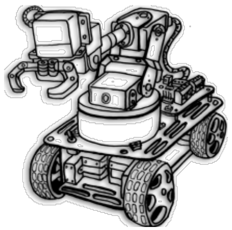
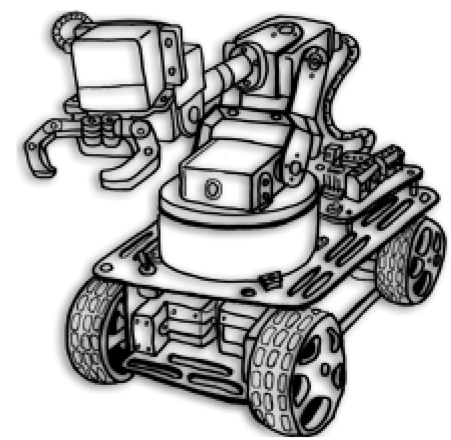


Robotics: Science & Systems

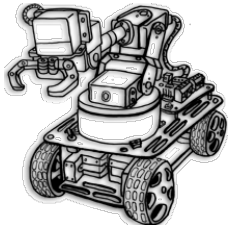
[Topic 3: Kinematics]



Prof. Sethu Vijayakumar

Course webpage: http://wcms.inf.ed.ac.uk/ipab/rss

THE UNIVERSITY of EDINBURGH
informatics

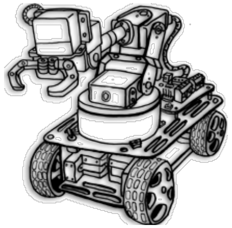


Kinematics?

- Move all the joints in a coordinated way such that the *end-effector* makes the desired movement

Three ingredients:

- when we know/set the joint angles, where is the end-effector?
- when we change the joint angles, how does the end-effector change position?
- when we *want* a certain change in end-effector position, how should we change the joint angles?



Notations

$q \in \mathbb{R}^n$	vector of joint angles (robot configuration)
$\dot{q} \in \mathbb{R}^n$	vector of joint angular velocities
$\delta q \in \mathbb{R}^n$	small step in joint angles
$y \in \mathbb{R}^d$	some "endeffector(s) feature(s)" e.g. position $\in \mathbb{R}^3$ or vector $\in \mathbb{R}^3$
$\phi : \; q \mapsto y$	kinematic map
$J(q) = \frac{\partial \phi}{\partial q} \in \mathbb{R}^{d \times n}$	Jacobian
$\lVert v \rVert_W^2 = v^\top W v$	squared norm of v w.r.t. metric W

Kinematics: 3 ingredients

Kinematic Map

$$\phi : \; q \mapsto y$$

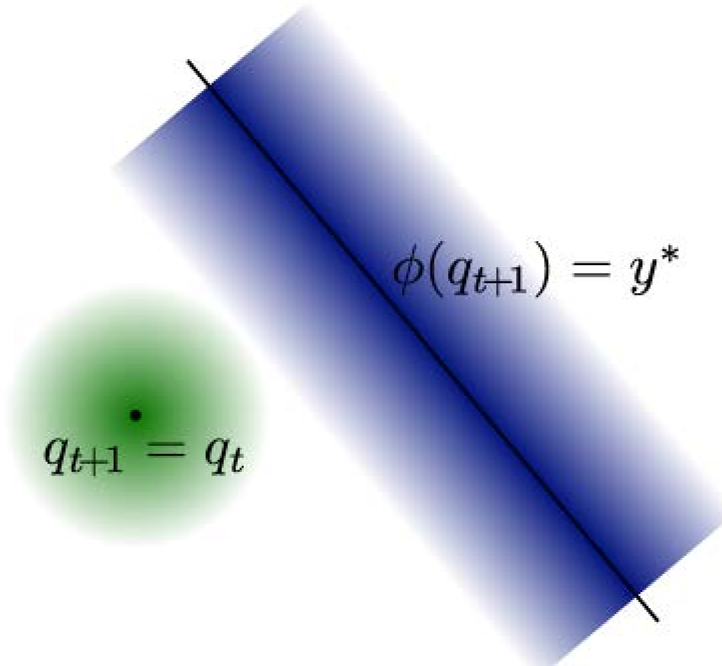
when we know/set the joint angles *q*, where is the end-effector *y*?

Jacobian

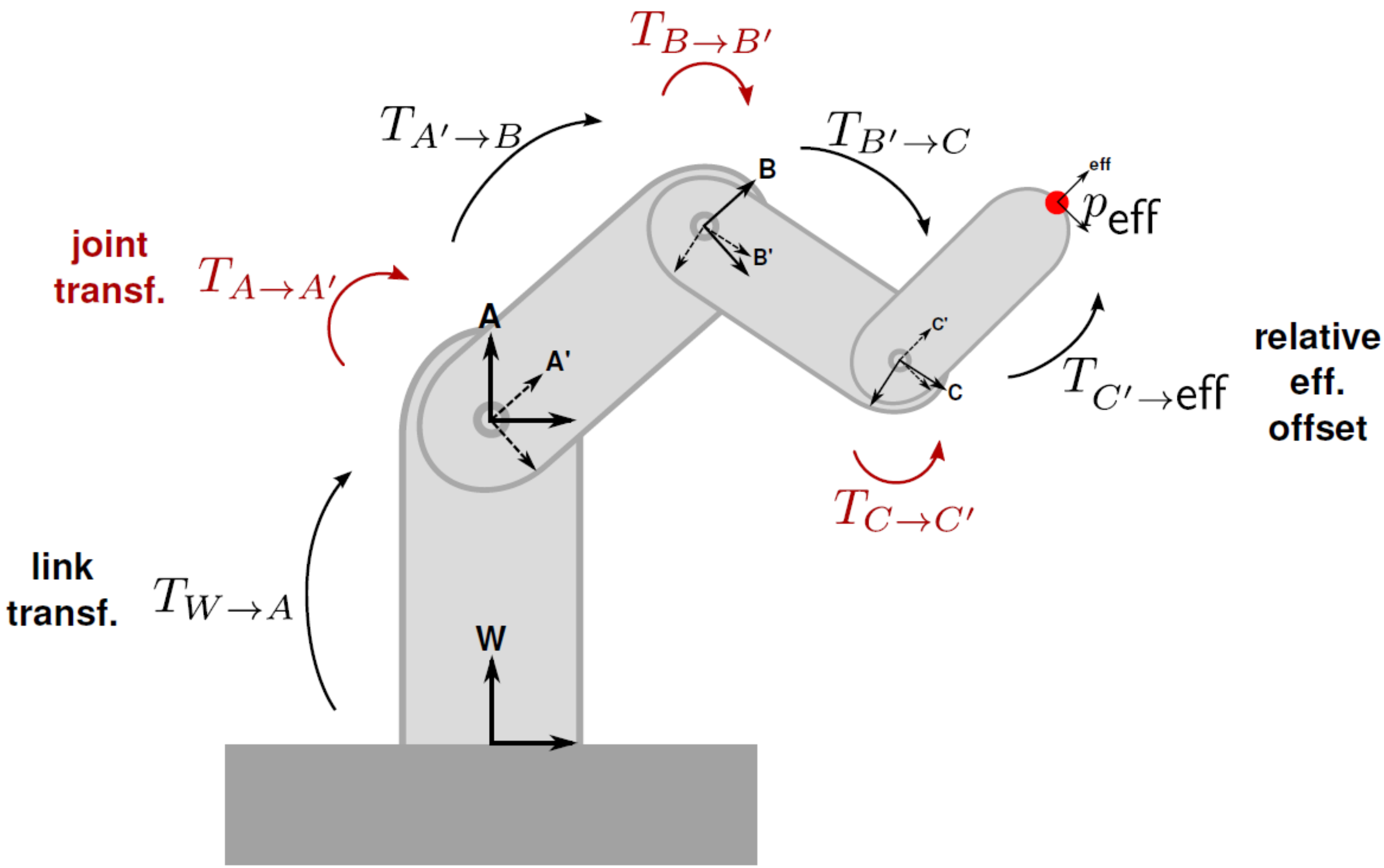
$$J : \; \delta q \mapsto \delta y$$

when we change the joint angles *δq*, how does the end-effector change position *δy*?

Optimality Criterion

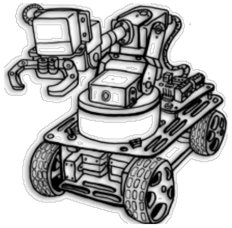


Kinematic Structures



A *kinematic structure* is a graph (usually tree or chain) of rigid **links** and **joints**

$$T_{W\to\text{eff}}(q) = T_{W\to A}\, T_{A\to A'}(q)\, T_{A'\to B}\, T_{B\to B'}(q)\, T_{B'\to C}\, T_{C\to C'}(q)\, T_{C'\to\text{eff}}$$



Joint Types

- link transformations: $T_{W \to A}$
- joint transformations: $T_{A \to A'}(q)$ depends on $q \in \mathbb{R}^n$

revolute joint: joint angle $q \in \mathbb{R}$ determines rotation about x-axis

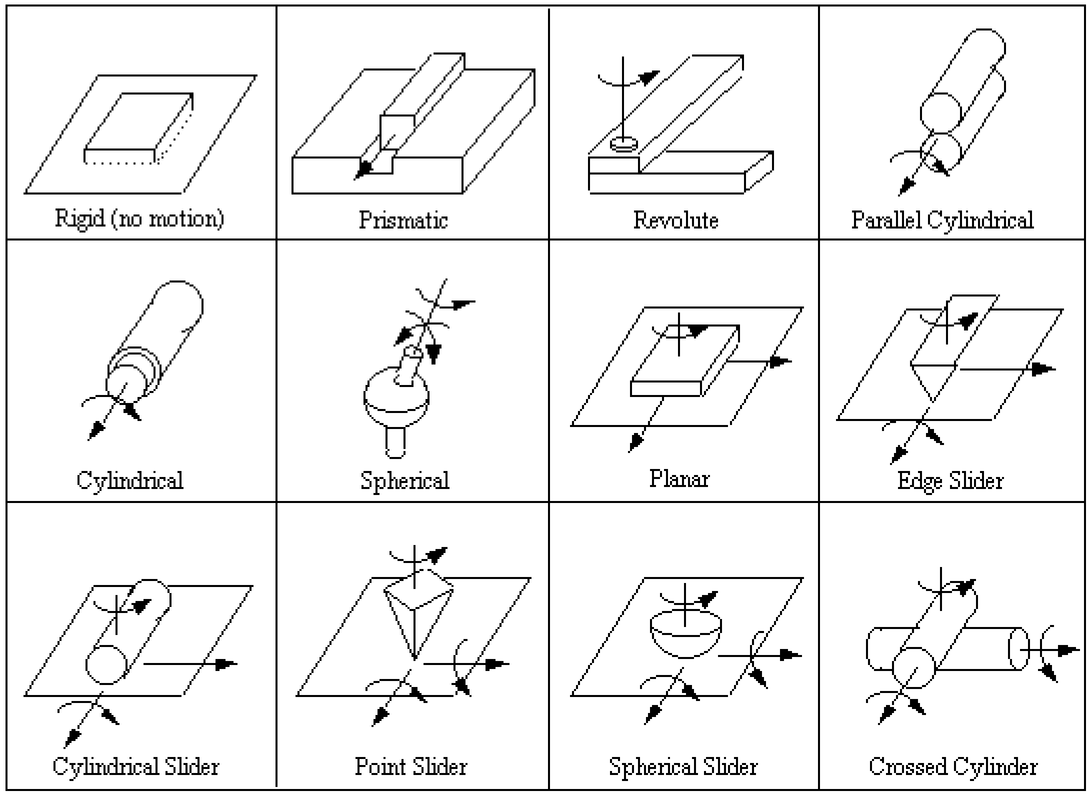
$$T_{A \to A'}(q) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(q) & -\sin(q) & 0 \\ 0 & \sin(q) & \cos(q) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

prismatic joints: offset $q \in \mathbb{R}$ determines translation along x-axis

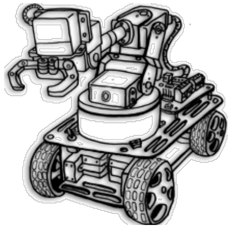
$$T_{A \to A'}(q) = \begin{pmatrix} 1 & 0 & 0 & q \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

others: 1DOF = screw, 2 DOF= cylindrical, spherical, universal

Kinematic Joint Types in 3D



Robinson 1989, Goodrich 1991, Ward 1992



Kinematic Map

For any joint angle vector $q \in \mathbb{R}^n$ we can compute $T_{W\to\text{eff}}(q)$ by *forward chaining* of transformations

$T_{W\to\text{eff}}(q)$ gives us the *pose* of the endeffector

Two basic ways to define a *kinematic map* $\phi: \; q \to y$ are

$$\phi_{\text{pos}}(q) = T_{W\to\text{eff}}(q)\text{.translation} \quad \in \mathbb{R}^3$$

and

$$\phi_{\text{vec}}(q) = [T_{W\to\text{eff}}(q)\text{.rotation}] \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \in \mathbb{R}^3$$

Kinematics: 3 ingredients

Kinematic Map

$$\phi : \; q \mapsto y$$

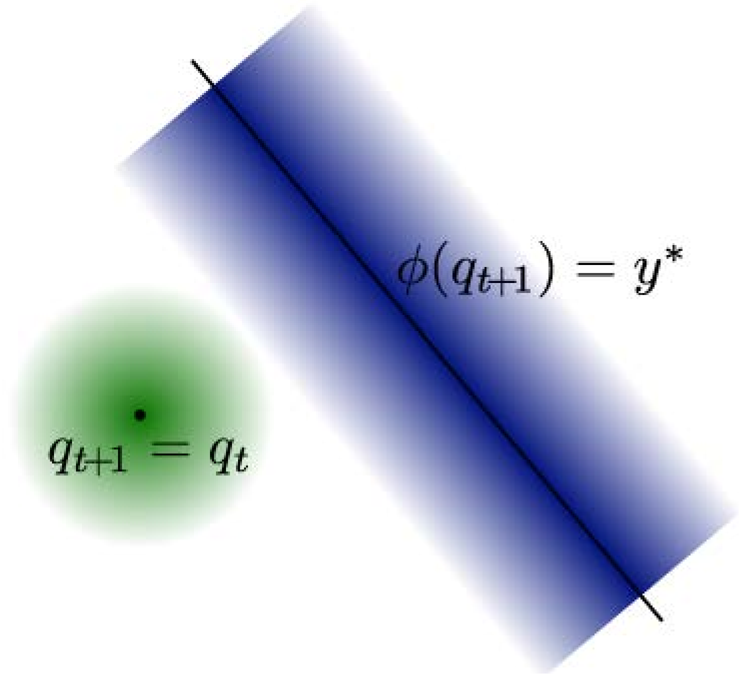
when we know/set the joint angles *q*, where is the end-effector *y*?

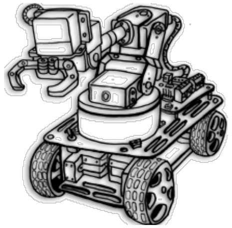
Jacobian

$$J : \; \delta q \mapsto \delta y$$

when we change the joint angles δq, how does the end-effector change position δy?

Optimality Criterion





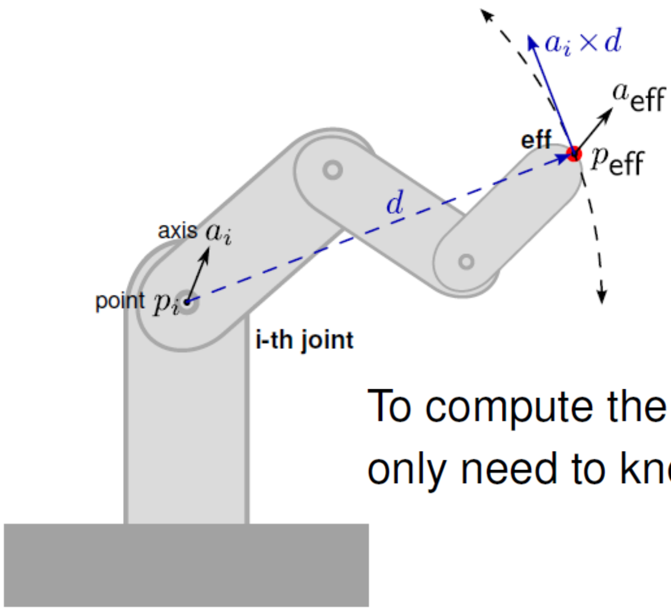
Jacobian

when we change the joint angles δq, how does the end-effector change position δy?

Given the kinematic map $y = \phi(q)$
what is the Jacobian $J(q) = \frac{\partial}{\partial q}\phi(q)$?

$$J(q) = \frac{\partial}{\partial q}\phi(q) = \begin{pmatrix} \frac{\partial \phi_1(q)}{\partial q_1} & \frac{\partial \phi_1(q)}{\partial q_2} & \dots & \frac{\partial \phi_1(q)}{\partial q_n} \\ \frac{\partial \phi_2(q)}{\partial q_1} & \frac{\partial \phi_2(q)}{\partial q_2} & \dots & \frac{\partial \phi_2(q)}{\partial q_n} \\ \vdots & & & \vdots \\ \frac{\partial \phi_d(q)}{\partial q_1} & \frac{\partial \phi_d(q)}{\partial q_2} & \dots & \frac{\partial \phi_d(q)}{\partial q_n} \end{pmatrix}$$

Jacobian



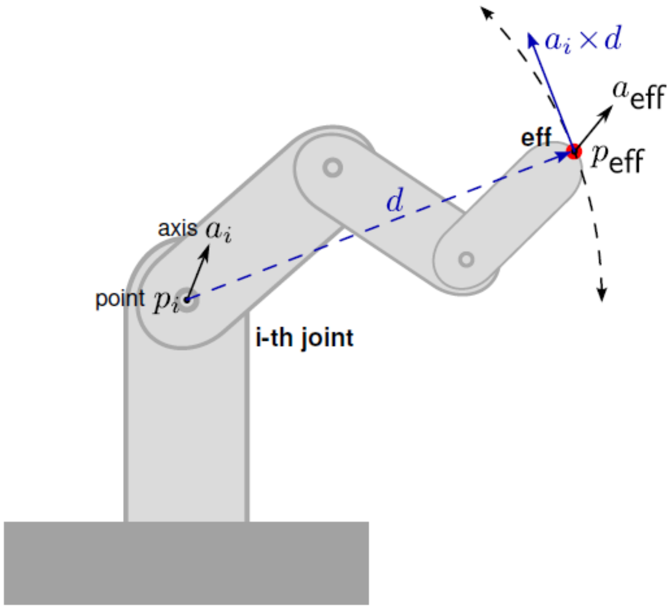
To compute the Jacobian of some endeffector position or vector, we only need to know the position and rotation axis of each joint.

We consider an infinitesimal variation δq_i of the ith joint and see how the endeffector's position $p_{\text{eff}} = \phi_{\text{pos}}(q)$ and attached vector $a_{\text{eff}} = \phi_{\text{vec}}(q)$ change. It must hold

$$\delta p_{\text{eff}} = J_{\text{pos}}(q)_{\cdot i} \delta q_i \qquad \delta a_{\text{eff}} = J_{\text{vec}}(q)_{\cdot i} \delta q_i$$

$a_i = [T_{W \to i}(q).\text{rot}] \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ is rotation axis and $p_i = [T_{W \to i}(q).\text{pos}]$ position of ith joint

Jacobian



Consider a variation δq_i
$\rightarrow$ the whole sub-tree rotates

$$\delta p_{\text{eff}} = \delta q_i [a_i \times (p_{\text{eff}} - p_i)]$$
$$\delta a_{\text{eff}} = \delta q_i [a_i \times a_{\text{eff}}]$$

$$J_{\text{pos}}(q) = \begin{pmatrix} [a_1 \times (p_{\text{eff}} - p_1)] & [a_2 \times (p_{\text{eff}} - p_2)] & \dots & [a_n \times (p_{\text{eff}} - p_n)] \end{pmatrix} \in \mathbb{R}^{3 \times n}$$

$$J_{\text{vec}}(q) = \begin{pmatrix} [a_1 \times a_{\text{eff}}] & [a_2 \times a_{\text{eff}}] & \dots & [a_n \times a_{\text{eff}}] \end{pmatrix} \in \mathbb{R}^{3 \times n}$$

Position Jacobian

Vector Jacobian

Kinematics: 3 ingredients

Kinematic Map

$$\phi : \; q \mapsto y$$

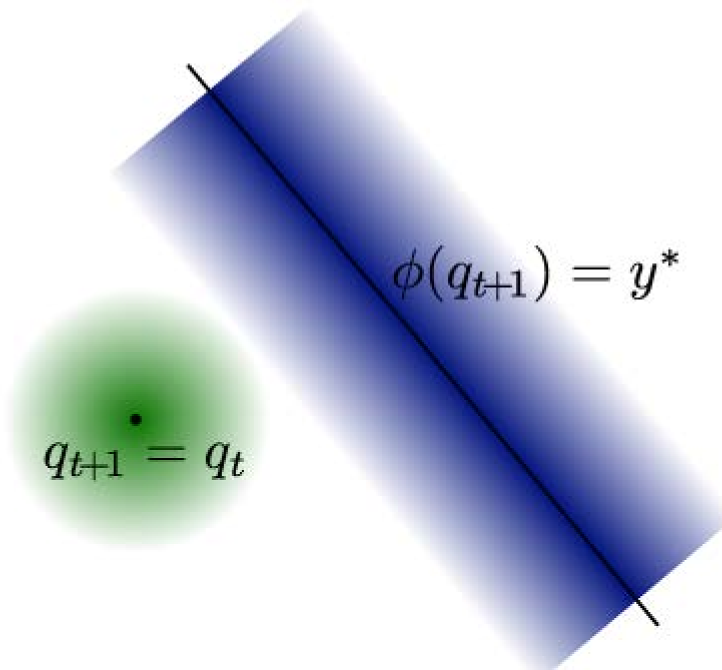
when we know/set the joint angles *q*, where is the end-effector *y*?

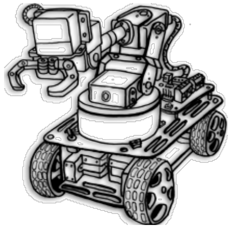
Jacobian

$$J : \; \delta q \mapsto \delta y$$

when we change the joint angles δq, how does the end-effector change position δy?

Optimality Criterion





Inverse Kinematics Problem

When we *want* a certain change δy in eff. position,
how do we have to change the joint angles δq?

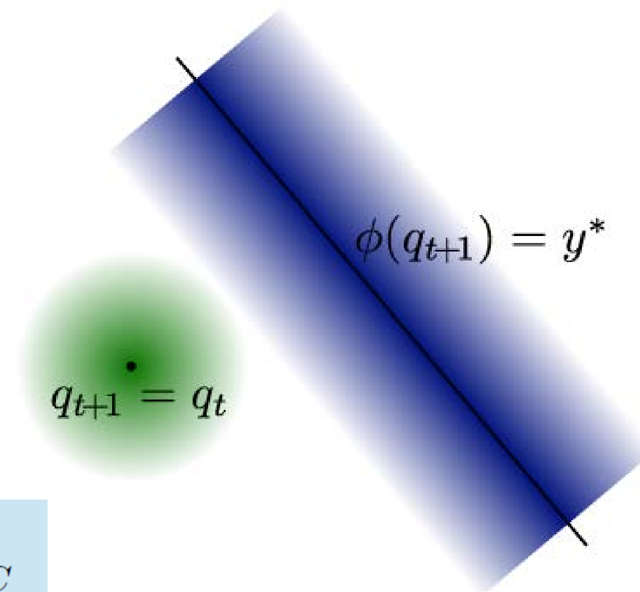
- The Jacobian gives us: $\delta y = J(q)\ \delta q$
- *Iff* the Jacobian were invertible: $\delta q = J(q)^{-1}\ \delta y$
  but typically is not invertible !! ($J \in \mathbb{R}^{d \times n}$ with $d \neq n$)

We formulate an optimality principle to choose δq given δy
– related to taking the pseudo-inverse $J^\sharp$ instead of the undefined J^{-1}

Inverse Kinematics: Optimality Principle

- Given current q_t and $y_t = \phi(q_t)$
  given desired y^*
  compute q_{t+1} *such that*

  1) $\phi(q_{t+1})$ is close to y^* $\leftrightarrow$ *move effector*
  2) q_{t+1} is close to q_t $\leftrightarrow$ *be lazy*



- Formalize as an objective function

$$f(q_{t+1}) \;=\; \|q_{t+1} - q_t\|_W^2 \;+\; \|\phi(q_{t+1}) - y^*\|_C^2$$

Inverse Kinematics: Optimality Principle

$$f(q_{t+1}) \;=\; \|q_{t+1} - q_t\|^2_W \;+\; \|\phi(q_{t+1}) - y^*\|^2_C$$

- When using the **local linearization** $\phi(q_{t+1}) \approx \phi(q_t) + J\,(q_{t+1} - q_t)$, the optimal next joint state q_{t+1} that minimizes $f(q_{t+1})$ is

$$q_{t+1} = q_t + J^\sharp\,(y^* - y_t)$$

$$\delta q = J^\sharp\,\delta y$$

$$J^\sharp = (J^\top C J + W)^{-1} J^\top C = W^{-1} J^\top (J W^{-1} J^\top + C^{-1})^{-1}$$

– for $C \to \infty$ and $W = \mathbf{I}$, $J^\sharp = J^\top (J J^\top)^{-1}$ is called *pseudo-inverse*
– W generalizes the metric in q-space
– C regularizes this pseudo-inverse

Kinematics: 3 ingredients

Kinematic Map

$$\phi : \; q \mapsto y$$

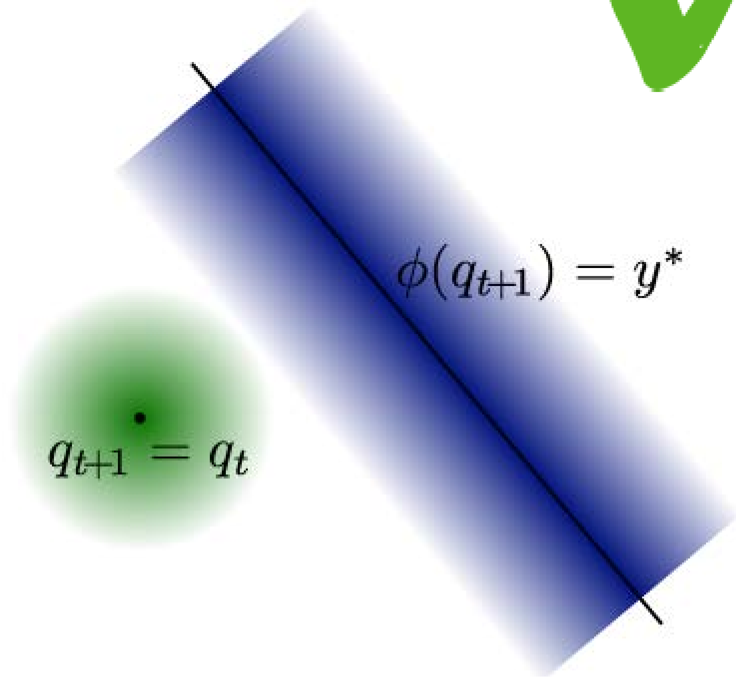
when we know/set the joint angles *q*, where is the end-effector *y*?

Jacobian

$$J : \; \delta q \mapsto \delta y$$

when we change the joint angles *δq*, how does the end-effector change position *δy*?

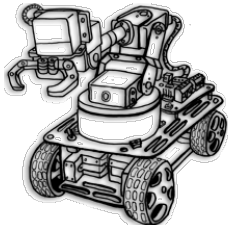
Optimality Criterion



Iterating Inverse Kinematics

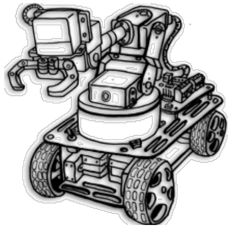
- Assume initial posture q_0. We want to reach a desired endeff position y^* in T steps:

Input: initial state q_0, desired y^*, methods ϕ_{pos} and J_{pos}
Output: trajectory $q_{0:T}$
Set $y_0 = \phi_{\text{pos}}(q_0)$ ▷ current (old) endeff position
for $t = 1 : T$ **do**
    $y \leftarrow \phi_{\text{pos}}(q_{t\text{-}1})$ ▷ current endeff position
    $J \leftarrow J_{\text{pos}}(q_{t\text{-}1})$ ▷ current endeff Jacobian
    $\hat{y} \leftarrow y_0 + (t/T)(y^* - y_0)$ ▷ interpolated endeff target
    $q_t = q_{t\text{-}1} + J^\sharp(\hat{y} - y)$ ▷ new joint positions
    Command q_t to all robot motors and compute all $T_{W\to i}(q_t)$
end for



Where are we?

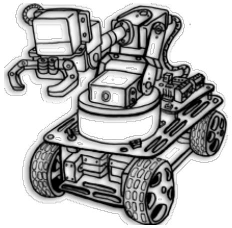
- We have derived the most basic motion generation principle in robotics – *inverse kinematics* – from:
  - an understanding of the robot geometry and kinematics
  - a basic notion of optimality
- In the remainder
  - inverse kinematics and motion rate control
  - singularity and singularity-robustness
  - null space, task space/operational space, joint space
  - extension to multiple task variables
  - extension to other task variables, collisions



Inverse Kinematics and Motion Rate Control

- The notion "kinematics" describes the mapping $\phi: \ q \to y$, which usually is a many-to-one function.
- The notion "inverse kinematics" in the strict sense describes some mapping $g: \ y \to q$ such that $\phi(g(y)) = y$, which usually is non-unique (and non-optimal in our setting).
- In practice, one often refers to $\delta q = J^\sharp \delta y$ as **inverse kinematics**.

- When iterating $\delta q = J^\sharp \delta y$ in a control cycle with time step τ (typically $\tau \approx 1 - 10$ msec), then $\dot{y} = \delta y / \tau$ and $\dot{q} = \delta q / \tau$ and $\dot{q} = J^\sharp \dot{y}$. Therefore the control cycle effectively controls the endefector velocity—this is why it is called **motion rate control**.



Null, Task, Operational, Joint, Configuration Space

- The space of all $q \in \mathbb{R}^n$ is called **joint/configuration space**
  The space of all $y \in \mathbb{R}^d$ is called **task/operational space**
  Usually $d < n$, which is called **redundancy**
- For a desired endeffector state y^* there exists a whole manifold (assuming ϕ is smooth) of joint configurations q:

$$\textbf{nullspace}(y^*) = \{q \mid \phi(q) = y^*\}$$

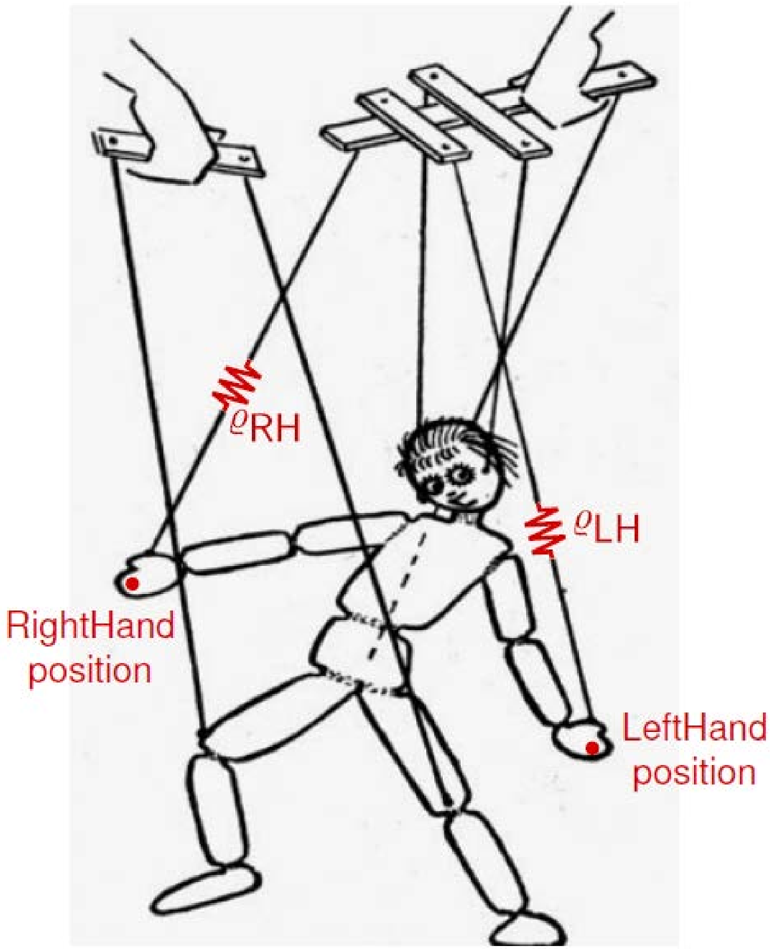
Null Space Motion

- Plain $\delta q = J^\sharp \, \delta y$ resolves redundancy based on the “be lazy” criterion. One can also add **null space motion**: an additional drift $h \in \mathbb{R}^n$ in the nullspace of the task:

$$\delta q = J^\sharp \, \delta y + (I - J^\sharp J) \, h$$

This corresponds to a cost term $\|q_{t+1} - q_t - h\|_W^2$ in $f(q_{t+1})$!

Multiple Tasks



Multiple Tasks

- Assume we have m simultaneous tasks; for each task i we have:
  - a kinematic mapping $y_i = \phi_i(q) \in \mathbb{R}^{d_i}$
  - a current value $y_{i,t} = \phi_i(q_t)$
  - a desired value y_i^*
  - a metric C_i or precision ϱ_i (related via $C_i = \varrho_i \, \mathbf{I}$)
- Each task contributes a term to the objective function

$$
\begin{aligned}
f(q_{t+1}) &= \|q_{t+1} - q_t\|_W^2 \\
&\quad + \|\phi_1(q_{t+1}) - y_1^*\|_{C_1}^2 \\
&\quad + \varrho_2 \, \|\phi_2(q_{t+1}) - y_2^*\|^2 \\
&\quad + \cdots
\end{aligned}
$$

Solution: Optimal joint step is:

$$
q_{t+1} = q_t + \Big[\sum_{i=1}^{m} J^\top C_i J + W\Big]^{-1} \Big[\sum_{i=1}^{m} J^\top C_i (y_i^* - y_{i,t})\Big]
$$

Multiple Tasks

- A much nicer way to write (and code) exactly the same:

$$f(q_{t+1}) = \; \|q_{t+1} - q_t\|^2_W + \Phi(q_{t+1})^\top \Phi(q_{t+1})$$

with the "big task vector" $\Phi(q_{t+1}) := \begin{pmatrix} M_1 \; (\phi_1(q_{t+1}) - y_1^*) \\ \sqrt{\varrho_2} \; (\phi_2(q_{t+1}) - y_2^*) \\ \vdots \end{pmatrix} \quad \in \mathbb{R}^{\sum_i d_i}$

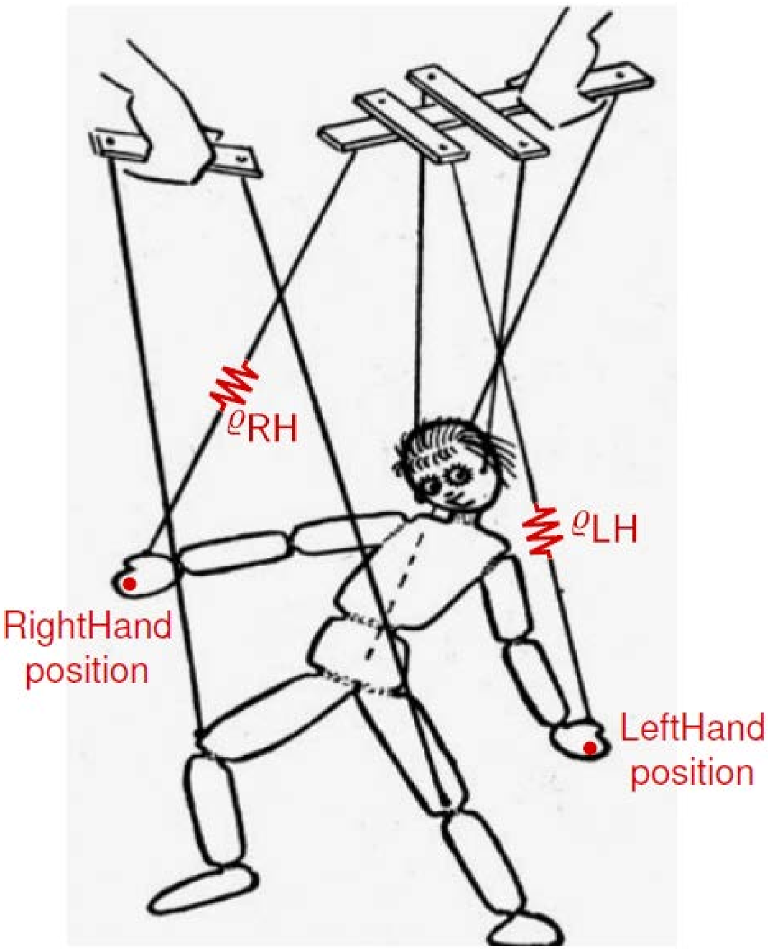
where M_1 is the Cholesky decomposition $C_1 = M_1^\top M_1$.

- The optimal joint step in now:

$$q_{t+1} = q_t - (J^\top J + W)^{-1} J^\top \, \Phi(q_t)$$

with $J \equiv \frac{\partial \Phi(q)}{\partial q}$ the "big Jacobian".

Multiple Tasks



- we learnt how to ‘puppeteer’ a robot
- we can handle many task variables (but it is hard to specify their precision)
- what are interesting **task variables**?

Homework

- Prove the results from slide 14

$$f(q_{t+1}) \;=\; \|q_{t+1} - q_t\|_W^2 \;+\; \|\phi(q_{t+1}) - y^*\|_C^2$$

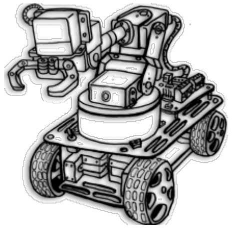
- When using the **local linearization** $\phi(q_{t+1}) \approx \phi(q_t) + J\,(q_{t+1} - q_t)$, the optimal next joint state q_{t+1} that minimizes $f(q_{t+1})$ is

$$q_{t+1} = q_t + J^\sharp\,(y^* - y_t)$$

$$\delta q = J^\sharp\,\delta y$$

$$J^\sharp = (J^\top C J + W)^{-1} J^\top C = W^{-1} J^\top (J W^{-1} J^\top + C^{-1})^{-1}$$

Hint: If you can derive the weighted least squares regression solution from first principles, this is not very different.



Motion Planning: Task Variables

- The following slides will define different types of task variables
- This is meant to give an idea of different possibilities – and as a reference.

Task Var. 1: Endeffector Position

Position of some point attached to link i	
dimension	$d = 3$
parameters	link index i, point offset v
kin. map	$\phi_{\text{pos}i,v}(q) = \text{pos}_i + \text{rot}_i\ v$
Jacobian	$J_{\text{pos}i,v}(q)_{1:3,k} = [k \prec i]\ a_k \times (\phi_{\text{pos}i}(q) - p_k)$

Notation:

– pos_i and rot_i denote position and rotation in $T_{W \to i}$

– a_k, p_k are axis and position of joint k

– $[k \prec i]$ indicates whether joint k is between root and link i

– $J_{\text{pos}i}(q)_{1:3,k}$ is the kth row

Task Var. 2: Endeffector Direction

Vector attached to link i	
dimension	$d = 3$
parameters	link index i, attached vector v
kin. map	$\phi_{\text{vec}i,v}(q) = \text{rot}_i\ v$
Jacobian	$J_{\text{vec}i,v}(q)_{1:3,k} = [k \prec i]\ a_k \times \phi_{\text{vec}i}(q)$

Notation:
– pos_i and rot_i denote position and rotation in $T_{W \to i}$
– a_k, p_k are axis and position of joint k
– $[k \prec i]$ indicates whether joint k is between root and link i
– $J_{\text{pos}i}(q)_{1:3,k}$ is the kth row

Task Var. 3: Endeffector Alignment

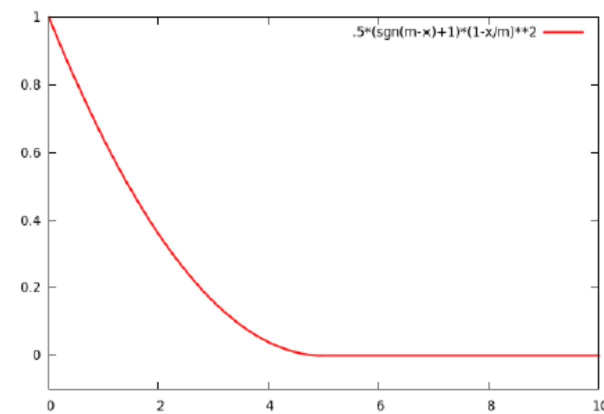
Alignment a vector attached to link i with a reference v^*	
dimension	$d = 1$
parameters	link index i, attached vector v, world reference v^*
kin. map	$\phi_{\text{align}i,v}(q) = v^{*\top}\, \phi_{\text{vec}i,v}$
Jacobian	$J_{\text{align}i,v}(q) = v^{*\top}\, J_{\text{vec}i,v}$

Note: $\phi_{\text{align}} = 1 \leftrightarrow$ align $\quad \phi_{\text{align}} = -1 \leftrightarrow$ anti-align $\quad \phi_{\text{align}} = 0 \leftrightarrow$ orthog.

Task Var. 4: Joint Limits

Penetration of joint limit constraints	
dimension	$d = 1$
parameters	joint limits $q_{\text{low}}, q_{\text{hi}}$, margin m
kin. map	$\phi_{\text{limits}}(q) = \frac{1}{m} \sum_{i=1}^{n} [q_{\text{low}} - q_i + m]^+ + [q_i - q_{\text{hi}} + m]^+$
Jacobian	$J_{\text{limits}}(q)_{1,i} = -\frac{1}{m}[q_{\text{low}} - q_i + m > 0] + \frac{1}{m}[q_i - q_{\text{hi}} + m > 0]$

$[x]^+ = x > 0 ? x : 0$ $\quad$ $[\cdots]$: indicator function



Task Var. 5: Collision Avoidance

Penetration of collision constraints	
dimension	$d = 1$
parameters	margin m
kin. map	$\phi_{\text{col}}(q) = \frac{1}{m} \sum_{k=1}^{K} [m - \lvert p_k^a - p_k^b \rvert]^+$
Jacobian	$J_{\text{col}}(q) = \frac{1}{m} \sum_{k=1}^{K} [m - \lvert p_k^a - p_k^b \rvert > 0]$ $(-J_{\text{pos}p_k^a} + J_{\text{pos}p_k^b})^\top \frac{p_k^a - p_k^b}{\lvert p_k^a - p_k^b \rvert}$

A collision detection engine returns a set $\{(a, b, p^a, p^b)_{k=1}^K\}$ of potential collisions between link a_k and b_k, with nearest points p_k^a on a and p_k^b on b.

Task Var. 6: Center of Gravity

Center of gravity of the whole kinematic structure	
dimension	$d = 3$
parameters	(none)
kin. map	$\phi_{\text{cog}}(q) = \sum_i \text{mass}_i \; \phi_{\text{pos}i, c_i}$
Jacobian	$J_{\text{cog}}(q) = \sum_i \text{mass}_i \; J_{\text{pos}i, c_i}$

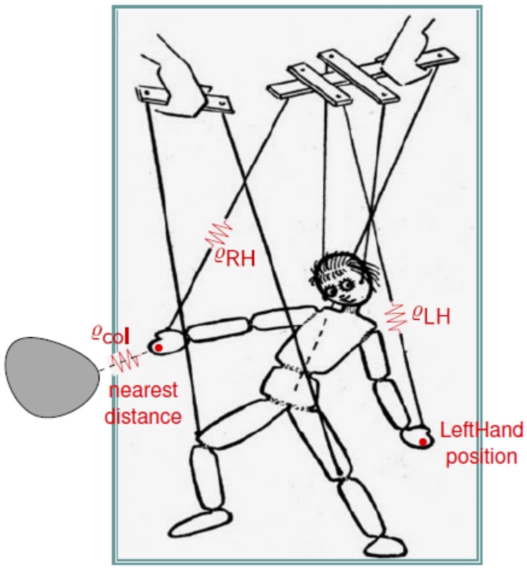
c_i denotes the center-of-mass of link i (in its own frame)

Task Var. 7: Joint Angles (Comfort)

The joint angles themselves	
dimension	$d = n$
parameters	(none)
kin. map	$\phi_{\text{qitself}}(q) = q$
Jacobian	$J_{\text{qitself}}(q) = \mathbf{I}_n$

Example: Set the target $y^* = 0$ and the precistion ϱ very low $\rightarrow$ this task describes posture comfortness in terms of deviation from the joints' zero position.

Task Variables: Conclusion

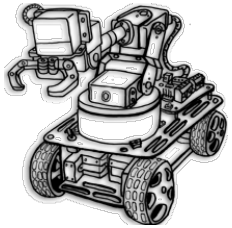


- There is much space for creativity in defining task variables: most of them are combinations of ϕ_{pos} and ϕ_{vec} and the Jacobians combine the basic ones.
- What is the *right* task variable to design/describe motion is a very hard problem. What task variables do humans plan in?
- In practise: Robot motion design requires cumbersome hand tuning of such task variables!

Trajectory Generation

So far, all our methods only look *one* step ahead:
$f(q_{t+1})$ is a cost function for the *next* joint step
$\delta q = J^\sharp \delta y$ desribed the *next* joint step

What if we want to have a nice *trajectory* that smoothly accelerates and comes to a halt at the target?



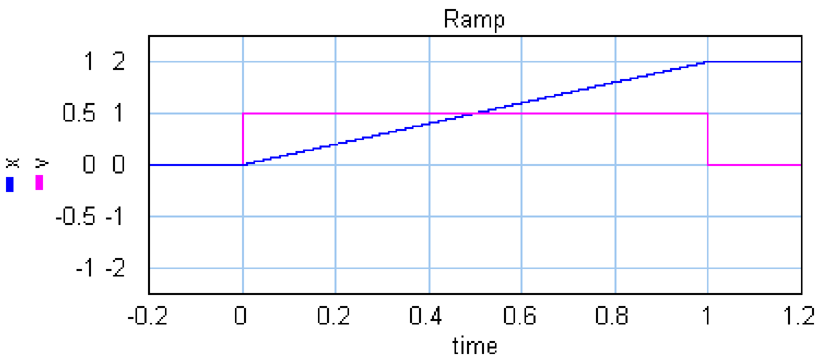
Trajectory Generation: Interpolation

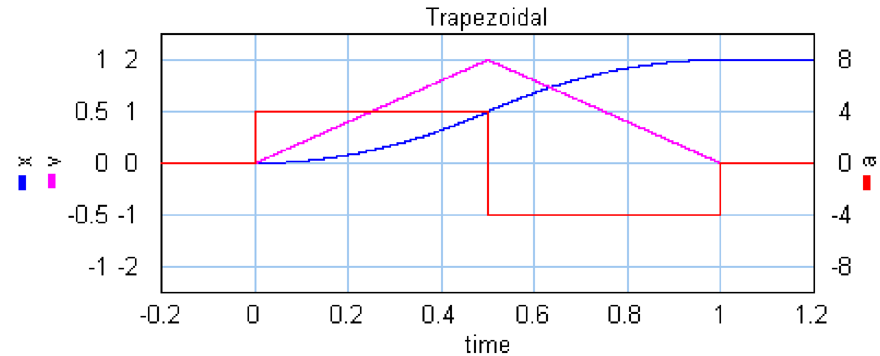
- A **trajectory** $q_{0:T}$ is a sequence of robot configurations $q_t \in \mathbb{R}^n$.
  – This corresponds to $T+1$ *time slices* but T *time steps* (or *transitions*)!
  – In software: typically stored as $(T+1) \times n$-matrix!

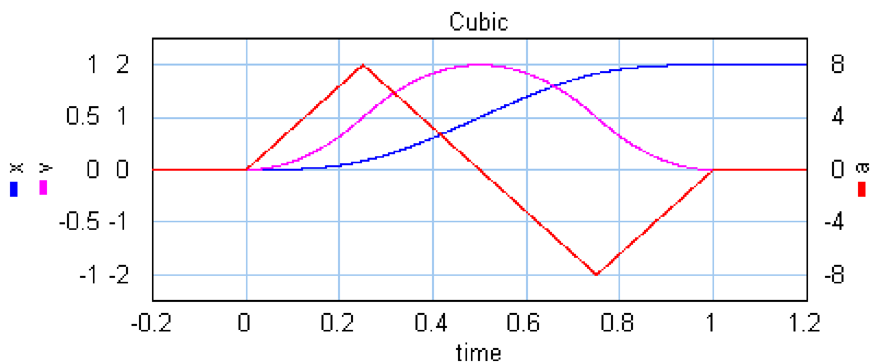
- The basic heuristic for trajectory generation: If you know a desired start point x_0 and target point x_T, interpolate on a straight line and choose a nice **motion profile**.

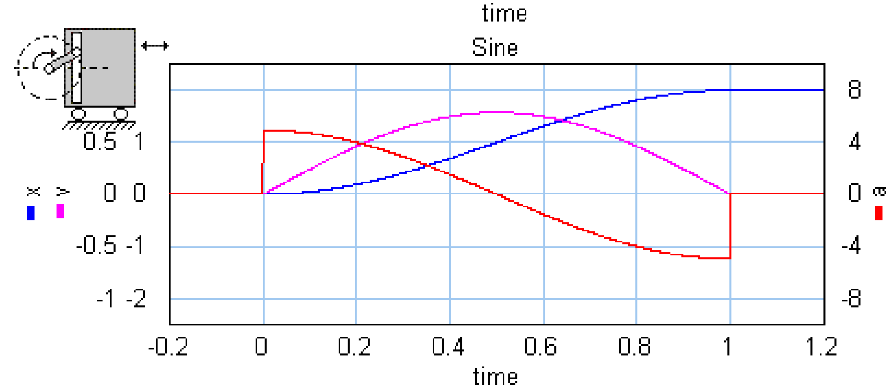
Heuristic Motion Profiles

- Assume initially $x = 0, \dot{x} = 0$. After 1 second you want $x = 1, \dot{x} = 0$. How do you move from $x = 0$ to $x = 1$ in one second?









The sine profile $x_t = x_0 + \frac{1}{2}[1 - \cos(\pi t/T)](x_T - x_0)$ is a compromise for low max-acceleration and max-velocity

Task Space Interpolation

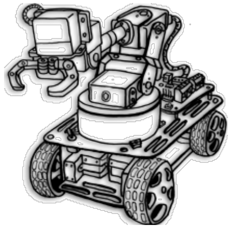
- Task Space Interpolation

  Given a initial task value y_0 and a desired final task value y_T, interpolate on a straight line with a some motion profile. This gives $y_{0:T}$.

- Joint Space Projection

  Given the task trajectory $y_{0:T}$, compute a corresponding joint trajectory $q_{0:T}$ using inverse kinematics

$$q_{t+1} = q_t + J^\sharp (y_{t+1} - \phi(q_t))$$



Joint Space Interpolation

- Optimize Final State Configuration

Given a desired final task value y_T, optimize a final joint state q_T to minimize the function

$$f(q_T) = \|q_T - q_0\|^2_{W/T} + \|y_T - \phi(q_T)\|^2_C$$

Note the step metric $\frac{1}{T}W$, which is consistent with T cost terms with metric W.

- Joint Space Interpolation

Given the initial configuration q_0 and the final q_T, interpolate on a straight line with a some motion profile. This gives $q_{0:T}$.