16-745 Optimal Control and Reinforcement Learning
Spring 2020
The Robotics Institute

Homework 1

Jacky Liang — qiaolian

1 Inverse Kinematics

To perform optimization-based inverse kinematics, I coded a forward kinematics function, an end-effector pose error evaluation function, and a function that computes distances to collisions.

1.1 Forward Kinematics

Given a snake robot with N links, each with length l_n and euler angles r_n, p_n, y_n, the transform from the $n+1$th joint to the world frame (origin at the 0th joint) can be computed as follows:

$$\tau_{n+1}^0 = \begin{bmatrix} R_n^0 R(r_n, p_n, y_n) & R_n^0 [l_n, 0, 0]^\top + T_n^0 \\ 0 & 1 \end{bmatrix} \tag{1}$$

Where τ_n^0 denotes the transform from the nth joint to the origin (similarly for R for rotation and T for translation). $R(r, p, y)$ is the rotation matrix corresponding to the euler angles r, p, y.

1.2 Pose Error Evaluation

The pose difference between two rigid transforms τ_n^0 and τ_m^0 is evaluated as follows:

$$\tau_n^m = (\tau_m^0)^{-1} \tau_n^0 \tag{2}$$

$$\Delta(\tau_0, \tau_1) = \|T_n^m\|_2^2 + \phi(\tau_n^m)^2 \tag{3}$$

where T_n^m is the translation component of τ_n^m, and $\phi(\tau_n^m)$ gives the angle of the axis-angle representation of the rotation component of τ_n^m.

1.3 Collision Avoidance

To compute collision penalty, I construct a line segment that corresponds to each of the links. Then, I compute the obstacle with the minimum distance for each of the line segment. Finally, the collision penalty is the negative of the minimum minimum obstacle distance across all the line segments.

The distance from a line segment (specified by its two endpoints s_0 and s_1) and a point in space p (the center of each obstacle sphere) is computed by:

$$t = \min(\max(\frac{(p - s_0)^\top s_1}{\|s_1 - s_0\|_2^2}, 0), 1) \tag{4}$$

$$D = \|s_1 + t(s_1 - s_0) - p\|_2 \tag{5}$$

1.4 Cost Function

The final cost function used for optimization-absed inverse kinematics is:

$$\Delta(\tau_0, \tau_1) - 0.4D \tag{6}$$

The value of 0.4 was chosen via grid search - smaller values did not avoid the obstacles, while larger values made the IK inaccurate.

1.5 Results

See Figure 1 for results. To generate the target poses, I first randomly sample link lengths of $N = 7$ and random joint angles - the resultant end-effector pose is recorded as a target. This way, the generated targets are guaranteed to be reachable.

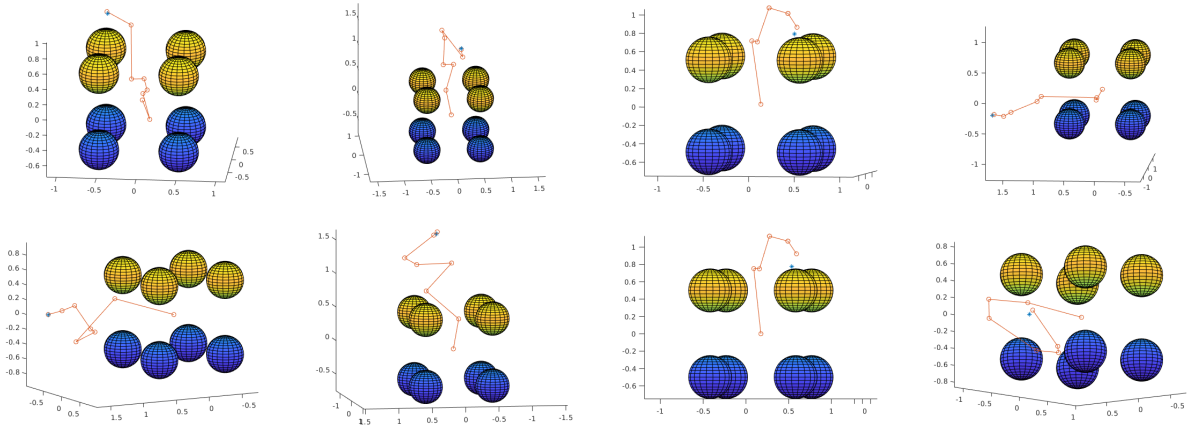

Figure 1: IK with collision avoidance

2 Analytic Derivatives

I did not code this section because I was not able to get the jacobian function in Matlab to work. Specifically, when I try to call jacobian I get the error message "Undefined function or variable 'jacobian'." I have verified that the symbolic math library is indeed installed on my Matlab. Googling for an hour did not lead to any fixes.

If I had access to the jacobian function, I would write the forward kinematics and the collision penalty functions with symbolic math in Matlab. Then, calling the jacobian function would give the analytic gradient. This would then be coded into a gradient function that would be passed into fmincon.

3 Optimizer Comparisons

Figure 2 plots comparison plots of Interior Point (IP), Sequential Quadratic Program (SQP), and Active Set (AS). I didn't experiment with Trust Region method because I couldn't get the analytical

gradeint to work.

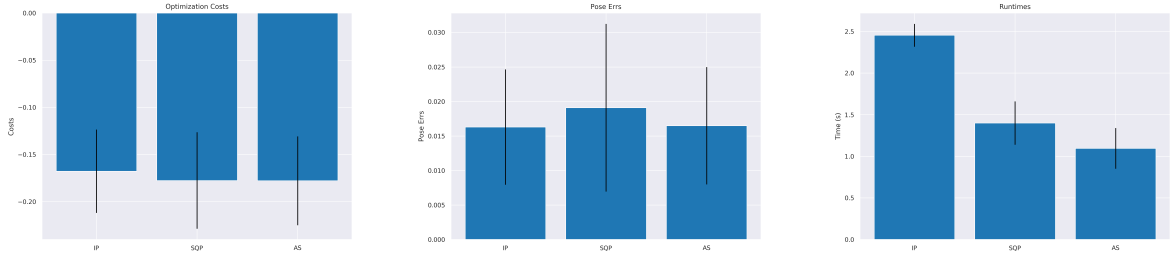

Figure 2: Comparison of 3 optimizers in optimization costs, final pose errors, and runtime. There are no discernable difference between the optimizers except runtime, where AS and SQP significantly outperform IP. Data aggregated across 10 runs. Length of black bar denote one standard deviation.

4 Local Minimums

Local minimums exist when the number of degree of freedoms is greater than 6. I used the case of $N = 7$. To generate different local minimums, choosing different initial conditions would suffice. Below I give the IK solver the same target pose but with random initial conditions sampled from the range of valid joint angles. The local minimum with the lowest cost can then be chosen:

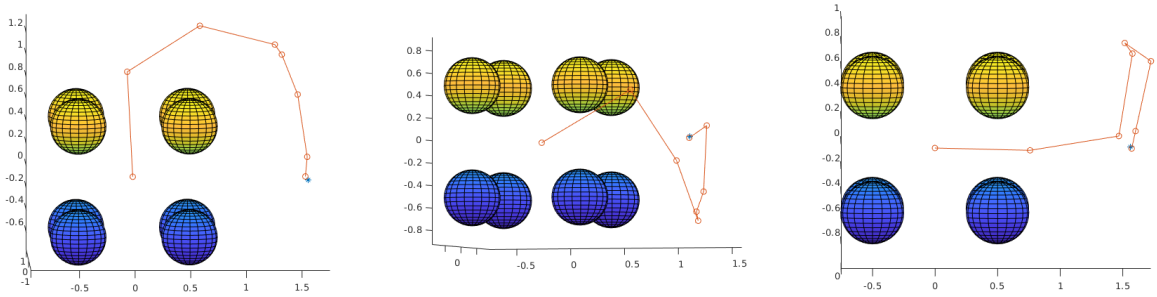

Figure 3: Local Minimums. The pose errors from left to right are $4.5 \times 10^{-5}, 2.6 \times 10^{-5}, 0.2 \times 10^{-5}$