**ECE 481 / 780 T01 Digital Control Systems / Sampled Data Control Systems**

**Gennaro Notomista**

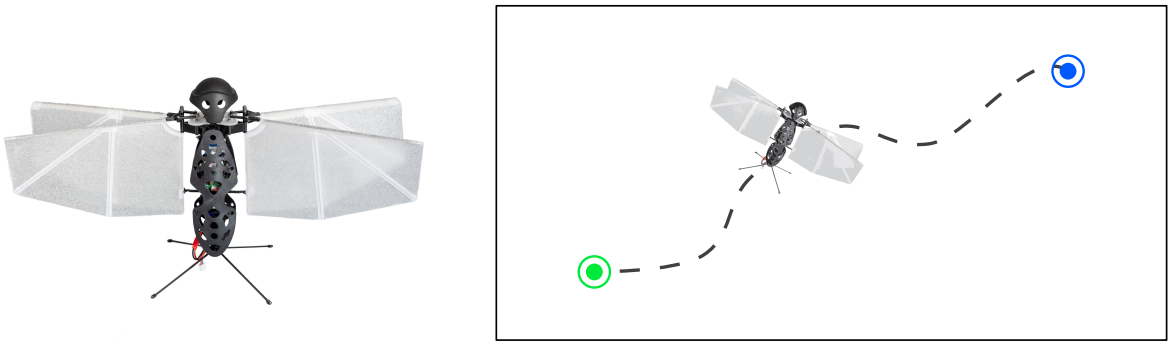
**PROJECT DESCRIPTION**

**Due date: Aug 4, 2025**

# Contents

# 1 ECE 481

## 1.1 Introduction and objective



(a) The flapping wing robot Flapper Nimble+.

(b) The robot tracking a predefined path from a starting location (green) to a goal location (blue).

Figure 1: Flapping wing robot and trajectory tracking scenario of the project.

**Objective**

The objective of this project is state estimation and control of a flapping wing robot. The estimator and controller will be based on eigenvalue placement for discrete time systems.

The approach will consist of the following tasks:

1. Modeling
2. State observer design
3. State feedback control design

## 1.2 Instructions

The robot is modeled using three linear decoupled triple integrators:

$$\begin{cases} \dddot{p}_x = j_x \\ \dddot{p}_y = j_y \\ \dddot{p}_z = j_z, \end{cases} \tag{1}$$

where $p_x$, $p_y$, and $p_z$ are the components of the position of the robot in the world reference frame (see Fig. 2b in the following), and $j_x$, $j_y$, and $j_z$ are the jerk inputs in the respective directions.

**Task 1**

Define the input vector $u = [j_x, j_y, j_z]^T$ and consider the output $y = [p_x, p_y, p_z]^T$. Define a suitable state vector and write the dynamics (1) in discrete time state space form using a sampling time $T = 0.1$s. Highlight the $A$, $B$, $C$, and $D$ matrices.

**Task 2**

Using the model defined in Task 1, design a state observer with the following specifications:

1. Overshoot $O.S.\% \leq 10\%$
2. Rise time $T_r \leq 0.5$s
3. Settling time $T_s^{1\%} \leq 2$s

**Task 3**

Using the model defined in Task 1, define a piecewise-constant-acceleration trajectory between arbitrary initial and final positions:

$$p_d : [t_0, t_1] \to \mathbb{R}^3 : t \mapsto \begin{bmatrix} p_{x,d}(t) \\ p_{y,d}(t) \\ p_{z,d}(t) \end{bmatrix}, \tag{2}$$

with $p_d(t_0)$ and $p_d(t_1)$ being the initial and final robot positions, respectively, and $\dot{p}_d(t_0) = \dot{p}_d(t_1) = \ddot{p}_d(t_0) = \ddot{p}_d(t_1) = 0$.
Design a state feedback controller to track the trajectory defined above with the following specifications:

1. No overshoot
2. Rise time $T_r \leq 1$s
3. Settling time $T_s^{1\%} \leq 5$s

**Notes**

1. The controlled system is a sampled data control system, where the system evolves in continuous time and interacts with the controller only at the sampling time instants
2. The state feedback controller designed in Task 3 must make use of the observed state computed using the algorithm designed in Task 2
3. In the definition of the trajectory to track, consider that the following constraints are enforced both in the simulator and on the real robot:
   - $-1.0 \leq p_x \leq 2.0$
   - $-2.0 \leq p_y \leq 2.0$
   - $0.4 \leq p_z \leq 1.5$
   - $|\dot{p}_x| \leq 0.1$
   - $|\dot{p}_y| \leq 0.1$
   - $|\dot{p}_z| \leq 0.1$
   - $|j_x| \leq 0.01$
   - $|j_y| \leq 0.01$
   - $|j_z| \leq 0.01$

   You do not need to explicitly consider these constraints in the control design process.

### 1.3 Performance evaluation

A successful implementation of the tasks described in the previous section consists of:

- A state observer capable of asymptotically estimating the state of the flapping wing robot, with estimation dynamics fulfilling the desired specifications
- A state feedback controller capable of making the flapping wing robot track a predefined trajectory with the desired specifications
- A report containing the detailed description of the approach adopted to solve the project tasks, alongside illustrative plots and references to the parts of the code which solve each part of the tasks
- The code written to solve the project tasks

Recall that the deliverables related to the project are as follows (including percentage of the overall course grade):

- Final project report: 20%
- Project code: 5%

The format and the structure of the report must be as follows:

- Maximum length: 4 pages
- Format: PDF
- Template: IEEE conference template (`https://www.ieee.org/conferences/publishing/templates.html`)
- Structure
  - Section I: Proposed approach
  - Section II: Results
  - Section III: Discussion

The reports must include also the detailed description of the work carried out by each member of the group, including what sections of the report were written by whom.

# 2 ECE 780 T01

## 2.1 Introduction and objective

**Objective**

The objective of this project is state estimation and Model Predictive Control (MPC) of a flapping wing robot.

The approach will consist of the following tasks:

1. Modeling
2. State observer design
3. MPC design

## 2.2 Instructions

The robot model is given by (1) in Section 1.2.

**Task 1**

Define the input vector $u = [j_x, j_y, j_z]^T$ and consider the output $y = [p_x, p_y, p_z]^T$. Define a suitable state vector and write the dynamics (1) in discrete time state space form using a sampling time $T = 0.1$s. Highlight the $A$, $B$, $C$, and $D$ matrices.

**Task 2**

Using the model defined in Task 1, design a state observer with the following specifications:

1. Overshoot $O.S.\% \leq 10\%$
2. Rise time $T_r \leq 0.5$s
3. Settling time $T_s^{1\%} \leq 2$s

**Task 3**

Using the model defined in Task 1, define a piecewise-constant-acceleration trajectory between arbitrary initial and final positions:

$$p_d : [t_0, t_1] \to \mathbb{R}^3 : t \mapsto \begin{bmatrix} p_{x,d}(t) \\ p_{y,d}(t) \\ p_{z,d}(t) \end{bmatrix}, \tag{3}$$

with $p_d(t_0)$ and $p_d(t_1)$ being the initial and final robot positions, respectively, and $\dot{p}_d(t_0) = \dot{p}_d(t_1) = \ddot{p}_d(t_0) = \ddot{p}_d(t_1) = 0$.
Design a model predictive controller of the form

$$\begin{aligned} \underset{u[0],\dots,u[N-1]}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \left( \|x[k] - x_d[k]\|_Q^2 + \|u[k]\|_R^2 \right) \\ \text{subject to} \quad & x[k+1] = Ax[k] + Bu[k], \\ & x[k] \in \mathcal{X}, \\ & u[k] \in \mathcal{U}, \\ & \qquad\qquad k = 0, \dots, N-1, \end{aligned} \tag{4}$$

where $Q, R \succ 0$, $x_d[k]$ denotes the desired state at time step $k$, and $x[k+1] = Ax[k]+Bu[k]$ is discrete time state space description of the system obtained in Task 1. The controlled system must be able to track the trajectory defined above with the following specifications:

1. No overshoot
2. Rise time $T_r \leq 1$s
3. Settling time $T_s^{1\%} \leq 5$s

**Notes**

1. The controlled system is a sampled data control system, where the system evolves in continuous time and interacts with the controller only at the sampling time instants
2. The model predictive controller designed in Task 3 must make use of the observed state computed using the algorithm designed in Task 2
3. The constraints $x[k] \in \mathcal{X}, u[k] \in \mathcal{U},\ k = 0, \dots, N-1$ in (4) need to be defined to satisfy the following inequalities on state and input:
   - $-1.0 \leq p_x \leq 2.0$
   - $-2.0 \leq p_y \leq 2.0$
   - $0.4 \leq p_z \leq 1.5$
   - $|\dot{p}_x| \leq 0.1$
   - $|\dot{p}_y| \leq 0.1$
   - $|\dot{p}_z| \leq 0.1$
   - $|j_x| \leq 0.01$
   - $|j_y| \leq 0.01$
   - $|j_z| \leq 0.01$

   These constraints need to be considered also in the definition of the trajectory to track.

### 2.3 Performance evaluation

A successful implementation of the tasks described in the previous section consists of:

- A state observer capable of asymptotically estimating the state of the flapping wing robot, with estimation dynamics fulfilling the desired specifications
- A model predictive controller capable of making the flapping wing robot track a predefined trajectory with the desired specifications
- A report containing the detailed description of the approach adopted to solve the project tasks, alongside illustrative plots and references to the parts of the code which solve each part of the tasks
- The code written to solve the project tasks

Recall that the deliverables related to the project are as follows (including percentage of the overall course grade):

- Final project report: 40%
- Project code: 10%

The format and the structure of the report must be as follows:

- Maximum length: 4 pages
- Format: PDF
- Template: IEEE conference template (`https://www.ieee.org/conferences/publishing/templates.html`)
- Structure
  - Section I: Proposed approach
  - Section II: Results
  - Section III: Discussion

The reports must include also the detailed description of the work carried out by each member of the group, including what sections of the report were written by whom.

### 2.4 Alternative for MASc and PhD students

The project will consist of the solution to a problem in the student's research area using the techniques covered during the course.
In addition to the final deliverables, a proposal should be submitted following the instructions below:

- Maximum length: 1 page
- Format: PDF
- Template: IEEE conference template (`https://www.ieee.org/conferences/publishing/templates.html`)
- Structure
  - Section I: Problem description
  - Section II: Novelty and/or impact
  - Section III: How robot dynamics and control techniques play a key role
  - Section IV: Technical challenges
  - Section V: Metric for success
  - Section VI: Timeline
- Deadline: **June 8**

The format and the structure of the final report must be as follows:

- Maximum length: 4 pages

- Format: PDF
- Template: IEEE conference template (`https://www.ieee.org/conferences/publishing/templates.html`)
- Structure
  - Section I: Introduction
  - Section II: Literature review
  - Section III: Materials and methods
  - Section IV: Results
  - Section V: Discussion

A short video (maximum 1 minute) to supplement the results may also be attached.

# 3 Code

The controller must be developed on your PC using Python starting from the files provided on LEARN, under Content/Project/Code.

## 3.1 Robot API

## 3.2 Simulator

You are encouraged to test your functions in simulation first. The test script `test_sim.py` shows how to use the functions provided in the simulator. These are a replica of the functions to interface with the real robot, so that the only line you will need to change to test your controller in the Robohub is the constructor of the `Flapper`. In particular:

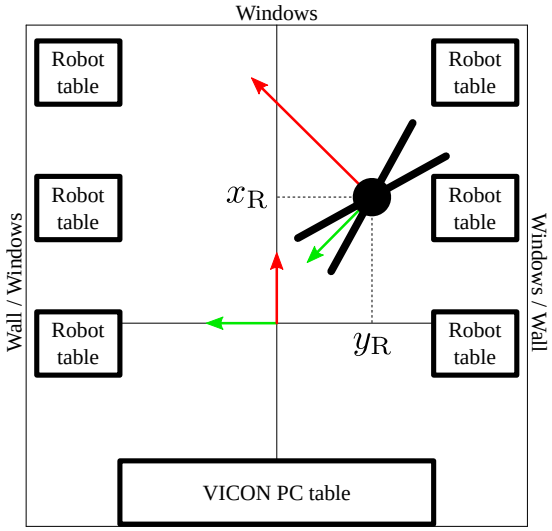
- `f = FlapperSim(robot_pose=np.array([1.0, 2.0, 0.8, 1.57]))`
  initializes the robot at pose $[p_x, p_y, p_z, \psi] = [1.0, 2.0, 0.8, 1.57]$, where $\psi$ is the yaw angle in radians. If `robot_pose` is not specified, the robot will be initialized at a random position and orientation.

- `f.step(u=np.array([1.0, 2.0, 3.0]))`
  moves the platform with an input jerk equal to 1.0, 2.0, and 3.0, in the $x$, $y$, and $z$ directions, respectively. See also the reference frame in Fig. 2b. `u` is a three-dimensional numpy array containing $j_x$, $j_y$, and $j_z$.

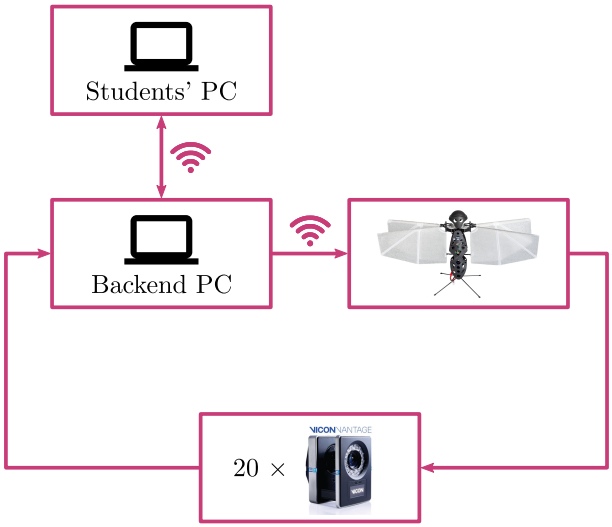
- `y = f.get_output_measurement()`
  returns the noisy measurement of the position of the robot. `y` is a three-dimensional numpy array containing measurements of $p_x$, $p_y$, and $p_z$.

# 4 Robohub schedule

(a) A humanoid, a mobile manipulator, and the ceiling camera system in the Robohub.




(b) Global reference frame in the Robohub (red and green arrows are $x$ and $y$ directions) and local reference frame of the robot (red and green arrows centered at the robot).




(c) Block diagram of the closed-loop control of the flapping wing robot in the Robohub.

Figure 2: The Robohub at University of Waterloo.

The Waterloo Robohub (Fig. 2a) is a collaborative robotics research facility located on the ground floor of Engineering 7. It hosts a diverse fleet of robots (humanoids, quadrupeds, manipulators, ground, and aerial mobile platforms) and it is equipped with an indoor positioning system comprised of a set of 20 Vicon Vantage V5 cameras. The Flapper Nimble+ robot is controlled according to the feedback control loop shown in Fig. 2c.
The schedule to perform project-related activities in the Robohub is reported in the table below (as well as in the course syllabus).

| **Date** | **Time** | **Suggested activity** | |
|---|---|---|---|
| | | **ECE 481** | **ECE 780 T01** |
| Jul 7 | 14:00–17:00 | State observer design / Quidditch | State observer design / Quidditch |
| Jul 14 | 14:00–17:00 | State feedback control design / Quidditch | MPC / Quidditch |
| Jul 21 | 14:00–17:00 | Observed-state feedback control design / Quidditch | Observed-state MPC / Quidditch |
| Jul 28 | 14:00–17:00 | Observed-state feedback control design / Quidditch | Observed-state MPC / Quidditch |