Project #2

Shao Hua, Huang
ICN5534 - Robotics

January 9, 2019

1 Language and Platform Description

1. I use C++ and Python3 to write my program.
   If possible, please use linux machine to run my code.
   (a) C++11 feature is required.
   (b) Eigen: C++ template library for linear algebra
   (c) python packages: numpy and matplotlib
2. Intall Dependencies
   Eigen3: `sudo apt install libeigen3-dev`
   numpy and matplotlib: `pip3 install numpy matplotlib`
3. Run

```
python3 project2.py
```

2 Program Architecture

1. project1.hpp and project1.cc: codes from project1
   - IsValidRange: check if the angle is not NaN and is in valid range
   - MakeVector6: make a 6 x 1 vector in Eigen
   - MakeA: given $(d_n, a_n, \alpha_n, \theta_n)$, make matrix $\mathbf{A}_n$
   - DoTask1: given (n, o, a, p), compute joint variables
   - DoTask2: given joint variables, compute (n, o, a, p) and $(x, y, z, \phi, \theta, \phi)$
   - PrintAnswer: print task1 answer for command line interface
2. project2.py
   - path_planning: path planning algorithm from textbook
   - plot_param: plot position/angle, velocity, accerlation parameters
   - plot_3dpath: plot final 3d path

3 Equations Derivation

$$\begin{cases} a_4 t_{acc}^4 + a_3 t_{acc}^3 + a_2 t_{acc}^2 + a_1 t_{acc} + a_0 = \mathbf{B} + \Delta\,\mathbf{C}\,\dfrac{t_{acc}}{\mathbf{T}} & (1) \\ a_4 t_{acc}^4 - a_3 t_{acc}^3 + a_2 t_{acc}^2 - a_1 t_{acc} + a_0 = \mathbf{B} + \Delta\,\mathbf{B} & (2) \\ 4a_4 t_{acc}^3 + 3a_3 t_{acc}^2 + 2a_2 t_{acc} + a_1 = \dfrac{\Delta\,\mathbf{C}}{\mathbf{T}} & (3) \\ -4a_4 t_{acc}^3 + 3a_3 t_{acc}^2 - 2a_2 t_{acc} + a_1 = -\dfrac{\Delta\,\mathbf{B}}{t_{acc}} & (4) \\ 12a_4 t_{acc}^2 + 3a_3 t_{acc} + 2a_2 = 0 & (5) \\ 12a_4 t_{acc}^2 - 3a_3 t_{acc} + 2a_2 = 0 & (6) \end{cases}$$

$$\text{from (5) and (6)} \Rightarrow a_3 = 0 \tag{7}$$

$$(1)-(2) \Rightarrow a_1 = \frac{1}{2t_{acc}}(\Delta\,\mathbf{C}\,\frac{t_{acc}}{\mathbf{T}} - \Delta\,\mathbf{B}) = \frac{1}{2t_{acc}}(\Delta\,\mathbf{C}\,\frac{t_{acc}}{\mathbf{T}} + \Delta\,\mathbf{B}) - \frac{1}{t_{acc}}\Delta\,\mathbf{B} \tag{8}$$

$$(3)*3 \Rightarrow 12a_4 t_{acc}^2 + 6a_2 = \frac{3}{2t_{acc}^2}(\Delta\,\mathbf{C}\,\frac{t_{acc}}{\mathbf{T}} + \Delta\,\mathbf{B}) \tag{9}$$

$$(9)-(5) \Rightarrow a_2 = \frac{3}{8t_{acc}^2}(\Delta\,\mathbf{C}\,\frac{t_{acc}}{\mathbf{T}} + \Delta\,\mathbf{B}) \tag{10}$$

$$\text{sub (10) to (5)} \Rightarrow a_4 = -\frac{1}{16t_{acc}^4}(\Delta\,\mathbf{C}\,\frac{t_{acc}}{\mathbf{T}} + \Delta\,\mathbf{B}) \tag{11}$$

$$\text{sub to (1)} \Rightarrow a_0 = \frac{3}{16}(\Delta\,\mathbf{C}\,\frac{t_{acc}}{\mathbf{T}} + \Delta\,\mathbf{B}) + \mathbf{B} \tag{12}$$

Let

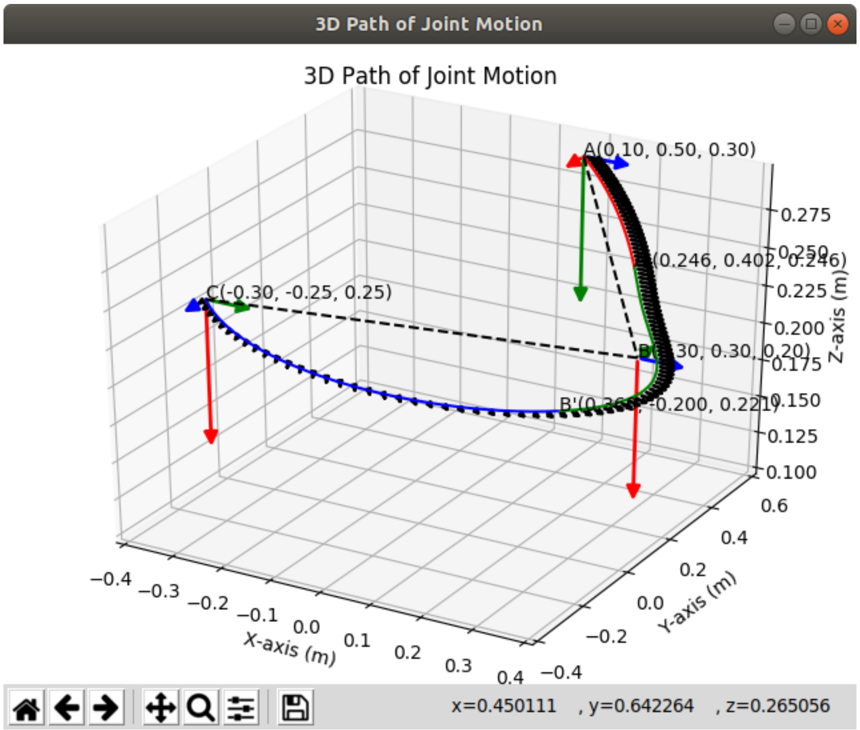
$$h = \frac{t + t_{acc}}{2t_{acc}}$$

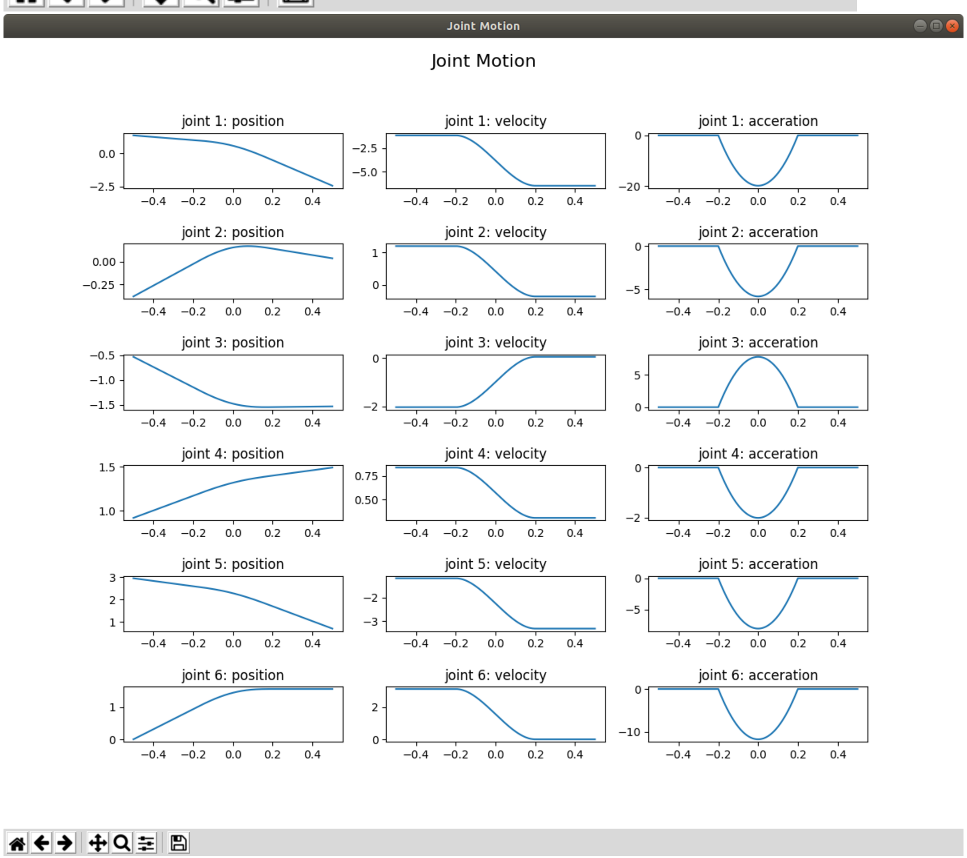
Then,

$$\begin{aligned} q(t) &= (\Delta\,\mathbf{C}\,\frac{t_{acc}}{\mathbf{T}} + \Delta\,\mathbf{B})(-\frac{t^4}{16t_{acc}^4} + \frac{3t^2}{8t_{acc}^2} + \frac{t}{2t_{acc}} + \frac{3}{16}) + \mathbf{B} - \frac{t}{t_{acc}}\Delta\,\mathbf{B} \\ &= (\Delta\,\mathbf{C}\,\frac{t_{acc}}{\mathbf{T}} + \Delta\,\mathbf{B})(2-h)h^3 + \mathbf{B} + (1-2h)\Delta\,\mathbf{B} \\ \dot{q}(t) &= (\Delta\,\mathbf{C}\,\frac{t_{acc}}{\mathbf{T}} + \Delta\,\mathbf{B})(-\frac{t^3}{4t_{acc}^4} + \frac{3t}{4t_{acc}^2} + \frac{1}{2t_{acc}}) - \frac{1}{t_{acc}}\Delta\,\mathbf{B} \\ &= \left[(\Delta\,\mathbf{C}\,\frac{t_{acc}}{\mathbf{T}} + \Delta\,\mathbf{B})(3-2h)h^2 - \Delta\,\mathbf{B}\right]\frac{1}{t_{acc}} \\ \ddot{q}(t) &= (\Delta\,\mathbf{C}\,\frac{t_{acc}}{\mathbf{T}} + \Delta\,\mathbf{B})(-\frac{3t^2}{4t_{acc}^4} + \frac{3}{4t_{acc}^2}) \\ &= (\Delta\,\mathbf{C}\,\frac{t_{acc}}{\mathbf{T}} + \Delta\,\mathbf{B})(1-h)\frac{3h}{t_{acc}^2} \end{aligned}$$

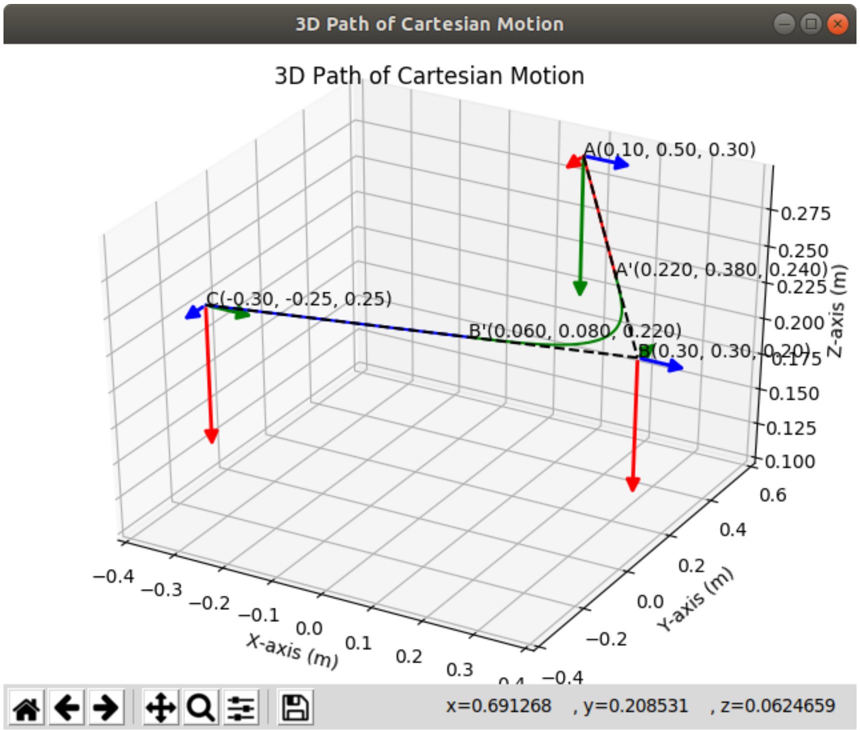
4 Result

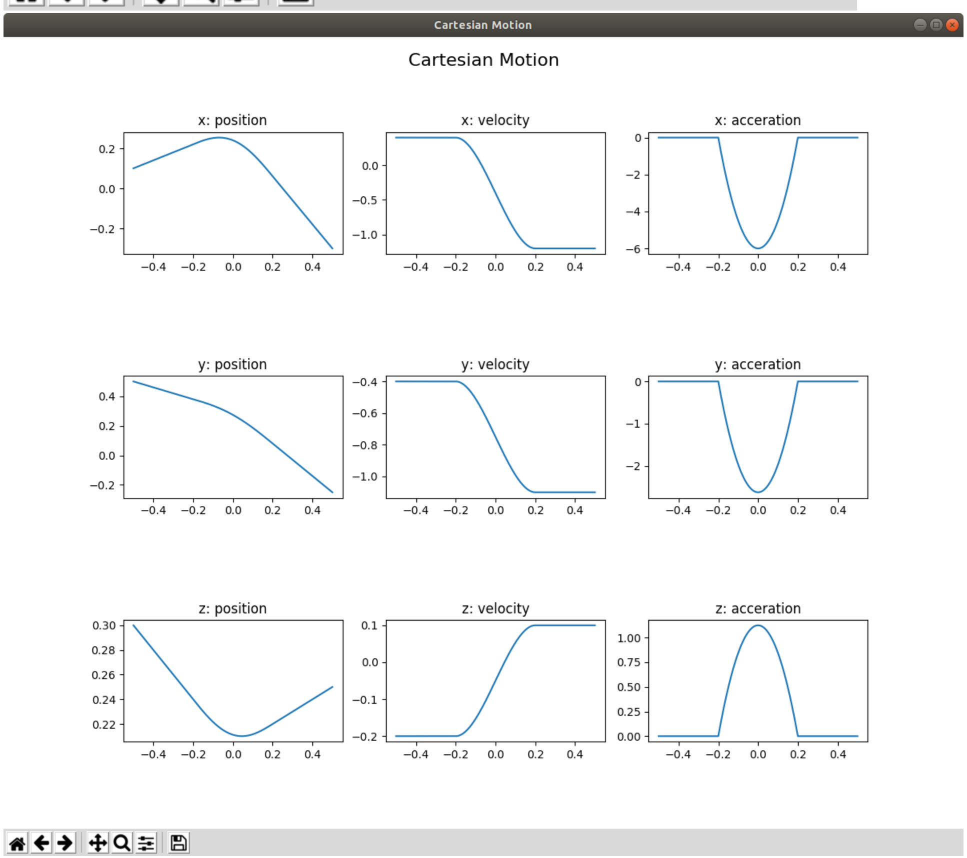
4.1 Joint Motion





4.2 Cartesian Motion





5 Difference Between Joint and Cartesian Motion

5.1 Joint Motion

1. Pros
    - Efficient in computation
    - No singularity problem
    - No configuration problem
    - Minimum time planning
2. Cons: The corresponding Cartesian locations may be complicated.

5.2 Cartesian Motion

1. Pros
    - Motion between path segments and points is well defined.
    - Different constraints, such as smoothness and shortest path, etc., can be imposed upon.
2. Cons
    - Computational load is high.
    - The motion breaks down when singularity occurs.