CZECH TECHNICAL UNIVERSITY IN PRAGUE

Robotics: Inverse Kinematics

Vladimír Petrík

vladimir.petrik@cvut.cz

16.10.2023

Kinematics tasks

- Forward kinematics (FK)
  - how to compute end-effector pose from the configuration
  - $\boldsymbol{x} = f_{\mathsf{fk}}(\boldsymbol{q})$
  - $\boldsymbol{x}$ is expressed in task-space, *i.e.* $SE(2)$, $SE(3)$, or $\mathbb{R}^2$, $\mathbb{R}^3$ for position only
  - $\boldsymbol{q} \in \mathbb{R}^N$ is configuration (joint space)

Kinematics tasks

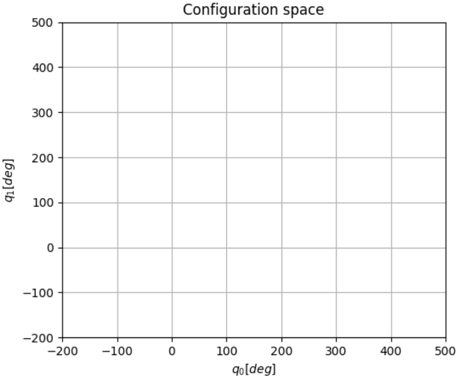
- Forward kinematics (FK)
  - how to compute end-effector pose from the configuration
  - $\boldsymbol{x} = f_{\mathsf{fk}}(\boldsymbol{q})$
  - $\boldsymbol{x}$ is expressed in task-space, *i.e.* $SE(2)$, $SE(3)$, or $\mathbb{R}^2$, $\mathbb{R}^3$ for position only
  - $\boldsymbol{q} \in \mathbb{R}^N$ is configuration (joint space)
- Differential kinematics
  - relates end-effector velocity to joint velocities
  - $\dot{\boldsymbol{x}} = J(\boldsymbol{q})\dot{\boldsymbol{q}}$

Kinematics tasks

- Forward kinematics (FK)
  - how to compute end-effector pose from the configuration
  - $\boldsymbol{x} = f_{\mathsf{fk}}(\boldsymbol{q})$
  - $\boldsymbol{x}$ is expressed in task-space, *i.e.* $SE(2)$, $SE(3)$, or $\mathbb{R}^2$, $\mathbb{R}^3$ for position only
  - $\boldsymbol{q} \in \mathbb{R}^N$ is configuration (joint space)
- Differential kinematics
  - relates end-effector velocity to joint velocities
  - $\dot{\boldsymbol{x}} = J(\boldsymbol{q})\dot{\boldsymbol{q}}$
- Inverse kinematics (IK)
  - how to compute robot configuration(s) for given end-effector configuration
  - $\boldsymbol{q} \in f_{\mathsf{ik}}(\boldsymbol{x})$

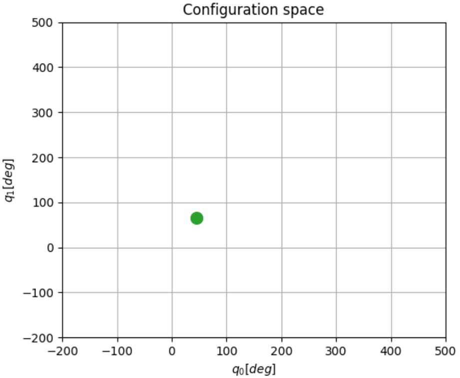
Example: Analytical IK for RR structure

- Task space: translation of end-effector, $\boldsymbol{x} \in \mathbb{R}^2$
- Configuration (joint) space: $\boldsymbol{q} \in \mathbb{R}^2$



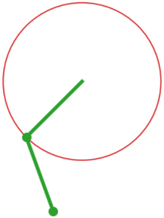
Example: Analytical IK for RR structure

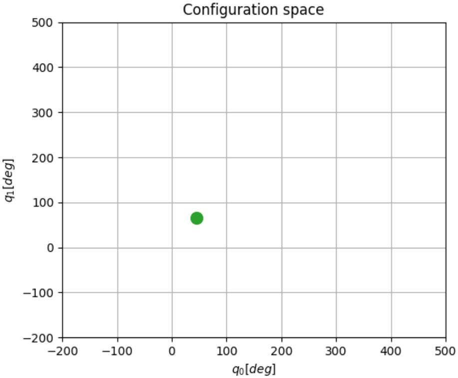
- Task space: translation of end-effector, $\boldsymbol{x} \in \mathbb{R}^2$
- Configuration (joint) space: $\boldsymbol{q} \in \mathbb{R}^2$



Example: Analytical IK for RR structure

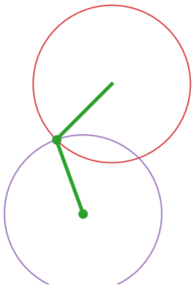
- Task space: translation of end-effector, $\boldsymbol{x} \in \mathbb{R}^2$
- Configuration (joint) space: $\boldsymbol{q} \in \mathbb{R}^2$

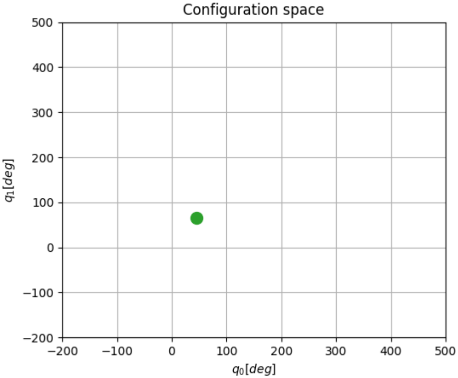




Example: Analytical IK for RR structure

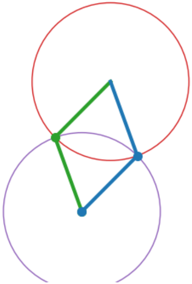
- Task space: translation of end-effector, $\boldsymbol{x} \in \mathbb{R}^2$
- Configuration (joint) space: $\boldsymbol{q} \in \mathbb{R}^2$

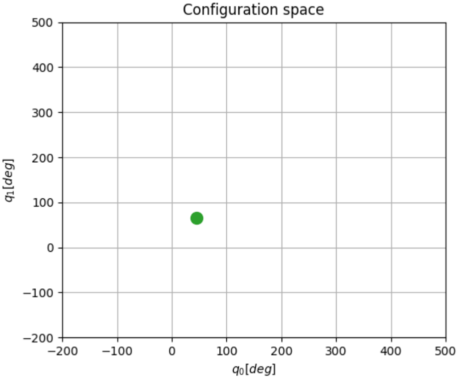




Example: Analytical IK for RR structure

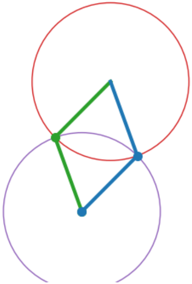
- Task space: translation of end-effector, $\boldsymbol{x} \in \mathbb{R}^2$
- Configuration (joint) space: $\boldsymbol{q} \in \mathbb{R}^2$

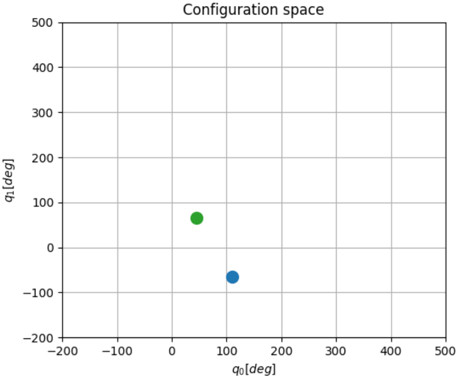




Example: Analytical IK for RR structure

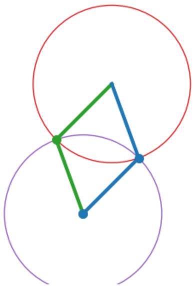
- Task space: translation of end-effector, $\boldsymbol{x} \in \mathbb{R}^2$
- Configuration (joint) space: $\boldsymbol{q} \in \mathbb{R}^2$

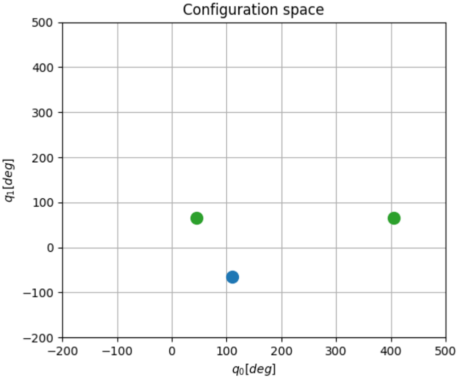




Example: Analytical IK for RR structure

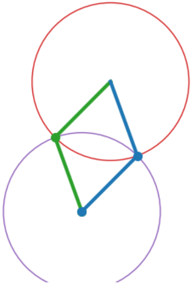
- Task space: translation of end-effector, $\boldsymbol{x} \in \mathbb{R}^2$
- Configuration (joint) space: $\boldsymbol{q} \in \mathbb{R}^2$

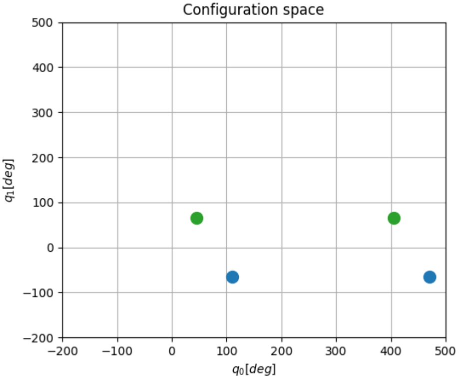




Example: Analytical IK for RR structure

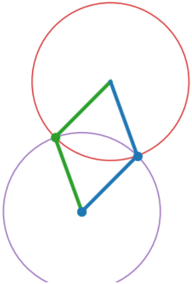
- Task space: translation of end-effector, $\boldsymbol{x} \in \mathbb{R}^2$
- Configuration (joint) space: $\boldsymbol{q} \in \mathbb{R}^2$

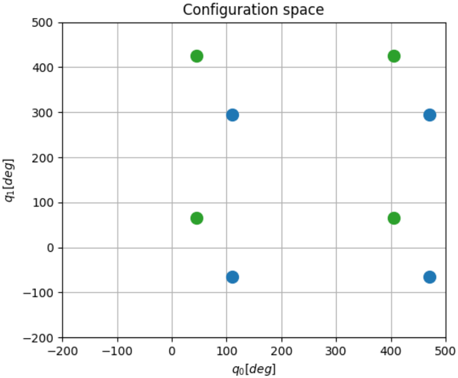




Example: Analytical IK for RR structure

- Task space: translation of end-effector, $\boldsymbol{x} \in \mathbb{R}^2$
- Configuration (joint) space: $\boldsymbol{q} \in \mathbb{R}^2$





Example: Analytical IK for RR structure

- Task space: translation of end-effector, $\boldsymbol{x} \in \mathbb{R}^2$
- Configuration (joint) space: $\boldsymbol{q} \in \mathbb{R}^2$
- Algorithm:
  - Compute position of all joints and end-effector
  - No solution, 1 solution, 2 solutions, or ∞ solutions
  - For each solution, compute joint configurations $\theta_i = \operatorname{atan2}(y, x) + 2k\pi$, $k \in \mathbb{Z}$
    $\begin{pmatrix} x & y \end{pmatrix}^\top = \boldsymbol{t}_{i,i+1}$, *i.e.* translation part of $T_{i,i+1}$

Numerical optimization

- Analytical solution is often unavailable
  - solution does not exist and we seek for the closest approximate
  - infinite solutions exist and we seek for configuration w.r.t. given criteria

Numerical optimization

- Analytical solution is often unavailable
  - solution does not exist and we seek for the closest approximate
  - infinite solutions exist and we seek for configuration w.r.t. given criteria
- We can use generic numerical algorithm, that iteratively reduce error
- **Newton–Raphson** method

Numerical optimization

- Analytical solution is often unavailable
  - solution does not exist and we seek for the closest approximate
  - infinite solutions exist and we seek for configuration w.r.t. given criteria
- We can use generic numerical algorithm, that iteratively reduce error
- **Newton–Raphson** method
  - solve $g(\theta) = 0, g : \mathbb{R} \to \mathbb{R}$

Numerical optimization

- Analytical solution is often unavailable
  - solution does not exist and we seek for the closest approximate
  - infinite solutions exist and we seek for configuration w.r.t. given criteria
- We can use generic numerical algorithm, that iteratively reduce error
- **Newton–Raphson** method
  - solve $g(\theta) = 0, g : \mathbb{R} \rightarrow \mathbb{R}$
  - taylor expansion of $g(\theta)$ at θ^0:
    $g(\theta) = g(\theta^0) + \frac{\partial g}{\partial \theta}(\theta^0)(\theta - \theta^0) +$ higher-order terms

Numerical optimization

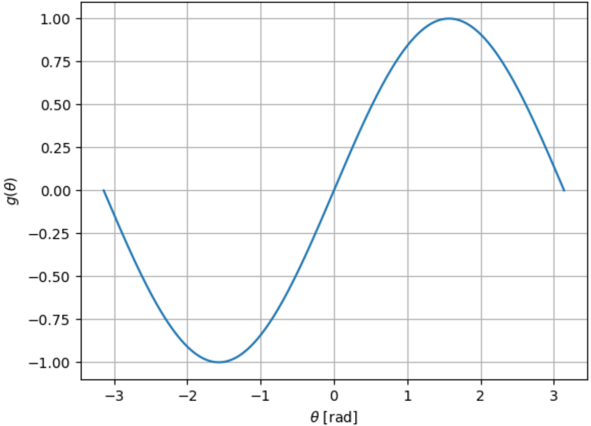
- Analytical solution is often unavailable
  - solution does not exist and we seek for the closest approximate
  - infinite solutions exist and we seek for configuration w.r.t. given criteria
- We can use generic numerical algorithm, that iteratively reduce error
- **Newton–Raphson** method
  - solve $g(\theta) = 0, g : \mathbb{R} \to \mathbb{R}$
  - taylor expansion of $g(\theta)$ at θ^0:
    $g(\theta) = g(\theta^0) + \frac{\partial g}{\partial \theta}(\theta^0)(\theta - \theta^0) +$ higher-order terms
  - set $g(\theta) = 0$, ignore higher-order terms, and solve for θ:
    $\theta \approx \theta^0 - \left(\frac{\partial g}{\partial \theta}(\theta^0)\right)^{-1} g(\theta^0)$

Numerical optimization

- Analytical solution is often unavailable
  - solution does not exist and we seek for the closest approximate
  - infinite solutions exist and we seek for configuration w.r.t. given criteria
- We can use generic numerical algorithm, that iteratively reduce error
- **Newton–Raphson** method
  - solve $g(\theta) = 0, g : \mathbb{R} \to \mathbb{R}$
  - taylor expansion of $g(\theta)$ at θ^0:
    $g(\theta) = g(\theta^0) + \frac{\partial g}{\partial \theta}(\theta^0)(\theta - \theta^0) +$ higher-order terms
  - set $g(\theta) = 0$, ignore higher-order terms, and solve for θ:
    $\theta \approx \theta^0 - \left(\frac{\partial g}{\partial \theta}(\theta^0)\right)^{-1} g(\theta^0)$
  - as we ignore higher-order terms, we need to iterate:
    $\theta^{k+1} = \theta^k - \left(\frac{\partial g}{\partial \theta}(\theta^k)\right)^{-1} g(\theta^k)$

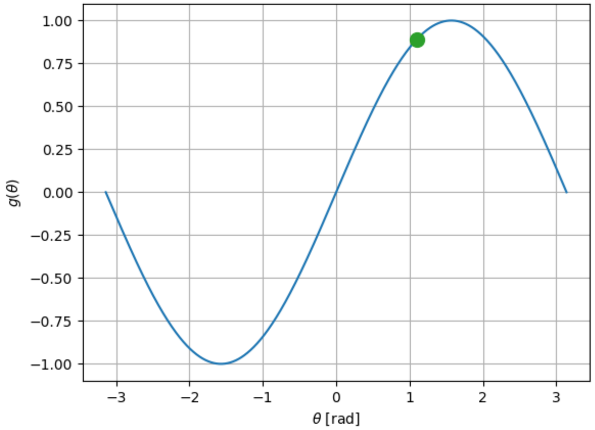
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.1$



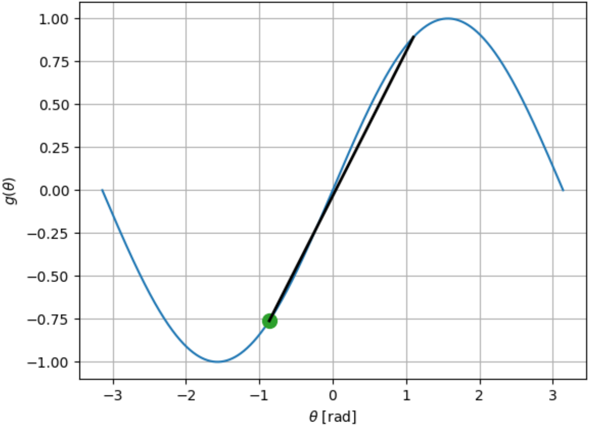
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.1$



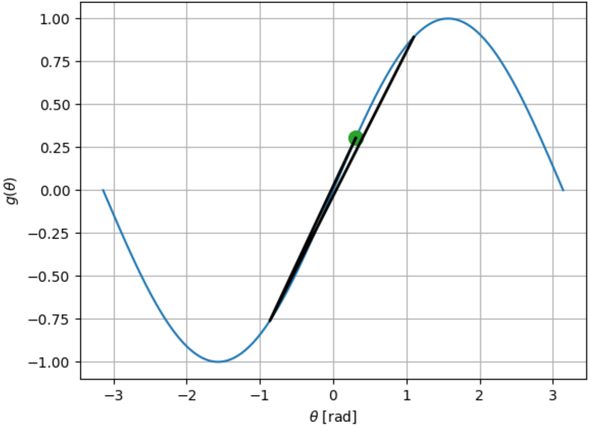
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.1$



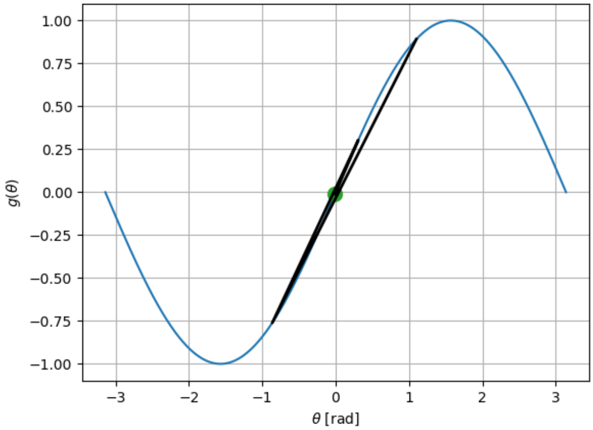
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.1$



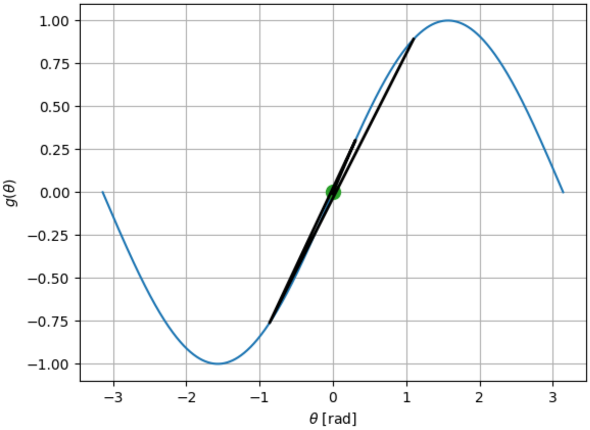
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.1$



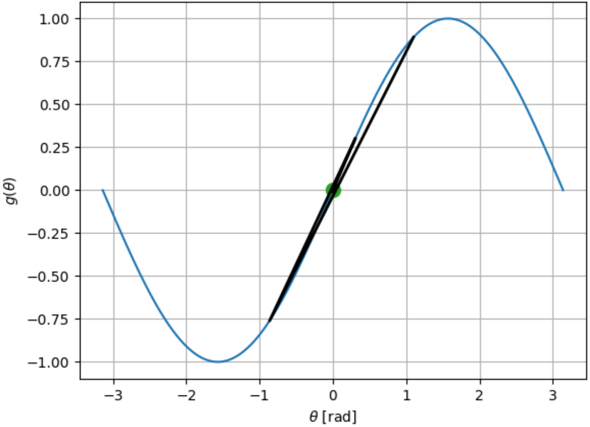
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.1$



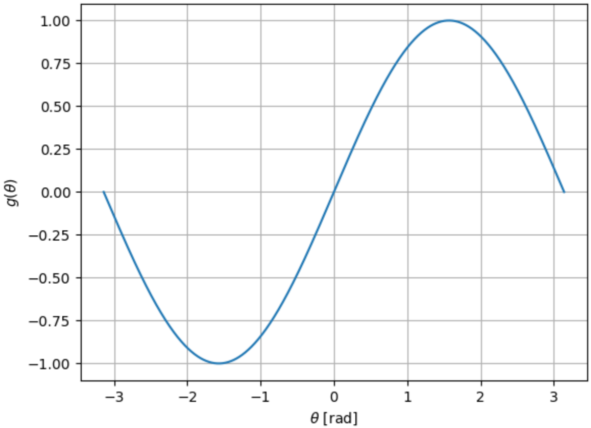
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.1$



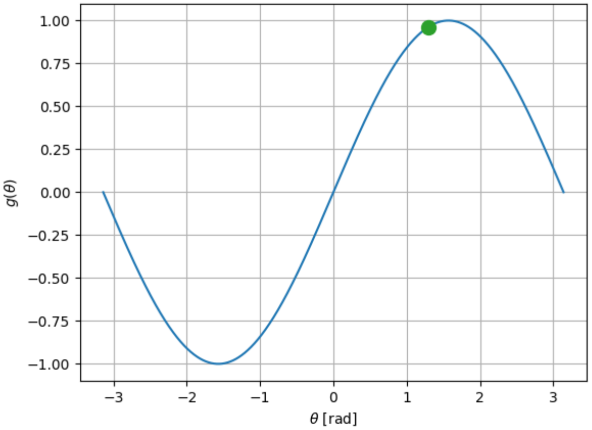
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.3$
- Quality of the solution depends on the initial guess



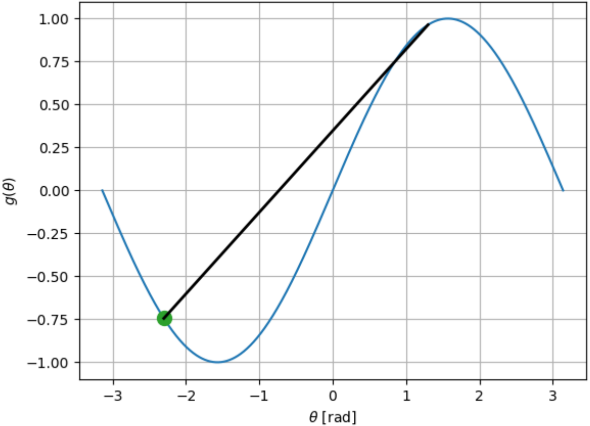
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.3$
- Quality of the solution depends on the initial guess



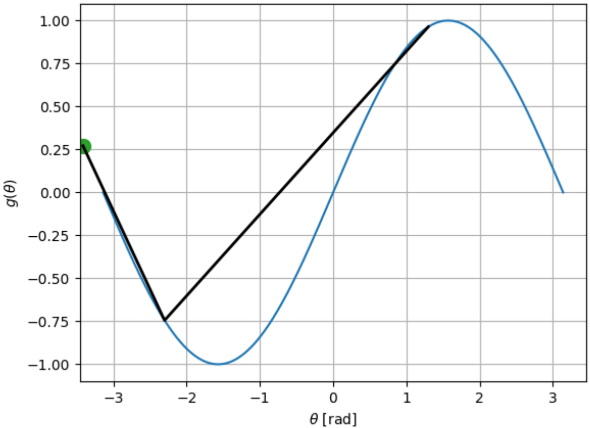
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.3$
- Quality of the solution depends on the initial guess



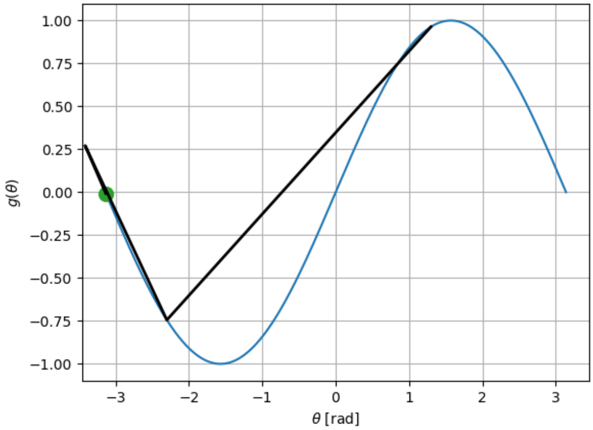
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.3$
- Quality of the solution depends on the initial guess



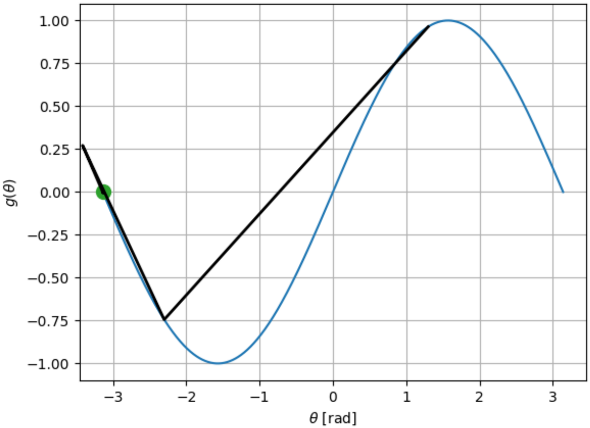
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.3$
- Quality of the solution depends on the initial guess



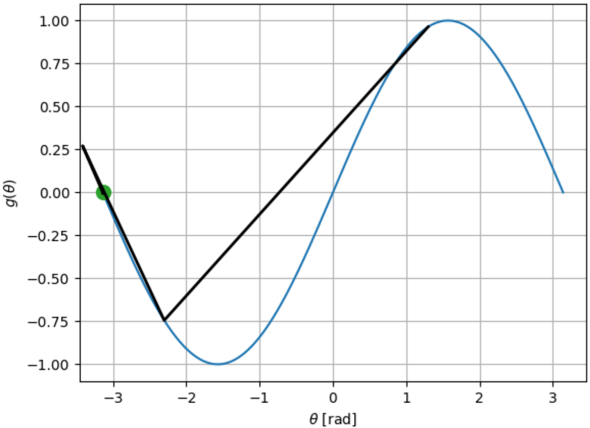
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.3$
- Quality of the solution depends on the initial guess



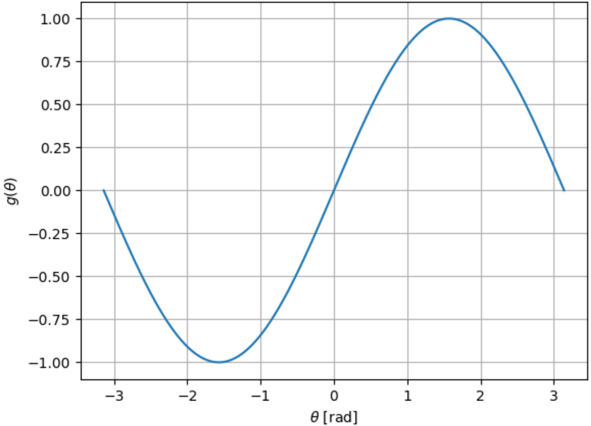
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.3$
- Quality of the solution depends on the initial guess



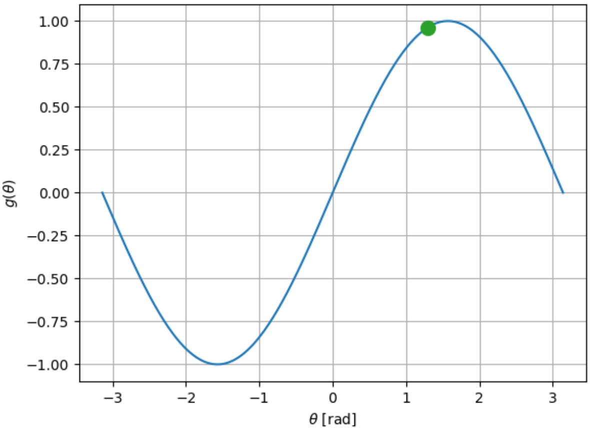
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.3$, $\alpha = 0.5$
- $\theta^{k+1} = \theta^k - \alpha \left(\frac{\partial g}{\partial \theta}(\theta^k) \right)^{-1} g(\theta^k)$



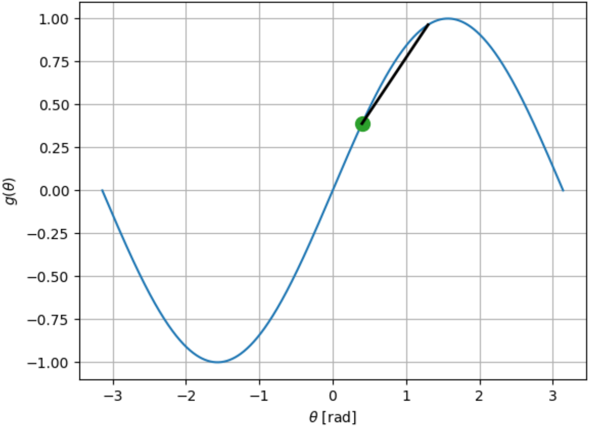
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.3$, $\alpha = 0.5$
- $\theta^{k+1} = \theta^k - \alpha \left(\frac{\partial g}{\partial \theta}(\theta^k) \right)^{-1} g(\theta^k)$



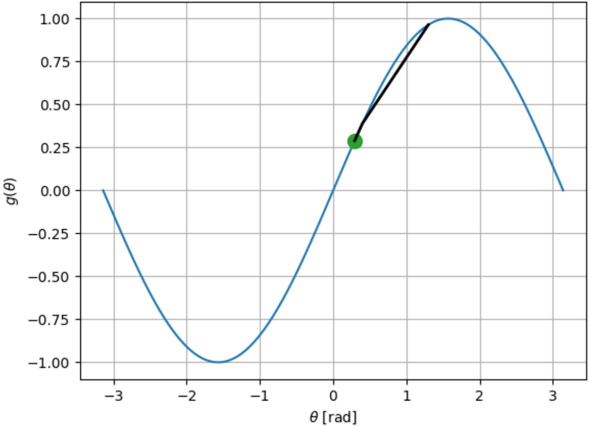
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.3$, $\alpha = 0.5$
- $\theta^{k+1} = \theta^k - \alpha \left(\frac{\partial g}{\partial \theta}(\theta^k) \right)^{-1} g(\theta^k)$



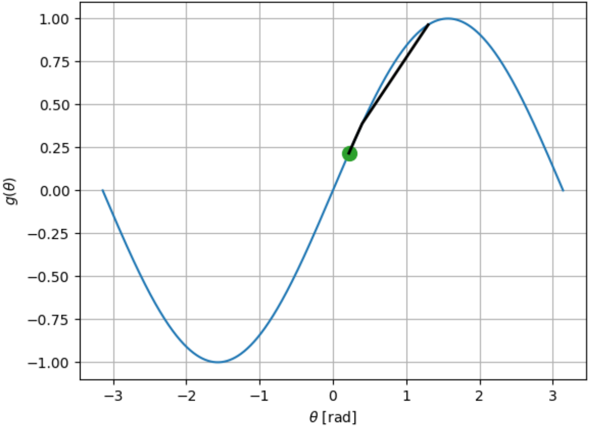
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.3$, $\alpha = 0.5$
- $\theta^{k+1} = \theta^k - \alpha \left(\frac{\partial g}{\partial \theta}(\theta^k)\right)^{-1} g(\theta^k)$



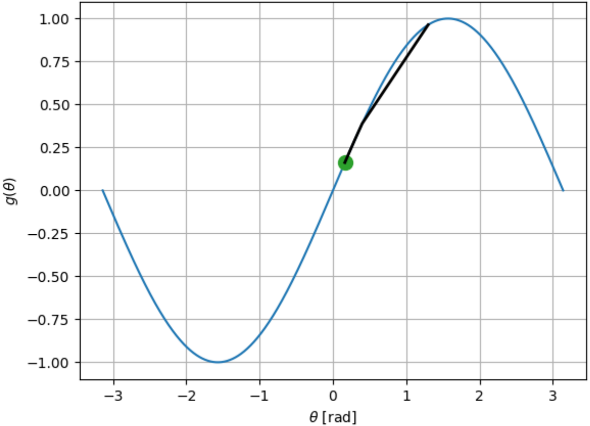
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.3$, $\alpha = 0.5$
- $\theta^{k+1} = \theta^k - \alpha \left(\frac{\partial g}{\partial \theta}(\theta^k) \right)^{-1} g(\theta^k)$



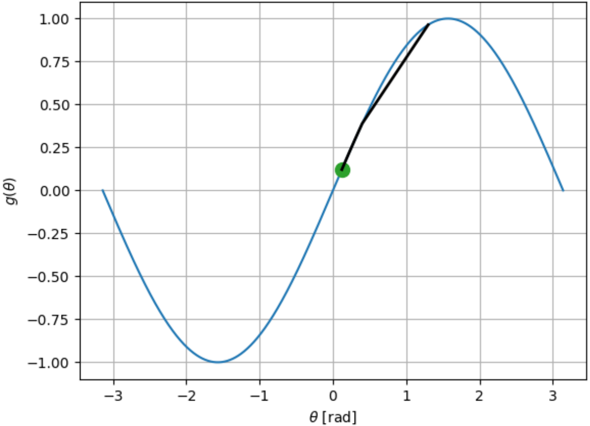
1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.3$, $\alpha = 0.5$
- $\theta^{k+1} = \theta^k - \alpha \left(\frac{\partial g}{\partial \theta}(\theta^k)\right)^{-1} g(\theta^k)$



1D Newton–Raphson method example

- $g(\theta) = \sin(\theta)$, find θ^* s.t. $g(\theta^*) = 0$, $\theta^0 = 1.3$, $\alpha = 0.5$
- $\theta^{k+1} = \theta^k - \alpha \left(\frac{\partial g}{\partial \theta}(\theta^k) \right)^{-1} g(\theta^k)$



How to find α?

- Line-search algorithm

How to find α?

- Line-search algorithm
- Find α s.t. $g(\theta^{k+1}) < g(\theta^{k})$

How to find α?

- Line-search algorithm
- Find α s.t. $g(\theta^{k+1}) < g(\theta^k)$
- Algorithm:
  - $\alpha^0 = 1$

How to find α?

- Line-search algorithm
- Find α s.t. $g(\theta^{k+1}) < g(\theta^k)$
- Algorithm:
  - $\alpha^0 = 1$
  - if $g(\theta^{k+1}) < g(\theta^k)$: break

How to find α?

- Line-search algorithm
- Find α s.t. $g(\theta^{k+1}) < g(\theta^k)$
- Algorithm:
  - $\alpha^0 = 1$
  - if $g(\theta^{k+1}) < g(\theta^k)$: break
  - $\alpha^{i+1} = \tau\alpha^i$, $0 < \tau < 1$, e.g. $\tau = 0.5$
  - repeat

How to find α?

- Line-search algorithm
- Find α s.t. $g(\theta^{k+1}) < g(\theta^k)$
- Algorithm:
  - $\alpha^0 = 1$
  - if $g(\theta^{k+1}) < g(\theta^k)$: break
  - $\alpha^{i+1} = \tau\alpha^i$, $0 < \tau < 1$, e.g. $\tau = 0.5$
  - repeat
- More sophisticated line-search algorithms exist

Numerical solution for RR IK

- Newton–Raphson method for n-dimensional case
  $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \left(\frac{\partial g}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^k) \right)^{-1} g(\boldsymbol{\theta}^k)$ solves $g(\boldsymbol{\theta}) = \mathbf{0}$

Numerical solution for RR IK

- Newton–Raphson method for n-dimensional case
  $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^{k} - \alpha \left(\frac{\partial g}{\partial \boldsymbol{\theta}} (\boldsymbol{\theta}^{k}) \right)^{-1} g(\boldsymbol{\theta}^{k})$ solves $g(\boldsymbol{\theta}) = \mathbf{0}$
- For manipulator kinematics:
  $g(\boldsymbol{q}) = \boldsymbol{x}_d - f_{\mathsf{fk}}(\boldsymbol{q})$, $\boldsymbol{x}_d \in \mathbb{R}^2$

Numerical solution for RR IK

- Newton–Raphson method for n-dimensional case
  $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \left(\frac{\partial g}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^k)\right)^{-1} g(\boldsymbol{\theta}^k)$ solves $g(\boldsymbol{\theta}) = \mathbf{0}$
- For manipulator kinematics:
  $g(\boldsymbol{q}) = \boldsymbol{x}_d - f_{\mathsf{fk}}(\boldsymbol{q})$, $\boldsymbol{x}_d \in \mathbb{R}^2$
- Following NR method (for $g(\boldsymbol{q}) = 0$):
- $\boldsymbol{x}_d = f_{\mathsf{fk}}(\boldsymbol{q}_d) \approx f_{\mathsf{fk}}(\boldsymbol{q}^0) + \frac{\partial f_{\mathsf{fk}}}{\partial \boldsymbol{q}}(\boldsymbol{q}^0)(\boldsymbol{q}_d - \boldsymbol{q}^0)$

Numerical solution for RR IK

- Newton–Raphson method for n-dimensional case
  $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \left(\frac{\partial g}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^k) \right)^{-1} g(\boldsymbol{\theta}^k)$ solves $g(\boldsymbol{\theta}) = \mathbf{0}$
- For manipulator kinematics:
  $g(\boldsymbol{q}) = \boldsymbol{x}_d - f_{\mathsf{fk}}(\boldsymbol{q})$, $\boldsymbol{x}_d \in \mathbb{R}^2$
- Following NR method (for $g(\boldsymbol{q}) = 0$):
- $\boldsymbol{x}_d = f_{\mathsf{fk}}(\boldsymbol{q}_d) \approx f_{\mathsf{fk}}(\boldsymbol{q}^0) + \frac{\partial f_{\mathsf{fk}}}{\partial \boldsymbol{q}}(\boldsymbol{q}^0)(\boldsymbol{q}_d - \boldsymbol{q}^0) = f_{\mathsf{fk}}(\boldsymbol{q}^0) + J(\boldsymbol{q}^0)(\boldsymbol{q}_d - \boldsymbol{q}^0)$

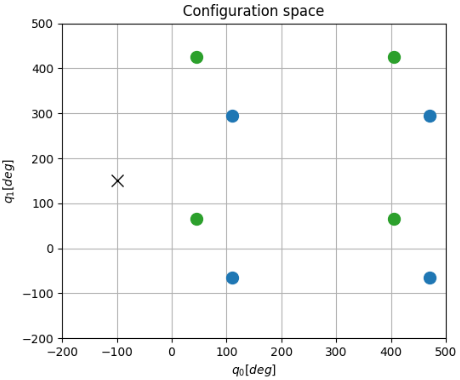
Numerical solution for RR IK

- Newton–Raphson method for n-dimensional case
  $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \left(\frac{\partial g}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^k) \right)^{-1} g(\boldsymbol{\theta}^k)$ solves $g(\boldsymbol{\theta}) = \mathbf{0}$
- For manipulator kinematics:
  $g(\boldsymbol{q}) = \boldsymbol{x}_d - f_{\mathsf{fk}}(\boldsymbol{q})$, $\boldsymbol{x}_d \in \mathbb{R}^2$
- Following NR method (for $g(\boldsymbol{q}) = 0$):
- $\boldsymbol{x}_d = f_{\mathsf{fk}}(\boldsymbol{q}_d) \approx f_{\mathsf{fk}}(\boldsymbol{q}^0) + \frac{\partial f_{\mathsf{fk}}}{\partial \boldsymbol{q}}(\boldsymbol{q}^0)(\boldsymbol{q}_d - \boldsymbol{q}^0) = f_{\mathsf{fk}}(\boldsymbol{q}^0) + J(\boldsymbol{q}^0)(\boldsymbol{q}_d - \boldsymbol{q}^0)$
  $\boldsymbol{q}_d \approx \boldsymbol{q}^0 + J(\boldsymbol{q}^0)^{-1}(\boldsymbol{x}_d - f_{\mathsf{fk}}(\boldsymbol{q}^0))$

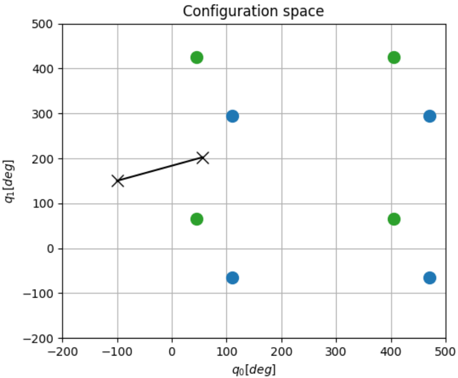
Numerical solution for RR IK

- Newton–Raphson method for n-dimensional case
  $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \left(\frac{\partial g}{\partial \boldsymbol{\theta}} (\boldsymbol{\theta}^k) \right)^{-1} g(\boldsymbol{\theta}^k)$ solves $g(\boldsymbol{\theta}) = \mathbf{0}$
- For manipulator kinematics:
  $g(\boldsymbol{q}) = \boldsymbol{x}_d - f_{\mathsf{fk}}(\boldsymbol{q})$, $\boldsymbol{x}_d \in \mathbb{R}^2$
- Following NR method (for $g(\boldsymbol{q}) = 0$):
- $\boldsymbol{x}_d = f_{\mathsf{fk}}(\boldsymbol{q}_d) \approx f_{\mathsf{fk}}(\boldsymbol{q}^0) + \frac{\partial f_{\mathsf{fk}}}{\partial \boldsymbol{q}}(\boldsymbol{q}^0)(\boldsymbol{q}_d - \boldsymbol{q}^0) = f_{\mathsf{fk}}(\boldsymbol{q}^0) + J(\boldsymbol{q}^0)(\boldsymbol{q}_d - \boldsymbol{q}^0)$
  $\boldsymbol{q}_d \approx \boldsymbol{q}^0 + J(\boldsymbol{q}^0)^{-1}(\boldsymbol{x}_d - f_{\mathsf{fk}}(\boldsymbol{q}^0))$
- Iteratively with line-search:
  $\boldsymbol{q}^{k+1} = \boldsymbol{q}^k + \alpha J(\boldsymbol{q}^k)^{-1}(\boldsymbol{x}_d - f_{\mathsf{fk}}(\boldsymbol{q}^k))$
- Intuition via differential kinematics:
  - what should be velocity in joint space s.t. we achieve given velocity in task-space
  - $\dot{\boldsymbol{q}} = J^{-1}\dot{\boldsymbol{x}}$

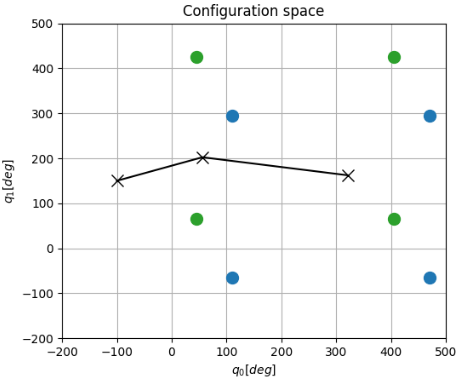
Numerical solution for RR IK #1



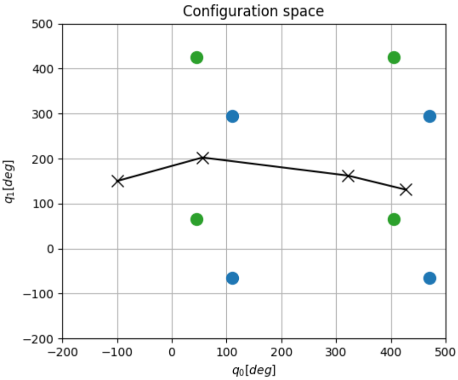
Numerical solution for RR IK #1



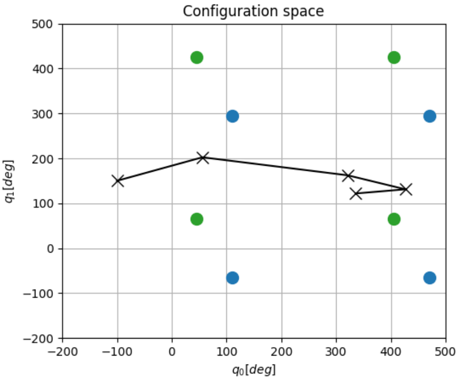
Numerical solution for RR IK #1



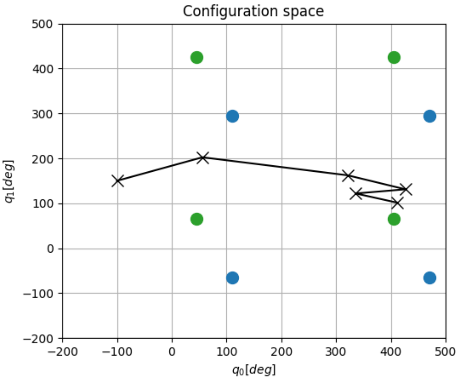
Numerical solution for RR IK #1



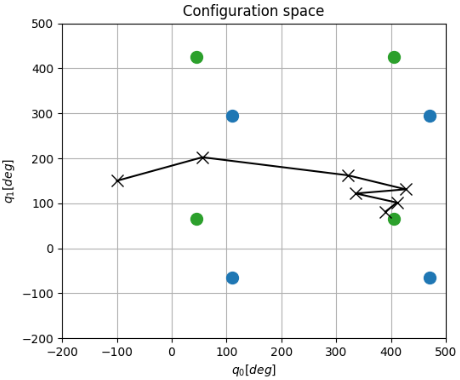
Numerical solution for RR IK #1



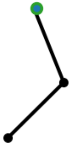
Numerical solution for RR IK #1

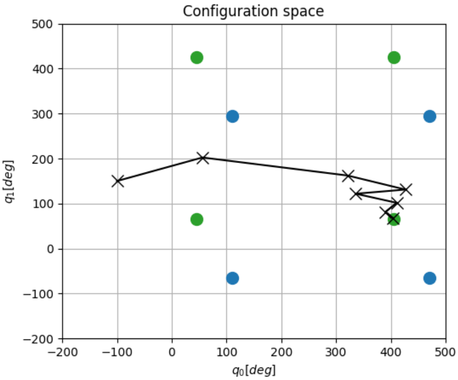


Numerical solution for RR IK #1

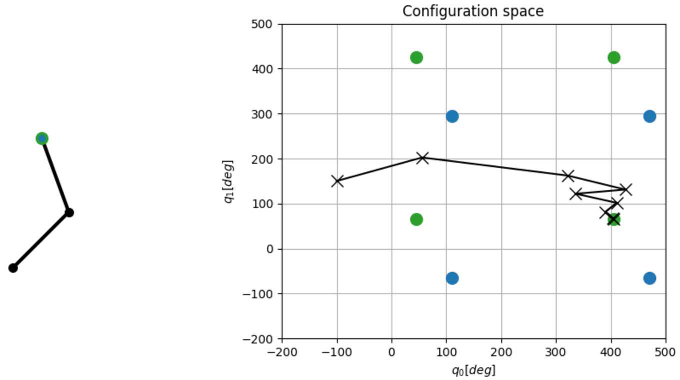


Numerical solution for RR IK #1

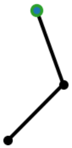


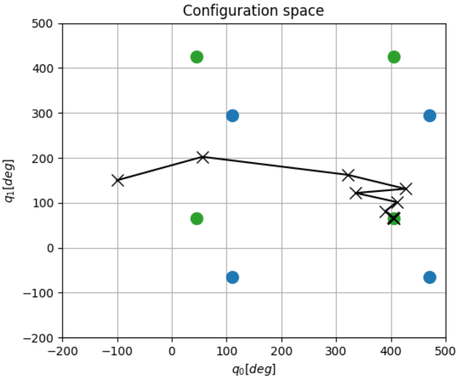


Numerical solution for RR IK #1

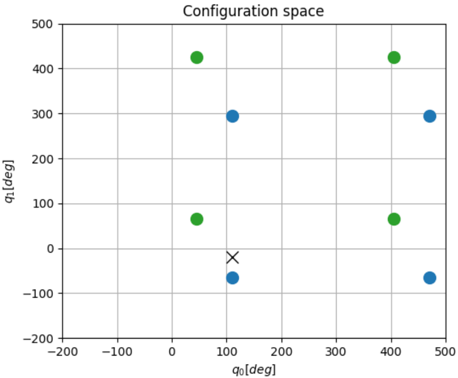


Numerical solution for RR IK #1

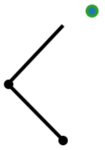


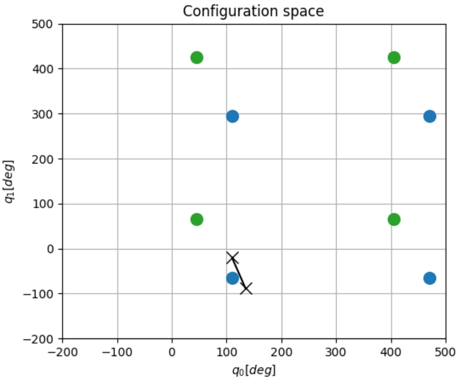


Numerical solution for RR IK #2

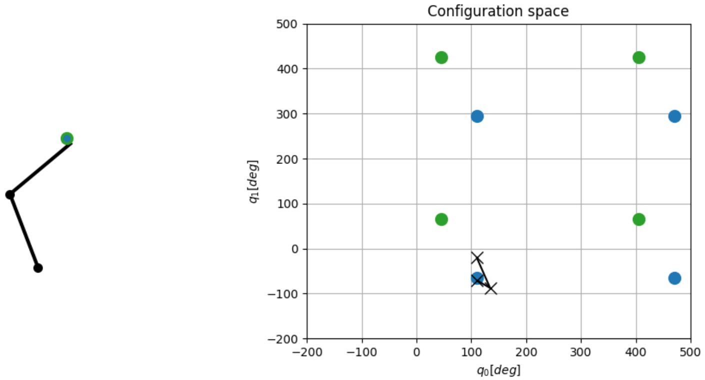


Numerical solution for RR IK #2

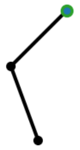


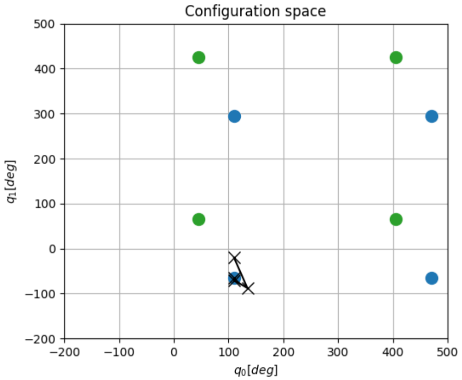


Numerical solution for RR IK #2

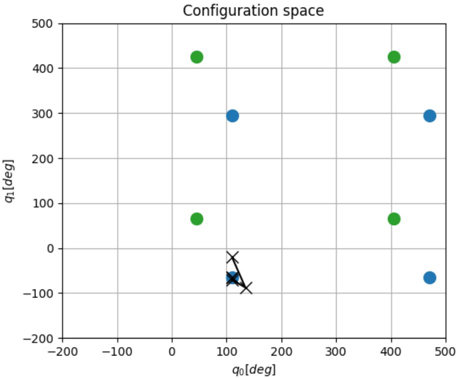


Numerical solution for RR IK #2

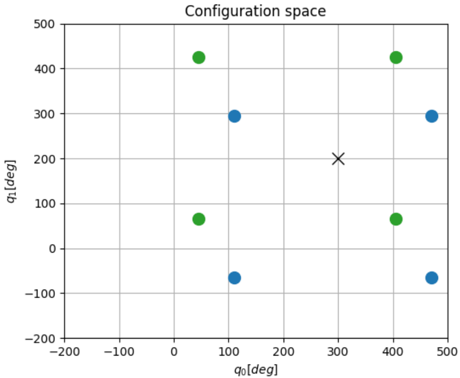




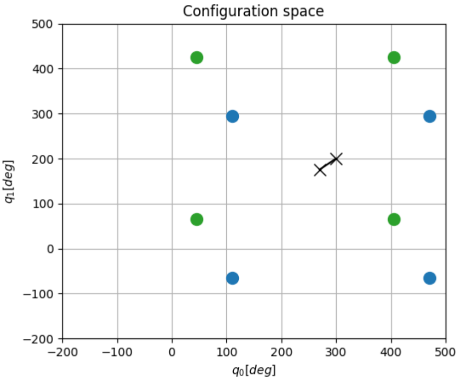
Numerical solution for RR IK #2



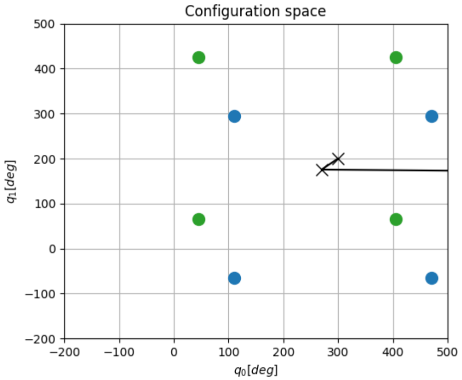
Numerical solution for RR IK #3



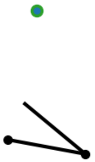
Numerical solution for RR IK #3

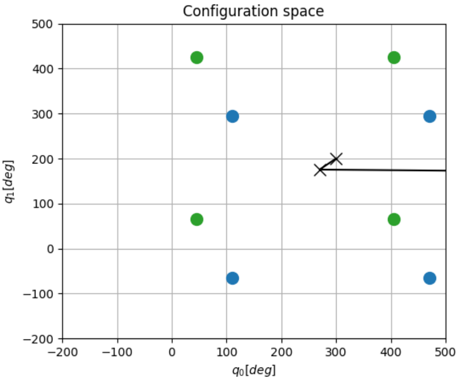


Numerical solution for RR IK #3

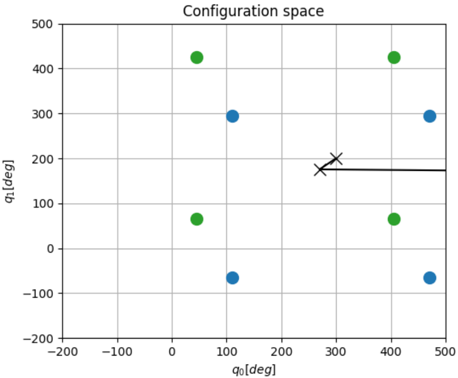


Numerical solution for RR IK #3

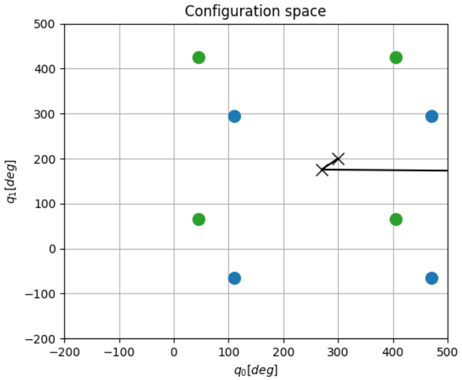




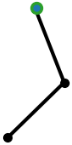
Numerical solution for RR IK #3

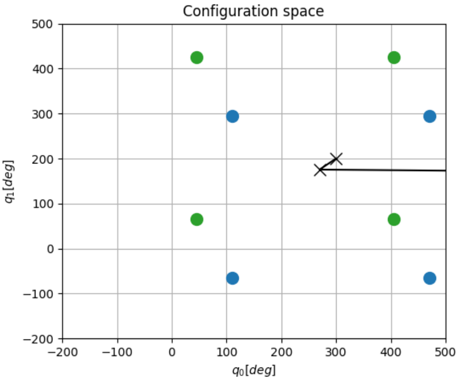


Numerical solution for RR IK #3

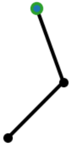


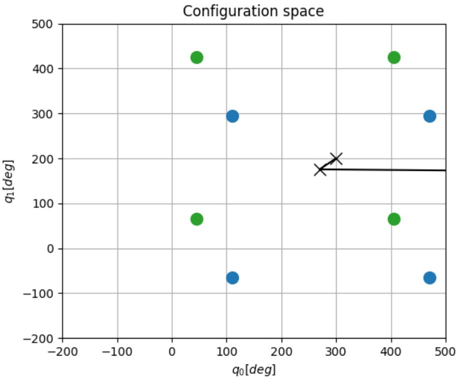
Numerical solution for RR IK #3



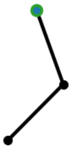


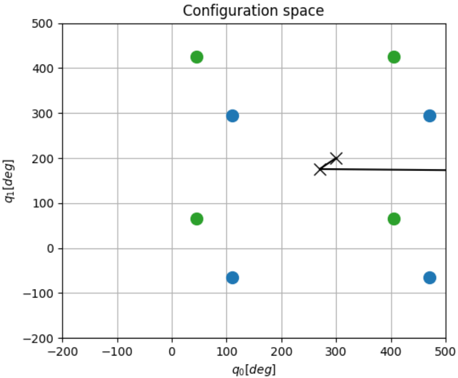
Numerical solution for RR IK #3





Numerical solution for RR IK #3





Numerical solution - takout message

- Numerical solution is easy to implement for general manipulators
- Initial guess is important
  - if we are *close* to the solution, FK is almost linear we will converge to the *closest* solution
  - if we are *too far away* we have no control about which solution is selected
  - tuning step-size might help
- We need to define stopping criteria
  - *e.g.* $\left\| \boldsymbol{x}_d - f_{\mathsf{fk}}(\boldsymbol{q}^k) \right\| < \varepsilon$

What if J is not invertible?

- Redundant robots, Underactuated robots, Singularity

What if J is not invertible?

- Redundant robots, Underactuated robots, Singularity
- Moore–Penrose pseudoinverse $J^\dagger$
- $\boldsymbol{q}^{k+1} = \boldsymbol{q}^k + \alpha J^\dagger(\boldsymbol{q}^k)(\boldsymbol{x}_d - f_{\mathsf{fk}}(\boldsymbol{q}^k))$

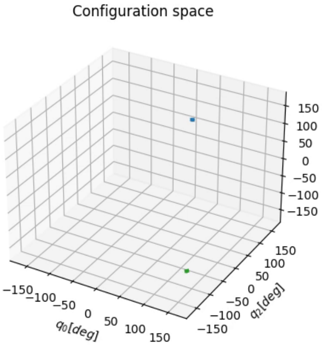
What if J is not invertible?

- Redundant robots, Underactuated robots, Singularity
- Moore–Penrose pseudoinverse $J^\dagger$
- $\boldsymbol{q}^{k+1} = \boldsymbol{q}^k + \alpha J^\dagger(\boldsymbol{q}^k)(\boldsymbol{x}_d - f_{\mathsf{fk}}(\boldsymbol{q}^k))$
- Redundant robots
  - infinite solutions to achieve same task space velocity
  - pseudoinverse will additionally minimize $\|\boldsymbol{q}\|$

What if J is not invertible?

- Redundant robots, Underactuated robots, Singularity
- Moore–Penrose pseudoinverse $J^\dagger$
- $\boldsymbol{q}^{k+1} = \boldsymbol{q}^k + \alpha J^\dagger(\boldsymbol{q}^k)(\boldsymbol{x}_d - f_{\mathsf{fk}}(\boldsymbol{q}^k))$
- Redundant robots
  - infinite solutions to achieve same task space velocity
  - pseudoinverse will additionally minimize $\|\boldsymbol{q}\|$
- Underactuated robots or singularity
  - no exact solution exist for task space velocity
  - pseudoinverse will minimize the error in task-space

IK solution for redundant robot



IK in $SE(2)$ for RRR

- Given desired pose $T^{D}_{\mathsf{RG}} \in SE(2)$
  - R - reference frame
  - G - gripper frame

IK in $SE(2)$ for RRR

- Given desired pose $T^D_{\mathsf{RG}} \in SE(2)$
  - R - reference frame
  - G - gripper frame
- Analytical solution
  - decouple problem into rotation (last joint) and position (other joints)
  - $\boldsymbol{t}_{RC} = T^D_{RG} \begin{pmatrix} -l_3 & 0 & 1 \end{pmatrix}^\top$
  - $\boldsymbol{t}_{RB}$ compute as for RR for translation task-space
  - use $\operatorname{atan2}$ to compute joint configurations

IK in $SE(2)$ for RRR

- Given desired pose $T^D_{\mathsf{RG}} \in SE(2)$
  - R - reference frame
  - G - gripper frame
- Analytical solution
  - decouple problem into rotation (last joint) and position (other joints)
  - $\boldsymbol{t}_{RC} = T^D_{RG} \begin{pmatrix} -l_3 & 0 & 1 \end{pmatrix}^\top$
  - $\boldsymbol{t}_{RB}$ compute as for RR for translation task-space
  - use $\operatorname{atan2}$ to compute joint configurations
- Numerical solution
  - error in reference frame:
    $e(\boldsymbol{q}) = \begin{pmatrix} x^D_{RG} - x_{RG}(\boldsymbol{q}) & y^D_{RG} - y_{RG}(\boldsymbol{q}) & \phi^D_{RG} - \phi_{RG}(\boldsymbol{q}) \end{pmatrix}^\top$

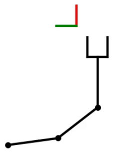
IK in $SE(2)$ for RRR

- Given desired pose $T^D_{\text{RG}} \in SE(2)$
  - R - reference frame
  - G - gripper frame
- Analytical solution
  - decouple problem into rotation (last joint) and position (other joints)
  - $\boldsymbol{t}_{RC} = T^D_{RG} \begin{pmatrix} -l_3 & 0 & 1 \end{pmatrix}^\top$
  - $\boldsymbol{t}_{RB}$ compute as for RR for translation task-space
  - use $\operatorname{atan2}$ to compute joint configurations
- Numerical solution
  - error in reference frame:
    $e(\boldsymbol{q}) = \begin{pmatrix} x^D_{RG} - x_{RG}(\boldsymbol{q}) & y^D_{RG} - y_{RG}(\boldsymbol{q}) & \phi^D_{RG} - \phi_{RG}(\boldsymbol{q}) \end{pmatrix}^\top$
  - NR step:
    $\boldsymbol{q}^{k+1} = \boldsymbol{q}^k + \alpha J^\dagger(\boldsymbol{q}^k)\boldsymbol{e}(\boldsymbol{q}^k)$

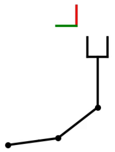
IK in $SE(2)$ for RRR

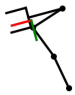
- Given desired pose $T^D_{\text{RG}} \in SE(2)$
  - R - reference frame
  - G - gripper frame
- Analytical solution
  - decouple problem into rotation (last joint) and position (other joints)
  - $\boldsymbol{t}_{RC} = T^D_{RG} \begin{pmatrix} -l_3 & 0 & 1 \end{pmatrix}^\top$
  - $\boldsymbol{t}_{RB}$ compute as for RR for translation task-space
  - use $\operatorname{atan2}$ to compute joint configurations
- Numerical solution
  - error in reference frame:
    $e(\boldsymbol{q}) = \begin{pmatrix} x^D_{RG} - x_{RG}(\boldsymbol{q}) & y^D_{RG} - y_{RG}(\boldsymbol{q}) & \phi^D_{RG} - \phi_{RG}(\boldsymbol{q}) \end{pmatrix}^\top$
  - NR step:
    $\boldsymbol{q}^{k+1} = \boldsymbol{q}^k + \alpha J^\dagger(\boldsymbol{q}^k)\boldsymbol{e}(\boldsymbol{q}^k)$

Numerical solution in $SE(2)$

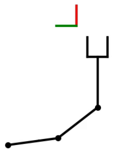


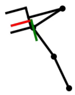
Numerical solution in $SE(2)$





Numerical solution in $SE(2)$





IK in $SE(3)$

- Numerical IK algorithm is almost the same
  - error needs to be computed via transformations
  - as in planar case, error needs to be represented in reference frame

IK in $SE(3)$

- Numerical IK algorithm is almost the same
  - error needs to be computed via transformations
  - as in planar case, error needs to be represented in reference frame
- Analytical solution might not exists for general 6 DoF manipulator

IK in $SE(3)$

- Numerical IK algorithm is almost the same
  - error needs to be computed via transformations
  - as in planar case, error needs to be represented in reference frame
- Analytical solution might not exists for general 6 DoF manipulator
- For 6 DoF spatial robot with revolute joints
  - solution can be decoupled if last three joint axes intersect each other
  - use last three joints to orient gripper
  - use the first three joints to position the flange

Example of importance of multiple solutions

Summary

- Inverse kinematics
  - analytical solution via geometrical analysis
    - leads to computation of intersections of geometrical primitives
  - numerical solution, Newton–Raphson method
    - Jacobian
    - pseudoinverse
- Number of solutions of inverse kinematics
  - no solution
  - multiple solutions
  - periodical solutions
  - infinite number of solutions

Laboratory

- Numerical IK in $SE(2)$
- Analytical IK in $SE(2)$ for RRR manipulator
- Analytical IK in $SE(2)$ for PRR manipulator [optional]