AuE 8220- Autonomy: Mobility and Manipulation

Homework 3: Interpolation, Inverse Kinematics

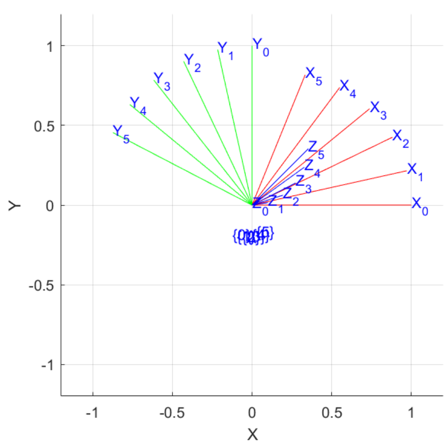
Authors: Tanmay Samak, Chinmay Samak, Riccardo Setti, Olamide Akinyele

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

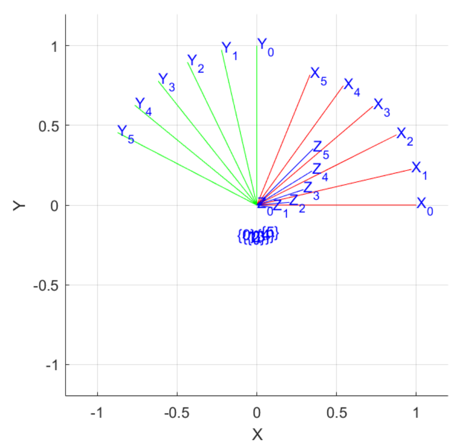
Problem 1:

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.

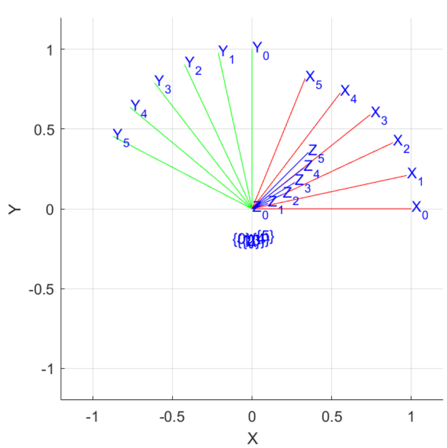
Results: For each of the four cases, we plot the orientation of the intermediate frames at t= [0:2:10] secs on the same graph; note that the initial and final frames overlap exactly in each case (as expected) but the intermediate frames slightly vary based on the interpolation scheme used.



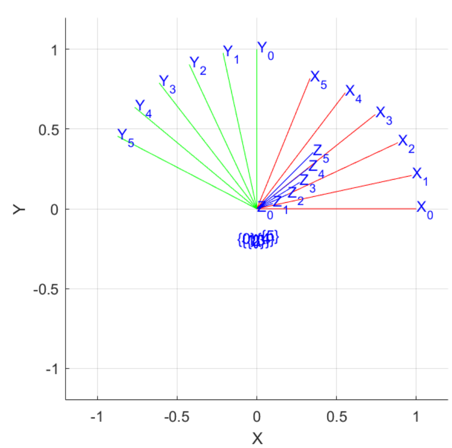
1-A



1-B



1-C



1-D

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:

$$R_{xyz} = R_{z,\psi} R_{y,\theta} R_{x,\phi} \quad \ldots\text{(Absolute/fixed axis representation)}$$

$$= \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix}$$

$$= \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi s\theta c\phi \\ s\psi c\theta & c\psi c\phi + s\psi s\theta s\phi & -c\psi s\phi + s\psi s\theta c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}$$

$$\phi = \operatorname{atan2}(r_{32}, r_{33}) \qquad \phi = \operatorname{atan2}(-r_{32}, -r_{33})$$

$$\theta = \operatorname{atan2}\left(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}\right) \qquad \theta = \operatorname{atan2}\left(-r_{31}, -\sqrt{r_{11}^2 + r_{21}^2}\right)$$

$$\psi = \operatorname{atan2}(r_{21}, r_{11}) \qquad \psi = \operatorname{atan2}(-r_{21}, -r_{11})$$

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.

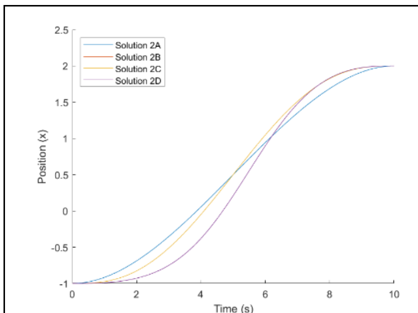
Code: `AuE8220_AMM_HW03_F22.m` lines 158-195

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

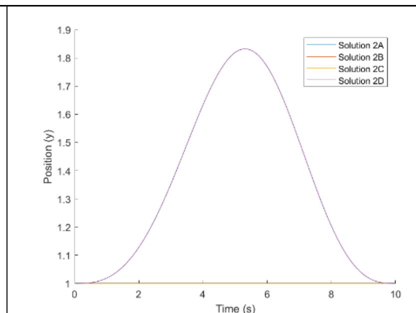
Problem 2:

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.

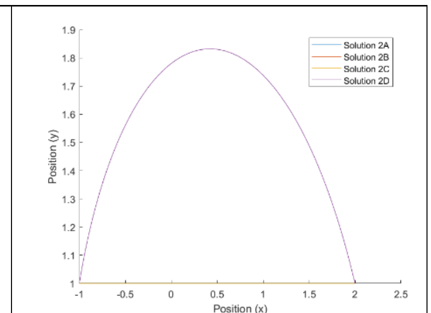
Results: First, for all interpolation schemes, we plot task space variables: (i) x vs. t; (ii) y vs. t; and (iii) y vs. x



x vs. t

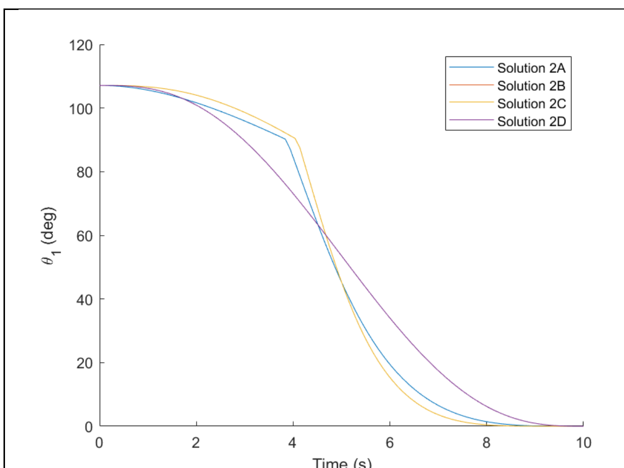


y vs. t

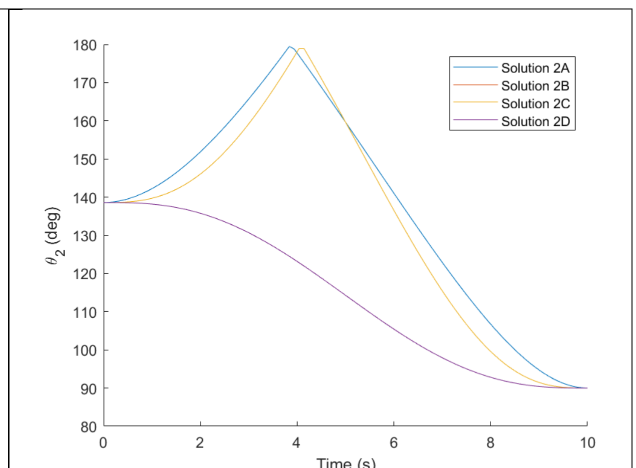


y vs. x

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:

Cubic polynomial fit in Cartesian coordinates

$$c(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

$$\dot{c}(t) = a_1 + 2a_2 t + 3a_3 t^2$$

Constraints:

$$c(t_i) = c_i \quad , \quad c(t_f) = c_f$$

$$\dot{c}(t_i) = 0 \quad , \quad \dot{c}(t_f) = 0$$

$$\therefore \begin{cases} c_i = a_0 + a_1 t_i + a_2 t_i^2 + a_3 t_i^3 \\ c_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \end{cases}$$

$$\begin{cases} 0 = a_1 + 2a_2 t_i + 3a_3 t_i^2 \\ 0 = a_1 + 2a_2 t_f + 3a_3 t_f^2 \end{cases}$$

This applies for $c \in \{x, y\}$ for 2R Planar Robot with $\begin{cases} x \equiv [a_0 \ a_1 \ a_2 \ a_3]^T \\ y \equiv [b_0 \ b_1 \ b_2 \ b_3]^T \end{cases}$

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.

2R-Planar Robot IK (Geometric Approach)

Elbow down	Elbow up
$\theta_2 = \cos^{-1}\left(\dfrac{x^2+y^2-a_1^2-a_2^2}{2a_1a_2}\right)$	$\theta_2 = -\cos^{-1}\left(\dfrac{x^2+y^2-a_1^2-a_2^2}{2a_1a_2}\right)$
$\theta_1 = \tan^{-1}\left(\dfrac{y}{x}\right) - \tan^{-1}\left(\dfrac{a_2\sin\theta_2}{a_1+a_2\cos\theta_2}\right)$	$\theta_1 = \tan^{-1}\left(\dfrac{y}{x}\right) + \tan^{-1}\left(\dfrac{a_2\sin\theta_2}{a_1+a_2\cos\theta_2}\right)$

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:

Quintic polynomial fit in joint coordinates:

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

$$\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4$$

$$\ddot{\theta}(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3$$

Constraints:

$$\theta(t_i) = \theta_i \quad , \quad \theta(t_f) = \theta_f$$

$$\dot{\theta}(t_i) = 0 \quad , \quad \dot{\theta}(t_f) = 0$$

$$\ddot{\theta}(t_i) = 0 \quad , \quad \ddot{\theta}(t_f) = 0$$

$$\begin{cases} \theta_i = a_0 + a_1 t_i + a_2 t_i^2 + a_3 t_i^3 + a_4 t_i^4 + a_5 t_i^5 \\ \theta_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5 \end{cases}$$

$$\begin{cases} 0 = a_1 + 2a_2 t_i + 3a_3 t_i^2 + 4a_4 t_i^3 + 5a_5 t_i^4 \\ 0 = a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4 \end{cases}$$

$$\begin{cases} 0 = 2a_2 + 6a_3 t_i + 12a_4 t_i^2 + 20a_5 t_i^3 \\ 0 = 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3 \end{cases}$$

This applies for $\theta \in \{\theta_1, \theta_2\}$ for 2R Planar Robot with $\begin{cases} \theta_1 = [a_0\ a_1\ a_2\ a_3\ a_4\ a_5]^T, \\ \theta_2 = [b_0\ b_1\ b_2\ b_3\ b_4\ b_5]^T \end{cases}$

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.

2R-Planar Robot FK

$$x = a_1 \cos\theta_1 + a_2 \cos(\theta_1 + \theta_2)$$

$$y = a_1 \sin\theta_1 + a_2 \sin(\theta_1 + \theta_2)$$

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