Topics Covered:

- Review of Geometric Approach to Inverse Kinematics
- Paden-Kahan Approach to Inverse Kinematics
- Quick Overview of an "Algebraic Approach"
- Numerical Approach to Inverse Kinematics

Additional Reading:

- MLS Chapter 3, Section 3; LP 6.2

Note: Matrix Exponentials do *not* communite (unless the actual matrices commute, i.e. $AB = BA$), this is why order matters for the Product of Exponentials:

$$e^A e^B \neq e^B e^A$$

Second, closed-form solutions are defined as expressions which are "formed with constants, variables and a finite set of basic functions connected by arithmetic operations ($\pm, \times, \setminus$ and integer powers) and function composition".

Review

Last week we discussed closed-form solutions to inverse kinematics.

We also presented the geometric approach where we obtained the formulas:

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

for the "righty" solution of an elbow manipulator, and

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

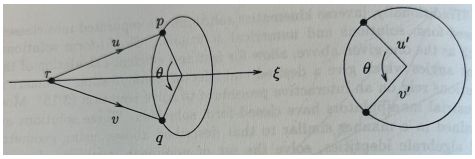
for the "lefty" solution.

Paden-Kahan Subproblems

The Paden-Kahan method of inverse kinematics uses the product of exponentials to break down the problem into subproblems. This method was presented by Paden in 1986 and built on the work of Kahan (done in 1983).

Overall, there are three subproblems:

1. Rotation about a single axis



This subproblem is defined as finding θ such that

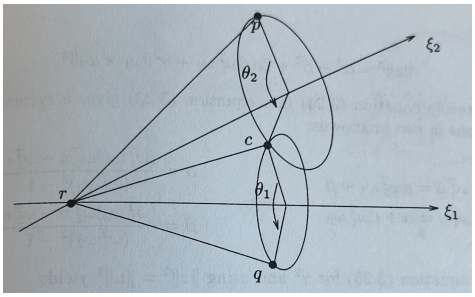
$$e^{\hat{\xi}\theta}p = q$$

This is done by defining $u = (p - r)$ with r being a point on the axis of ξ, and $v = (q - r)$. Then, by projecting u and v onto the plane perpendicular to ω, a formula for θ can be obtained:

$$\theta = \text{atan2}(\omega^\top(u' \times v'), u'^\top v')$$

where $u' = u - \omega\omega^\top u$ and $v' = v - \omega\omega^\top v$.

2. Rotation about two subsequent axes



This subproblem is defined as finding θ_1, θ_2 such that

$$e^{\hat{\xi}_1\theta_1}e^{\hat{\xi}_2\theta_2}p = q$$

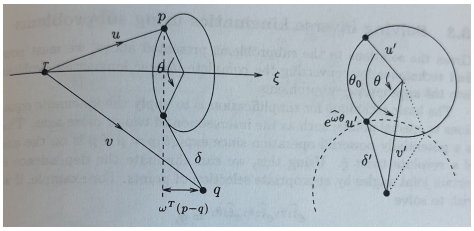
Note that if the two axes of rotation coincide, this problem reduces to subproblem 1 with $\theta = \theta_1 + \theta_2$. If the two axes are not parallel, then let c be some point of intersection such that:

$$e^{-\hat{\xi}_1\theta_1}q = c = e^{\hat{\xi}_2\theta_2}p$$

Then, a formula for θ_1 and θ_2 can be obtained by solving subproblem 1 with:

$$e^{\hat{\xi}_2\theta_2}p = c$$
$$e^{-\hat{\xi}_1\theta_1}q = c$$

3. Rotation to a given distance



Lastly, this subproblem is defined as finding θ such that

$$\|q - e^{\hat{\xi}\theta}p\| = \delta$$

with the solution of this subproblem being:

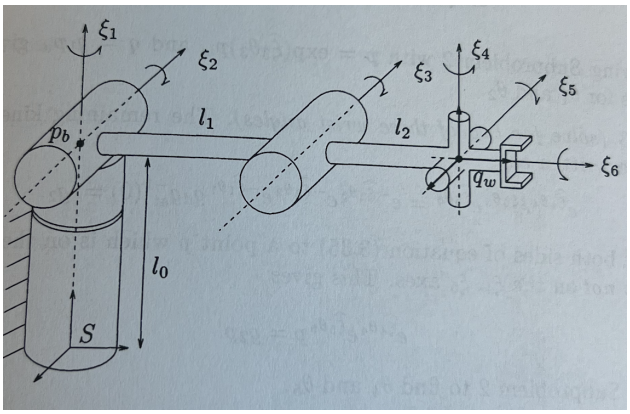
$$\theta = \text{atan2}(\omega^\top(u' \times v'), u'^\top v') \pm \cos^{-1}\left(\frac{\|u'\|^2 + \|v'\|^2 - \delta'^2}{2\|u'\|\|v'\|}\right)$$

and $\delta'^2 = \delta^2 - |\omega^\top(p-q)|^2$.

Together, these subproblems can be used to solve the inverse kinematics problem for most common manipulators.

For example, let's consider the following elbow manipulator:

Elbow Manipulator Example



The inverse kinematics problem for this manipulator wishes to solve:

$$g_{we}(\theta) = e^{\hat{\xi}_1\theta_1} \cdots e^{\hat{\xi}_6\theta_6} g_{we}(0) = g_d$$

with $g_d \in SE(3)$ being the desired end-effector pose.

1. First, we can isolate the exponential maps by post-multiplying by $g_{we}^{-1}(0)$:

   $$e^{\hat{\xi}_1\theta_1} \cdots e^{\hat{\xi}_6\theta_6} = g_d g_{we}^{-1}(0) := g_1$$

   Then, we can use the third subproblem to solve for θ_3. We will first obtain the expression for subproblem 3 by applying both sides of the exponential map equation to some point $p_w \in \mathbb{R}^3$ which is the position of the wrist frame:

   $$e^{\hat{\xi}_1\theta_1} \cdots e^{\hat{\xi}_3\theta_3} p_w = g_1 p_w$$

   Now, taking p_b to be the point of intersection for the first two axes, we can write:

   $$\begin{aligned} e^{\hat{\xi}_1\theta_1} e^{\hat{\xi}_2\theta_2} e^{\hat{\xi}_3\theta_3} p_w - p_b &= g_1 p_w - p_b \\ e^{\hat{\xi}_1\theta_1} e^{\hat{\xi}_2\theta_2} (e^{\hat{\xi}_3\theta_3} p_w - p_b) &= g_1 p_w - p_b \\ \|e^{\hat{\xi}_3\theta_3} p_w - p_b\| &= \|g_1 p_w - p_b\| \end{aligned}$$

   (Distance between points is preserved by rigid motion)

   Thus, taking $\delta = \|g_1 p_w - p_b\|$, $p = p_w$ and $q = p_b$, we can solve for θ_3 using the third subproblem.

2. Now, applying the second subproblem with $p = e^{\hat{\xi}_3\theta_3}$ and $q = g_1 p_w$, we can solve for θ_1 and θ_2.

3. Next, we can solve for θ_4 and θ_5 using subproblem 2. To obtain the expression for the subproblem, we will write the remaining kinematics as:

   $$e^{\hat{\xi}_4\theta_4} e^{\hat{\xi}_5\theta_5} e^{\hat{\xi}_6\theta_6} = e^{-\hat{\xi}_3\theta_3} e^{-\hat{\xi}_2\theta_2} e^{-\hat{\xi}_1\theta_1} g_d g_{we}^{-1}(0) := g_2$$

   Taking some point p to be a point on the axis of ξ_6 but *not* ξ_4 or ξ_5, we can write:

   $$e^{\hat{\xi}_4\theta_4} e^{\hat{\xi}_5\theta_5} p = g_2 p$$

   which allows us to use subproblem 2 with p obtained through forward kinematics.

4. Finally, we can solve for θ_6 by isolating the exponential map:

   $$e^{\hat{\xi}_6\theta_6} = e^{-\hat{\xi}_5\theta_5} e^{-\hat{\xi}_4\theta_4} g_2 p := q$$

   and applying the above expression to subproblem 1.

   After completing these steps, there are a maximum of 8 possible solutions, due to multiple solutions for Step 1, Step 2, and Step 3.

The Algebraic Approach

Main idea:

- try to solve the simultaneous equations
- elimination theory
  - successively eliminate variables until a single equation in one unknown is reached.
  - for manipulator forward kinematics equations, there exists a systematic approach to carry out this procedure.
  - back-substitution gives solution

Numerical Approach

In many real-world applications, the last three axes do not exactly intersect at a common point. In these cases, the analytic inverse kinematic solutions can be used as initial guesses to an iterative numerical procedure.

These iterative approaches are often based on "root finding" methods, where the goal is to find the "roots" of a nonlinear function. Also, optimization-based techinques are advantageous in situations where an exact solution may not exist, or if infinite solutions exist (as is the case for redundant manipulators which we will discuss later).

Newton-Raphson Method

The Newton-Raphson method is a fundamental approach to nonlinear root-finding. It solves some equation $g(\theta) = 0$ numerically by iteratively updating some guess $\theta_k + 1$ using the formula:

$$\theta_{k+1} = \theta_k - \left(\frac{\partial g}{\partial \theta}(\theta_k)\right)^{-1} g(\theta_k)$$

where $\theta_k \in \mathbb{R}^n$ is the estimate of the input θ at iteration k. This formula comes from the first-order Taylor expansion of $g(\theta)$ around θ_k:

$$g(\theta) \approx g(\theta_k) + \frac{\partial g}{\partial \theta}(\theta_k)(\theta - \theta_k)$$

and setting $g(\theta) = 0$. The iterations are repeated until some stopping criterion is met. This criterion is typically defined as a small change in θ between iterations, or a small change in $g(\theta)$, i.e. $|g(\theta_k) - g(\theta_{k+1})|/|g(\theta_k)| \leq \epsilon$.

We can apply the Newton-Raphson method to inverse kinematics by defining $g(\theta_d) = x_d - f(\theta)$ and letting θ_0 be some initial guess. Notably, in this expression, we see the Jacobian appear:

$$x_d = f(\theta_d) = f(\theta_0) + \underbrace{\left.\frac{\partial f}{\partial \theta}\right|_{\theta_0}}_{J(\theta_0)} (\theta_d - \theta_0)$$

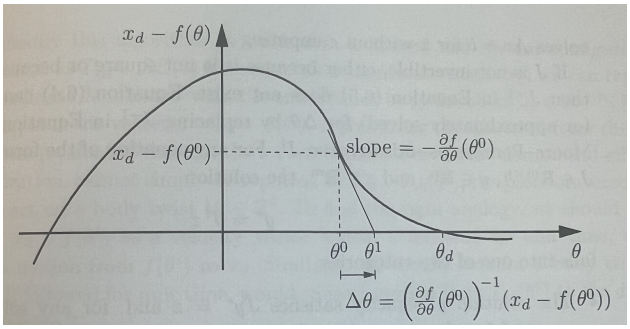
where $J(\theta_0) \in \mathbb{R}^{m \times n}$ is the (coordinate) Jacobian evaluated at θ_0.

By rearranging this expression, we can directly write our update rule as:

$$\delta\theta = J^{\dagger}(\theta_0)(x_d - f(\theta_0))$$

with $J^{\dagger}$ being the pseudoinverse, which we will also cover more in depth later in the course.

Overall, we can see the effect of θ_0 by considering the following example:



If θ_0 is chosen to be close to the solution, then eventually θ_k will converge to θ_d. However, if θ_0 is chosen to the left of the plateau of $x_d - f(\theta)$, then it will likely converge to the other root of $x_d - f(\theta)$.

Modern Optimization Techinques

Most modern optimization solvers use some technique similar to Newton-Raphson. However, they often include additional features which allow for the addition of objective functions and constraints. But in all cases, good initial guesses are crucial for the success of the algorithm.

Typically, the optimization problem is set up as:

$$\begin{aligned} \min_{\theta} \quad & f(\theta) \\ \text{s.t.} \quad & c(\theta) = 0 \end{aligned}$$

which can be coded in python as:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize as opt

def objective(theta):
    return f(theta)
def constraint(theta):
    return c(theta)

# Example function to minimize
def f(theta):
    return (theta - 3) ** 2 + 4

# Example constraint function
def c(theta):
    return theta - 2

# Initial guess
theta0 = 0.5

# Perform the optimization
result =
    opt.minimize(objective, theta0, constraints={'type': 'eq', 'fun': constraint})

# Print the result
print("Optimal theta:", result.x)
print("Objective function value at optimal theta:", result.fun)
```

or in MATLAB as:

```
% Initial guess
theta0 = 0.5;

% Perform the optimization
[result, fval] = fmincon(@objective, theta0, [], [], [], [], [], [], @constraint);

% Print the result
fprintf('Optimal theta: %f\n', result);
fprintf('Objective function value at optimal theta: %f\n', fval);

% Define the objective function
function val = objective(theta)
    val = f(theta);
end

% Define the constraint function
function [c, ceq] = constraint(theta)
    c = []; % No inequality constraints
    ceq = theta - 2; % Equality constraint
end

% Example function to minimize
function val = f(theta)
```

```
    val = (theta - 3) ^ 2 + 4;
end
```