Homework #1 – ENPM662 – Fall 2024

1. Two-Wheeled Differential Drive Robot Modeling

Write a Python program to plot the 2D trajectory of the center point C of a two-wheeled differential drive robot, given the initial pose (x_i, y_i, θ_i), the left and right wheel angular speeds w_l and w_r, respectively, and the duration T. Assume values for the initial pose, w_l and w_r and T. The wheels have a diameter of 0.6 m, and the distance between the wheels (wheelbase) is 1.2 m.

Assume that:

- The robot experiences no slipping.
- The robot remains perfectly upright with no roll or pitch.

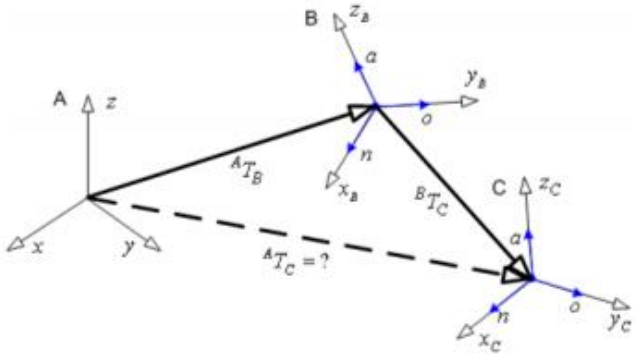
Deliverables for problem 1:

- Python code. No need to submit your code via GitHub. Please ensure that your code is submitted as a separate file, not within a zip archive.
- A read-me file declaring the values for the initial pose and wheels angular velocities considered for the robot.
- Plots showing the 2D trajectory of the center point C.

2. Given two Known frame transformations as

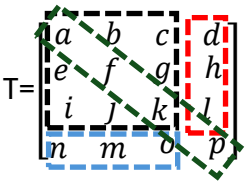
$$ {}^{A}T_{B} = \begin{bmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{B}T_{C} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

Find frame transformation ${}^{A}T_{C}$.



Homework #1 – ENPM662 – Fall 2024

3. For the following given transformation matrix:

T= 

Consider a square ABCD with the following vertices:

A=(0,0),B=(1,0),C=(1,1),D=(0,1)

Select specific numerical values for the elements of the matrix T, then apply the transformation to a square. Demonstrate how each of the four groups of matrix elements affects the square, specifically in terms of rotation, translation, scaling, and perspective transformation. For each group of elements, provide a separate numerical example and illustrate the result by drawing the mapped square after applying the corresponding transformation.

4. Design a Planar RRR Robot (with Three Revolute Joints)

1. **Design and Dimensions**:
    - Design a planar 3-link robot with three revolute joints (RRR configuration).
    - Assign specific dimensions to each link (e.g., lengths l_1, l_2, and l_3).

a. Derive Forward Kinematics (Position and Velocity):

- Using the geometric approach, derive the forward kinematics equations for position (x, and y) of the robot's end-effector and orientation (θ) of the robot's end-effector with respect to horizontal x axis.
- Express the position of the end-effector in terms of joint angles θ_1 , θ_2 , θ_3, as well as l_1, l_2, and l_3.

b. Derive Velocity Kinematics and Inverse Velocity Kinematics:

- Using position kinematics equations, derive the velocity kinematics equations, expressing the end-effector's velocities ($\dot{x}$, $\dot{y}$, and $\dot{\theta}$)
- Then calculate the inverse velocity kinematics equations in matrix format, given the end-effector velocities ($\dot{x}$, $\dot{y}$, and $\dot{\theta}$) and joint values (θ_1 , θ_2 , θ_3) , to calculate $\dot{\theta}_1$, $\dot{\theta}_2$, $\dot{\theta}_3$.

Note: Use Python's SymPy library to compute the derivatives and perform matrix operations.

c. Simulate Circular Trajectory:

- Assume that the robot's end-effector follows a circular trajectory within its workspace, with a radius of 20 cm. The robot should draw a full circle in 30 seconds (hint: use polar coordinates to define the equation of circle) and make sure that the end-effector always remain horizontal (i.e., $\theta = 0$) in this specific scenario.
- Write a python code to compute the joint velocities over time as the end-effector follows this circular path.
- Plot the joint angles and joint velocities as functions of time. Ensure that the robot's workspace constraints are respected in your calculations. (Modify your robot dimension, links' lengths, to ensure the circle is within your robot's workspace)

Deliverables for problem 4:

- Step-by-step derivation of the forward and inverse kinematics equations.
- Python code to simulate and visualize the end-effector's circular trajectory. No need to submit your code via GitHub. Please ensure that your code is submitted as a separate file, not within a zip archive.
- Plots showing the evolution of joint angles and joint velocities over time.