**Inverse Kinematics**

A solve for manipulator configuration from end effector config

\[
\begin{align*}
X &= l_1 \cos \Theta_1 + l_2 \cos (\Theta_1 + \Theta_2) \\
Y &= l_1 \cos \Theta_1 + l_2 \cos (\Theta_1 + \Theta_2)
\end{align*}
\]

**Solution**

- could be more than 1 answer!
- could be 0 solutions!
- general case is solving a set of nonlinear equations
  - given \(x\) and \(y\), find \(\Theta_1\) and \(\Theta_2\)

\[
\begin{align*}
\alpha &= \sqrt{x^2 + y^2} \\
S &= \tan^{-1}\left(\frac{y}{x}\right) = \text{numpy.arctan2}(y, x) \\
\beta &= \pm \arccos\left(\frac{x^2 + y^2 - d^2}{2xy}\right) \\
\Theta_1 &= S - \beta \quad \text{or} \quad S + \beta \\
\Theta_2 &= \pi - \Theta_1 \quad \text{or} \quad \pi - \Theta_1
\end{align*}
\]

avoid NaNs! not 1 -> short, long or \(\Theta_1\) doesn't exist, everything pick solution closest to current location!

**Warm-up: Partial Derivatives**

\[
F(x, y) = \cos(3xy - \Theta) + 4x - \exp(y)
\]

\[
\begin{align*}
\frac{\partial F}{\partial x} &= -\sin(3xy - \Theta) (3y) + 4 \\
\frac{\partial F}{\partial y} &= -\sin(3xy - \Theta) (3x) - \exp(y)
\end{align*}
\]

**Differential Kinematics**

Consider the derivative of E.E. pose w.r.t. respect to manipulator configuration.
EE position: \( p = (x, y) \) 
\( \text{note: } x, y \text{ are functions of } \theta_1 \text{ and } \theta_2 \)

configuration: \( q = (\theta_1, \theta_2) \)

\[
p(q) = \begin{bmatrix} x(\theta_1, \theta_2) \\ y(\theta_1, \theta_2) \end{bmatrix}
\]

**Defn:** the kinematic Jacobian as the matrix of partial derivatives

\[
J(q) = \begin{bmatrix} \frac{dx}{d\theta_1} & \frac{dx}{d\theta_2} \\ \frac{dy}{d\theta_1} & \frac{dy}{d\theta_2} \end{bmatrix}
\]

as short hand: \( J(q) = [\frac{dp}{dq}] \)

**input:** manipulator configuration \( q \) \n**output:** EE position \( p \)

**Analytic construction of Jacobian:** just take partial derivatives

\[
x(\theta_1, \theta_2) = l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2)
\]

\[
y(\theta_1, \theta_2) = l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2)
\]

\[
J(q) = \begin{bmatrix} -l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \end{bmatrix}
\]

**Geometric Construction:**

* Each column of \( J(q) \) corresponds to a different single manipulator degree of freedom → can deduce this geometrically

* revolute joint in 2D is vector perpendicular to displacement of EE from joint anchor

\[
\begin{aligned}
J(q) &= \begin{bmatrix} x & -y \\ y & -(x-y) \end{bmatrix} \\
\end{aligned}
\]

* in 3D, Jacobian column is (displacement) \( \times (x \times y) \)

**Jacobian maps velocities in manipulator configuration to end effector velocities**

\[
\text{vector of EE velocities } \dot{p} = J(q) \dot{q}
\]

linear map at any point \( q \)
Principle of Virtual Work:

- a change of coordinates should conserve energy (work)
- work = force x distance (linear)
  - torque x angular displacement (angular)

rate of work: force x linear velocity
  (power) torque x angular velocity

thus,

\[ \dot{F} \cdot F = \dot{q} \cdot \Gamma \]

let \( F \) be a force vector @ EE

\[ \dot{r} \text{ be a vector of joint torques} \]

\[ \dot{r} \cdot F = J(q) \dot{q} \cdot F = \dot{\Gamma} \]

\[ \Rightarrow (J(q))^\top \dot{F} = \dot{\Gamma} \]

\[ \Rightarrow \dot{J}(q)^\top \dot{F} = \dot{\Gamma} \]

\[ \dot{J}(q)^\top \dot{F} = \dot{\Gamma} \]

(conclusion: \( J(q) \) acts as a linear map from EE forces to joint torques)

\[ \dot{\Gamma} = J(q)^\top \dot{F} \]

Note: if \( x \cdot y = x \cdot z \) then \( x \times \ldots \times y = 2 \times \ldots \times z \)

\[ \dot{p} = J(q) \dot{q} \]

\[ \frac{dp}{dt} = J(q) \frac{dq}{dt} \]

\[ \frac{dp}{dt} = \left[ \frac{\partial \Gamma}{\partial q^i} \right] \frac{dq}{dt} \]

\[ \Delta p = \frac{\partial \Gamma}{\partial q^i} \Delta q \]

\[ \Delta q = J(q)^{-1} \Delta p \]

\[ \{ \Delta p = \left[ \frac{\partial \Gamma}{\partial q^i} \right] \Delta q \} \]

Suppose \( J(q) \) is invertible

Then, \( \Delta q \approx J(q)^{-1} \Delta p \)

We can use this to do iterative numerical motion.

Small step size

\[ \Delta p \approx \alpha (P_{n+1} - P) \]

\[ \Delta q = J(q)^{-1} \]

\[ q = \tilde{q} + \Delta q \]

and repeat

\[ \text{Compute } 1k \text{ via Jacobian} \]

\[ \text{Compute } p \text{ from } q \text{ (inject)} \]

\[ \text{Compute } \Delta p = \alpha (P_{n+1} - P) \text{ if } \|P_{n+1} - P\| \text{ is small, then done} \]

\[ \text{Compute } J(q) \text{ and its inverse} \]
What if Jacobian doesn't have an inverse

\[ J = \begin{bmatrix} -2 & 1 & 0 \\ 2 & 1 & 3 \end{bmatrix} \]
\[ \Theta_2 = 0 \quad \text{is singular! (not invertible)} \]

Solution to many problems:

\[ \Delta \theta = (J(\theta)^T J(\theta) + \lambda I)^{-1} J(\theta)^T \Delta \theta \]

Feedback Control:

Let \( \theta \) be the observed/measured state of the robot.
Let \( u \) be a set of controls.

For the robot: \( \theta = (x, y, \Theta) \), \( u = (x, u, \Theta) \)

A feedback control scheme (continuously adjusts \( u \) based on \( \theta \) (aka. closed loop control))

vs. open-loop control (no feedback)

Open loop examples: hitting a pitch (a second to judge \( v \rightarrow \) swing)

Closed loop examples: picking up water bottle

- thermostat (controlled heating/cooling)
- cruise control
- avionics (electronics to control airplanes + rockets)
Linear Systems:

\[ \dot{\mathbf{q}} = A \mathbf{q} + B \mathbf{u} \]

state time derivatives \( n \times 1 \rightarrow n \times n \rightarrow n \times 1 \) state

ie. \( \mathbf{q} = [x] \) (temperature) \( \dot{x} = 0 \mathbf{q} + 1 \mathbf{u} \)

\( \mathbf{u} = \dot{x} \) (\( \Delta \) temperature)

**Control law**: \( \mathbf{U} = \begin{cases} \text{Full Blust} & \text{if } \mathbf{x} \neq \mathbf{x}_d \\ 0 \text{ OFF} & \text{otherwise} \end{cases} \)

\( \rightarrow \) "Bang-Bang" - basically binary control

**Pros**: the only choice for some systems

**Cons**: prone to overshoot, jerky - too much wear and tear

**Hysteresis**: turn on until above \( \tau_c \), don't turn on again until below \( \tau_c \) where \( \tau_c < \tau_t \)

- \( \tau_c \) - turn off threshold
- \( \tau_t \) - turn on threshold

---

**Proportional Control** - Turning Robot

**Control law**: \( \mathbf{U} = k_p (\mathbf{q}_{ref} - \mathbf{q}) \)

// Turning Robot

\( \theta = 90^\circ \)

\( \mathbf{q} = [\theta] \)

\( \mathbf{u} = [\dot{\theta}] \)

**Smooth as approach setpoint**

(exponential convergence)

- Not smooth near though! Big startup transient!

\(~\text{maybe start some to fix this}\)~

- Limit slope to limit max \( \mathbf{u} \)
- Min turning rate to make sure it finishes reasonably

ie. \( \mathbf{q} = [x] \)

\( \mathbf{u} = [F] \)

\( \ddot{x} = \frac{F}{m} \)

---

F \rightarrow \mathbf{x} \rightarrow \dot{x} = A \mathbf{q} + B \mathbf{u} \n
\[
\begin{bmatrix}
\dot{x} \\
\ddot{x}
\end{bmatrix} =
\begin{bmatrix}
0 & 0 \\
1 & 0
\end{bmatrix}
\begin{bmatrix}
x \\
\dot{x}
\end{bmatrix}
+ \begin{bmatrix}
\frac{1}{m}
\end{bmatrix}
\begin{bmatrix}
F
\end{bmatrix}
\]
Proportional control law:
\[ F = k_p (x_d - \bar{x}) \]
(non-damped spring) // horrible idea for controls // blunter just Hooke's Law for undamped oscillation forced

\[ q = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \]

Let's add damping:
\[ F = k_p (x_d - \bar{x}) + k_d (\dot{\bar{x}} - \dot{x}) \]
desired acceleration velocity \( \bar{x} \)

\[ u \] \[ PD \text{ control: proportional derivative control} \]

\[ \text{under damped } \rightarrow \text{increase } k_d \]
2nd order system

\[ \text{over damped } \rightarrow \text{decrease } k_d \]
2nd order system

* not continuous feedback \( \rightarrow \) quick snapshots
* don't make \( k_p \) too large, otherwise it's moving too much per screenshot

Let's say \( \dot{x} = \dot{x}_{\text{cmd}} + \bar{E} \)
\( \bar{E} > 0 \), nonzero offset
\( \dot{x}_{\text{cmd}}, \) command acceleration

\[ P \text{ control: proportional control} \]

\[ u = k_p (x_d - \bar{x}) + k_d (\dot{x}_d - \dot{x}) + k_i \int (x_d - \bar{x}) dt \]
\( k_i \) error term
When to use what:

- Bang-Bang: only on/off decision to make
  * Note: use hysteresis/dead band

- Proportional control: only if can control velocity directly
  - PD or PID: 2nd order system akin control proportional to $\dot{a}$, use PID only if need to control steady-state offset

[can do all this in the Laplace domain...]

x coupling complicates controls

10/18 HW 6 Questions?

1. **Problem 1**
   - **Question**: Position Jacobian
   - **Answer**: $\begin{bmatrix} \frac{dx}{d\theta_1} & \frac{dx}{d\theta_2} & \frac{dx}{d\theta_3} \\ \frac{dy}{d\theta_1} & \frac{dy}{d\theta_2} & \frac{dy}{d\theta_3} \\ \frac{dz}{d\theta_1} & \frac{dz}{d\theta_2} & \frac{dz}{d\theta_3} \end{bmatrix}$

2. **Problem 2**
   - **Equations**: $x_p = x_p + \ell \cos \theta$, $y_p = y_p + \ell \sin \theta$
   - **Solution**: $\begin{bmatrix} x_p & y_p \end{bmatrix} = \begin{bmatrix} x_0 + \ell \cos \theta \\ y_0 + \ell \sin \theta \end{bmatrix}$

3. **Problem 3**
   - **Equations**: $\dot{q} = \left[ \begin{array}{c} v_1(\theta) \\ \vdots \\ v_m(\theta) \end{array} \right][\begin{bmatrix} \dot{\theta} \\ \dot{\theta} \end{bmatrix}]$
   - **Solution**: $\left[ \begin{array}{c} \dot{\theta} \\ \dot{\theta} \end{array} \right] = \left[ \begin{array}{c} v_1(\theta) \theta_0(\theta) \end{array} \right][\begin{bmatrix} \frac{\partial v_1}{\partial \theta} \\ \vdots \\ \frac{\partial v_m}{\partial \theta} \end{bmatrix}]$