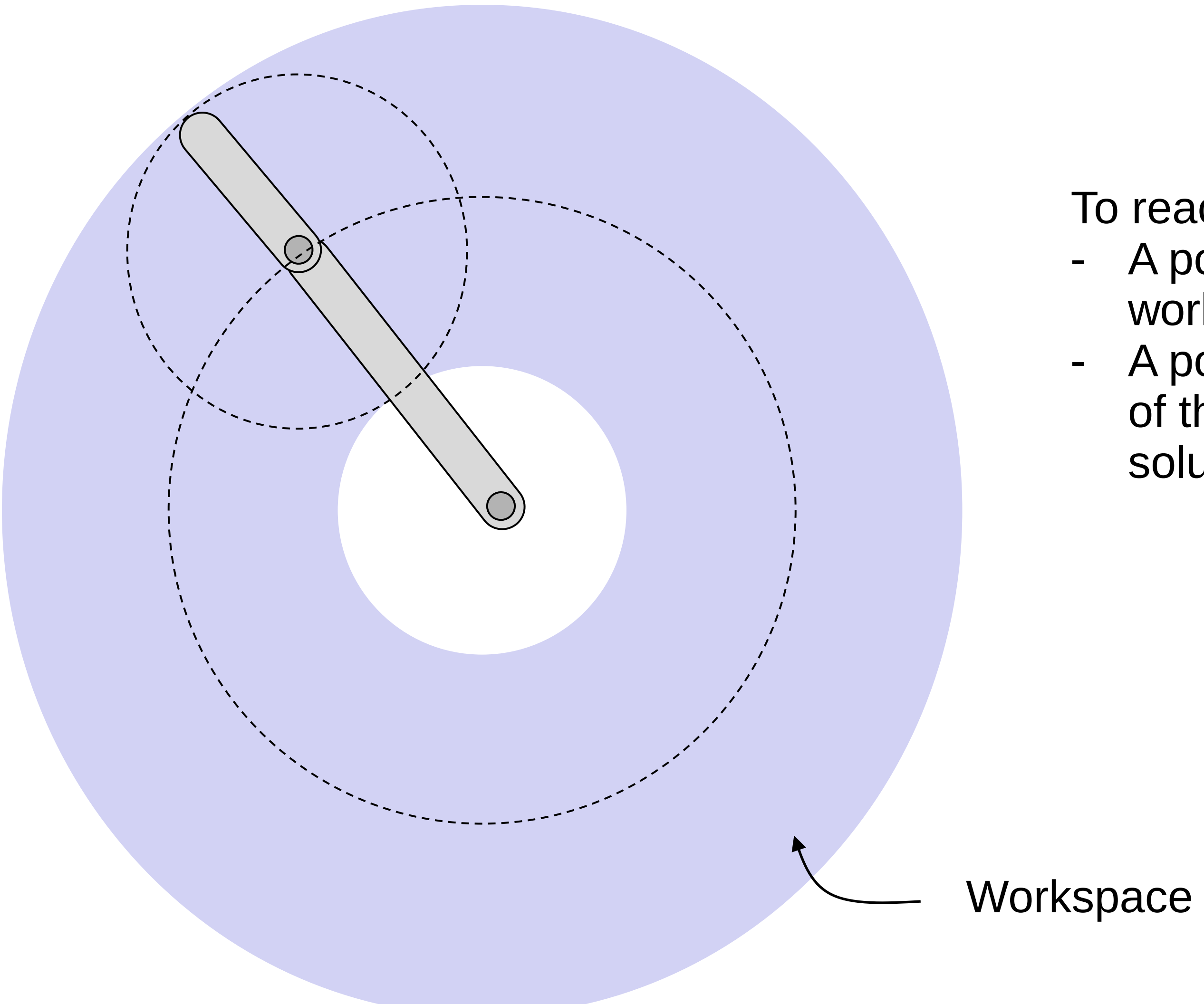
- **Kinematics** – is the study of motion without regard to forces.
  - Study of correspondence between actuator **mechanisms** (joint variables) and resulting **motion** of effectors.
- **Forward Kinematics** (FK): for the given angular movement at each joint, where will the end-effector reach?
- **Inverse Kinematics** (IK): for a desired position of the end-effector, how much should each joint rotate?



Inverse kinematics: three approaches

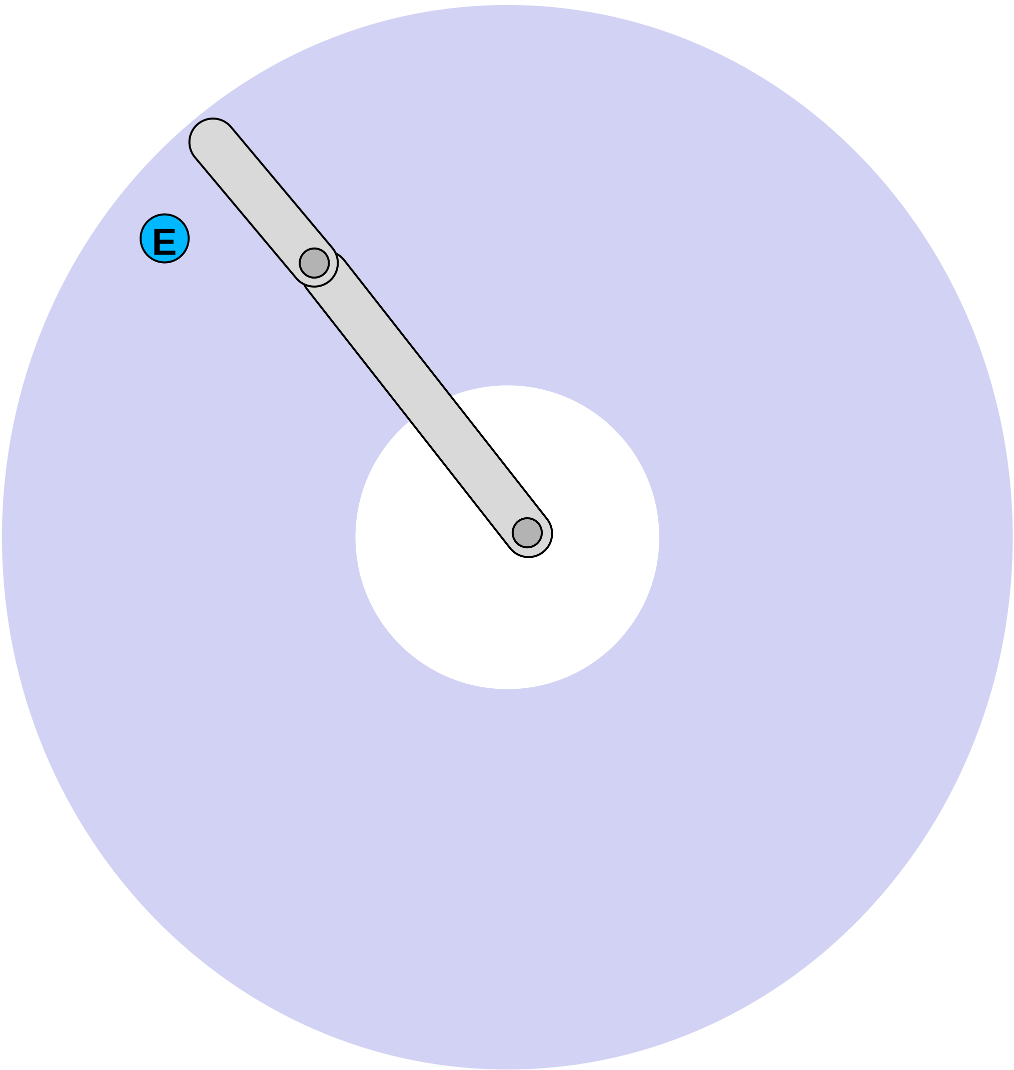
- Three approaches:
  - Analytical: geometry
  - Analytical: algebra, i.e. solving equations usually from FK
  - Numerical: iteratively find the solution using optimization algorithm
- More difficult than FK
- May have 0, 1 or multiple solutions, possibly infinite solutions
- Analytical closed-form solution(s) not always possible, however it can find all possible solutions
- Iterative numerical approach will find only one solution depending on the initial guess

2R planar open chain manipulator



To reach:
- A position outside the workspace, no solution
- A position at the boundary of the workspace, one solution

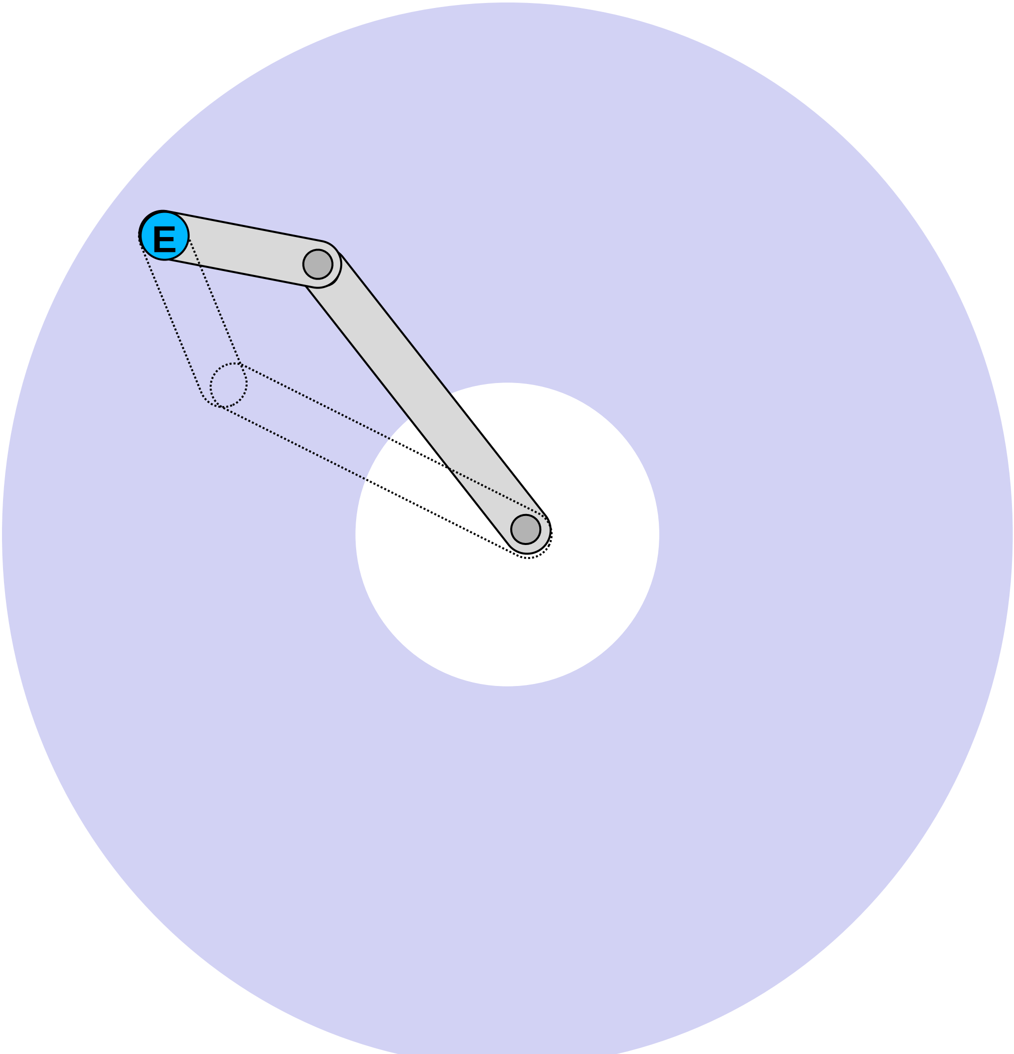
2R planar open chain manipulator



To reach:

- A position outside the workspace, no solution
- A position at the boundary of the workspace, one solution
- A position within the workspace, multiple solutions

2R planar open chain manipulator

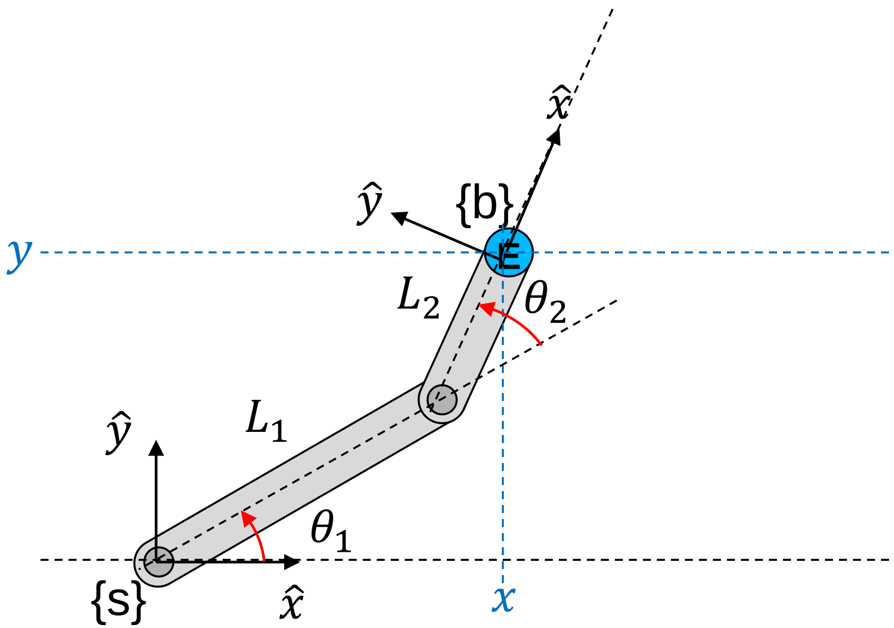


To reach:

- A position outside the workspace, no solution
- A position at the boundary of the workspace, one solution
- A position within the workspace, multiple solutions

2R planar robot: geometry

IK: Determine θ_1, θ_2 given pose of {b} in {s}



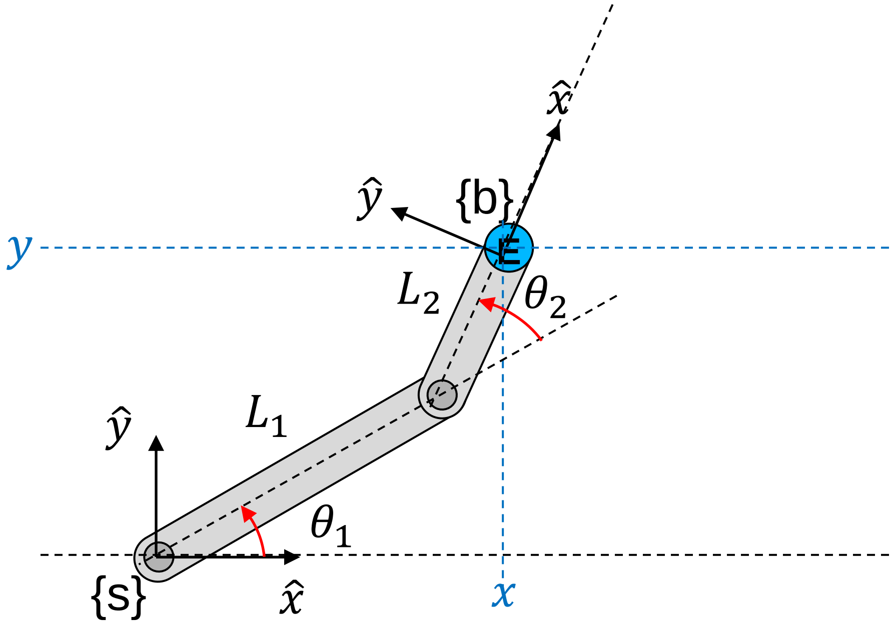
Given desired end-effector position E=(x, y).

Let's consider the position only. Usually, orientation can be treated separately especially if wrist joint is spherical, i.e. the wrist joint determine the orientation.

Determine values of $\theta = (\theta_1, \theta_2)$.

2R planar robot: geometry

IK: Determine θ_1, θ_2 given pose of {b} in {s}

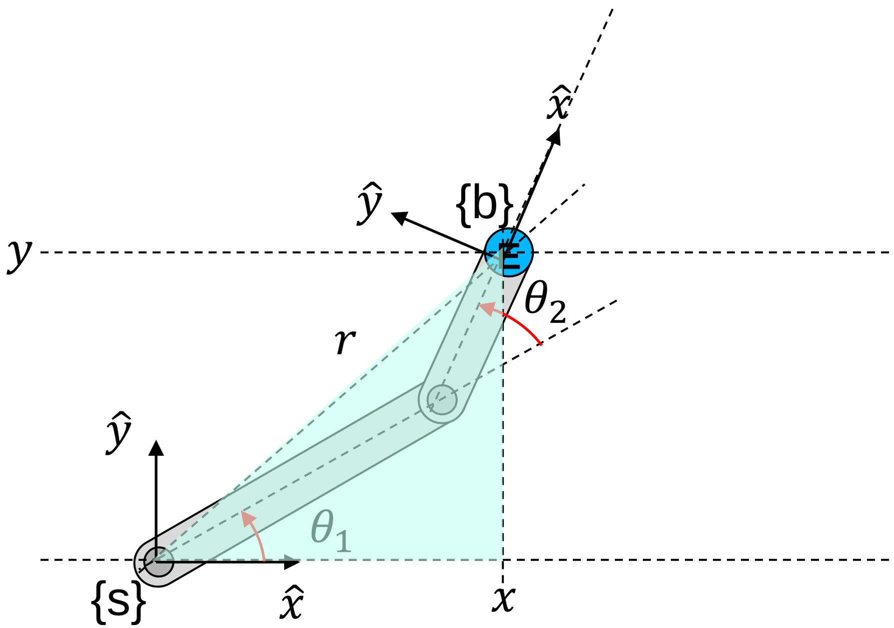


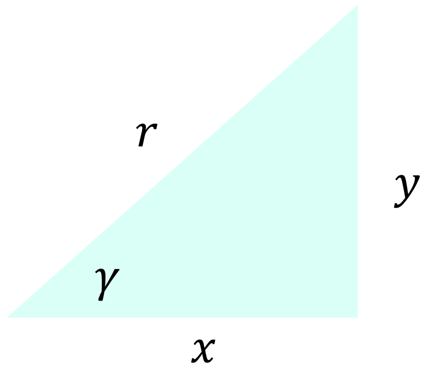
Given desired end-effector position E=(x, y).

Determine values of $\theta = (\theta_1, \theta_2)$.

2R planar robot: geometry

IK: Determine θ_1, θ_2 given pose of {b} in {s}





Given desired end-effector position E=(x, y).

Determine values of $\theta = (\theta_1, \theta_2)$.

$r^2 = x^2 + y^2$ (Pythagoras Theorem)

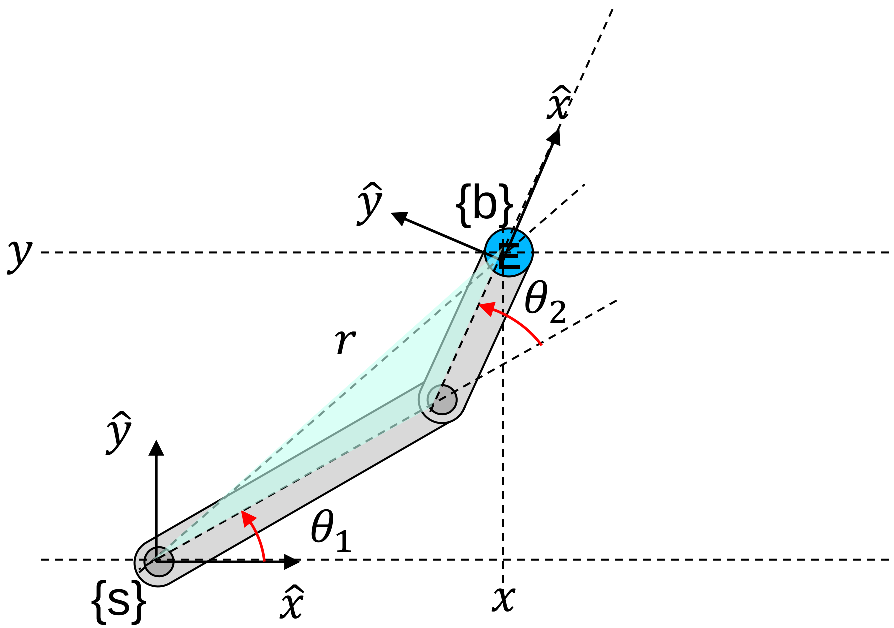
$$\gamma = \tan^{-1}\frac{y}{x}$$

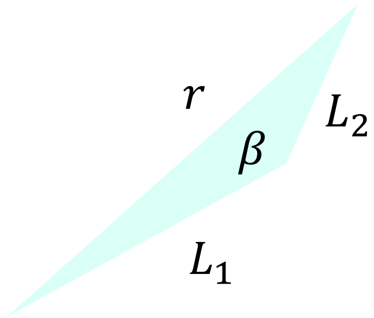
To consider the quadrant of γ:

$$\gamma = \text{atan2}(y, x)$$

2R planar robot: geometry

IK: Determine θ_1, θ_2 given pose of {b} in {s}





Given desired end-effector position E=(x, y).

Determine values of $\theta = (\theta_1, \theta_2)$.

$$r^2 = x^2 + y^2$$
$$\gamma = \text{atan2}(y, x)$$

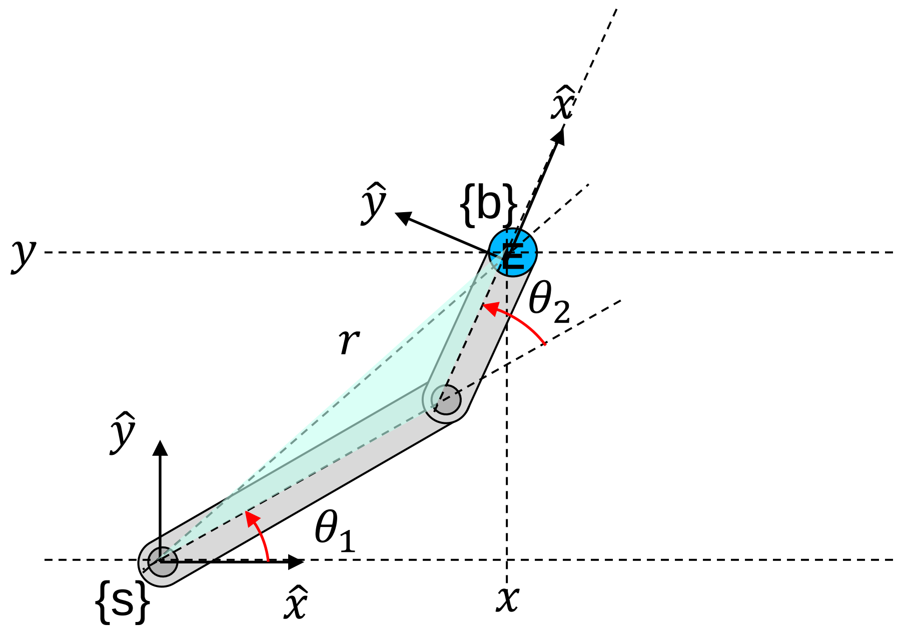
$$r^2 = {L_1}^2 + {L_2}^2 - 2L_1L_2 \cos\beta$$
(Cosine Rule)
$$x^2 + y^2 = {L_1}^2 + {L_2}^2 - 2L_1L_2 \cos\beta$$

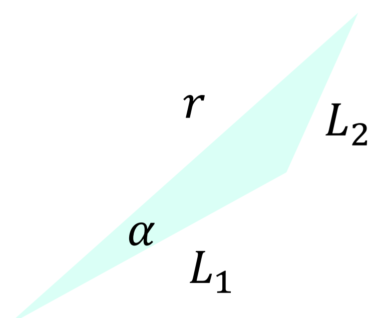
Rearrange:

$$\beta = \cos^{-1}\left(\frac{{L_1}^2 + {L_2}^2 - x^2 - y^2}{2L_1L_2}\right)$$

2R planar robot: geometry

IK: Determine θ_1, θ_2 given pose of {b} in {s}





Given desired end-effector position E=(x, y).

Determine values of $\theta = (\theta_1, \theta_2)$.

$$r^2 = x^2 + y^2$$
$$\gamma = \operatorname{atan2}(y, x)$$
$$\beta = \cos^{-1}\left(\frac{{L_1}^2 + {L_2}^2 - x^2 - y^2}{2L_1L_2}\right)$$

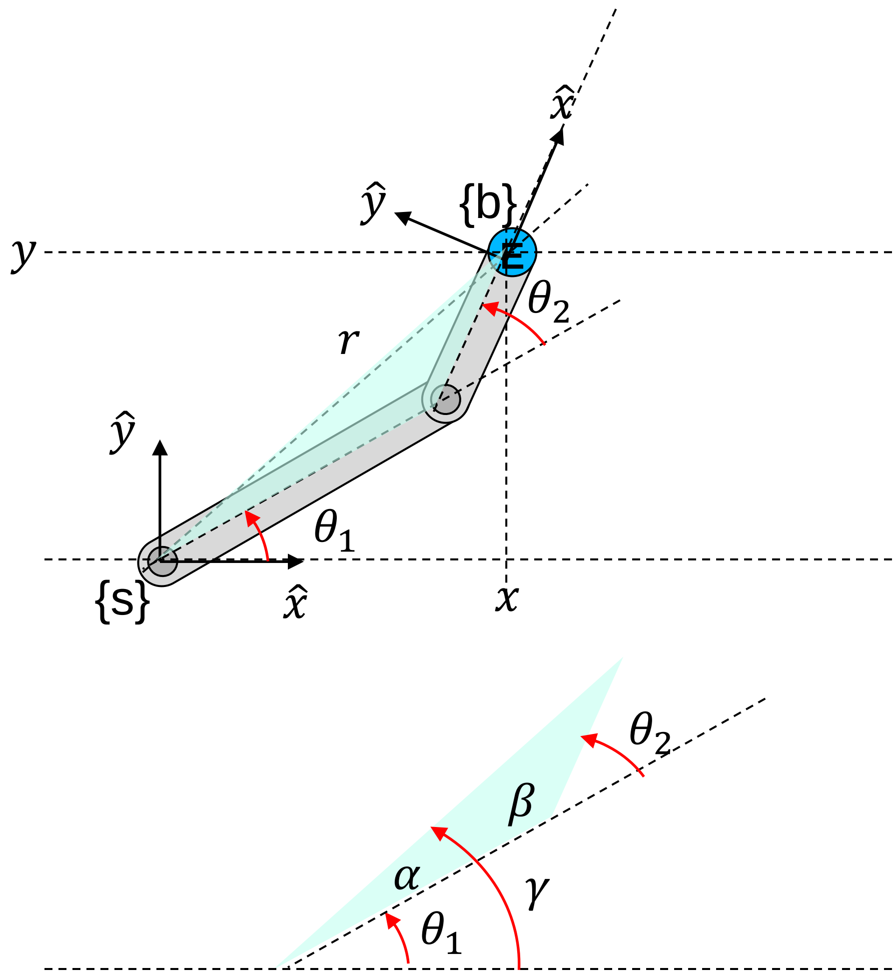
$${L_2}^2 = {L_1}^2 + r^2 - 2L_1 r \cos\alpha$$
$${L_2}^2 = {L_1}^2 + x^2 + y^2 - 2L_1\sqrt{x^2 + y^2}\cos\alpha$$

Rearrange:

$$\alpha = \cos^{-1}\left(\frac{{L_1}^2 - {L_2}^2 + x^2 + y^2}{2L_1\sqrt{x^2 + y^2}}\right)$$

2R planar robot: geometry

IK: Determine θ_1, θ_2 given pose of {b} in {s}



Given desired end-effector position E=(x, y).

Determine values of $\theta = (\theta_1, \theta_2)$.

$$\gamma = \operatorname{atan2}(y, x)$$

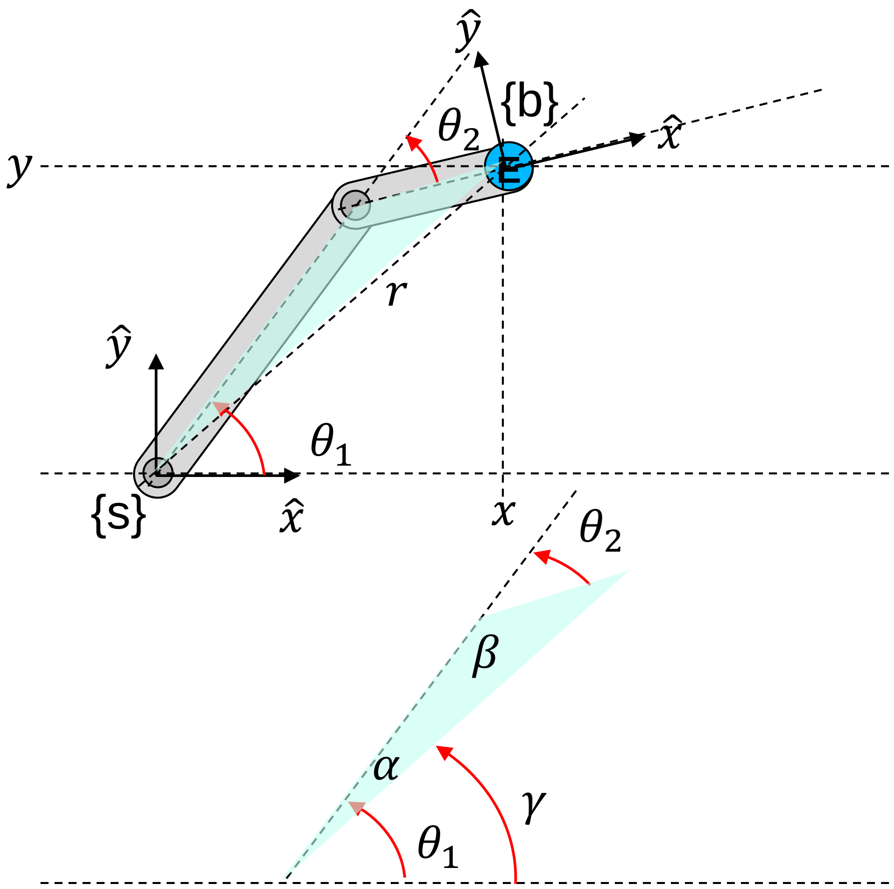
$$\beta = \cos^{-1}\left(\frac{{L_1}^2 + {L_2}^2 - x^2 - y^2}{2L_1L_2}\right)$$

$$\alpha = \cos^{-1}\left(\frac{{L_1}^2 - {L_2}^2 + x^2 + y^2}{2L_1\sqrt{x^2 + y^2}}\right)$$

$$\theta_1 = \gamma - \alpha \qquad \theta_2 = \pi - \beta$$

2R planar robot: geometry

IK: Determine θ_1, θ_2 given pose of {b} in {s}



Given desired end-effector position E=(x, y).

Determine values of $\theta = (\theta_1, \theta_2)$.

$$\gamma = \operatorname{atan2}(y, x)$$

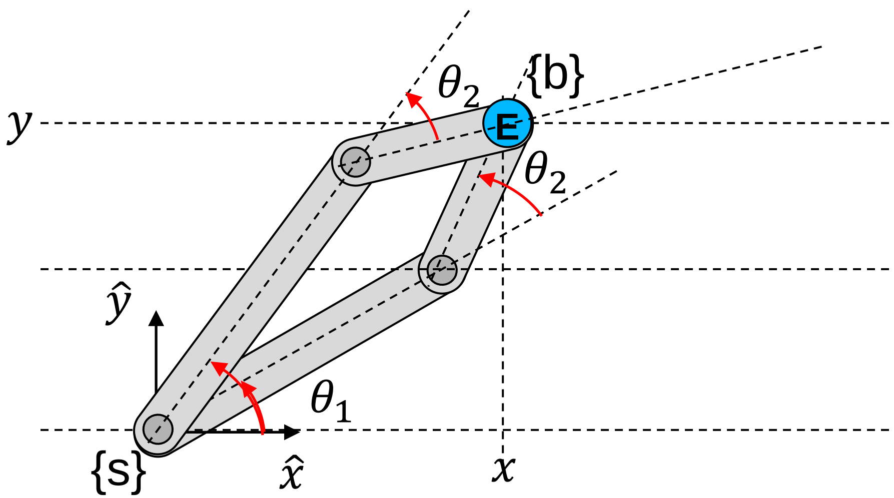
$$\beta = \cos^{-1}\left(\frac{{L_1}^2 + {L_2}^2 - x^2 - y^2}{2L_1L_2}\right)$$

$$\alpha = \cos^{-1}\left(\frac{{L_1}^2 - {L_2}^2 + x^2 + y^2}{2L_1\sqrt{x^2 + y^2}}\right)$$

$$\theta_1 = \gamma + \alpha \qquad \theta_2 = -(\pi - \beta)$$

2R planar robot: geometry

IK: Determine θ_1, θ_2 given pose of {b} in {s}



Given desired end-effector position E=(x, y).

Determine values of $\theta = (\theta_1, \theta_2)$.

$$\gamma = \operatorname{atan2}(y, x)$$

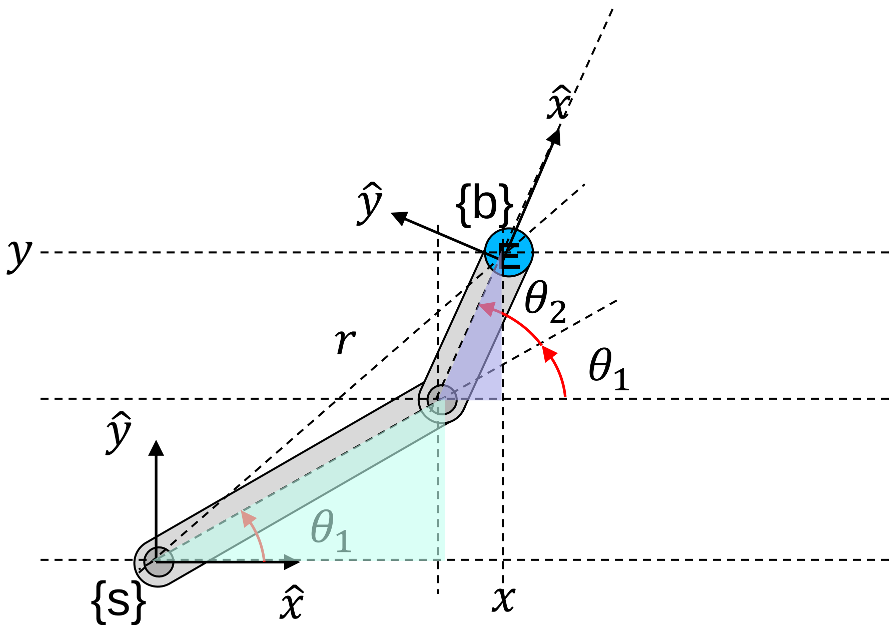
$$\beta = \cos^{-1}\left(\frac{L_1{}^2 + L_2{}^2 - x^2 - y^2}{2L_1L_2}\right)$$

$$\alpha = \cos^{-1}\left(\frac{L_1{}^2 - L_2{}^2 + x^2 + y^2}{2L_1\sqrt{x^2 + y^2}}\right)$$

Righty: $\theta_1 = \gamma - \alpha \quad \theta_2 = \pi - \beta$
Lefty: $\theta_1 = \gamma + \alpha \quad \theta_2 = \beta - \pi$

2R planar robot: algebra

IK: Determine θ_1, θ_2 given pose of {b} in {s}

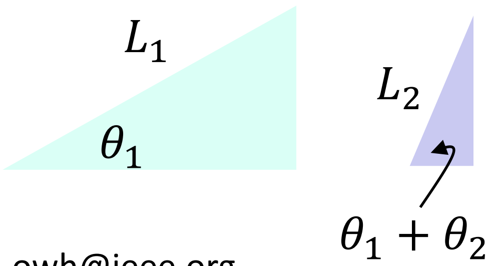


Given desired end-effector position E=(x, y).

Determine values of $\theta = (\theta_1, \theta_2)$.
Forward kinematics:

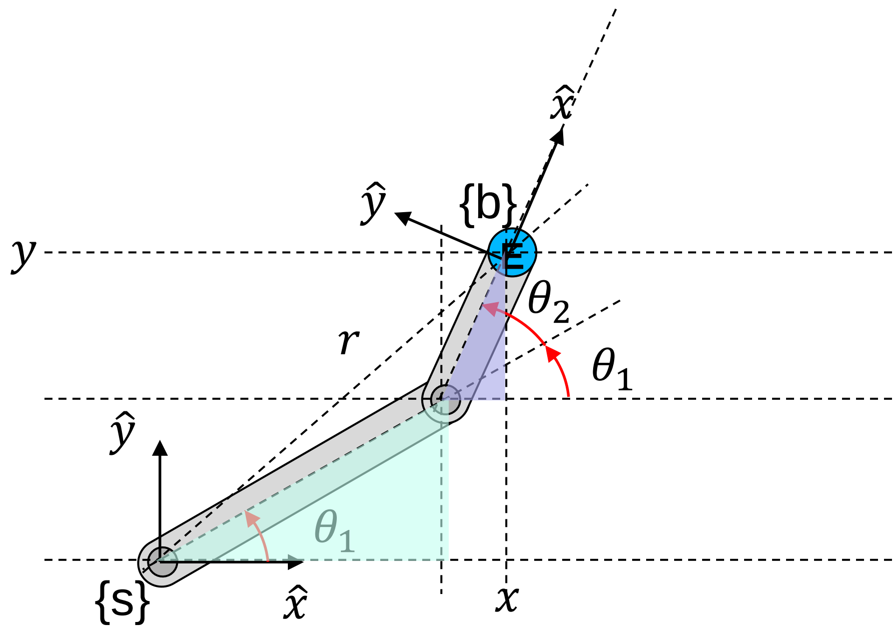
$$x = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2)$$
$$y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2)$$

Two equations, two unknowns, solve by algebra.



2R planar robot: algebra

IK: Determine θ_1, θ_2 given pose of {b} in {s}



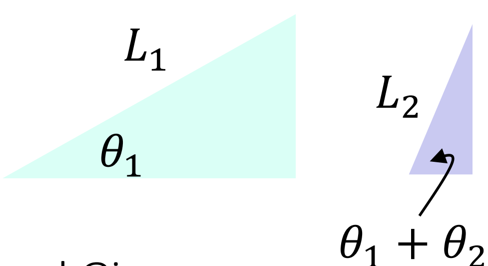
Given desired end-effector position E=(x, y).

Determine values of $\theta = (\theta_1, \theta_2)$.
Forward kinematics:

$$x = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2)$$
$$y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2)$$

Two equations, two unknowns, solve by algebra.

$$x^2 + y^2$$
$$= (L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2))^2$$
$$+ (L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2))^2$$

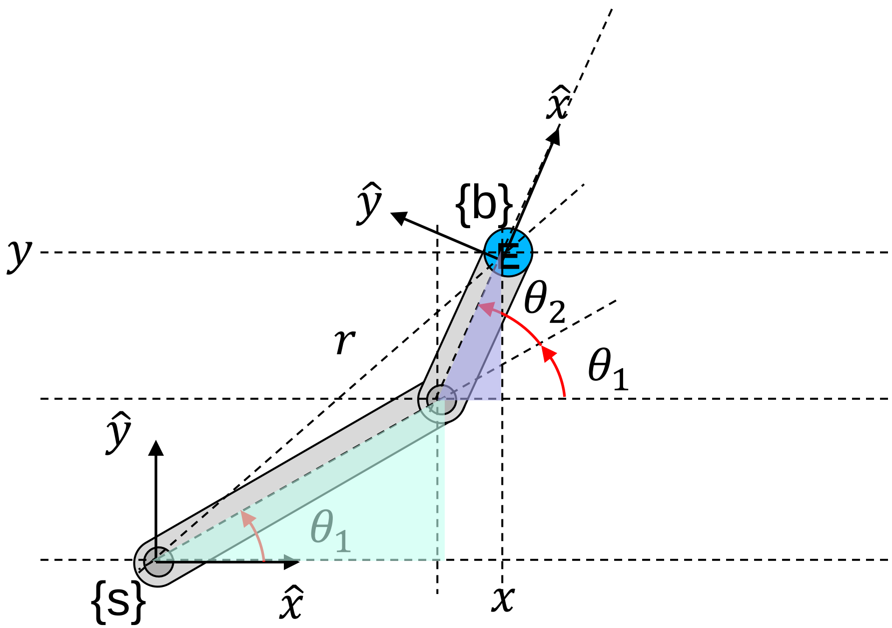


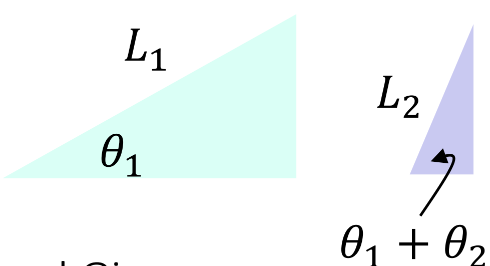
$$x^2 + y^2 = {L_1}^2 + {L_2}^2 + 2L_1L_2 \cos\theta_2$$

$$\theta_2 = \cos^{-1}\left(\frac{x^2 + y^2 - {L_1}^2 - {L_2}^2}{2L_1L_2}\right)$$

2R planar robot: algebra

IK: Determine θ_1, θ_2 given pose of {b} in {s}





Forward kinematics:

$$x = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2)$$
$$y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2)$$

Two equations, two unknowns, solve by algebra.

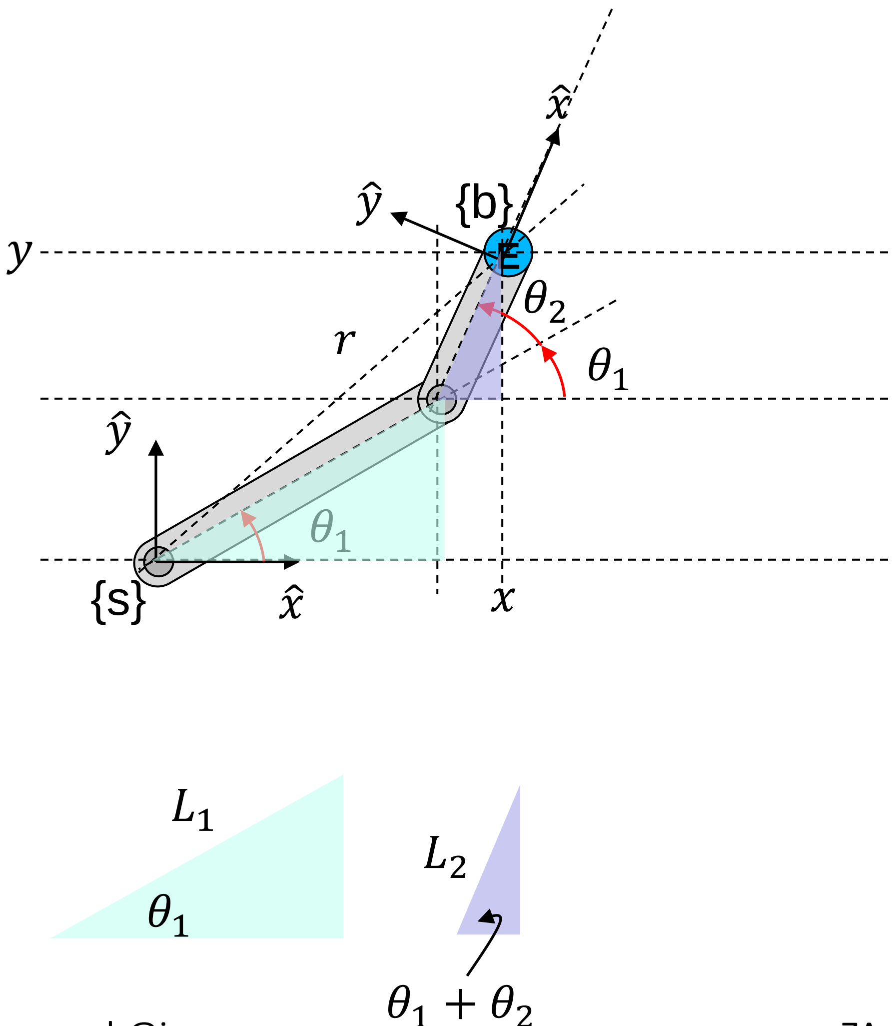
$$\begin{aligned} x &= L_1 c_1 + L_2 c_{12} \\ &= L_1 c_1 + L_2(c_1 c_2 - s_1 s_2) \\ y &= L_1 s_1 + L_2 s_{12} \\ &= L_1 s_1 + L_2(s_1 c_2 + c_1 s_2) \end{aligned}$$

$$x = (L_1 + L_2 c_2) c_1 - L_2 s_2 s_1$$
$$y = (L_1 + L_2 c_2) s_1 + L_2 s_2 c_1$$

$$x = (L_1 + L_2 c_2) \cos\theta_1 - L_2 s_2 \sin\theta_1$$
$$y = (L_1 + L_2 c_2) \sin\theta_1 + L_2 s_2 \cos\theta_1$$

2R planar robot: algebra

IK: Determine θ_1, θ_2 given pose of {b} in {s}



Forward kinematics:

$$x = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2)$$
$$y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2)$$

Two equations, two unknowns, solve by algebra.

$$x = (L_1 + L_2 c_2)\cos\theta_1 - L_2 s_2 \sin\theta_1$$
$$y = (L_1 + L_2 c_2)\sin\theta_1 + L_2 s_2 \cos\theta_1$$

$$x = A\cos\theta_1 - B\sin\theta_1$$
$$y = A\sin\theta_1 + B\cos\theta_1$$

θ_1 can be solved by:

For $\quad a\cos\theta + b\sin\theta = c,$

$$\theta = \tan^{-1}\frac{c}{\sqrt{a^2+b^2-c}} - \tan^{-1}\frac{a}{b}$$

6R PUMA-Type robot: analytical e.g.

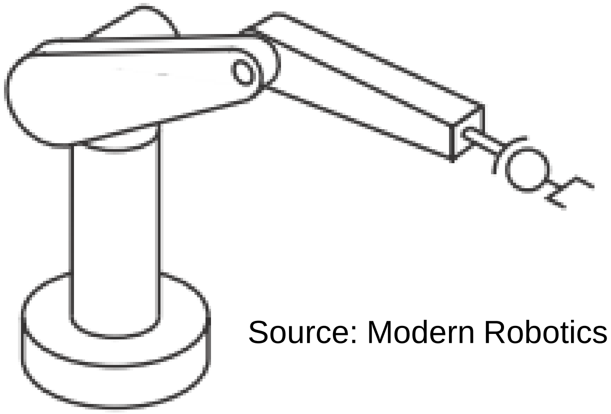
The pose of the end effector of an 6R robot can be represented as a homogeneous transformation matrix

$$T(\theta) = e^{[\mathcal{S}_1]\theta_1} e^{[\mathcal{S}_2]\theta_2} e^{[\mathcal{S}_3]\theta_3} e^{[\mathcal{S}_4]\theta_4} e^{[\mathcal{S}_5]\theta_5} e^{[\mathcal{S}_6]\theta_6} M$$

For a desired pose of the end-effector T_{sd}, the IK problem is to find solution $\theta \in \mathbb{R}^6$ satisfying $T(\theta) = T_{sd}$.

For this robot with a spherical wrist, the position and orientation can be decoupled (**kinematics decoupling**): solve θ_1, θ_2, θ_3 for **inverse position**, then solve θ_4, θ_5, θ_6 for **inverse orientation**.

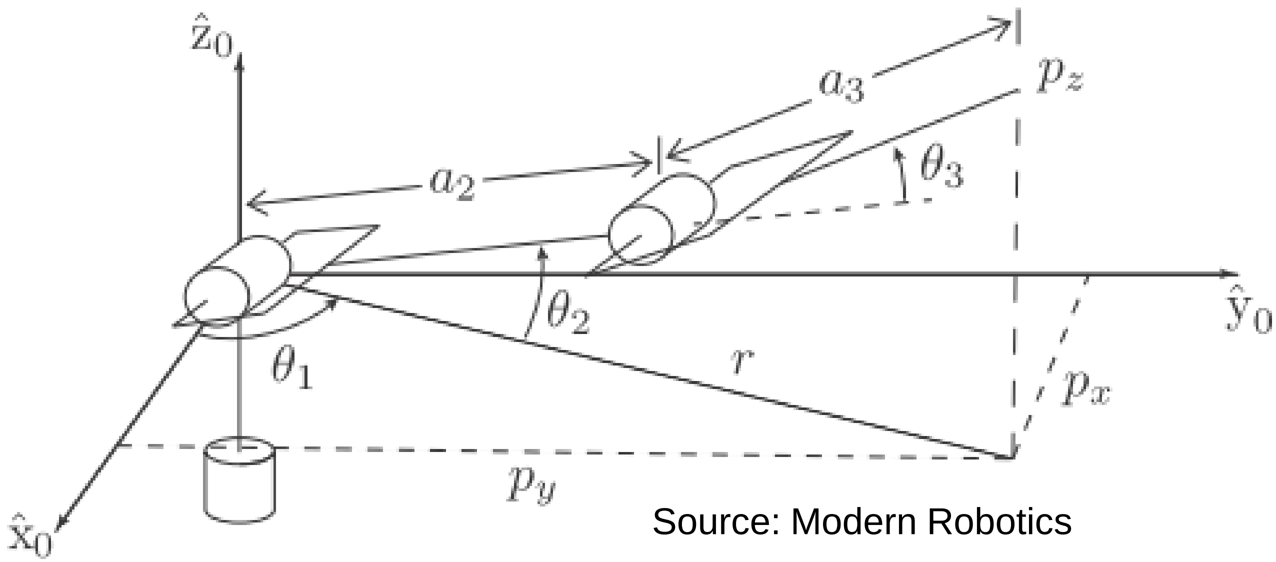
This example uses **geometry** to solve for inverse position problem, and **algebra** to solve for inverse orientation problem.



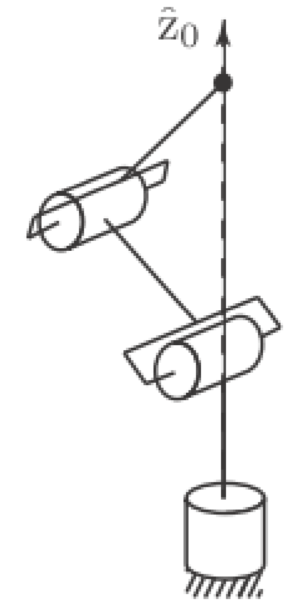
Source: Modern Robotics

6R PUMA-Type robot: analytical e.g.

We use **geometry** to solve the **inverse position IK** problem. We want to find θ_1, θ_2, θ_3 to achieve the desired position of the end-effector $\boldsymbol{p}_d = (p_x, p_y, p_z)$.



Source: Modern Robotics



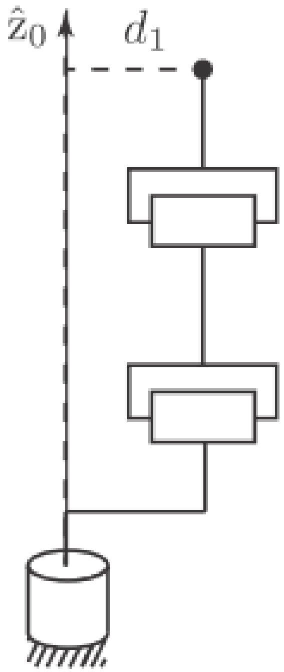
Source: Modern Robotics

If $p_x, p_y \neq 0$

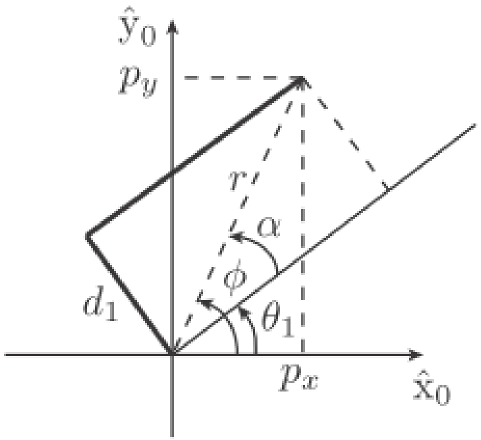
$$\theta_1 = \text{atan2}(p_y, p_x) \qquad \theta_1 = \text{atan2}(p_y, p_x) + \pi$$

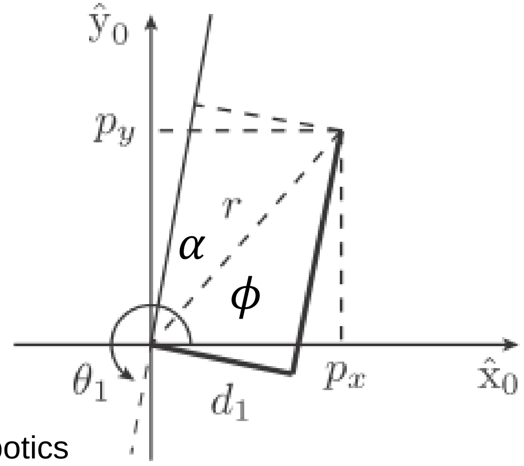
Singularity when $p_x, p_y = 0$, infinite solutions for θ_1.

6R PUMA-Type robot: analytical e.g.

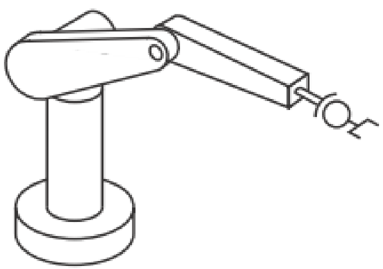


If there is shoulder displacement, $d_1 \neq 0$,





Source: Modern Robotics

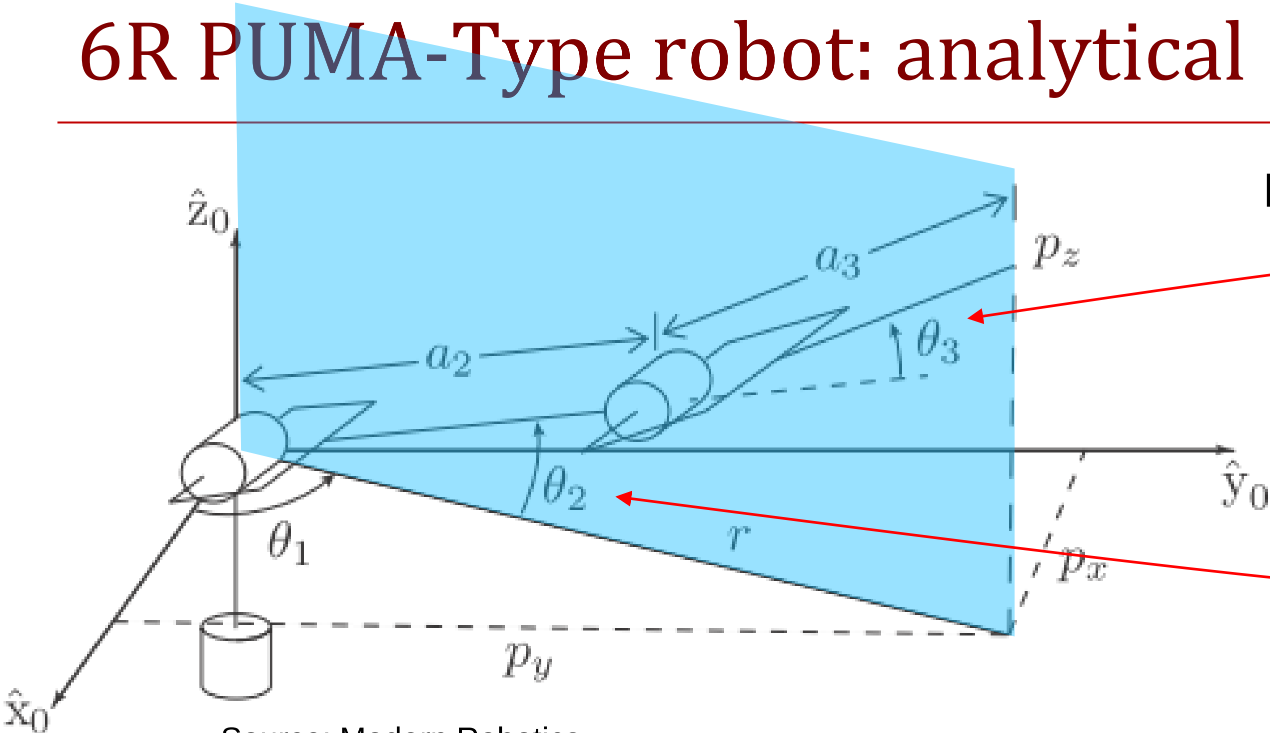


$$\theta_1 = \phi - \alpha$$
$$\phi = \text{atan2}(p_y, p_x)$$
$$\alpha = \text{atan2}\left(d_1, \sqrt{r^2 - {d_1}^2}\right)$$

$$\theta_1 = \pi + \phi + \alpha$$
$$\phi = \text{atan2}(p_y, p_x)$$
$$\alpha = \text{atan2}\left(-\sqrt{r^2 - {d_1}^2}, d_1\right)$$

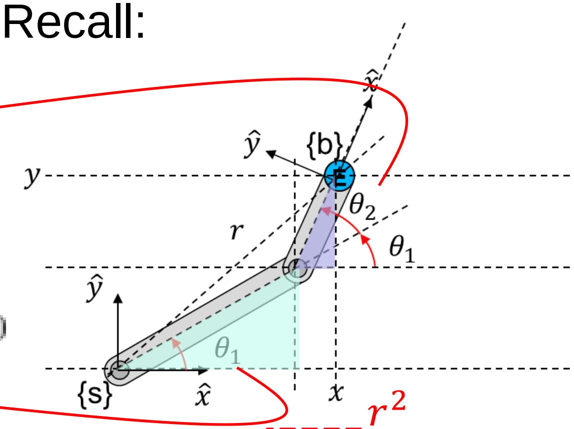
Source: Modern Robotics

6R PUMA-Type robot: analytical e.g.



Source: Modern Robotics

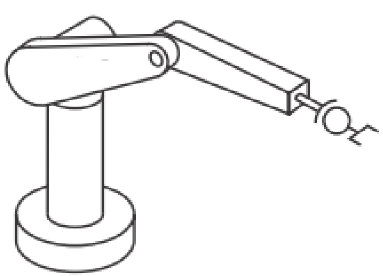
Recall:



$$\theta_2 = \cos^{-1}\left(\frac{x^2 + y^2 - L_1{}^2 - L_2{}^2}{2L_1L_2}\right)$$

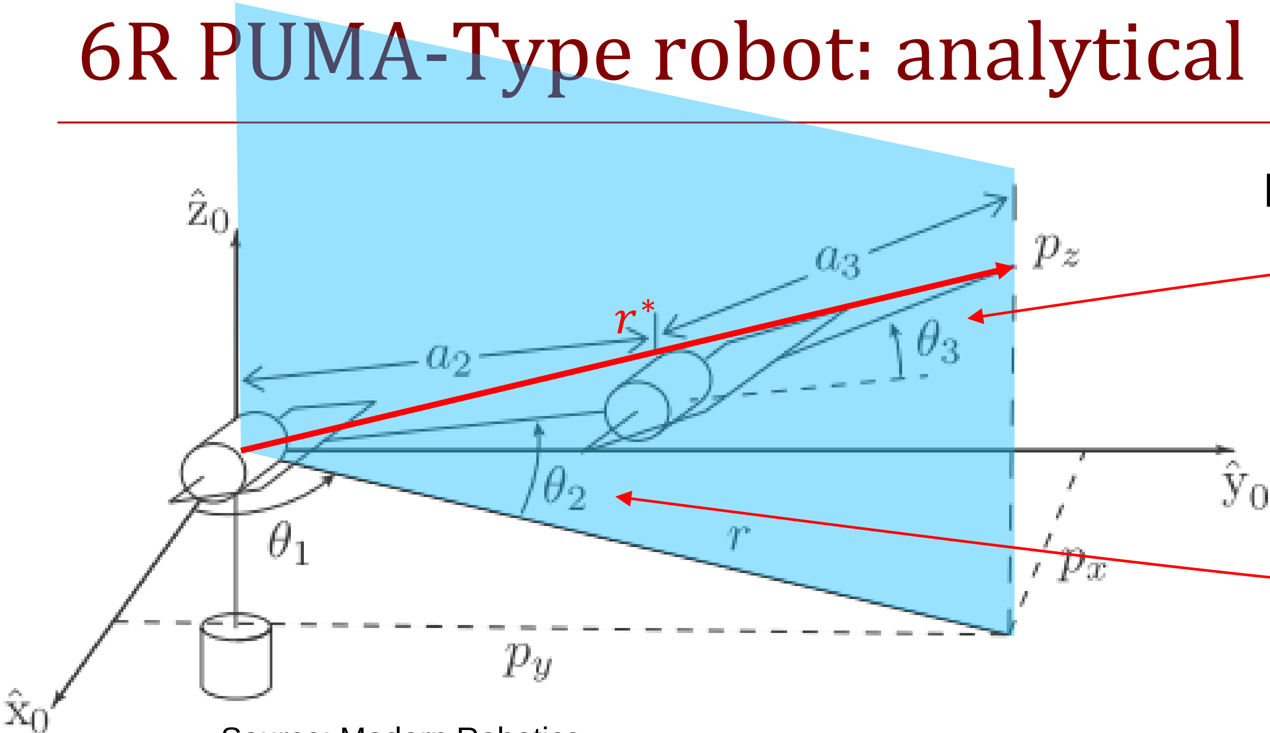
Solving for θ_2 and θ_3 is similar to solving the 2R planar IK on $r - z_0$ plane. Adapting the solution for 2R planar robot, we have:

$$\theta_3 = \cos^{-1}\left(\frac{r^{*2} - a_2{}^2 - a_3{}^2}{2a_2a_3}\right)$$



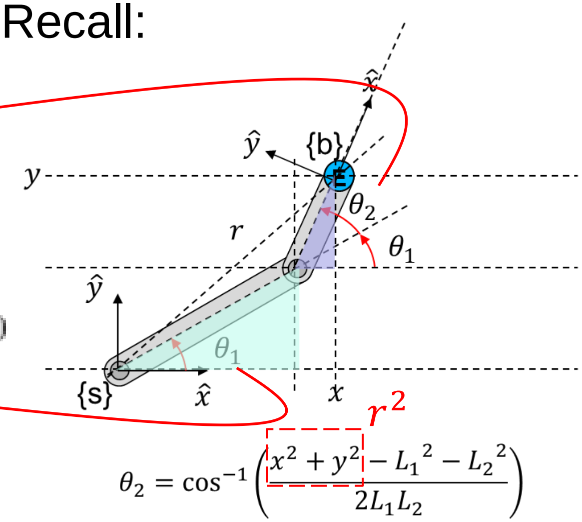
Likewise, we can adapt accordingly for to determine θ_2.

6R PUMA-Type robot: analytical e.g.



Source: Modern Robotics

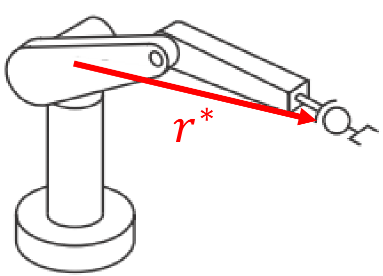
Recall:



$$r^{*2} = p_x{}^2 + p_y{}^2 - d_1{}^2 + p_z{}^2$$

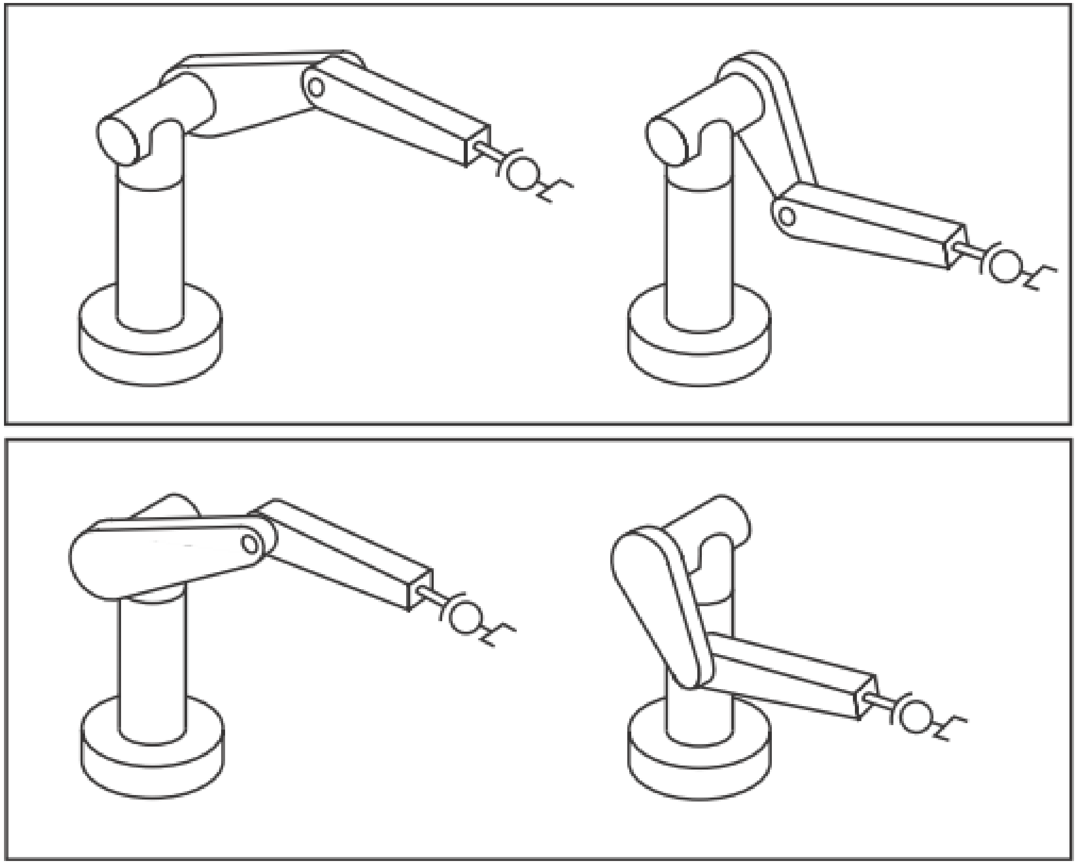
Solving for θ_2 and θ_3 is similar to solving the 2R planar IK on $r - z_0$ plane. Adapting the solution for 2R planar robot, we have:

$$\theta_3 = \cos^{-1}\left(\frac{r^{*2} - a_2{}^2 - a_3{}^2}{2a_2a_3}\right)$$



Likewise, we can adapt accordingly for to determine θ_2.

6R PUMA-Type robot: analytical e.g.



Source: Modern Robotics

Four possible inverse (position) kinematics solutions for the 6R PUMA-type arm with shoulder offset

6R PUMA-Type robot: analytical e.g.

Recall: for a desired pose of the end-effector T_{sd}, the IK problem is to find solution $\theta \in \mathbb{R}^6$ satisfying $T(\theta) = T_{sd}$.

Note $T_{sd} = \begin{bmatrix} R_d & \boldsymbol{p}_d \\ \boldsymbol{0} & 1 \end{bmatrix}$, where $\boldsymbol{p}_d$ is the desired position and R_d is the desired orientation of the end-effector.

Once we have found θ_1, θ_2, θ_3 for $\boldsymbol{p}_d$, we can solve the **inverse orientation** problem by **algebra**. We use the FK equation.

$$T(\theta) = T_{sd} = e^{[\mathcal{S}_1]\theta_1} e^{[\mathcal{S}_2]\theta_2} e^{[\mathcal{S}_3]\theta_3} e^{[\mathcal{S}_4]\theta_4} e^{[\mathcal{S}_5]\theta_5} e^{[\mathcal{S}_6]\theta_6} M$$
$$e^{-[\mathcal{S}_3]\theta_3} e^{-[\mathcal{S}_2]\theta_2} e^{-[\mathcal{S}_1]\theta_1} T_{sd} M^{-1} = e^{[\mathcal{S}_4]\theta_4} e^{[\mathcal{S}_5]\theta_5} e^{[\mathcal{S}_6]\theta_6}$$
$$L = e^{[\mathcal{S}_4]\theta_4} e^{[\mathcal{S}_5]\theta_5} e^{[\mathcal{S}_6]\theta_6}$$

Notice the left-hand side are now known, defined as L. We can solve for θ_4, θ_5, θ_6.

6R PUMA-Type robot: analytical e.g.

Recall: for a desired pose of the end-effector T_{sd}, the IK problem is to find solution $\theta \in \mathbb{R}^6$ satisfying $T(\theta) = T_{sd}$.

Note $T_{sd} = \begin{bmatrix} R_d & \boldsymbol{p}_d \\ \mathbf{0} & 1 \end{bmatrix}$, where $\boldsymbol{p}_d$ is the desired position and R_d is the desired orientation of the end-effector.

Alternatively, knowing the screw axes of $\mathcal{S}_4$, $\mathcal{S}_5$, $\mathcal{S}_6$, we can form the rotation matrix. For the 6R PUMA robot, the ω-components are

$$\omega_4 = (0,0,1),$$
$$\omega_5 = (0,1,0),$$
$$\omega_6 = (1,0,0).$$

Which results in a combined rotation of $Rot(\hat{z},\theta_4)Rot(\hat{y},\theta_5)Rot(\hat{x},\theta_6)$. We can solve for θ_4, θ_5, θ_6.

$$Rot(\hat{z},\theta_4)Rot(\hat{y},\theta_5)Rot(\hat{x},\theta_6) = R_d$$

Numerical inverse kinematics

Forward kinematics gives pose as a function of the joint variables:

$$\xi = f(\theta)$$

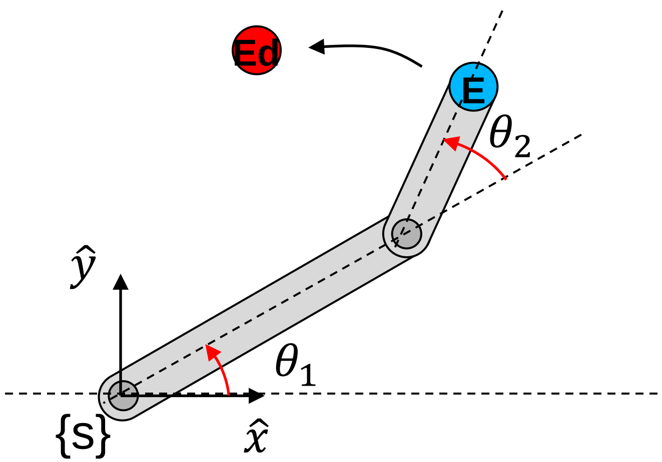
E.g.

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) \\ L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2) \end{pmatrix}$$

For more complex robots, it may not be possible to solve the FK equations analytically to obtain the desired joint variable values in a given IK problem.

An alternative approach is to solve it using **iterative numerical approach**.

Numerical inverse kinematics



Try different values of θ so that

$$\xi \to \xi_d$$

This is done in a systematic way by changing the values of θ in the direction to minimize the error of

$$\xi_d - \xi = \xi_d - f(\theta) \to 0$$

We expect if we have found $\theta = \theta_d$, the error

$$\xi_d - f(\theta_d) = 0$$

This is basically what optimization algorithms do: find the set of parameters (variable values) that minimize (or maximize) the cost or error (or objective or utility).

$$\theta^* = \arg\min_{\theta}\big(\xi_d - f(\theta)\big)$$

If we manage to minimize the error to zero, $\theta^* = \theta_d$.

Newton-Raphson method

We basically want to find $\theta = \theta_d$ such that

$$\xi_d - f(\theta\quad) = 0$$

I.e. we want to solve the above equation, in other words to **find the root** to the above equation.

Newton-Raphson root finding method is a method we can use to solve an equation $g(\theta) = 0$ numerically provided g is differentiable.

It gives an effective way to change the value of θ such that the error equation will head towards zero.

Newton-Raphson method: scalar example

Consider a single coordinate pose (scalar), $\xi_d = x_d$. We want to find the root of below equation numerically:

$$x_d - f(\theta) = 0$$

x_d is the desired value, $f(\theta)$ is the actual function value at θ.

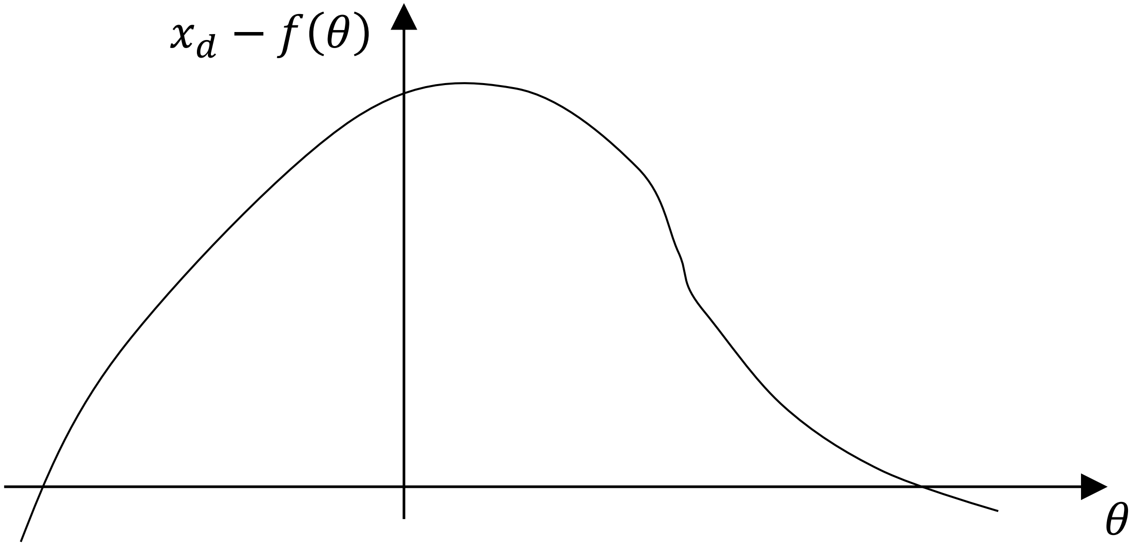
Naively, we can try all possible values of θ until we reach a point of $x_d - f(\theta) = 0$. At this point, $\theta = \theta_d$.

Newton-Raphson method: scalar example

Consider a single coordinate pose (scalar), $\xi_d = x_d$. We want to find the root of below equation numerically:

$$x_d - f(\theta) = 0$$

x_d is the desired value, $f(\theta)$ is the actual function value at θ.



Naively, we can try all possible values of θ until we reach a point of $x_d - f(\theta) = 0$. At this point, $\theta = \theta_d$.

We effectively drawn a graph of $x_d - f(\theta)$ against θ.

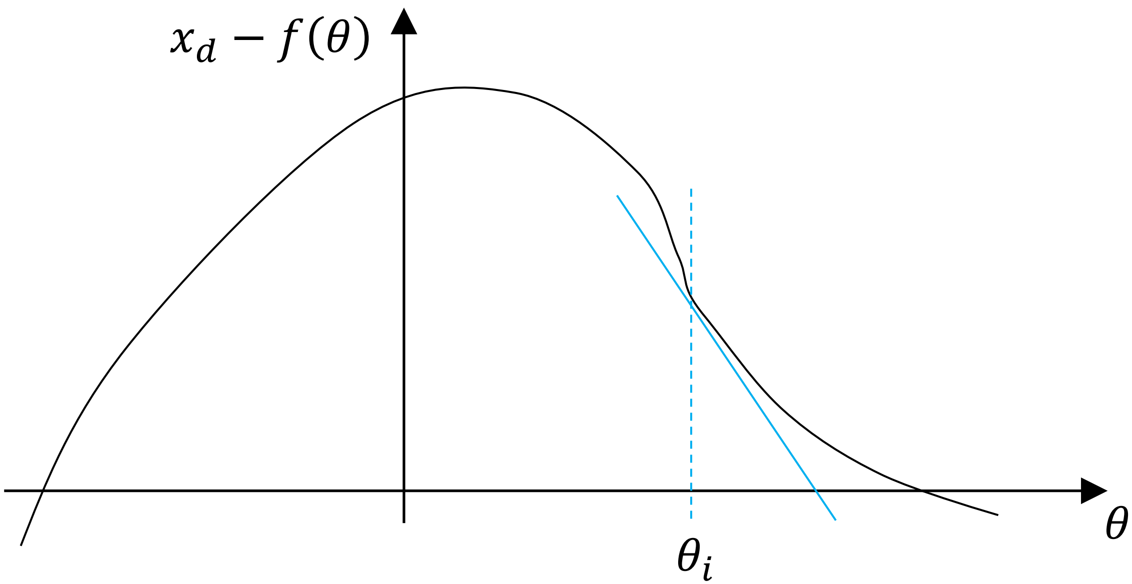
What range of θ should we try?

Newton-Raphson method: scalar example

Consider a single coordinate pose (scalar), $\xi_d = x_d$. We want to find the root of below equation numerically:

$$x_d - f(\theta) = 0$$

x_d is the desired value, $f(\theta)$ is the actual function value at θ.



If $x_d - f(\theta)$ is differentiable, we can find its gradient at every value of θ.

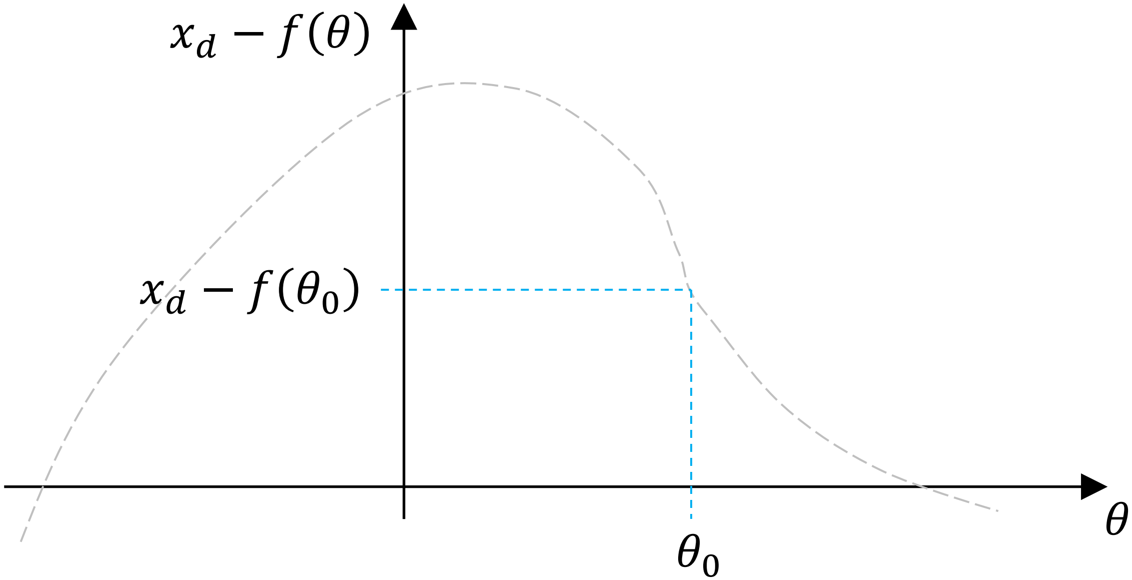
We can use this gradient or slope to guide us in choosing the value of θ in the next iteration.

Newton-Raphson method: scalar example

Consider a single coordinate pose (scalar), $\xi_d = x_d$. We want to find the root of below equation numerically:

$$x_d - f(\theta) = 0$$

x_d is the desired value, $f(\theta)$ is the actual function value at θ.



Let's not try all values of θ randomly.

Let's start with any value, an initial guess, θ_0.

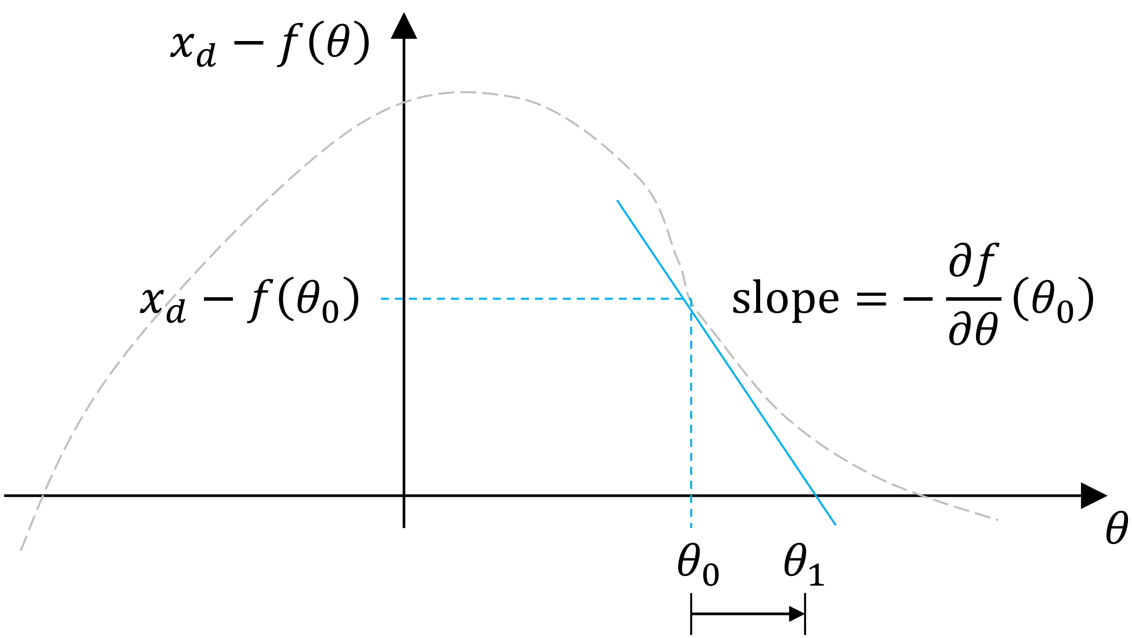
Compute FK $f(\theta_0)$ and error function $x_d - f(\theta_0)$.

Newton-Raphson method: scalar example

Consider a single coordinate pose (scalar), $\xi_d = x_d$. We want to find the root of below equation numerically:

$$x_d - f(\theta) = 0$$

x_d is the desired value, $f(\theta)$ is the actual function value at θ.



Not zero?

We need to try another value of θ. We will use the slope at θ_0 to help decide the next value of θ to try.

Compute the slope of $x_d - f(\theta_0)$. Since x_d is a constant

$$\text{slope} = -\frac{\partial f}{\partial \theta}(\theta_0)$$

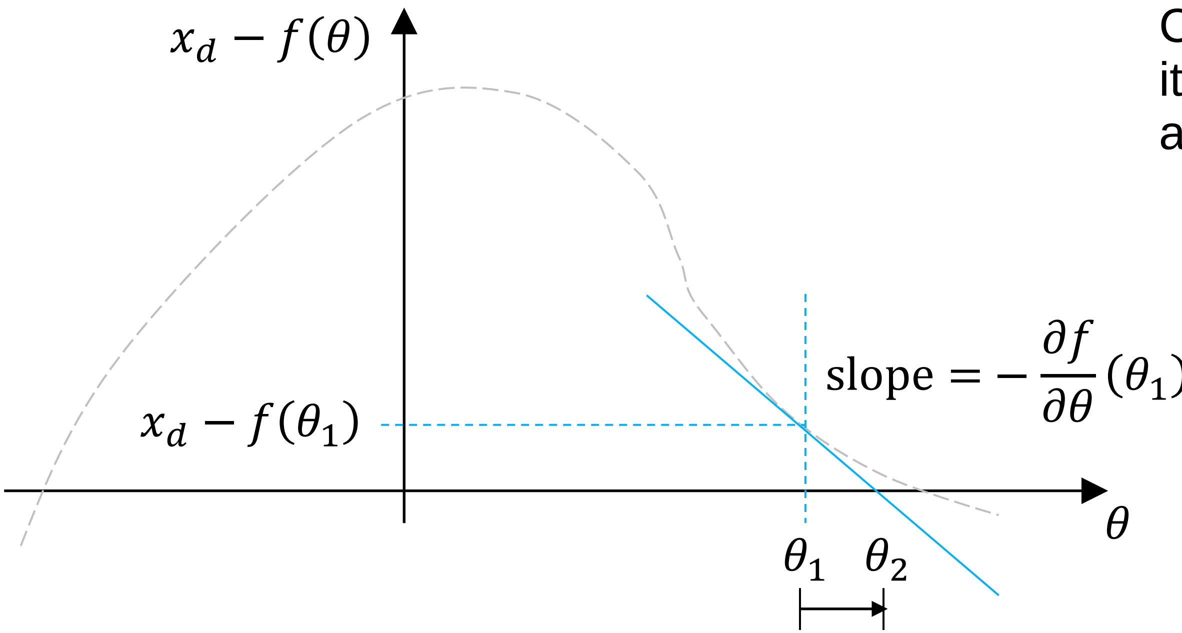
$$\Delta\theta = \left(\frac{\partial f}{\partial \theta}(\theta_0)\right)^{-1} \left(x_d - f(\theta_0)\right)$$

Newton-Raphson method: scalar example

Consider a single coordinate pose (scalar), $\xi_d = x_d$. We want to find the root of below equation numerically:

$$x_d - f(\theta) = 0$$

x_d is the desired value, $f(\theta)$ is the actual function value at θ.



Continue the process to next iteration with θ_1 that leads to θ_2 and so on.

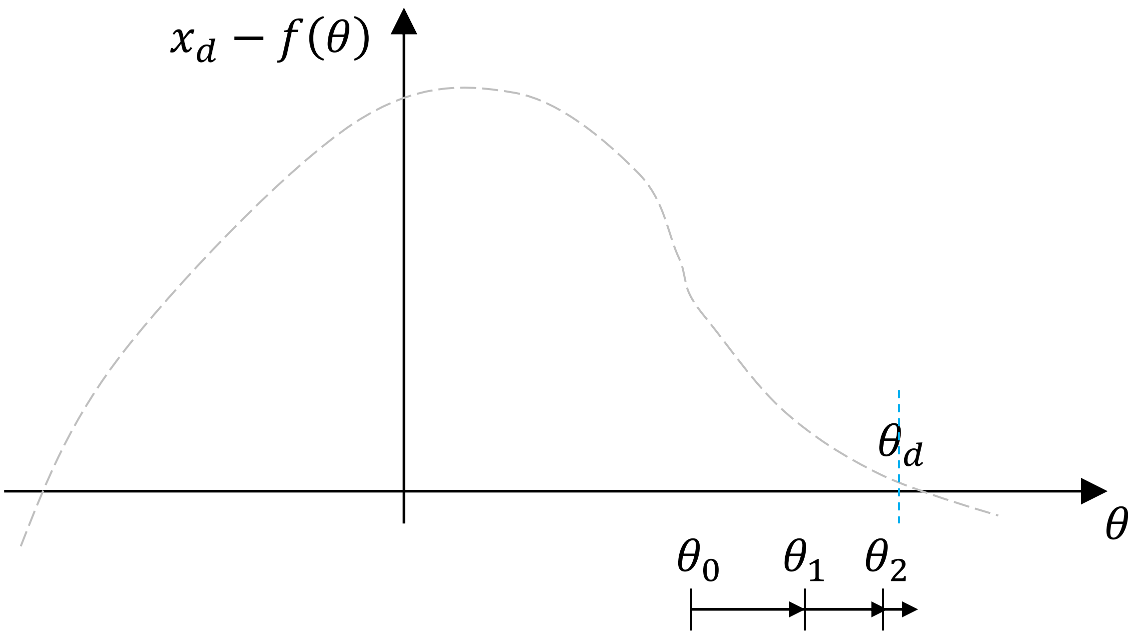
$$\Delta\theta = \left(\frac{\partial f}{\partial \theta}(\theta_1)\right)^{-1}\left(x_d - f(\theta_1)\right)$$

Newton-Raphson method: scalar example

Consider a single coordinate pose (scalar), $\xi_d = x_d$. We want to find the root of below equation numerically:

$$x_d - f(\theta) = 0$$

x_d is the desired value, $f(\theta)$ is the actual function value at θ.



Continue the process to next iteration with θ_1 that leads to θ_2 and so on.

Until we reach

$$x_d - f(\theta) = 0$$

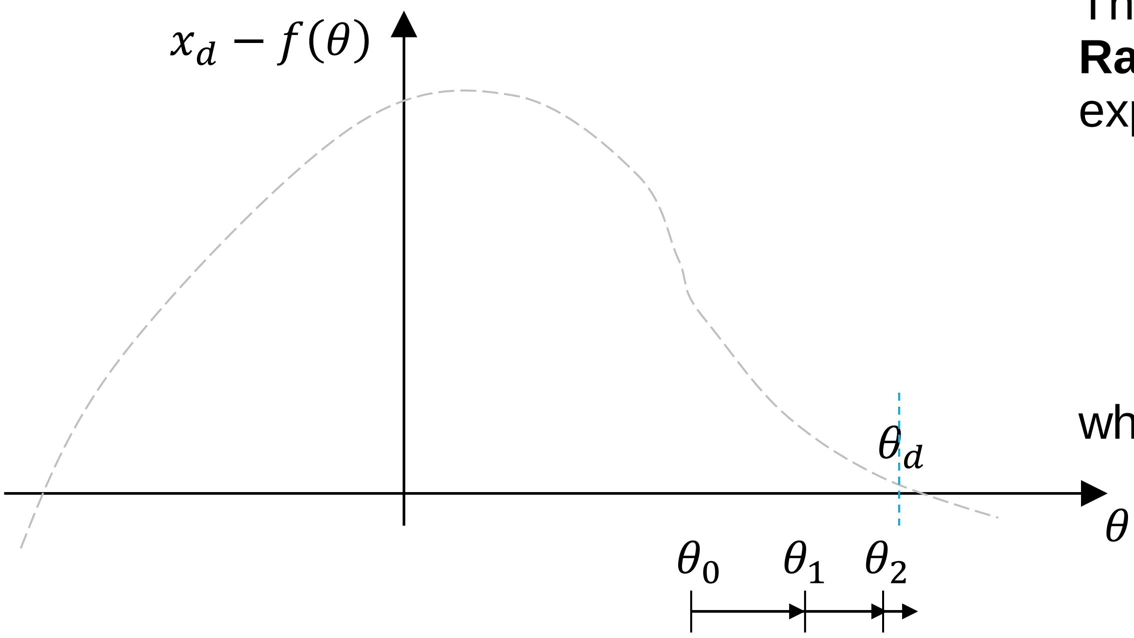
The value of θ would be the desired θ_d.

Newton-Raphson method: scalar example

Consider a single coordinate pose (scalar), $\xi_d = x_d$. We want to find the root of below equation numerically:

$$x_d - f(\theta) = 0$$

x_d is the desired value, $f(\theta)$ is the actual function value at θ.



The update of θ in **Newton-Raphson method** can be expressed as:

$$\theta^{i+1} = \theta^i - \left(\frac{\partial f}{\partial \theta}(\theta^i)\right)^{-1} g(\theta^i)$$

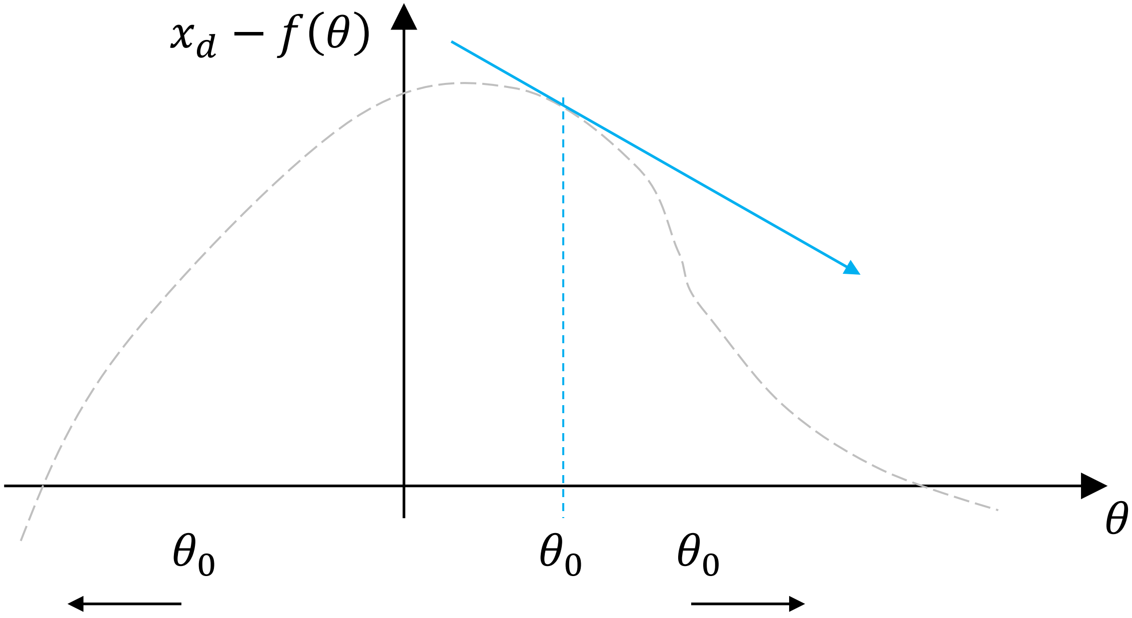
where $g(\theta^i) = x_d - f(\theta^i)$.

Newton-Raphson method: scalar example

Consider a single coordinate pose (scalar), $\xi_d = x_d$. We want to find the root of below equation numerically:

$$x_d - f(\theta) = 0$$

x_d is the desired value, $f(\theta)$ is the actual function value at θ.



The solution depends on initial guess of θ_0.

This guess should be near to a solution. Otherwise, the process may not converge.

Newton-Raphson method: vector

We generalize the formulation to non-scalar pose, i.e. $\xi = f(\theta)$ is a vector (not just an x coordinate). We can write the FK function $f(\theta)$ differentiable at θ^i as a **Taylor series**:

$$\xi = f(\theta) = f\left(\theta^i\right) + \frac{\partial f}{\partial \theta}\left(\theta^i\right)\left(\theta - \theta^i\right) + higher\ order\ terms$$

At $\theta = \theta_d$, we have:

$$\xi_d = f(\theta_d) = f\left(\theta^i\right) + \underbrace{\frac{\partial f}{\partial \theta}\left(\theta^i\right)}_{J\left(\theta^i\right)}\underbrace{\left(\theta_d - \theta^i\right)}_{\Delta\theta} + higher\ order\ terms$$

Recall $J(\theta) = \frac{\partial f(\theta)}{\partial \theta}$ → $J\left(\theta^i\right)$

Think of this as the FK function of the desired pose $\xi_d = f(\theta_d)$ at joint variables θ_d expressed as a function of current iteration joint variables θ^i and the slope $\frac{\partial f}{\partial \theta}\left(\theta^i\right)$ at this point.

We can rearrange and write above equation as below:

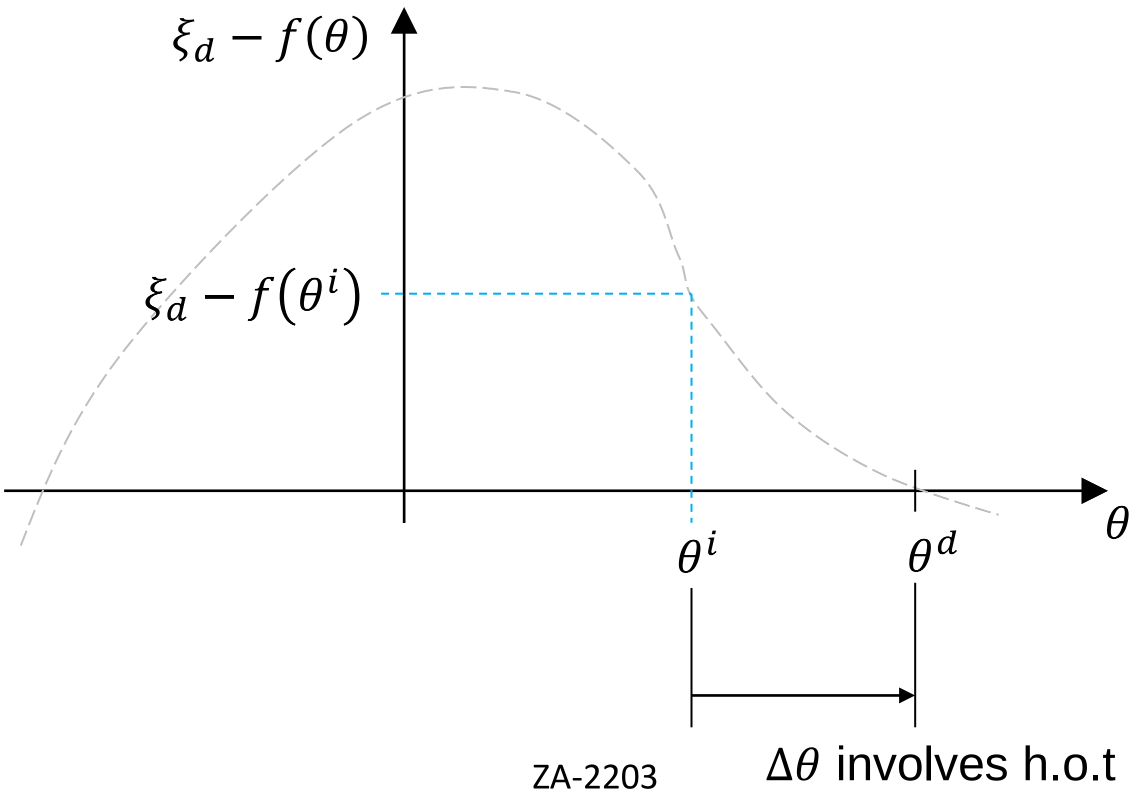
$$\xi_d - f\left(\theta^i\right) = J\left(\theta^i\right)\Delta\theta + higher\ order\ terms$$

Newton-Raphson method: vector

We can use the Taylor series to help us determine the $\Delta\theta$ to determine the next θ^{i+1}.

$$\xi_d - f(\theta^i) = J(\theta^i)\Delta\theta + higher\ order\ terms\ (h.o.t)$$

Using scalar ξ_d for illustration purpose.

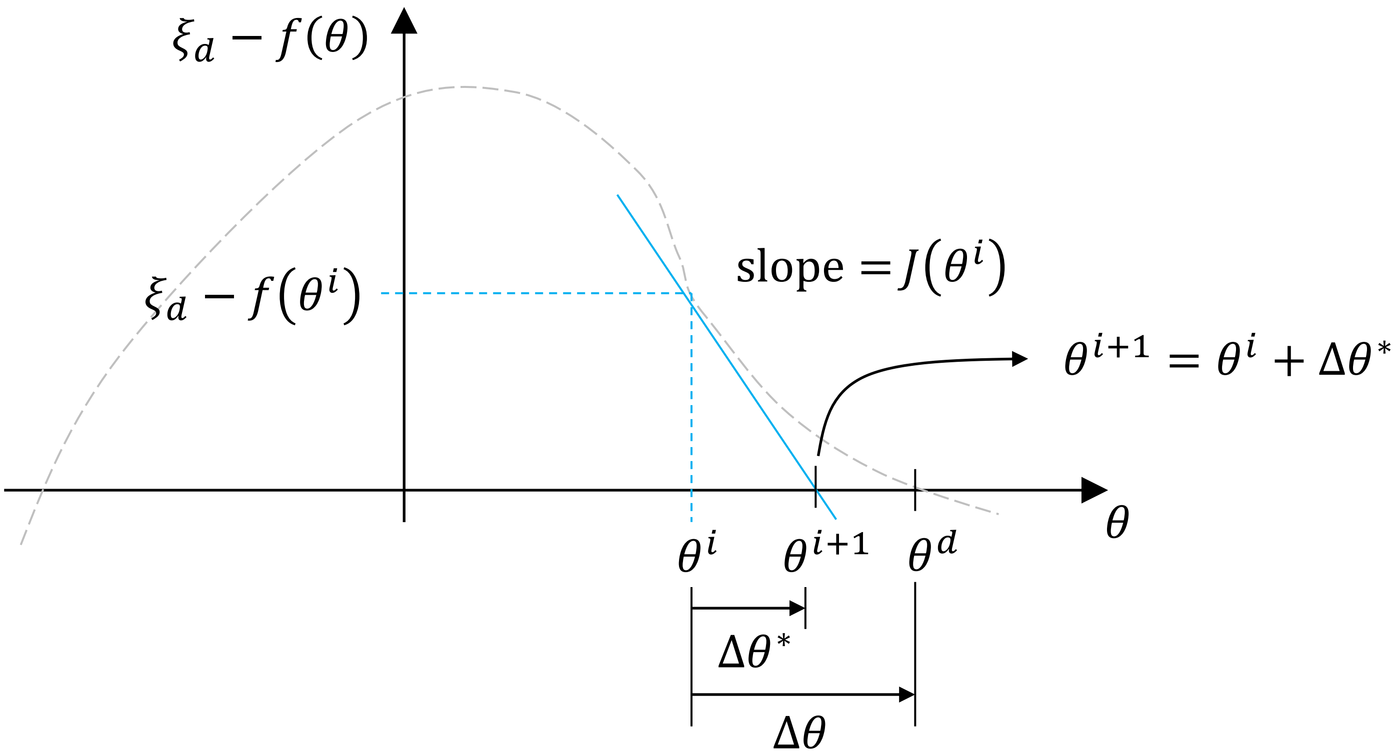


Newton-Raphson method: vector

If we ignore higher order terms, the $\Delta\theta$ we will obtain is not exactly $\theta_d - \theta^i$,

$$\xi_d - f(\theta^i) = J(\theta^i)\Delta\theta^*$$

where $\Delta\theta^* = \theta^{i+1} - \theta^i$.



Newton-Raphson method: vector

If we ignore higher order terms, the $\Delta\theta$ we will obtain is not exactly $\theta_d - \theta^i$,

$$\xi_d - f(\theta^i) = J(\theta^i)\Delta\theta^*$$

where $\Delta\theta^* = \theta^{i+1} - \theta^i$. Rearrange to determine $\Delta\theta^*$,

$$\Delta\theta^* = \frac{\xi_d - f(\theta^i)}{J(\theta^i)} = J^{-1}(\theta^i)\left(\xi_d - f(\theta^i)\right)$$

$J(\theta^i)$ may not always be invertible, e.g. if it is not a square matrix and at singularity point. We can use a mathematical tool, **matrix pseudoinverse** to obtain the inverse of $J(\theta^i)$ for such cases. It will also work if $J(\theta^i)$ is invertible. We donate pseudoinverse of $J(\theta^i)$ as $J^+(\theta^i)$.

$$\Delta\theta^* = J^+(\theta^i)\left(\xi_d - f(\theta^i)\right)$$

We can update $\theta^{i+1} = \theta^i + \Delta\theta^*$ iteratively until we reach $\theta^{i+1} = \theta_d$ where $\xi_d - f(\theta_d) = 0$.

$$\theta^{i+1} = \theta^i + J^+(\theta^i)\left(\xi_d - f(\theta^i)\right)$$

Newton-Raphson numerical IK: ξ_d steps

Given a desired pose expressed in minimal coordinates (usually position and orientation) $\xi_d \in \mathbb{R}^m$ and the FK of the robot $\xi = f(\theta)$, we solve the IK problem to determine the required joint variable values $\theta_d \in \mathbb{R}^n$ by following the steps below:

1. Set $i = 0$, make an initial guess $\theta^0 \in \mathbb{R}^n$. Decide the value of ϵ (max error).
2. Compute error $e = \xi_d - f(\theta^0)$.
3. While $||e|| > \epsilon$ for some small value of ϵ:
    3.1 Compute next value $\theta^{i+1} = \theta^i + J^+(\theta^i)e$
    3.2 $i = i + 1$
    3.3 Compute error $e = \xi_d - f(\theta^{i+1})$

Newton-Raphson numerical IK: T_{sd}

In the case when the desired pose of the end-effector is given in the form of a **homogeneous transformation matrix** T_{sd}, we can modify the Newton-Raphson numerical IK steps accordingly.

$T_{sd} \in \mathbb{R}^{4\times4}$

$T(\theta) = e^{[\mathcal{S}_1]\theta_1} \cdots M$

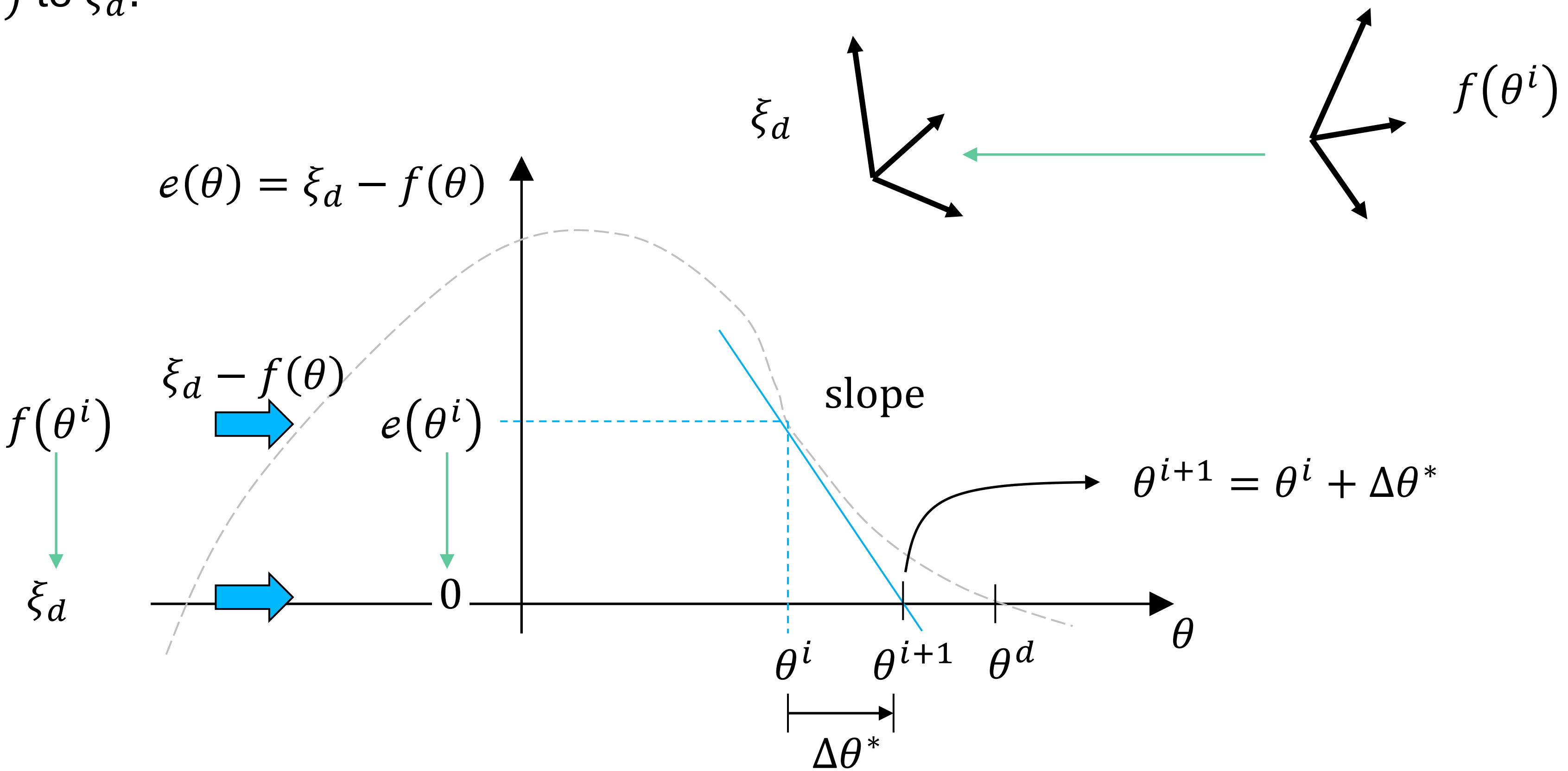
Given a desired pose expressed in minimal coordinates (usually position and orientation) $\xi_d \in \mathbb{R}^m$ and the FK of the robot $\xi = f(\theta)$, we solve the IK problem to determine the required joint variable values $\theta_d \in \mathbb{R}^n$ by following the steps below:

1. Set $i = 0$, make an initial guess $\theta^0 \in \mathbb{R}^n$. Decide the value of ϵ.
2. Compute error $e = \xi_d - f(\theta^0)$.
3. While $\|e\| > \epsilon$ for some small value of ϵ:
    - 3.1 Compute next value $\theta^{i+1} = \theta^i + J^+(\theta^i)e$
    - 3.2 $i = i + 1$
    - 3.3 Compute error $e = \xi_d - f(\theta^{i+1})$

~~$e = T_{sd} - T(\theta)$~~

Representing error as velocity in unit time

We can interpret $e(\theta^i)$ as the **change in pose** required to get from $f(\theta^i)$ to ξ_d. Given $\Delta\theta^*$ is a change in θ in **one unit time** step to change from $f(\theta^i)$ to ξ_d, we can interpret $e(\theta^i)$ as the required **change in pose in one unit time** from $f(\theta^i)$ to ξ_d.

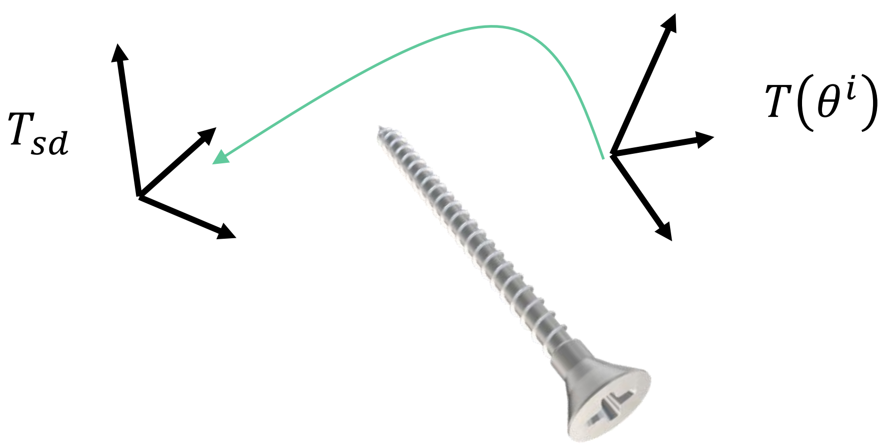


Representing error as velocity in unit time

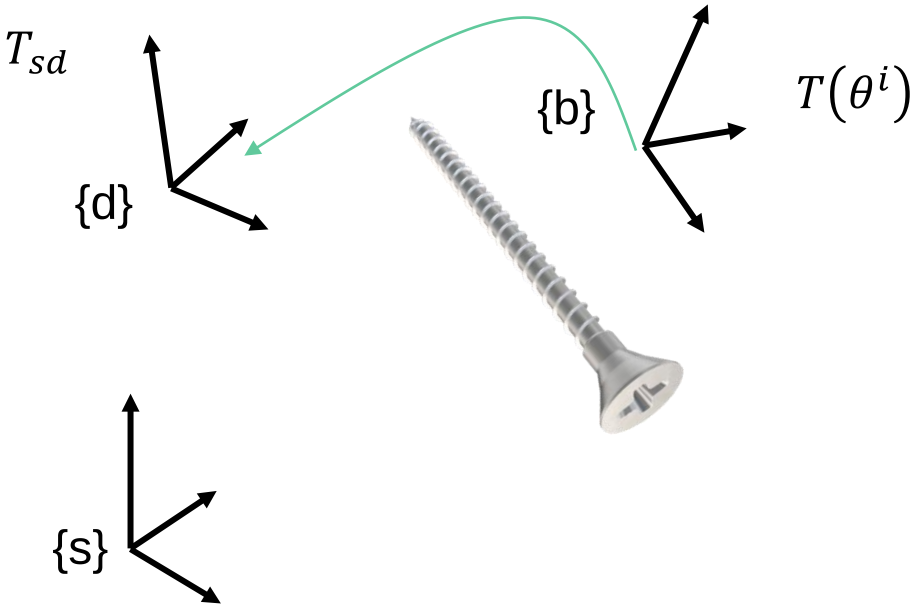
We can interpret $e(\theta^i)$ as the required **change in pose in one unit time** from $f(\theta^i)$ to ξ_d.

We can think of representing $e(\theta^i)$ as the **velocity that if the end-effector follow it in unit time**, it will change from $T(\theta^i)$ ($\equiv f(\theta^i)$) to T_{sd} ($\equiv \xi_d$). In other words, we can find the **twist** $\mathcal{V}$ that will change the effector pose from $T(\theta^i)$ to T_{sd} in unit time.

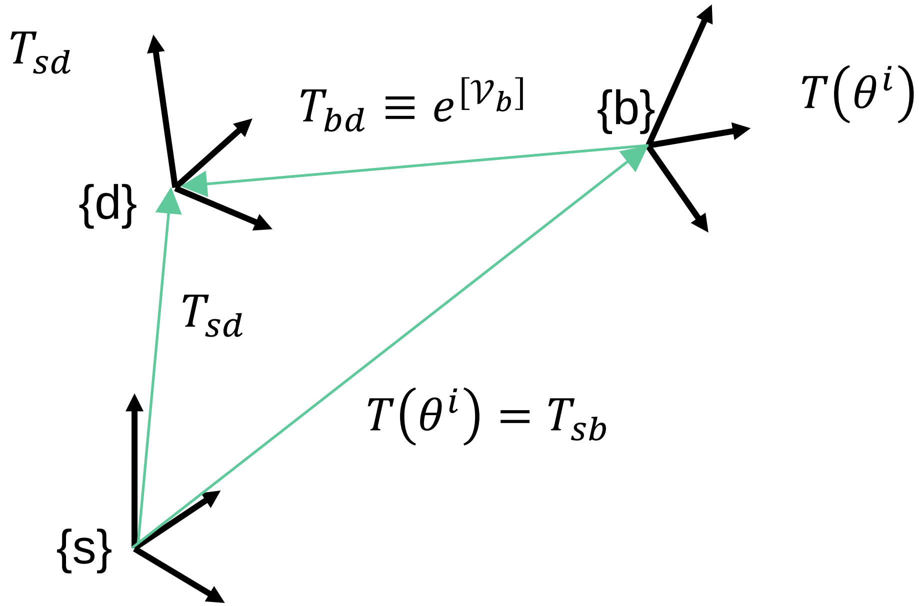
Recall twist $\mathcal{V}$ is the velocity of the pose (angular and linear combined).



Representing error as velocity in unit time



Representing error as velocity in unit time



$$T_{bd} = T_{bs}T_{sd} = T_{sb}{}^{-1}T_{sd}$$

$$[\mathcal{V}_b] = \log e^{[\mathcal{V}_b]} = \log T_{bd} = \log\left(T_{sb}{}^{-1}T_{sd}\right) = \log\left(T^{-1}\left(\theta^i\right)T_{sd}\right)$$

Newton-Raphson numerical IK: T_{sd} steps

Given a desired pose expressed in minimal coordinates (usually position and orientation) $T_{sd} \in \mathbb{R}^{4\times 4}$ and the FK of the robot $T(\theta) = e^{[\mathcal{S}_1]\theta_1} \cdots M$ or $T(\theta) = Me^{[\mathcal{B}_1]\theta_1} \cdots$, we solve the IK problem to determine the required joint variable values $\theta_d \in \mathbb{R}^n$ by following the steps below:

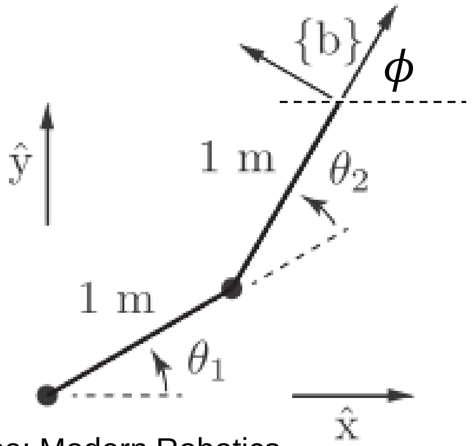
1. Set $i = 0$, make an initial guess $\theta^0 \in \mathbb{R}^n$. Decide the values of ϵ_w and ϵ_v (max errors).
2. Compute error $[\mathcal{V}_b] = \log(T^{-1}(\theta^0)T_{sd})$.
3. While $\|\omega_b\| > \epsilon_w$ or $\|v_b\| > \epsilon_v$ for some small value of ϵ_w and ϵ_v:
    3.1 Compute next value $\theta^{i+1} = \theta^i + {J_b}^+\left(\theta^i\right)\mathcal{V}_b$
    3.2 $i = i + 1$
    3.3 Compute error $[\mathcal{V}_b] = \log\left(T^{-1}\left(\theta^{i+1}\right)T_{sd}\right)$

$\mathcal{V}_b \in \mathbb{R}^6$ is the body twist. $J_b \in \mathbb{R}^{6\times n}$ is the body Jacobian.
Alternative, we can use space twist $\mathcal{V}_s$ and space Jacobian J_s.

Initial guess for Newton-Raphson

- The result of Newton-Raphson depends on the initial guess.
- The initial guess θ^0 should be as close to a solution θ_d as possible in order for the numerical IK to converge.
- We can start the robot from an initial home configuration where both the actual end-effector configuration and the joint angles are known and ensuring that the requested end-effector position T_{sd} changes slowly (moves in small steps) relative to the frequency of the calculation of the inverse kinematics.
- Then, for the rest of the robot's run, the calculated θ_d at the previous arm move serves as the initial guess θ^0 for the new T_{sd} at the next arm move.

Planar 2R robot: numerical IK example



Source: Modern Robotics

Given desired end-effector position is

$$(x, y) = (0.366\,m, 1.366\,m)$$

and end-effector orientation is

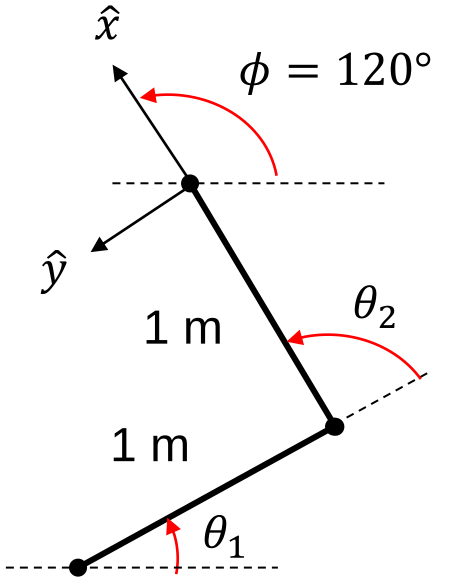
$$\phi = 120°$$

Determine $\theta_d = (\theta_1, \theta_2)$ to achieve the above pose. Use transformation matrix for your solution.

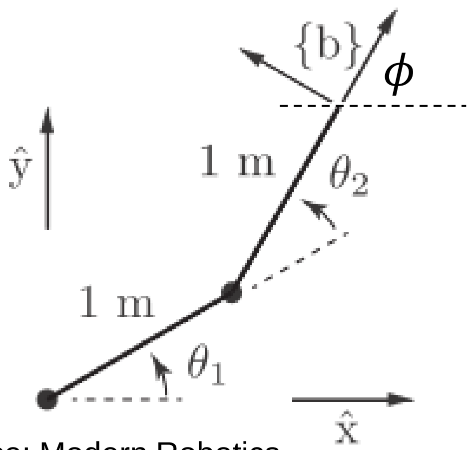
We need to have T_{sd} and the FK equation.

$$T_{sd} = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \text{ where } R = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } p = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

$$T_{sd} = \begin{bmatrix} -0.5 & -0.866 & 0 & 0.366 \\ 0.866 & -0.5 & 0 & 1.366 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Planar 2R robot: numerical IK example



Source: Modern Robotics

Given desired end-effector position is

$$(x, y) = (0.366\, m, 1.366\, m)$$

and end-effector orientation is

$$\phi = 120°$$

Determine $\theta_d = (\theta_1, \theta_2)$ to achieve the above pose. Use transformation matrix for your solution.

We need to have T_{sd} and the FK equation.

$$M = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathcal{B}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 2 \\ 0 \end{bmatrix}, \quad \mathcal{B}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Recall:

$$[\mathcal{B}] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix}$$

$$[\omega] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$$

$$[\mathcal{B}_1] = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad [\mathcal{B}_2] = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Planar 2R robot: numerical IK example

Desired pose: $T_{sd} = \begin{bmatrix} -0.5 & -0.866 & 0 & 0.366 \\ 0.866 & -0.5 & 0 & 1.366 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Forward kinematics: $T(\theta) = Me^{[\mathcal{B}_1]\theta_1}e^{[\mathcal{B}_2]\theta_2}$

$$M = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [\mathcal{B}_1] = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad [\mathcal{B}_2] = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Given $T_{sd} \in \mathbb{R}^{4\times4}$ and the FK of the robot $T(\theta) = Me^{[\mathcal{B}_1]\theta_1}e^{[\mathcal{B}_2]\theta_2}$.

1. Set $i = 0$, make an initial guess $\theta^0 \in \mathbb{R}^n$. Decide the values of ϵ_w and ϵ_v (max errors).
2. Compute error $[\mathcal{V}_b] = \log(T^{-1}(\theta^0)T_{sd})$.
3. While $\|\omega_b\| > \epsilon_w$ or $\|v_b\| > \epsilon_v$ for some small value ofϵ_w and ϵ_v:
    3.1 Compute next value $\theta^{i+1} = \theta^i + J_b{}^+(\theta^i)\mathcal{V}_b$
    3.2 $i = i + 1$
    3.3 Compute error $[\mathcal{V}_b] = \log\left(T^{-1}\left(\theta^{i+1}\right)T_{sd}\right)$

Planar 2R robot: numerical IK example

1. Set $i = 0$, make an initial guess $\theta^0 \in \mathbb{R}^n$. Decide the values of ϵ_w and ϵ_v (max errors).
Let $\theta^0 = (0,30°)$, and
allowable error in angular $\epsilon_w = 0.001\ rad\ (0.057°)$ and linear $\epsilon_v = 10^{-4}\ m\ (100\ microns)$.

2. Compute FK $T(\theta^0) = Me^{[\mathcal{B}_1]\theta_1}e^{[\mathcal{B}_2]\theta_2}$, then compute error $[\mathcal{V}_b] = \begin{bmatrix}\omega_b \\ v_b\end{bmatrix} = \log(T^{-1}(\theta^0)T_{sd})$.
3. If $\|\omega_b\| > \epsilon_w$ or $\|v_b\| > \epsilon_v$, then update $\theta^{i+1} = \theta^i + J_b{}^+(\theta^i)\mathcal{V}_b$.
Iterate until error is less than the set values.

The above computations are best done with computer.

$\boldsymbol{i}$	$(\boldsymbol{\theta_1}, \boldsymbol{\theta_2})$	$(\boldsymbol{x}, \boldsymbol{y})$	$\boldsymbol{\mathcal{V}_b} = (\boldsymbol{\omega_{zb}}, \boldsymbol{v_{xb}}, \boldsymbol{v_{yb}})$	$\|\boldsymbol{\omega_b}\|$	$\|\boldsymbol{v_b}\|$
0	$(0.00, 30.00°)$	$(1.866, 0.500)$	$(1.571, 0.498, 1.858)$	1.571	1.924
1	$(34.23°, 79.18°)$	$(0.429, 1.480)$	$(0.115, -0.074, 0.108)$	0.115	0.131
2	$(29.98°, 90.22°)$	$(0.363, 1,364)$	$(-0.004, 0.000, -0.004)$	0.004	0.004
3	$(30.00°, 90.00°)$	$(0.366, 1.366)$	$(0.000, 0.000, 0.000)$	0.000	0.000

Summary (1/2)

- **Inverse Kinematics** (IK) is finding the required joint variable values θ_d to achieve a given desired pose (position and orientation) ξ_d (expressed as vector of coordinates) or T_{sd} (expressed as homogeneous transformation matrix).
- There may be **0** (not reachable), **1** (at the boundary of the workspace) or **multiple solutions**.
- IK can be solved **analytically** or **numerically**.
- **Analytical** approach uses geometry and solves FK equations in $\xi = f(\theta)$.
- Analytical approach can find all possible solutions so that we can choose which solution to use (e.g. righty, lefty, elbow-up, elbow-down).

Summary (2/2)

- However, analytical approach may not always be possible, or may be very difficult for complex mechanisms.
- **Numerical** approach uses Newton-Raphson method to iteratively predict the value of θ^{i+1} until it found the$\theta^{i+1} = \theta_d$ such that the error $\xi_d - f\left(\theta^i\right)$ (or twist $\mathcal{V}_b$) is zero.
- **Newton-Raphson numerical IK** problem can be represented in the forms for **vector** or **homogeneous transformation matrix**.
- The **initial guess** θ^0 for the Newton-Raphson numerical IK needs to be close to a solution.
- If we move the robotic arm in **small steps**, in each step, the previous known configuration of θ can serve as a good initial guess θ^0 for the next move step.

Reading List

- Read Chapter 6 of Modern Robotics

To Do List

- Watch Chapter 6 videos of Modern Robotics on Coursera, or on YouTube https://www.youtube.com/playlist?list=PLggLP4f-rq02vX0OQQ5vrCxbJrzamYDfx