Practice Set 19

Robotics & Automation
Dylan Losey, Virginia Tech

Using your textbook and what we covered in lecture, try solving the following problems. For some problems you may find it convenient to use Matlab (or another programming language of your choice). The solutions are on the next page.

Problem 1

Given the transformation T, find $X = \log T$

$$T = \begin{bmatrix} 0 & -1 & 0 & 4 \\ 1 & 0 & 0 & 0.5 \\ 0 & 0 & 1 & 3.1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Problem 2

In our numerical solution, why does the `while loop` end when $\|X_d - X\| \leq \epsilon$? Why not require that $X_d = X$? If you increased ϵ, how would your solution change?

Problem 3

- How does the initial choice of θ (i.e., θ_0) affect the numerical solution?
- Imagine that you have a robot with multiple inverse kinematics solutions, and you are looking for the one closest to your current joint position (i.e., minimizing the distance traveled in joint space). What value of θ_0 should you choose?

Problem 1

Problem 1

Given the transformation T, find $X = \log T$

$$T = \begin{bmatrix} 0 & -1 & 0 & 4 \\ 1 & 0 & 0 & 0.5 \\ 0 & 0 & 1 & 3.1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

First check whether R is the identity matrix. It is not, so:

$$X = \begin{bmatrix} \hat{\omega}\theta \\ p \end{bmatrix} \tag{3}$$

We know the position vector p directly from the transformation matrix. To get the unit vector axis $\hat{\omega}$ and angle θ, use our formulas for rotation matrices:

$$\theta = \cos^{-1}\left(0.5 \cdot (\text{trace}(R) - 1)\right) = \pi/2 \tag{4}$$

$$[\hat{\omega}] = \frac{1}{2\sin\theta}(R - R^T), \quad \hat{\omega} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{5}$$

Putting this together, we get that X is:

$$X = \begin{bmatrix} 0 \\ 0 \\ \pi/2 \\ 4 \\ 0.5 \\ 3.1 \end{bmatrix} \tag{6}$$

Below you can find an example Matlab function for converting a rotation matrix R to the axis-angle representation $\omega = \hat{\omega}\theta$. You may need to modify this function to deal with other angles where $\theta = 0$.

```
function omega = R2axisangle(R)
    if norm(R - eye(3)) < 1e-3
        omega = [0;0;0];
    else
        theta = acos(0.5 * (trace(R) - 1));
        omega_hat = 1/(2*sin(theta)) * (R - R');
        omega_hat = [omega_hat(3,2);
            omega_hat(1,3); omega_hat(2,1)];
        omega = omega_hat * theta;
    end
end
```

Problem 2

In our numerical solution, why does the `while loop` end when $\|X_d - X\| \leq \epsilon$? Why not require that $X_d = X$? If you increased ϵ, how would your solution change?

> We are finding a numerical solution and often that solution is not exact. Because of machine precision or our choice of step size α we may never get $\|X_d - X\| = 0$. Increasing ϵ decreases your computation time, but it also allows for more error between the desired transformation matrix and the output transformation matrix. If you just need to move the robot into a target region, perhaps high values of ϵ are reasonable. When you need to complete a precision task (e.g., robotic surgery), you may be willing to wait longer for a solution that satisfies a lower ϵ.

Problem 3

- How does the initial choice of θ (i.e., θ_0) affect the numerical solution?
- Imagine that you have a robot with multiple inverse kinematics solutions, and you are looking for the one closest to your current joint position (i.e., minimizing the distance traveled in joint space). What value of θ_0 should you choose?

> Different initial guesses of θ_0 may lead to different inverse kinematics solutions (e.g., elbow-up or elbow-down).
>
> If we leave $b = 0$, then a robot using our numerical approach will find the inverse kinematics solution *closest* to its current joint position θ. If you want to find that closest solution, initialize θ_0 as the robot's actual joint position θ.