AER1517 Control for Robotics

Assignment 3: Path Integral Policy Improvement

Prof. Angela Schoellig

Institute for Aerospace Studies
UNIVERSITY OF TORONTO

VECTOR INSTITUTE

DYNAMIC SYSTEMS LAB

General Information

- Download handout and script templates from Quercus
  - Problem 3.1 Quadrotor Agile Maneuver with Path Integral Policy Improvement (PI2)
- Due on April 13 (Monday) 23:59
- Submission through *Gradescope*
  - A single PDF with solutions and requested scripts
  - Both typed and scanned handwritten solutions are accepted
- Office hours: April 2 and April 9 (Thursdays) 14:00 via Zoom
- Send emails for any questions not covered in the office hours

Problem 3.1 Agile Quadrotor Maneuver w/ PI2 | Overview

DYNAMIC SYSTEMS LAB

- Recap: In Assignment 1, we implemented a model-based optimal controller, ILQC, that enabled a quadrotor to fulfill the task of flying to a goal state and/or passing through a predefined via-point
  - Assumed perfect knowledge of system dynamics and saw good performance in simulation
  - In reality, we typically do not have perfect knowledge about the system dynamics, and this often results in sub-optimal performance
- In this assignment, we will explore a learning-based method that allows the quadrotor to achieve similar goals as in Assignment 1 but *not* relying on perfect knowledge of the system dynamics

Problem 3.1 Agile Quadrotor Maneuver w/ PI2 | Overview

DYNAMIC
SYSTEMS LAB

- Path Integral Policy Improvement (PI2)
  - Provides reference trajectory and controller
  - Does not require and use domain knowledge (e.g., model of the system)
  - But, performance relies on good initialization
- Overall idea: Combine model-based approach (i.e., ILQC) and learning-based approach (i.e., PI2)
  - Initialize PI2 with a trajectory and controller obtained from ILQC
  - Improve performance using PI2 algorithm

DYNAMIC SYSTEMS LAB

Initialization in main_p1_pi2.m

- Define task
- Load *nominal* model
- Define cost
- LQR and ILQC design (from Assignment 1)
- Load *test* model (model w/ perturbed parameters)
- Visualization (ILQC)

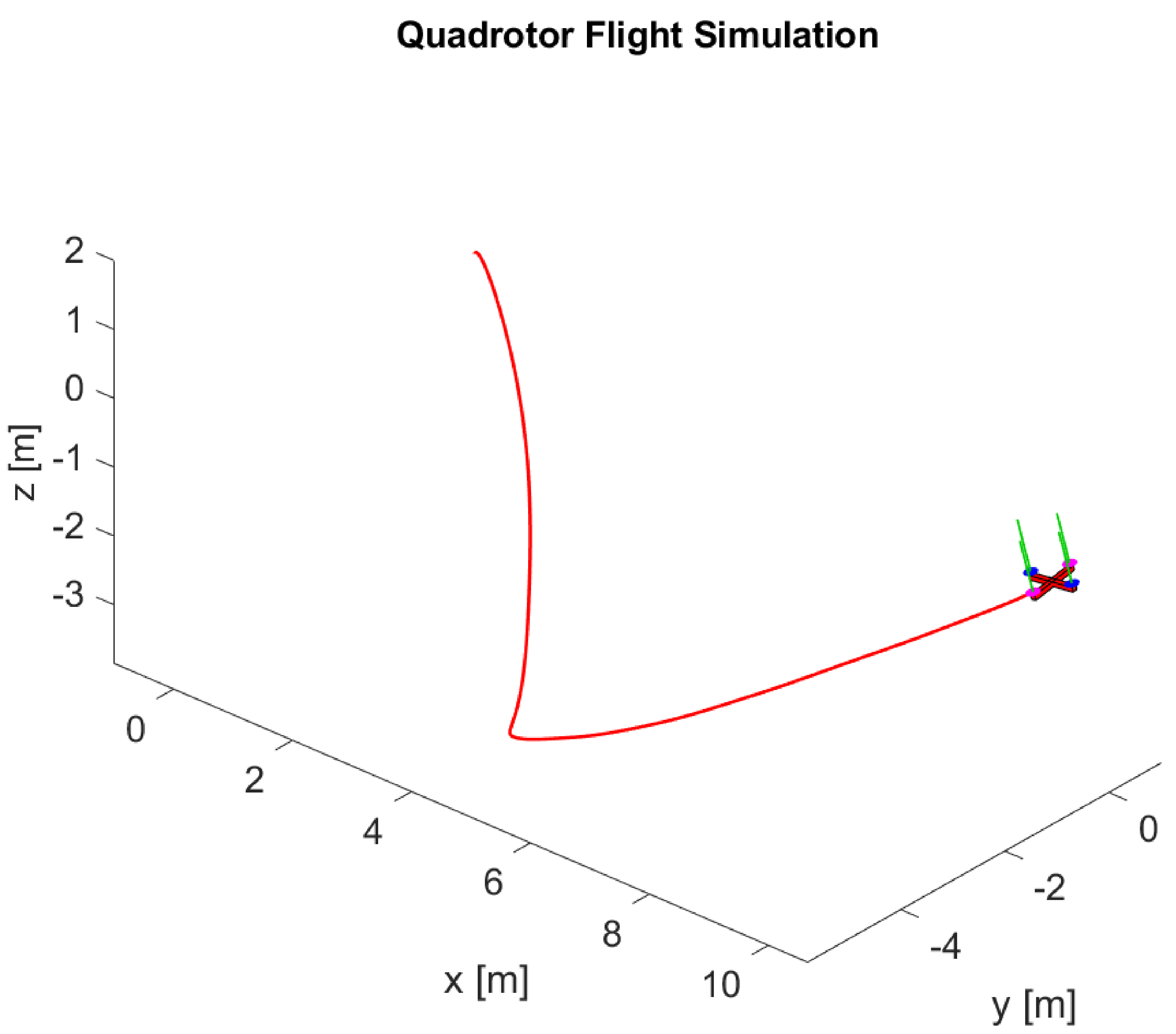
Editor - D:\Git Repos\ta__courses__aer1517\assignments\assignment_3\scripts\for_students\main_p1_pi2.m

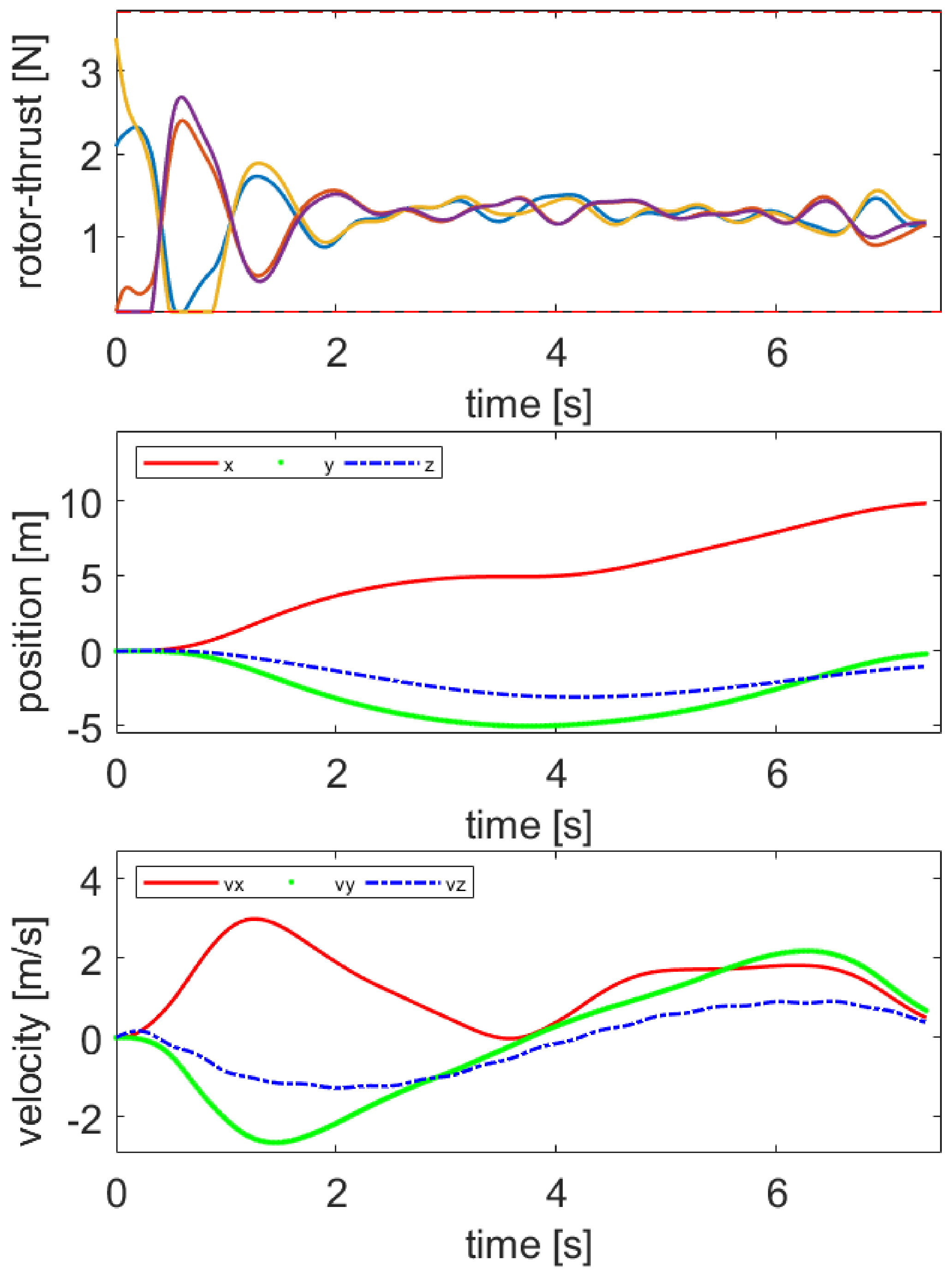
EDITOR PUBLISH VIEW

main_p1_pi2.m

```
%% General
% Add subdirectories
addpath(genpath(pwd));

% Define task
Task = Task_Design();

% Load the nominal model of the quadcopter
load('Quadrotor_Model.mat','Model');

% Define cost function
Task.cost = Cost_Design( Model.param.mQ, Task );

%% Initial controller design
% [Todo] Fill in your LQR_Design and ILQC_Design from Assignment 1 here
% [Initial_Controller, Cost_LQR] = ...;
% [ILQC_Controller, Cost] = ...;

%% Transform controller from ILQC representation to basis function representation
ReducedController = BaseFcnTrans(ILQC_Controller,Task.n_gaussian);

%% Load the perturbed "real" quadrotor model
clear Model;
load('Quadrotor_Model_perturbed.mat','Model_perturbed');

%% Visualization of initial guess with policy in base function representation on perturbed system
Task.noise_insertion_method = '';
Test_sim_out = Sample_Rollout(Model_perturbed, Task, ReducedController);
fprintf('Final Quadcopter Position: xQ = %.3f, yQ = %.3f, zQ = %.3f \n',Test_sim_out.x(1:3,end));
fprintf('Final Quadcopter Velocity: xQ = % .3f, yQ = %.3f, zQ = %.3f \n',Test_sim_out.x(7:9,end));
Visualize(Test_sim_out,Model_perturbed.param,'plot_mode',1);

%% Start PI Learning
% [Todo] Complete PI2_Update implementation, which is called by the
% function PIs_Learning
t_cpu = cputime;
[LearnedController,AllCost,AllController] = PIs_Learning(Model_perturbed,...
```

Problem 3.1 Agile Quadrotor Maneuver w/ PI2 | Example Result





ILQC Initialization

DYNAMIC SYSTEMS LAB

PI2 starter code in main_p1_pi2.m

- PIs_Learning
  (Todo: PI2_Update)
- Visualization (PI2)
- Plot cost

Editor - D:\Git Repos\ta_courses_aer1517\assignments\assignment_3\scripts\for_students\main_p1_pi2.m

EDITOR PUBLISH VIEW

main_p1_pi2.m

```matlab

%% Transform controller from ILQC representation to basis function representation
ReducedController = BaseFcnTrans(ILQC_Controller,Task.n_gaussian);

%% Load the perturbed "real" quadrotor model
clear Model;
load('Quadrotor_Model_perturbed.mat','Model_perturbed');

%% Visualization of initial guess with policy in base function representation on perturbed system
Task.noise_insertion_method = '';
Test_sim_out = Sample_Rollout(Model_perturbed, Task, ReducedController);
fprintf('Final Quadcopter Position: xQ = %.3f, yQ = %.3f, zQ = %.3f \n',Test_sim_out.x(1:3,end));
fprintf('Final Quadcopter Velocity: xQ = % .3f, yQ = %.3f, zQ = %.3f \n',Test_sim_out.x(7:9,end));
Visualize(Test_sim_out,Model_perturbed.param,'plot_mode',1);

%% Start PI Learning
% [Todo] Complete PI2_Update implementation, which is called by the
% function PIs_Learning
t_cpu = cputime;
[LearnedController,AllCost,AllController] = PIs_Learning(Model_perturbed,...
    Task, ReducedController);
t_cpu = cputime - t_cpu;
fprintf('CPU time: %f \n',t_cpu);
fprintf('The PI2 algorithm took %fs to converge \n\n',t_cpu);

%% Visualization of PI2 final trajectory
Task.random=[];
Test_sim_out = Sample_Rollout(Model_perturbed, Task, LearnedController);
fprintf('Final Quadcopter Position: xQ = %.3f, yQ = %.3f, zQ = %.3f \n',Test_sim_out.x(1:3,end));
fprintf('Final Quadcopter Velocity: xQ = %.3f, yQ = %.3f, zQ = %.3f \n',Test_sim_out.x(7:9,end));
Visualize(Test_sim_out,Model_perturbed.param,'plot_mode',1);

%% Cost plot
figure;
plot(AllCost);
xlabel('PI2 iteration number');
ylabel('Cost');
title('Quadrotor PI Learning Curve');
```

script Ln 8 Col 5

PI2 Algorithm

PIs_Learning (Provided)

Algorithm 10 General PI2 Algorithm

given

The cost function $J = \Phi(\mathbf{x}(t_f)) + \int_s^{t_f} \left(q(t, \mathbf{x}), \frac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u}\right) \, dt$

A linear model for function approximation

Initialize $\boldsymbol{\theta}_i$ for each input i

Initialize exploration noise standard deviation c

while maximum number of iteration not reached **do**

Create K rollouts of the system with the perturbed parameter $\boldsymbol{\theta}_i + \boldsymbol{\epsilon}_i$ for each input i, where $\boldsymbol{\epsilon}_{i,j} \sim \mathcal{N}(0, c^2)$

Parameter update

$\Delta\boldsymbol{\theta} \leftarrow$ `PI2_Update(Task, batch_sim_out, batch_cost)`

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \omega\Delta\boldsymbol{\theta}$

Decrease c for noise annealing

end while

PI2_Update (To be Implemented)

for the ith control input **do**

for each time s **do**

Calculate the Return starting from s for the kth rollout

$$R(\tau^k(s)) = \Phi(\mathbf{x}(t_f)) + \int_s^{t_f} \left(q(t, \mathbf{x}), \frac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u}\right) \, dt$$

Calculate α starting from s for the kth rollout

$$\alpha^k(s) = \exp\left(-\frac{1}{\lambda}R(\tau^k(s))\right) / \sum_{k'=1}^{K} R(\tau^{k'}(s))$$

Calculate the time-varying parameter increment $\Delta\boldsymbol{\theta}_i(s)$

$$\Delta\boldsymbol{\theta}_i(s) = \sum_{k=1}^{K} \alpha^k(s) \frac{\boldsymbol{\Upsilon}(s)\boldsymbol{\Upsilon}(s)^T}{\boldsymbol{\Upsilon}(s)^T\boldsymbol{\Upsilon}(s)} \boldsymbol{\epsilon}_i^k(s)$$

end for

for the jth column of $\Delta\boldsymbol{\theta}_i$ matrix, $\Delta\boldsymbol{\theta}_{i,j}$, **do**

Calculate the time-averaged parameter vector

$$\Delta\boldsymbol{\theta}_{i,j} = \left(\int_{t_0}^{t_f} \Delta\boldsymbol{\theta}_{i,j} \circ \boldsymbol{\Upsilon}(s) \, ds\right) \cdot / \int_{t_0}^{t_f} \boldsymbol{\Upsilon}(s) \, ds,$$

where $\circ$ and $\cdot/$ denote element-wise multiplication and division.

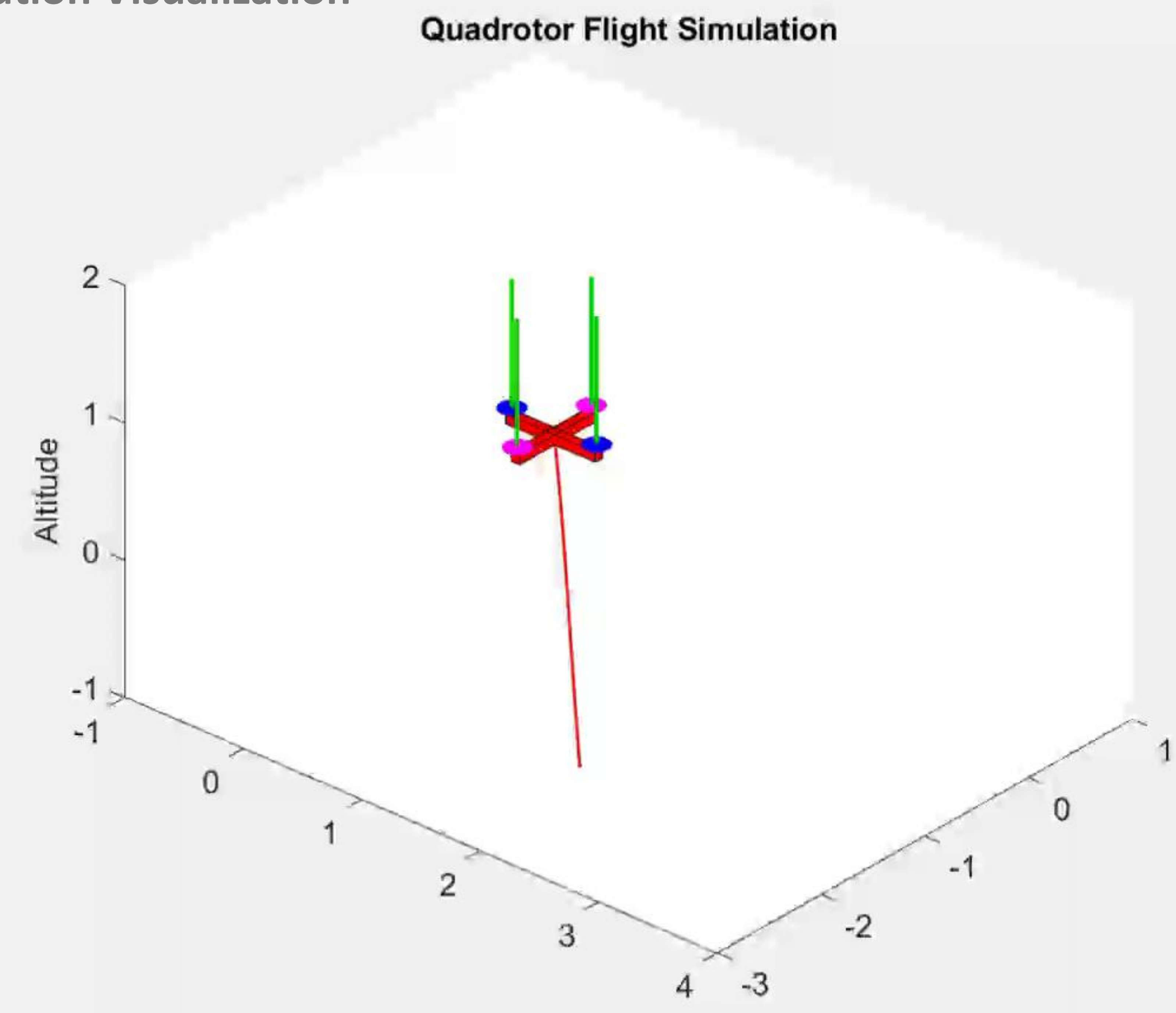
end for

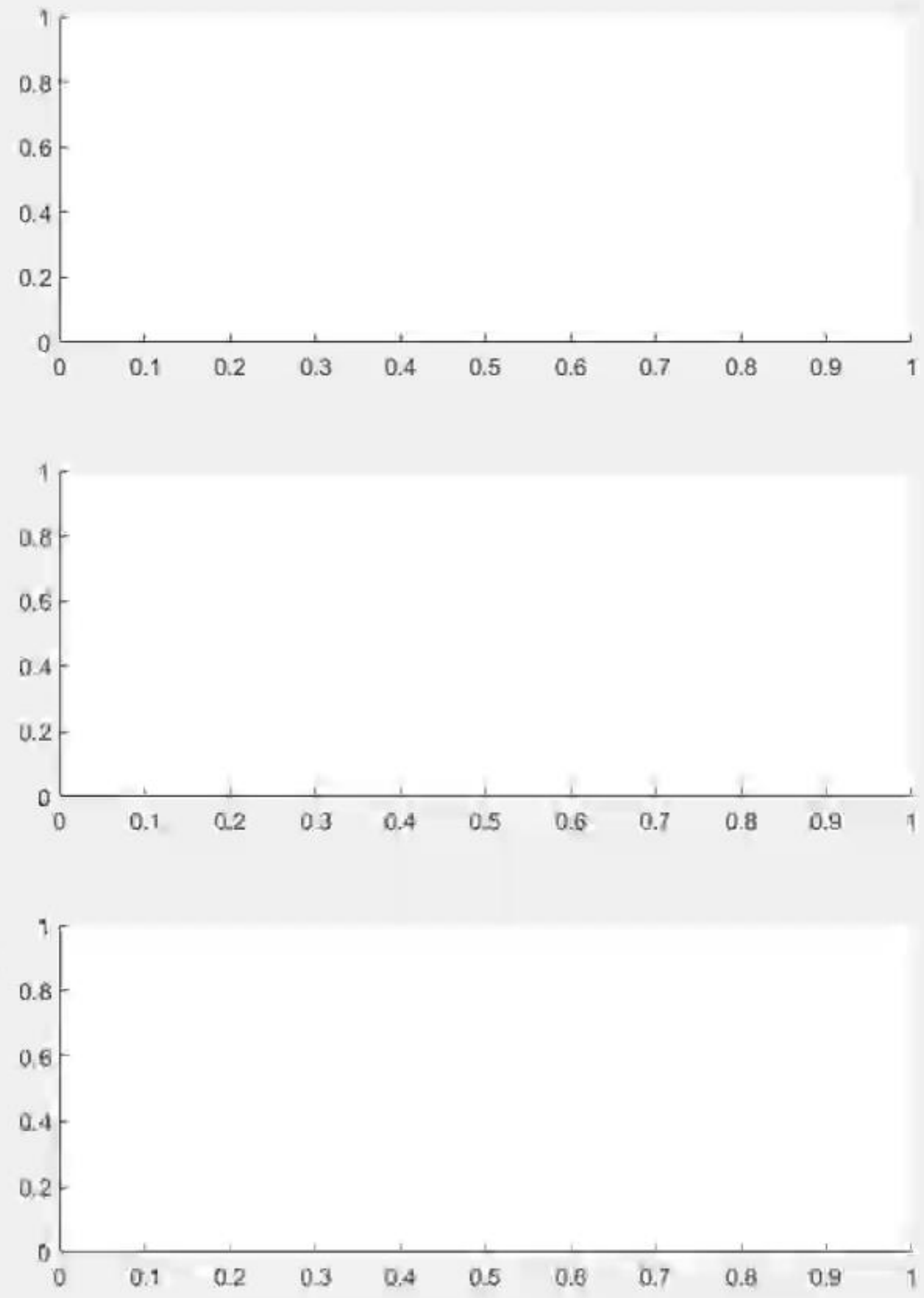
end for

return $\Delta\boldsymbol{\theta}$

Problem 3.1 Agile Quadrotor Maneuver w/ PI2 | Example Result

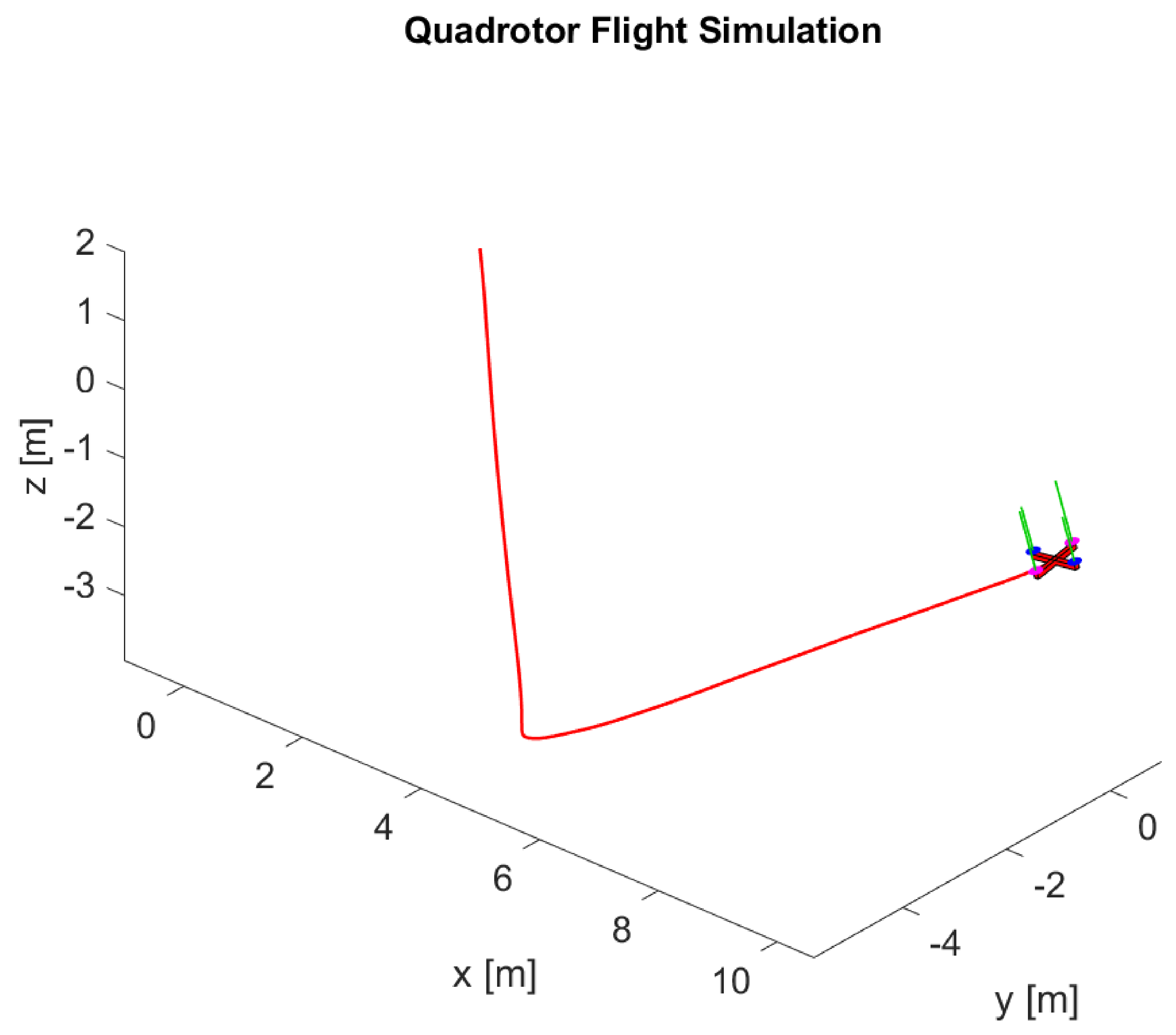
PI2 Solution Visualization

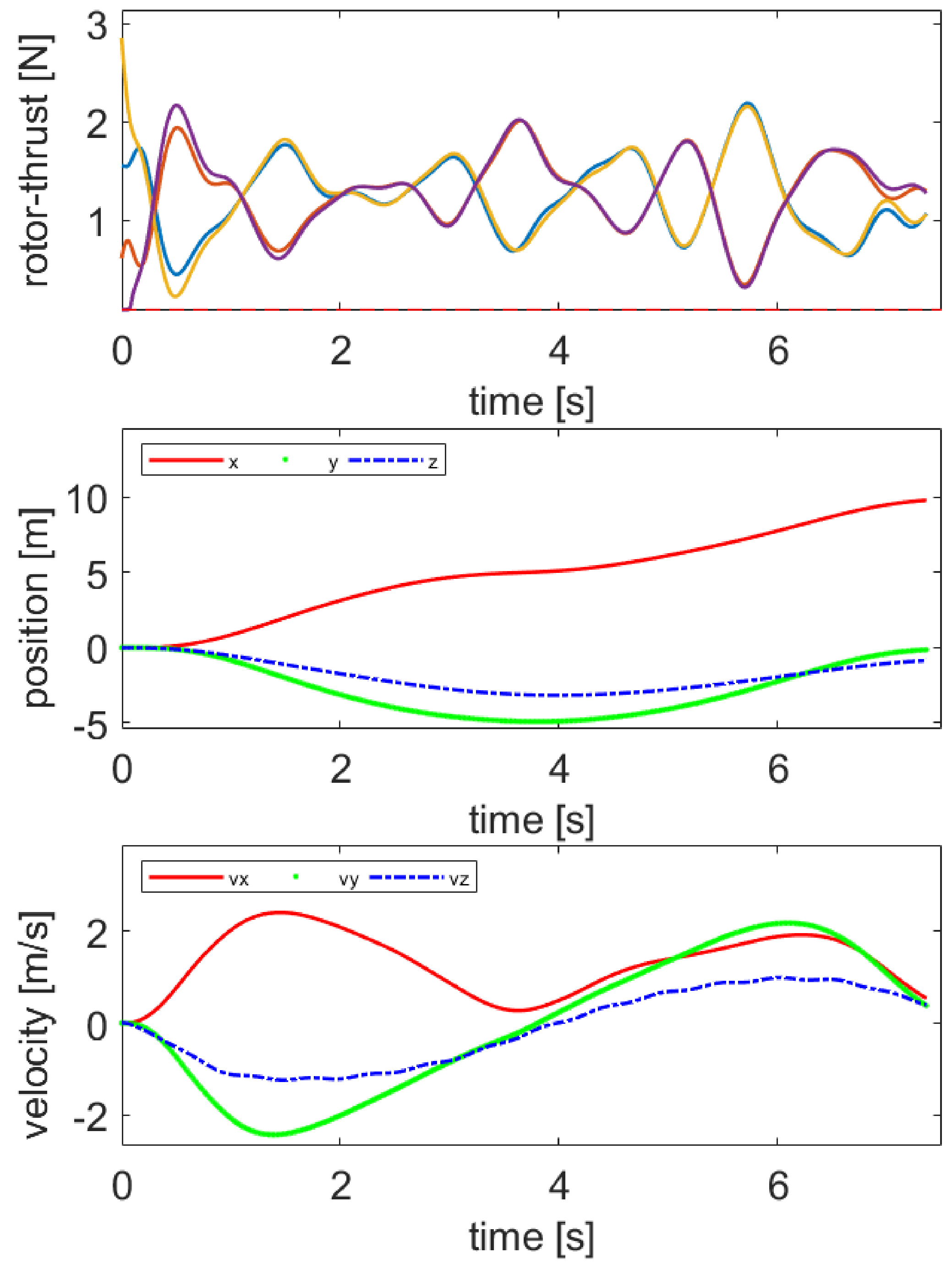




Link to video: https://drive.google.com/file/d/155NH1ptAQOZCHrX4SaGbrKXTPVmkysX3/view?usp=sharing

Problem 3.1 Agile Quadrotor Maneuver w/ PI2 | Example Result

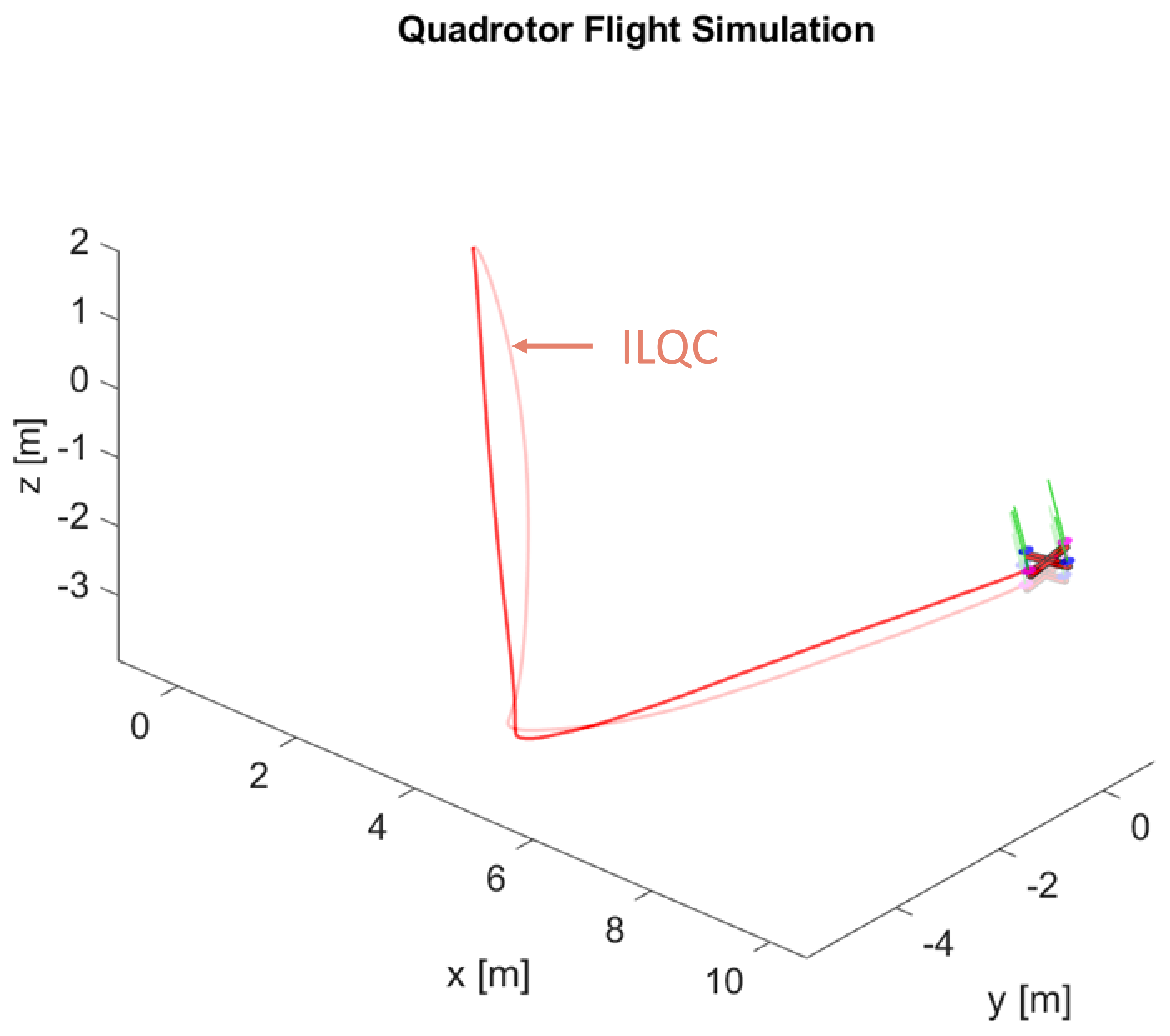




PI2 Solution

Problem 3.1 Agile Quadrotor Maneuver w/ PI2 | Example Result

DYNAMIC SYSTEMS LAB



PI2 Solution

Performance improvement of PI2 controller over 30 iterations