AuE 8220- Autonomy: Mobility and Manipulation

Homework 3: Interpolation, Inverse Kinematics

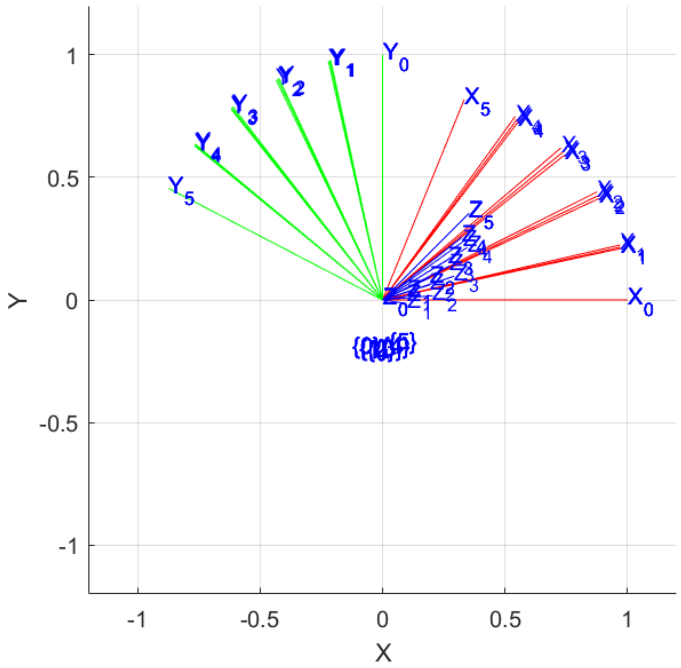
Authors: Tanmay Samak, Chinmay Samak, Riccardo Setti, Olamide Akinyele

NOTE: For running standalone GUIs, navigate to `HW03\Submission\GUI\GUI Executable` and run `Problem_1_GUI.exe` (OR) `Problem_2_GUI.exe`

Problem 1:

GUI: Run `HW03\Submission\GUI\GUI Executable\Problem_1_GUI.exe` for standalone execution. Refer `HW03\Submission\GUI\GUI Source\Problem_1_GUI.m` for source code and `HW03\Submission\GUI\GUI Build` for built packages.

Results: For each of the four cases, we plot the orientation of the intermediate frames at t= [0:2:10] secs on the same graph; note that the initial and final frames overlap exactly in each case (as expected) but the intermediate frames slightly vary based on the interpolation scheme used.



Animation MP4 files are located in `HW03\Submission\Results\Problem 1` directory

1-A

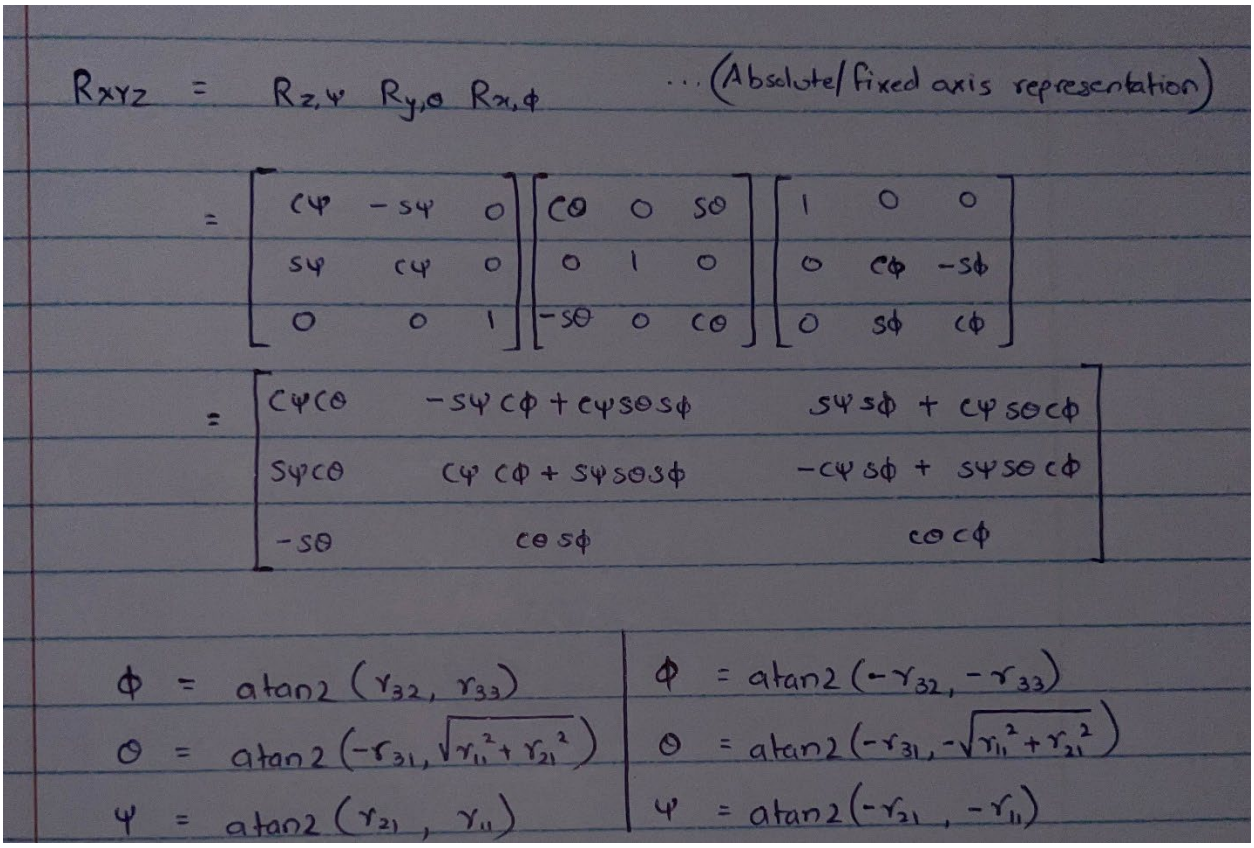
Concept: We linearly interpolate individual angles (relative ZYZ) and then compute rotation matrix for each of the interpolated angles.

Code: `AuE8220_AMM_HW03_F22.m` lines 9-56

GUI: Click on `Solve Problem 1-A` button to generate results

1-B

Concept: We first convert Rzyz relative Euler angle representation to Rxyz (roll, pitch, yaw) absolute/fixed angle representation:



We then choose only one (positive) set of absolute XYZ angles and linearly interpolate them. Finally, we compute rotation matrix for each of the interpolated angles.

Code: `AuE8220_AMM_HW03_F22.m` lines 57-124

GUI: Click on Solve Problem 1-B button to generate results

1-C

Concept: We first compute and convert the initial and final rotation matrices to homogenous transformation matrices using the `r2t()` function, and then interpolate between them using the `trinterp()` function, wherein the rotation is interpolated using quaternion spherical linear interpolation (slerp). Since this is different than interpolating the Euler angles, we get slightly different plots for 1-A and 1-C.

Code: `AuE8220_AMM_HW03_F22.m` lines 125-157

GUI: Click on Solve Problem 1-C button to generate results

1-D

Concept: We first compute and convert the initial and final rotation matrices to RPY representation using the `tr2rpy()` function, and then interpolate between them using the `trinterp()` function, wherein the rotation is interpolated using quaternion spherical linear interpolation (slerp). Since this is different than interpolating the Euler angles, we get slightly different plots for 1-B and 1-D.

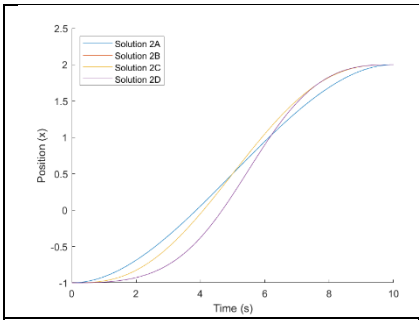
Code: `AuE8220_AMM_HW03_F22.m` lines 158-195

GUI: Click on Solve Problem 1-D button to generate results

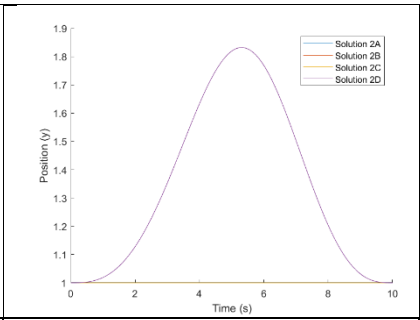
Problem 2:

GUI: Run `HW03\Submission\GUI\GUI Executable\Problem_2_GUI.exe` for standalone execution. Refer `HW03\Submission\GUI\GUI Source\Problem_2_GUI.m` for source code and `HW03\Submission\GUI\GUI Build` for built packages.

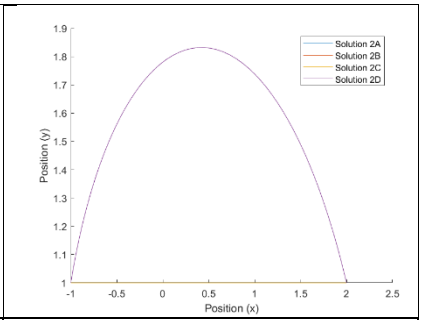
Results: First, for all interpolation schemes, we plot task space variables: (i) x vs. t; (ii) y vs. t; and (iii) y vs. x



x vs. t

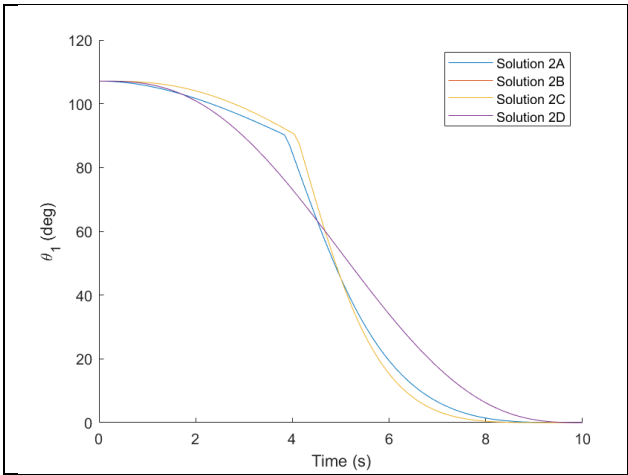


y vs. t

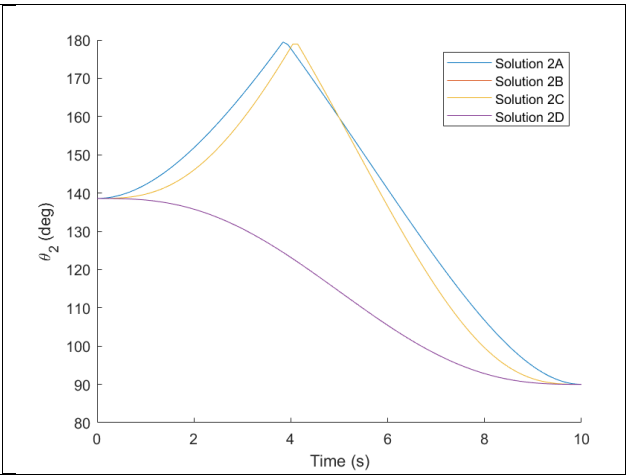


y vs. x

Similarly, for all interpolation schemes, we also plot joint space variables: (iv) θ_1 vs. t; and (v) θ_2 vs. t



θ_1 vs. t

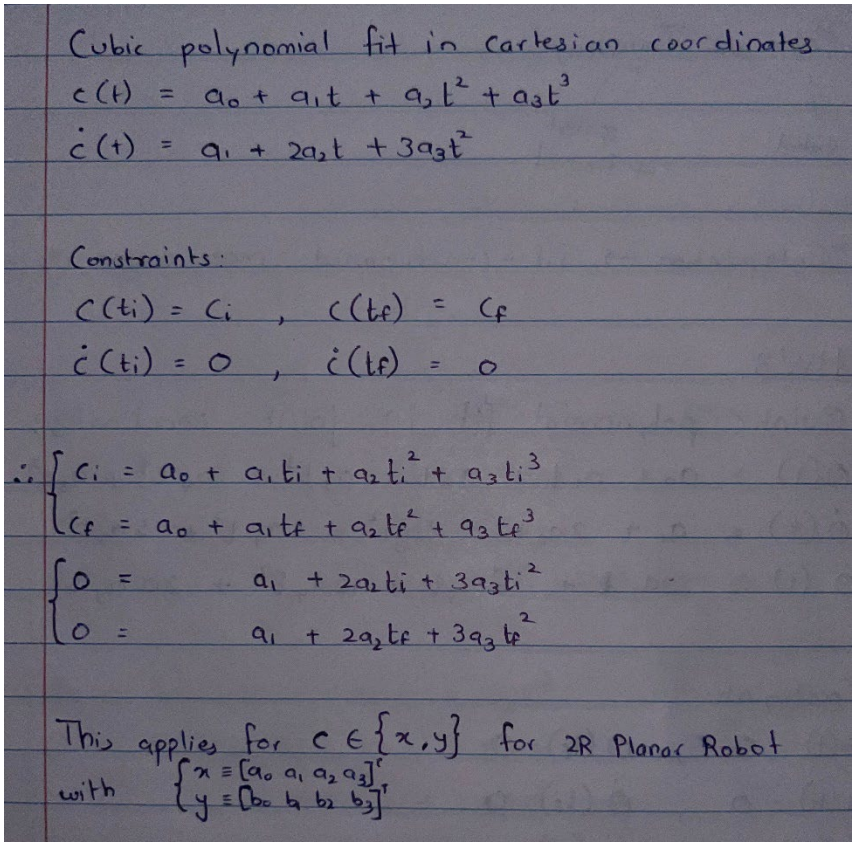


θ_2 vs. t

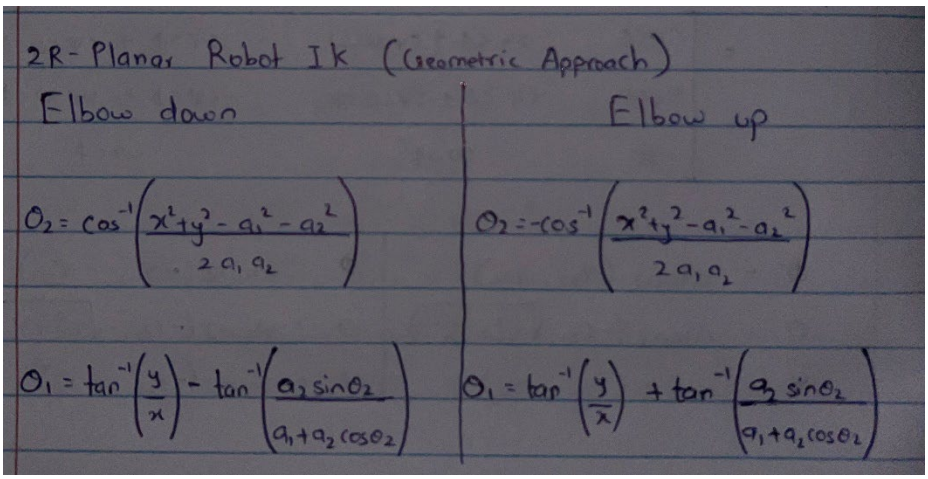
Animation MP4 files are located in `HW03\Submission\Results\Problem 2` directory

2-A

Concept: We perform cubic polynomial interpolation for Cartesian coordinates:



We convert the above equations into matrix representation, and interpolate x and y coordinates individually. Finally, we compute elbow-down inverse-kinematics of 2R planar robot (using geometric approach) for each of the interpolated coordinates.

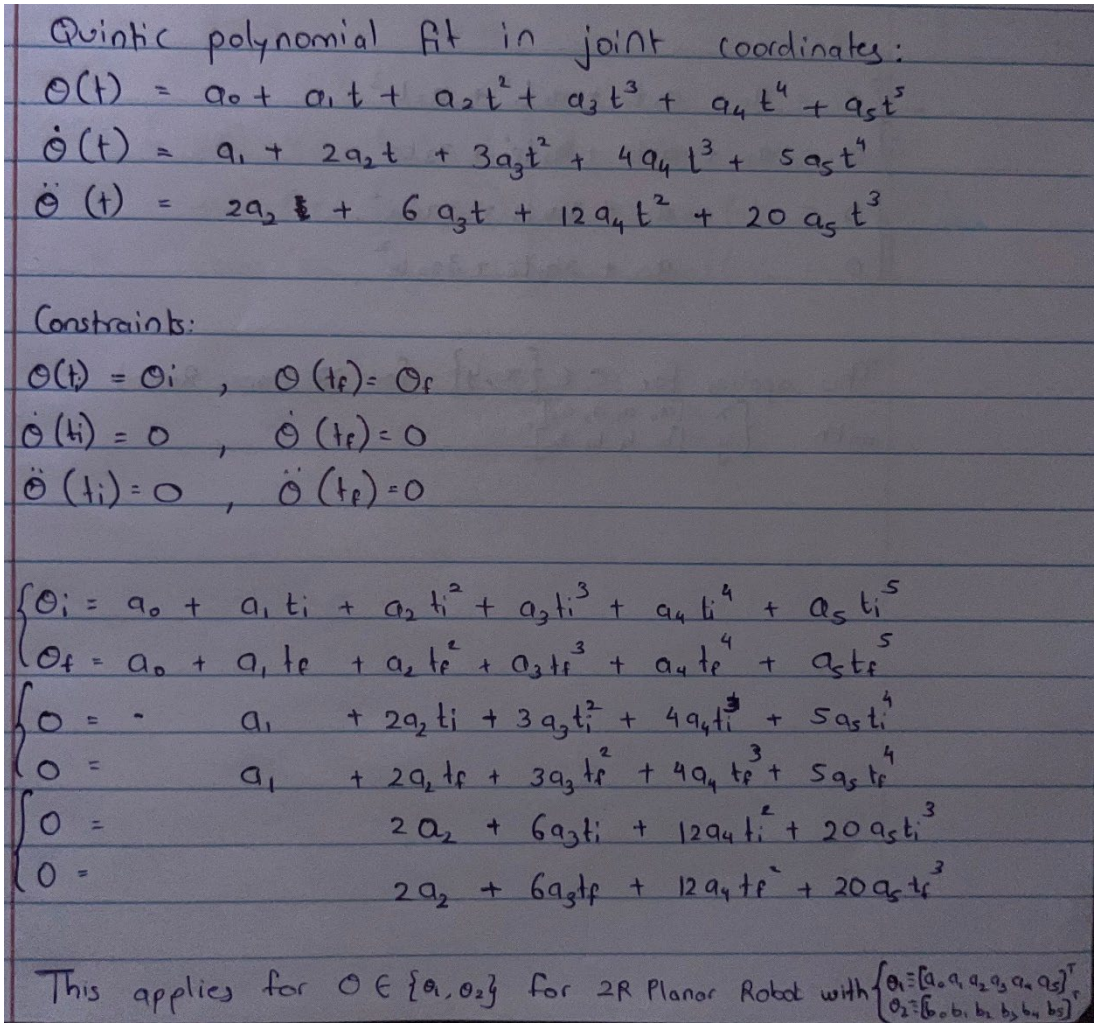


Code: `AuE8220_AMM_HW03_F22.m` lines 196-245; 346-430

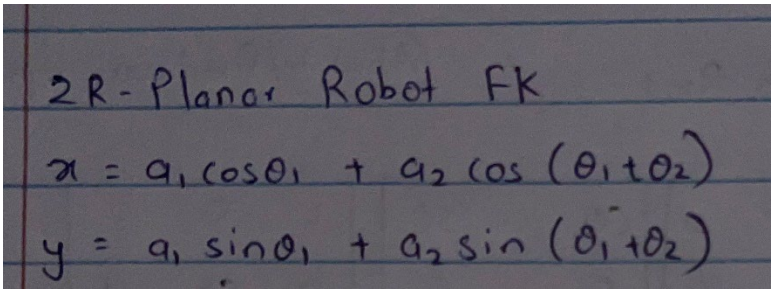
GUI: Click on `Solve Problem 2-A` button to generate results

2-B

Concept: We perform quintic polynomial interpolation for joint coordinates:



We convert the above equations into matrix representation, and interpolate θ_1 and θ_2 individually. Finally, we compute forward-kinematics of 2R planar robot (using geometric approach) for each of the interpolated coordinates.



Code: `AuE8220_AMM_HW03_F22.m` lines 246-302; 346-430

GUI: Click on Solve Problem 2-B button to generate results

2-C

Concept: We perform quintic polynomial interpolation between the Cartesian coordinates using the `tpoly()` function. **Note:** Peter Corke's Robotics Toolbox does not support cubic polynomial interpolation. This causes a slight difference between plots for 2-A and 2-C, since 2-A uses cubic polynomial interpolation which is less smooth than the quintic polynomial interpolation used in 2-C.

Finally, we compute elbow-down inverse-kinematics of 2R planar robot using the `ikine()` function for each of the interpolated coordinates. **Note:** since the `ikine()` function uses numerical approach, we cannot guarantee a particular configuration (elbow-up/down), hence we initialize the joint parameters to the geometrical elbow-down ones calculated in 2-A, to force the solver to solve for elbow-down configuration only.

Code: `AuE8220_AMM_HW03_F22.m` lines 303-324; 346-430

GUI: Click on `Solve Problem 2-C` button to generate results

2-D

Concept: We perform quintic polynomial interpolation between the joint coordinates using the `jtraj()` function. **Note:** there is no visible difference between plots for 2-B and 2-D, since both of these solutions use quintic polynomial interpolation.

Finally, we compute forward-kinematics of 2R planar robot using the `fkine()` function for each of the interpolated coordinates.

Code: `AuE8220_AMM_HW03_F22.m` lines 325-345; 346-430

GUI: Click on `Solve Problem 2-D` button to generate results