CMU RI 16-745: Dynamic Optimization: Assignment 1

Part 1:

Let f : q ∈ R^n → SE(3). It represents the map of a kinematic chain.

The n means the number of joints and f means the position and orientation of the end tip, which is denoted as G. Here we are trying to find q ∈ R^n such that f (q) = G.

To solve the inverse kinematics problem for a target point, we have to solve the following equation:

T1* A * Ty * B * T2 = G

A = [Ra ta; 000 1]

B = [Rb tb; 000 1]

G = [Rg tg 000 1]

Let R denote a revolute joint.

Here A and B are the transformation matrices from the distal frame of S to the proximal frame F. By extracting the Euler angles from rotation matrices, we can have the values of θs.

The position of S2 relative to S1 is given by

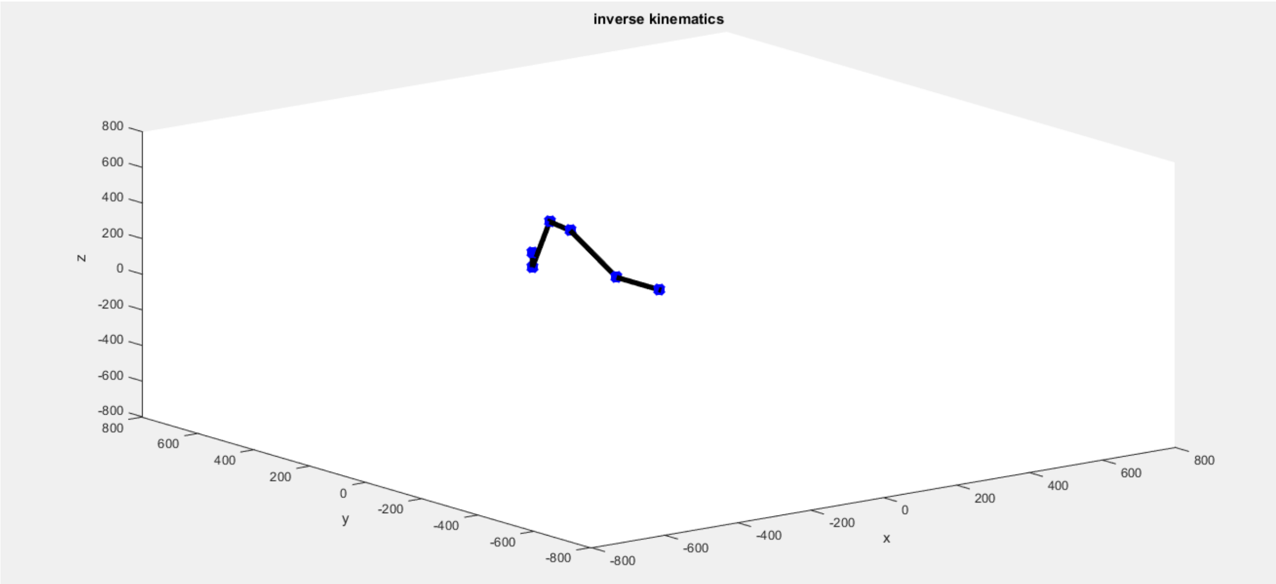
T1*A*Ty*B*T2[0, 0, 0, 1]T(transpose) = R1*Ra*Ry*tb + R1*ta

This equation gives

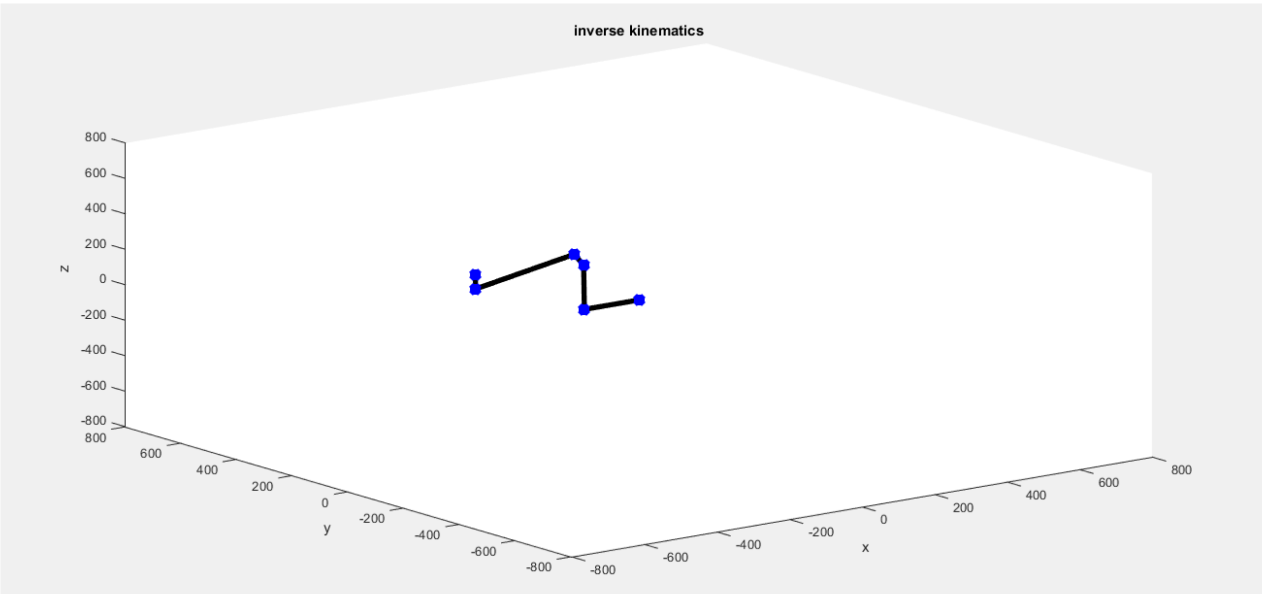
2transpose (ta)* Ra*Ry*tb = transpose (tg)* tg − transpose (ta)* ta − transpose (tb)* tb

Because it is a trigonometric equation of the form a cos(θ) + b sin(θ) = c, we can solve it with straightforward trigonometric methods. Generally speaking we will have two solutions, but given the constraints usually only one of them will work.

For obstacle avoidance, I used a matrix to represent the obstacles, 0 if the obstacle exist, otherwise, the value would be 1. However, the system doesn't work well, probably the unit I used were too large, and therefore the solutions often do not exist.



Goal point [500 200 100]

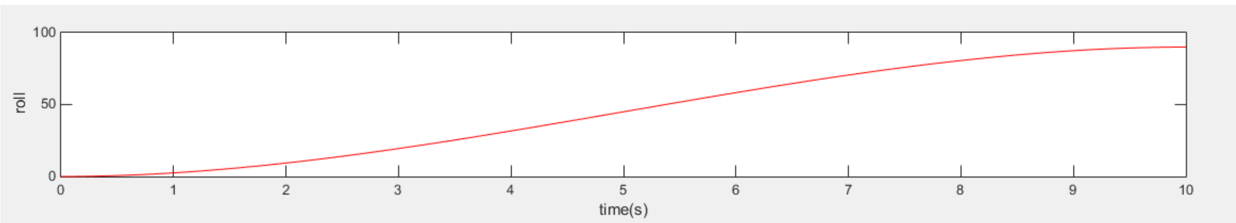


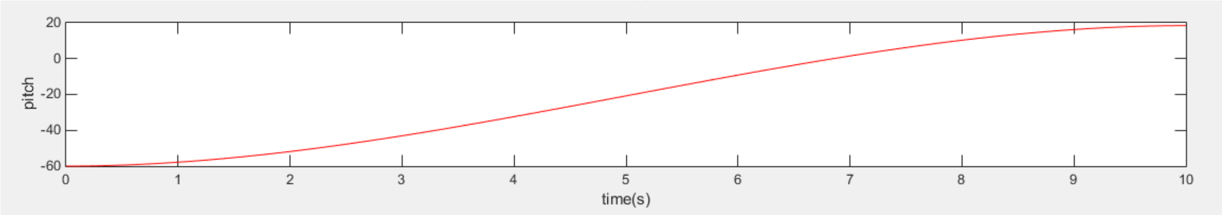
Goal point [600 -100 200]

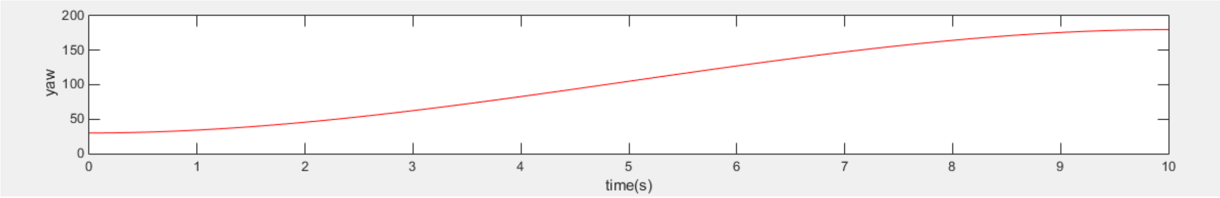
Part 2: Analytic Derivatives:

For each set of adjacent intersection points (ϕj, ϕj+1) in the sequence determine if the corresponding curve segment θ2i(ϕ)ϕj<ϕ<ϕj+1 lies within the range θ2 min and θ2 max. To accomplish this task, we can start by checking the derivative of θ2.

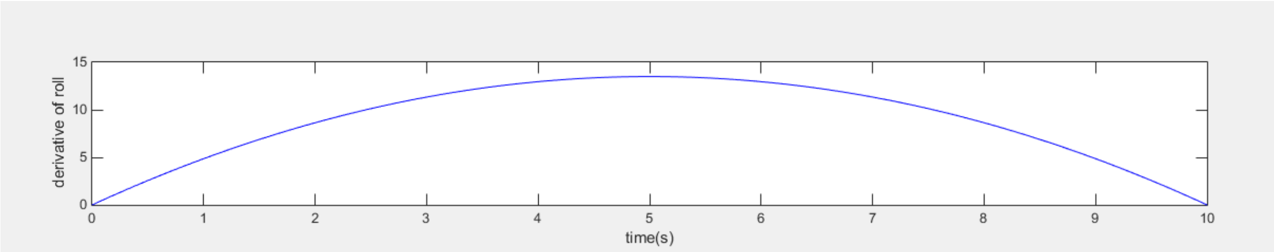
The original angles are list below,

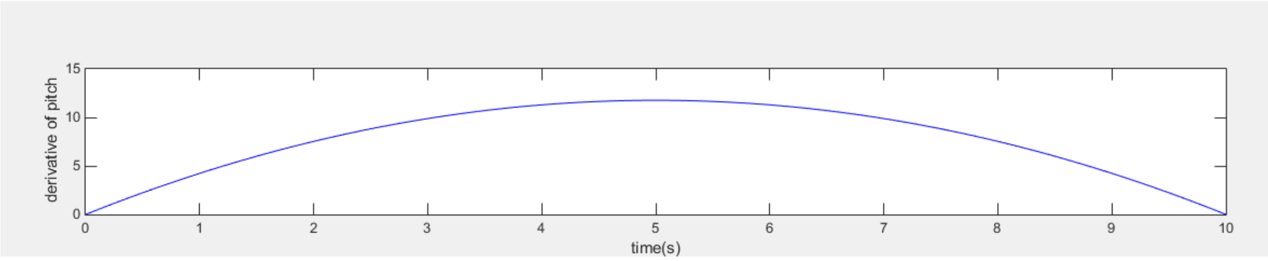


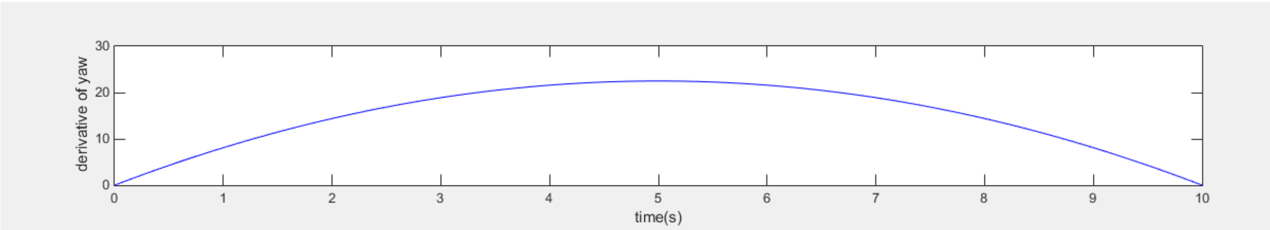




The following are the derivative of angles







By checking the sign of the derivative we can see if the function is increasing or decreasing.

Part 3:

An usual approach is using min f (θ1,...) to solve the problem. Because the constraints are nonlinear, the optimization problem is very difficult to deal with. In these cases, Matlab's fmincon helps a lot.

The outcome is slightly better than the original performance, around 130% faster.

Part 4:

After having the best local minimum, this value could be used to find the 2nd and 3rd best answer. We can filter the best local minimum out and the 2nd best will be chose during the second iteration. With the same concept, 3rd best answer could be found after 2nd best answer was found.

Though I had the idea, the code needs further adjustment to make each part fit.