Making Flips with Quadrotors in Constrained Environments

Elie Hatem

Advisors: Dr. Sébastien Briot & Dr. Isabelle Fantoni

19/01/2021

CENTRALE NANTES

Advanced Robotics

1. Introduction
2. System Modeling
3. Control of Quadrotors
4. Multi-Flip Maneuvers with Quadrotors
5. Trajectory Optimization
6. Planning
7. References

1 Introduction

2 System Modeling

3 Control of Quadrotors

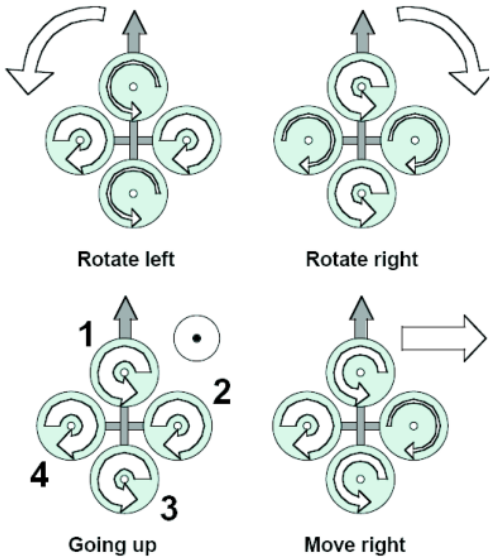
4 Multi-Flip Maneuvers with Quadrotors

5 Trajectory Optimization

6 Planning

7 References

(a) DJI Phantom quadcopter (UAV) [12].



(b) Quadrotor Concept. Width of the arrows is proportional to the angular speed of the propellers [1].

Figure: Commercial quadrtotor platform (left) and quadrotor concept (right).

Properties of the quadrotor:

- Under-actuated system.
- Controls all DOFs.

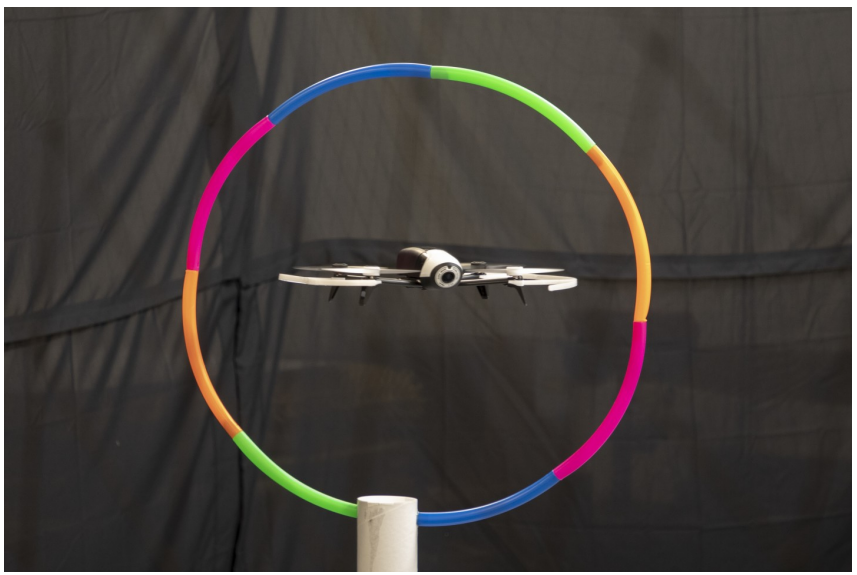
Remark

Rotational and translational dynamics are coupled.

Goal of the master thesis:

- Study of multi-flip maneuvers.
- Use different control methods.
- Performing the maneuvers in a constrained environment.

(a) Quadrotor performing a triple flip.[8]



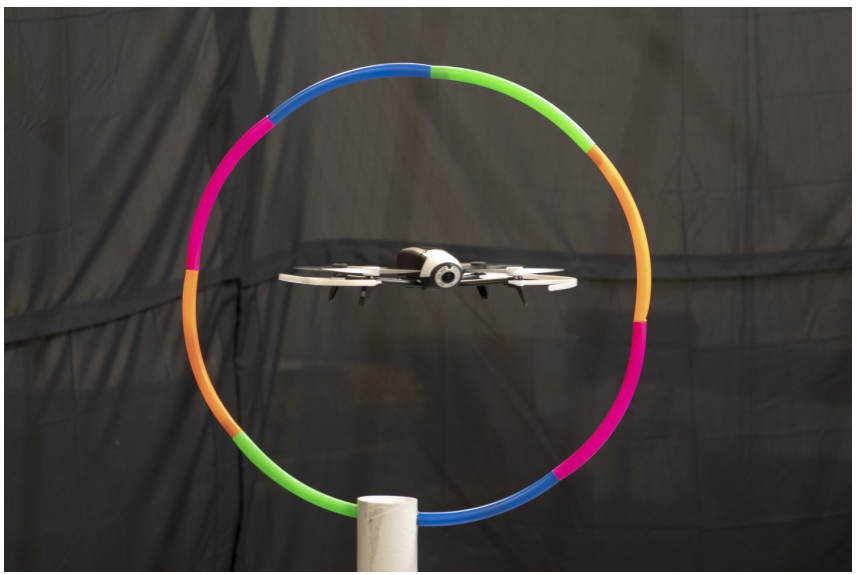
(b) Quadrotor going though a loop [3].

Figure: Representation of the issues to be tackled in this master thesis.

Goal of the master thesis:

- Study of multi-flip maneuvers.
- Use different control methods.
- Performing the maneuvers in a constrained environment.

(a) Quadrotor performing a triple flip.[8]



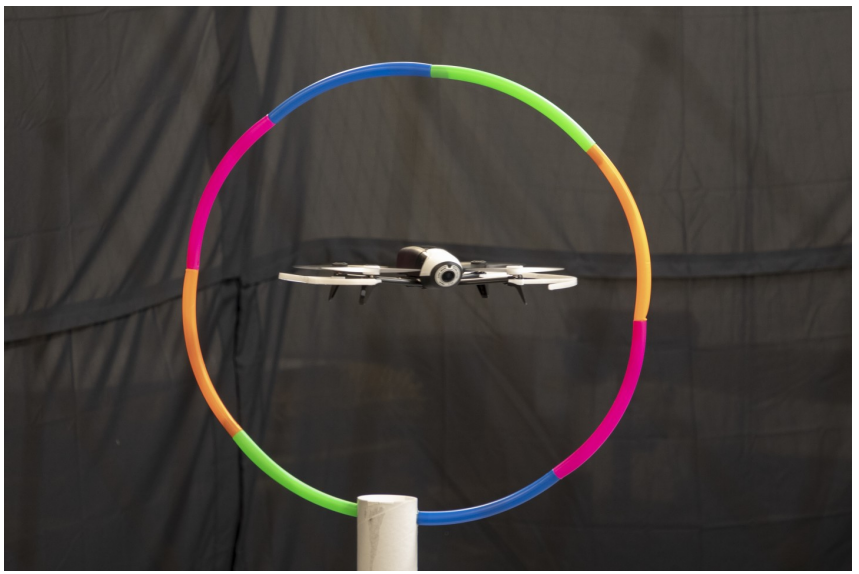
(b) Quadrotor going though a loop [3].

Figure: Representation of the issues to be tackled in this master thesis.

Goal of the master thesis:

- Study of multi-flip maneuvers.
- Use different control methods.
- Performing the maneuvers in a constrained environment.

(a) Quadrotor performing a triple flip.[8]



(b) Quadrotor going though a loop [3].

Figure: Representation of the issues to be tackled in this master thesis.

1 Introduction

2 System Modeling

3 Control of Quadrotors

4 Multi-Flip Maneuvers with Quadrotors

5 Trajectory Optimization

6 Planning

7 References

Principal **forces** and **torques** are generated by the propellers' rotations.

- **Forces** - Decomposed into 2 parts:
  - Weight.
  - Resulting lift.

$$\vec{F_\xi} = \vec{P} + \sum \vec{f_i} \tag{1}$$

which results in [5]:

$$\vec{F_\xi} = {}^{O}\begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + b\,{}^{U}\begin{bmatrix} 0 \\ 0 \\ \omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2 \end{bmatrix} \tag{2}$$

- **Torques** - 2 parts:
  - Relation between the velocities of the rotors.
  - Gyroscopic effects due to the change of orientation direction of the rotors.

$$\tau = \tau_a + \tau_G \tag{3}$$

which results in [5]:

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = \tau_a + \tau_G = \begin{bmatrix} l(f_4 - f_2) - qJ_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ l(f_3 - f_1) + pJ_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix} \tag{4}$$

Motors are controlled with torques and thrust using 4 control inputs.

$$u_1 = u \text{ (thrust)}$$
$$u_2 = \tau_\phi$$
$$u_3 = \tau_\theta$$
$$u_4 = \tau_\psi$$

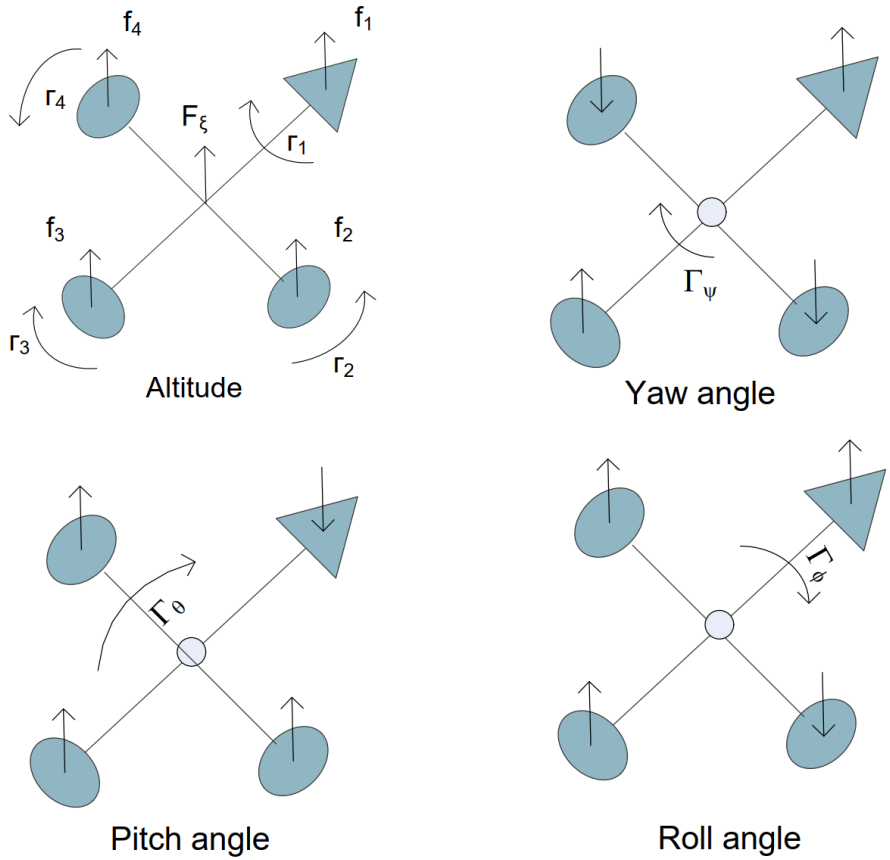


Figure: Quadrotor model with motor torques and Euler angles [5].

First, some assumptions are made:

- Rigid structure.
- Symmetric structure.
- CoG and origin of body-fixed frame coincide.
- Propellers are rigid.
- Thrust and drag are proportional to the square of the velocity of the propeller.

First, some assumptions are made:

- Rigid structure.
- Symmetric structure.
- CoG and origin of body-fixed frame coincide.
- Propellers are rigid.
- Thrust and drag are proportional to the square of the velocity of the propeller.

First, some assumptions are made:

- Rigid structure.
- Symmetric structure.
- CoG and origin of body-fixed frame coincide.
- Propellers are rigid.
- Thrust and drag are proportional to the square of the velocity of the propeller.

First, some assumptions are made:

- Rigid structure.
- Symmetric structure.
- CoG and origin of body-fixed frame coincide.
- Propellers are rigid.
- Thrust and drag are proportional to the square of the velocity of the propeller.

First, some assumptions are made:

- Rigid structure.
- Symmetric structure.
- CoG and origin of body-fixed frame coincide.
- Propellers are rigid.
- Thrust and drag are proportional to the square of the velocity of the propeller.

The dynamic model:

- Computed using **Euler-Lagrange** or **Newton-Euler** formalism.
- Decomposed to **2 different parts** [5]:
  - **Translational** model:

$$\begin{cases} m\ddot{x} = (\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)u \\ m\ddot{y} = (-\sin\phi\cos\psi + \sin\psi\sin\theta\cos\phi)u \\ m\ddot{z} = \cos\theta\cos\phi u - mg \end{cases} \tag{5}$$

  - **Rotational** model:

$$\begin{cases} \dot{p} = qr\frac{(I_{yy}-I_{zz})}{I_{xx}} + \frac{l}{I_{xx}}(f_4 - f_2) - \frac{1}{I_{xx}}qJ_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ \dot{q} = pr\frac{(I_{zz}-I_{xx})}{I_{yy}} + \frac{l}{I_{yy}}(f_3 - f_1) + \frac{1}{I_{yy}}pJ_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ \dot{r} = pq\frac{(I_{xx}-I_{yy})}{I_{zz}} + \frac{1}{I_{zz}}d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{cases} \tag{6}$$

The dynamic model:

- Computed using **Euler-Lagrange** or **Newton-Euler** formalism.
- Decomposed to **2 different parts** [5]:
  - **Translational** model:

$$\begin{cases} m\ddot{x} = (\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)u \\ m\ddot{y} = (-\sin\phi\cos\psi + \sin\psi\sin\theta\cos\phi)u \\ m\ddot{z} = \cos\theta\cos\phi u - mg \end{cases} \tag{5}$$

  - **Rotational** model:

$$\begin{cases} \dot{p} = qr\frac{(I_{yy}-I_{zz})}{I_{xx}} + \frac{l}{I_{xx}}(f_4 - f_2) - \frac{1}{I_{xx}}qJ_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ \dot{q} = pr\frac{(I_{zz}-I_{xx})}{I_{yy}} + \frac{l}{I_{yy}}(f_3 - f_1) + \frac{1}{I_{yy}}pJ_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ \dot{r} = pq\frac{(I_{xx}-I_{yy})}{I_{zz}} + \frac{1}{I_{zz}}d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{cases} \tag{6}$$

The dynamic model:

- Computed using **Euler-Lagrange** or **Newton-Euler** formalism.
- Decomposed to **2 different parts** [5]:
  - **Translational** model:

$$\begin{cases} m\ddot{x} = (\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)u \\ m\ddot{y} = (-\sin\phi\cos\psi + \sin\psi\sin\theta\cos\phi)u \\ m\ddot{z} = \cos\theta\cos\phi u - mg \end{cases} \tag{5}$$

  - **Rotational** model:

$$\begin{cases} \dot{p} = qr\frac{(I_{yy}-I_{zz})}{I_{xx}} + \frac{l}{I_{xx}}(f_4 - f_2) - \frac{1}{I_{xx}}qJ_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ \dot{q} = pr\frac{(I_{zz}-I_{xx})}{I_{yy}} + \frac{l}{I_{yy}}(f_3 - f_1) + \frac{1}{I_{yy}}pJ_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ \dot{r} = pq\frac{(I_{xx}-I_{yy})}{I_{zz}} + \frac{1}{I_{zz}}d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{cases} \tag{6}$$

The dynamic model:

- Computed using **Euler-Lagrange** or **Newton-Euler** formalism.
- Decomposed to **2 different parts** [5]:
  - **Translational** model:

$$\begin{cases} m\ddot{x} = (\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)u \\ m\ddot{y} = (-\sin\phi\cos\psi + \sin\psi\sin\theta\cos\phi)u \\ m\ddot{z} = \cos\theta\cos\phi u - mg \end{cases} \tag{5}$$

  - **Rotational** model:

$$\begin{cases} \dot{p} = qr\frac{(I_{yy}-I_{zz})}{I_{xx}} + \frac{I}{I_{xx}}(f_4 - f_2) - \frac{1}{I_{xx}}qJ_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ \dot{q} = pr\frac{(I_{zz}-I_{xx})}{I_{yy}} + \frac{I}{I_{yy}}(f_3 - f_1) + \frac{1}{I_{yy}}pJ_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ \dot{r} = pq\frac{(I_{xx}-I_{yy})}{I_{zz}} + \frac{1}{I_{zz}}d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{cases} \tag{6}$$

For small pitch and roll angles of order 2, a simplified dynamic model is obtained:

- **Translational** model:

$$\begin{cases} m\ddot{x} = (\phi \sin\psi + \theta \cos\psi)u \\ m\ddot{y} = (-\phi \cos\psi + \theta \sin\psi)u \\ m\ddot{z} = u - mg \end{cases} \tag{7}$$

- **Rotational** model:

$$\begin{cases} \ddot{\phi} = \dot{\theta}\dot{\psi}\frac{(I_{yy}-I_{zz})}{I_{xx}} + \frac{l}{I_{xx}}(f_4 - f_2) - \frac{1}{I_{xx}}\dot{\theta}J_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ \ddot{\theta} = \dot{\psi}\dot{\phi}\frac{(I_{zz}-I_{xx})}{I_{yy}} + \frac{l}{I_{yy}}(f_3 - f_1) + \frac{1}{I_{yy}}\dot{\phi}J_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ \ddot{\psi} = \dot{\theta}\dot{\phi}\frac{(I_{xx}-I_{yy})}{I_{zz}} + \frac{1}{I_{zz}}d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{cases} \tag{8}$$

1 Introduction

2 System Modeling

3 Control of Quadrotors

4 Multi-Flip Maneuvers with Quadrotors

5 Trajectory Optimization

6 Planning

7 References

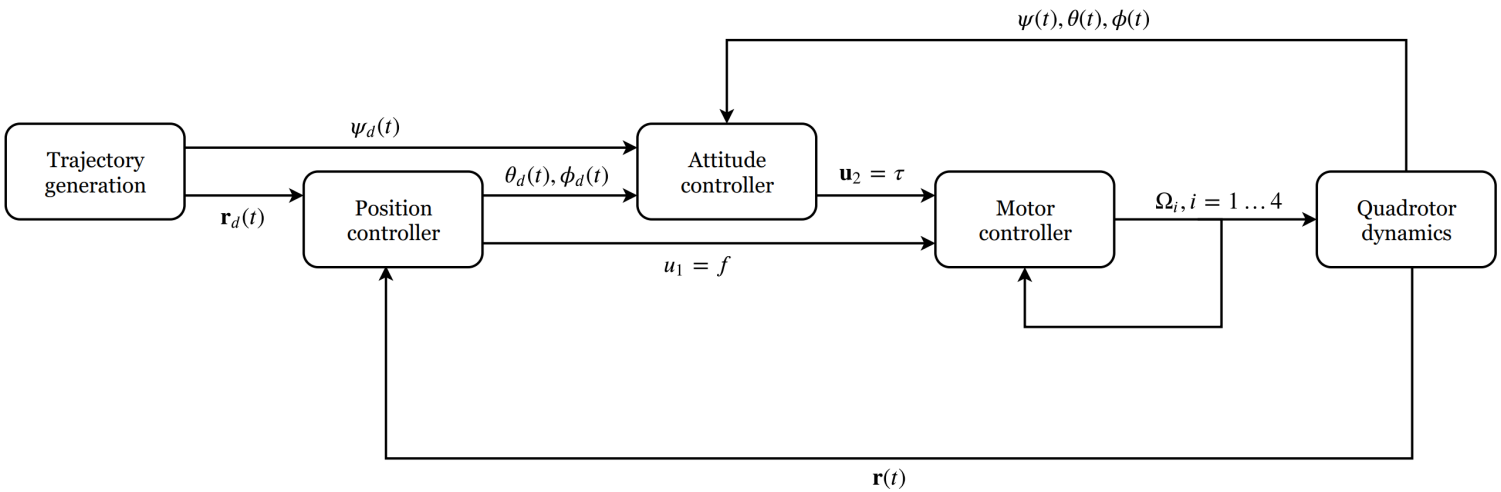


Figure: General control architecture for the position and attitude of a quadrotor [4].

- **Position controller**: drives translational dynamics errors to 0.
- **Attitude controller**: drives rotational dynamics errors to 0.
- **Motor controller**: receives the control intputs $\boldsymbol{u} = \begin{bmatrix} f & \boldsymbol{\tau} \end{bmatrix}^\top$ and maps them to Ω_i^* for each rotor.

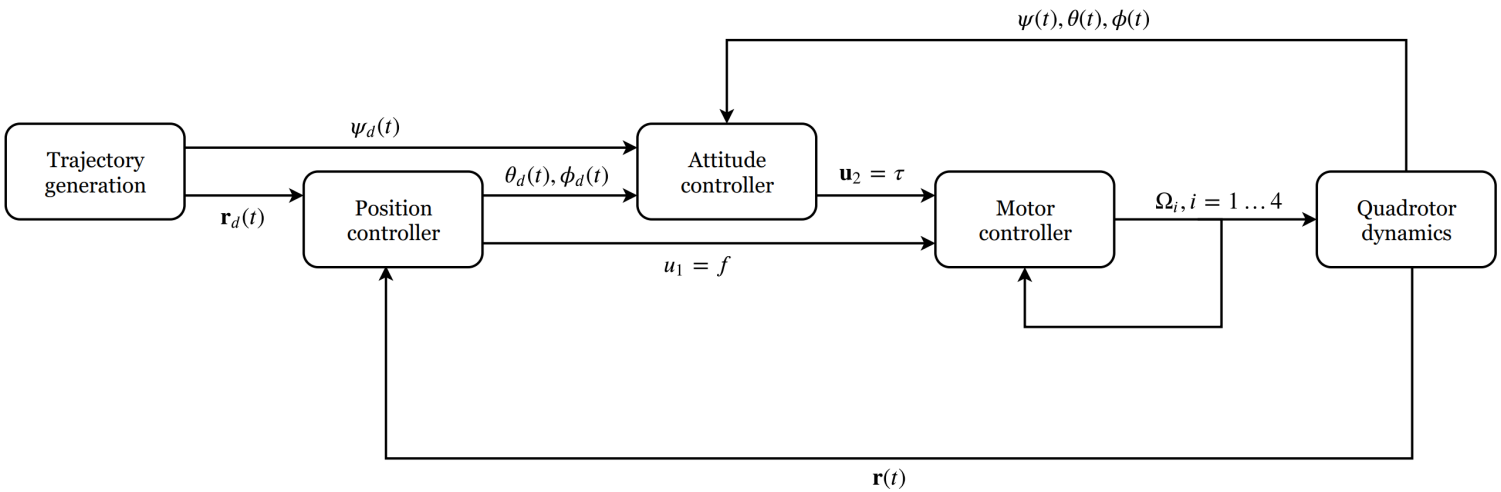


Figure: General control architecture for the position and attitude of a quadrotor [4].

- **Position controller**: drives translational dynamics errors to 0.
- **Attitude controller**: drives rotational dynamics errors to 0.
- **Motor controller**: receives the control intputs $\boldsymbol{u} = \begin{bmatrix} f & \tau \end{bmatrix}^\top$ and maps them to Ω_i^* for each rotor.

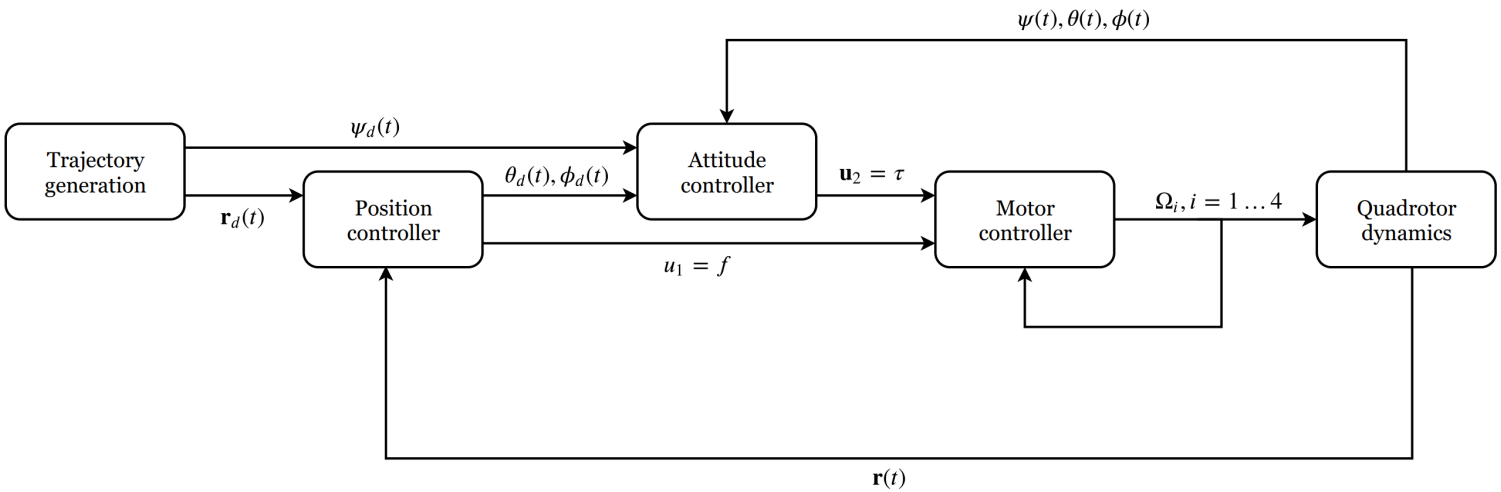


Figure: General control architecture for the position and attitude of a quadrotor [4].

- **Position controller**: drives translational dynamics errors to 0.
- **Attitude controller**: drives rotational dynamics errors to 0.
- **Motor controller**: receives the control intputs $\boldsymbol{u} = \begin{bmatrix} f & \boldsymbol{\tau} \end{bmatrix}^\mathsf{T}$ and maps them to Ω_i^* for each rotor.

What is MPC?

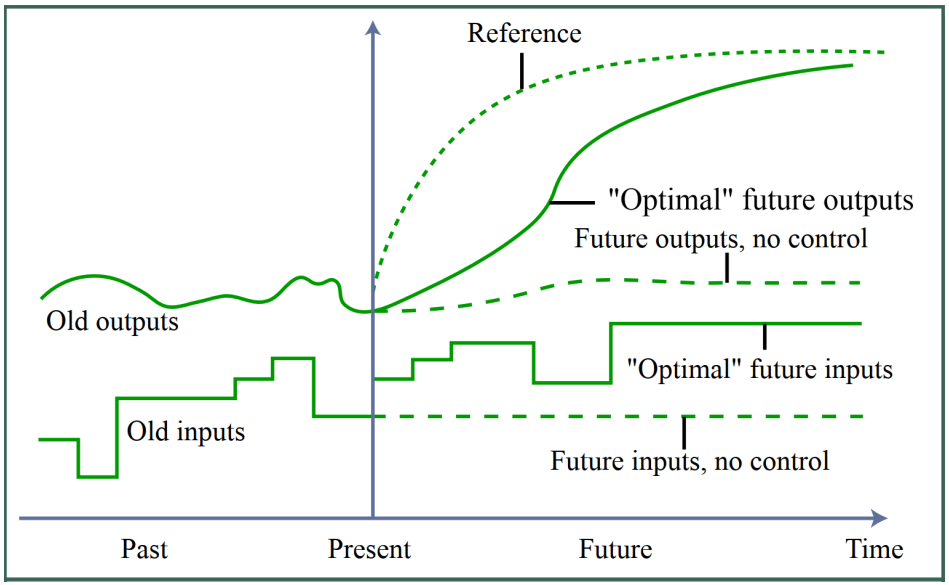


Figure: Basic idea of MPC [6].

- It is a **feedback control** algorithm.
- It uses a model to **precit** future outputs.
- It **solves an online optimization problem** to select the optimal control.

What is MPC?

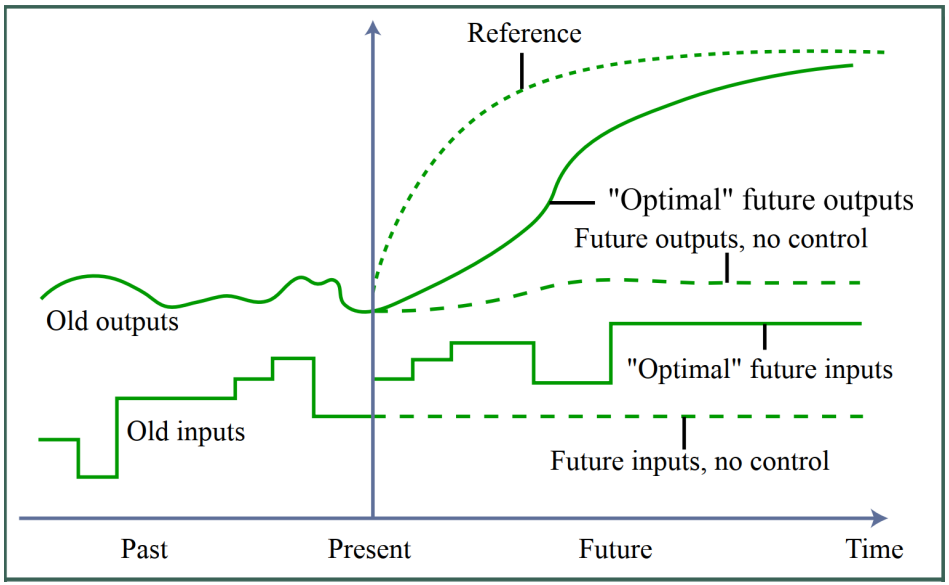


Figure: Basic idea of MPC [6].

- It is a **feedback control** algorithm.
- It uses a model to **precit** future outputs.
- It **solves an online optimization problem** to select the optimal control.

What is MPC?

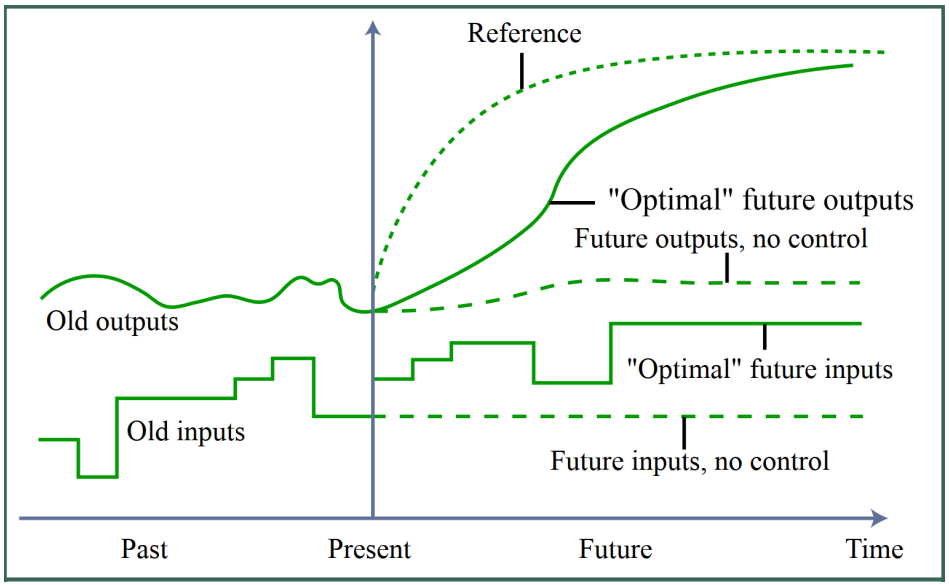


Figure: Basic idea of MPC [6].

- It is a **feedback control** algorithm.
- It uses a model to **precit** future outputs.
- It **solves an online optimization problem** to select the optimal control.

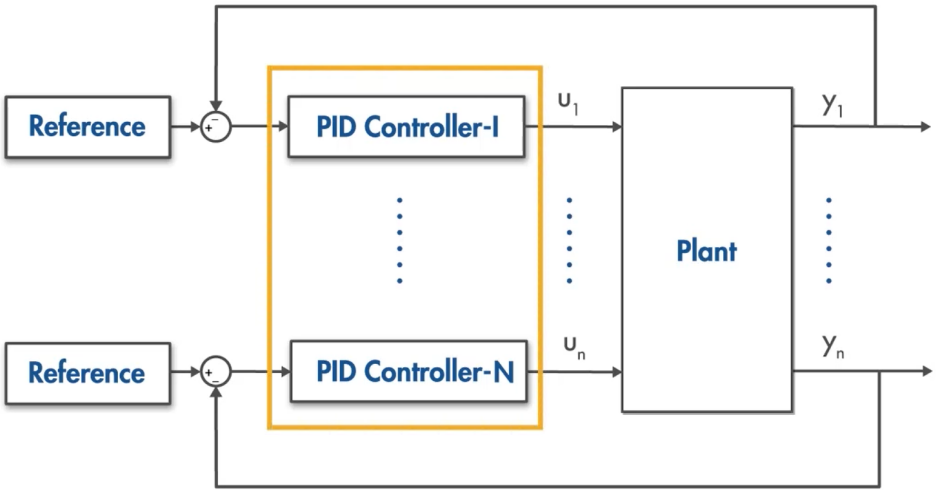


Figure: MIMO system with PIDs [10].

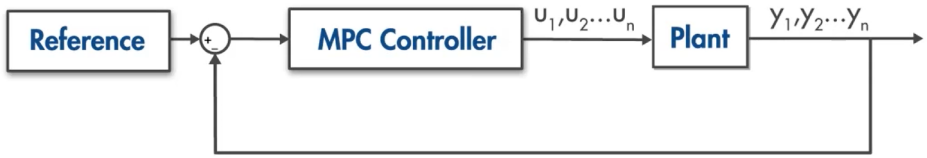


Figure: MIMO system with MPC controller [10].

- Can **control MIMO** systems.
- Explicitly **accounts for constraints**.
- Can handle **nonlinear and time-varying plant dynamics**.

Remark

MPC requires a **powerful**, **fast** processor with a **large** memory.

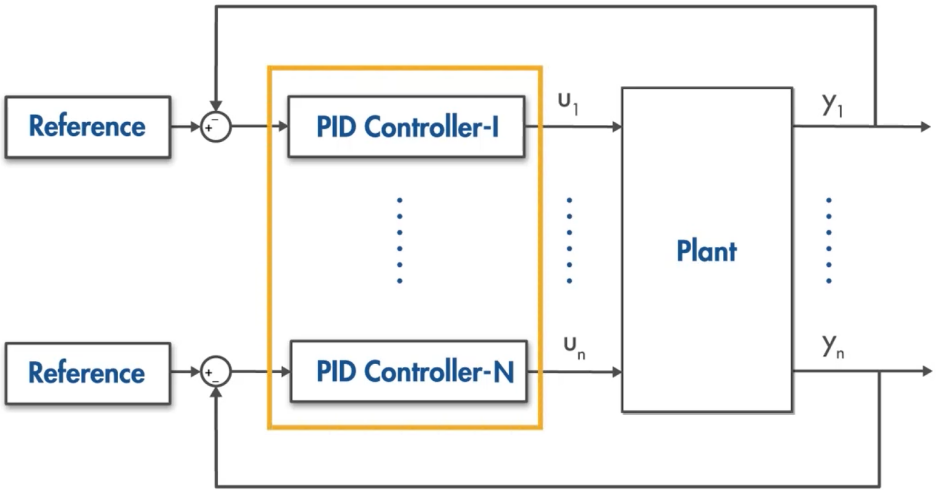


Figure: MIMO system with PIDs [10].

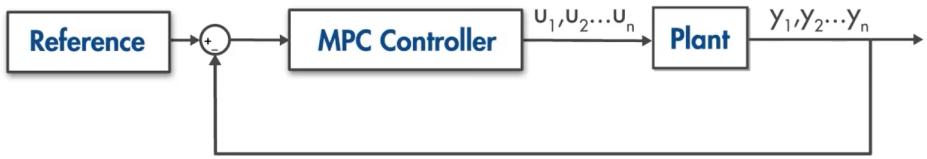


Figure: MIMO system with MPC controller [10].

- Can **control MIMO** systems.
- Explicitly **accounts for constraints**.
- Can handle **nonlinear and time-varying plant dynamics**.

Remark

MPC requires a **powerful**, **fast** processor with a **large** memory.

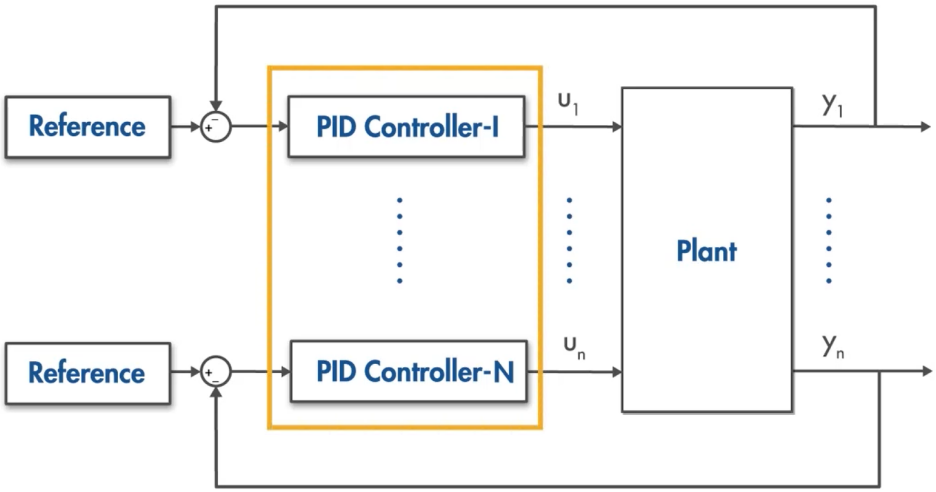


Figure: MIMO system with PIDs [10].

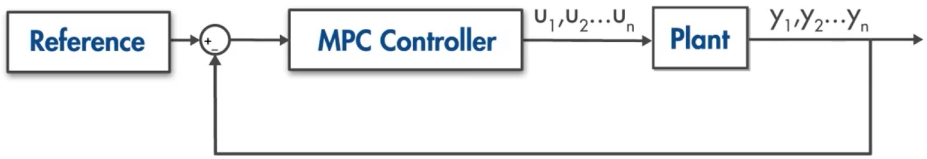


Figure: MIMO system with MPC controller [10].

- Can **control MIMO** systems.
- Explicitly **accounts for constraints**.
- Can handle **nonlinear and time-varying plant dynamics**.

Remark

MPC requires a **powerful**, **fast** processor with a **large** memory.

MPC Design parameters:

- Sample time
- Prediction horizon
- Control horizon
- Constraints
- Weights

Choosing proper values for these parameters is important as they affect:

- The controller performance.
- The computational complexity of the MPC algorithm.

MPC Design parameters:

- Sample time
- Prediction horizon
- Control horizon
- Constraints
- Weights

Choosing proper values for these parameters is important as they affect:

- The controller performance.
- The computational complexity of the MPC algorithm.

MPC Design parameters:

- Sample time
- Prediction horizon
- Control horizon
- Constraints
- Weights

Choosing proper values for these parameters is important as they affect:

- The controller performance.
- The computational complexity of the MPC algorithm.

MPC Design parameters:

- Sample time
- Prediction horizon
- Control horizon
- Constraints
- Weights

Choosing proper values for these parameters is important as they affect:

- The controller performance.
- The computational complexity of the MPC algorithm.

MPC Design parameters:

- Sample time
- Prediction horizon
- Control horizon
- Constraints
- Weights

Choosing proper values for these parameters is important as they affect:

- The controller performance.
- The computational complexity of the MPC algorithm.

MPC Design parameters:

- Sample time
- Prediction horizon
- Control horizon
- Constraints
- Weights

Choosing proper values for these parameters is important as they affect:

- The controller performance.
- The computational complexity of the MPC algorithm.

MPC Design parameters:

- Sample time
- Prediction horizon
- Control horizon
- Constraints
- Weights

Choosing proper values for these parameters is important as they affect:

- The controller performance.
- The computational complexity of the MPC algorithm.

General MPC methods:

- Linear time-invariant MPC.
- Adaptive MPC.
- Gain-Scheduled MPC.
- Nonlinear MPC.

General MPC methods:

- Linear time-invariant MPC.
- Adaptive MPC.
- Gain-Scheduled MPC.
- Nonlinear MPC.

General MPC methods:

- Linear time-invariant MPC.
- Adaptive MPC.
- Gain-Scheduled MPC.
- Nonlinear MPC.

General MPC methods:

- Linear time-invariant MPC.
- Adaptive MPC.
- Gain-Scheduled MPC.
- Nonlinear MPC.

In Adaptive MPC:

- A linear model is computed on the fly as the operating conditions change.
- At each time step, the internal plant model used by the MPC is updated with this linear model.

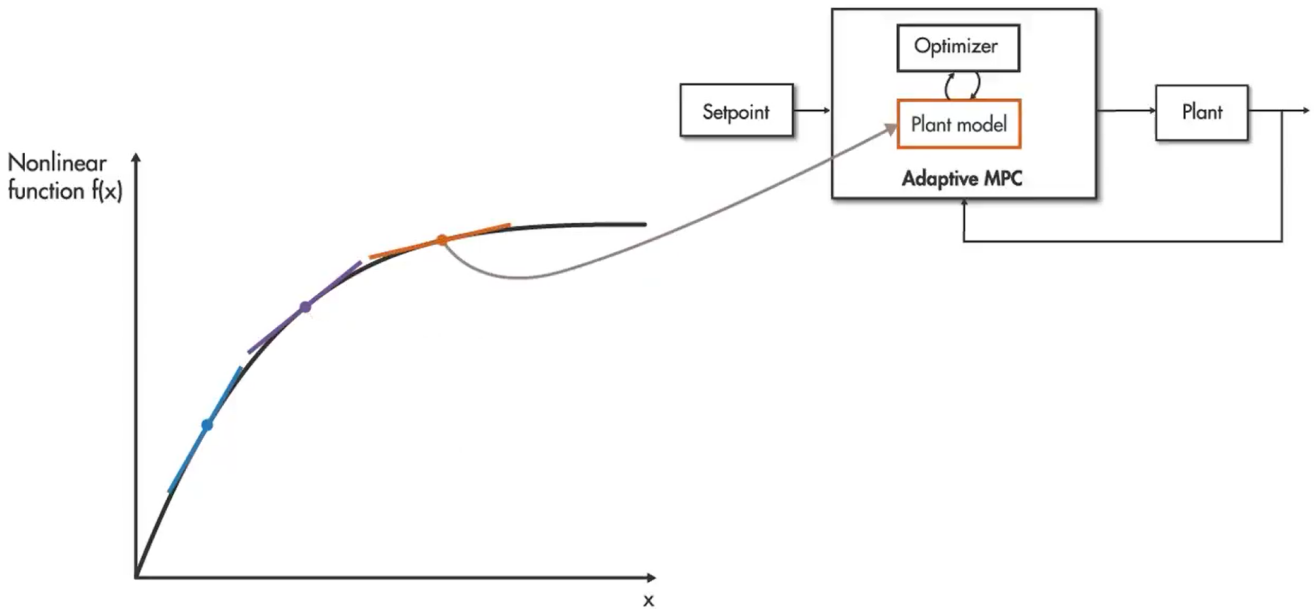


Figure: Example of Adaptive MPC [9].

Remark

The structure of the optimization problem remains the same across different operating points.

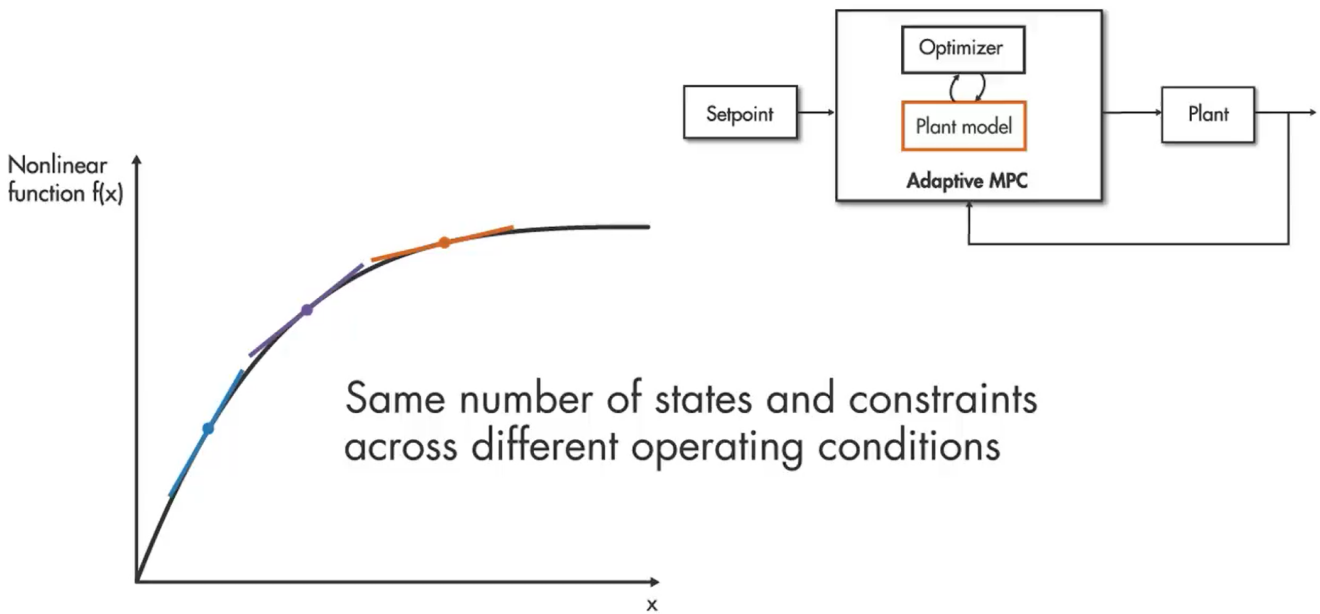


Figure: Example of Adaptive MPC [9].

In Gain-Scheduled MPC:

- Linearization is done offline at the operating points of interest.
- A linear MPC controller is then designed for each operating point.

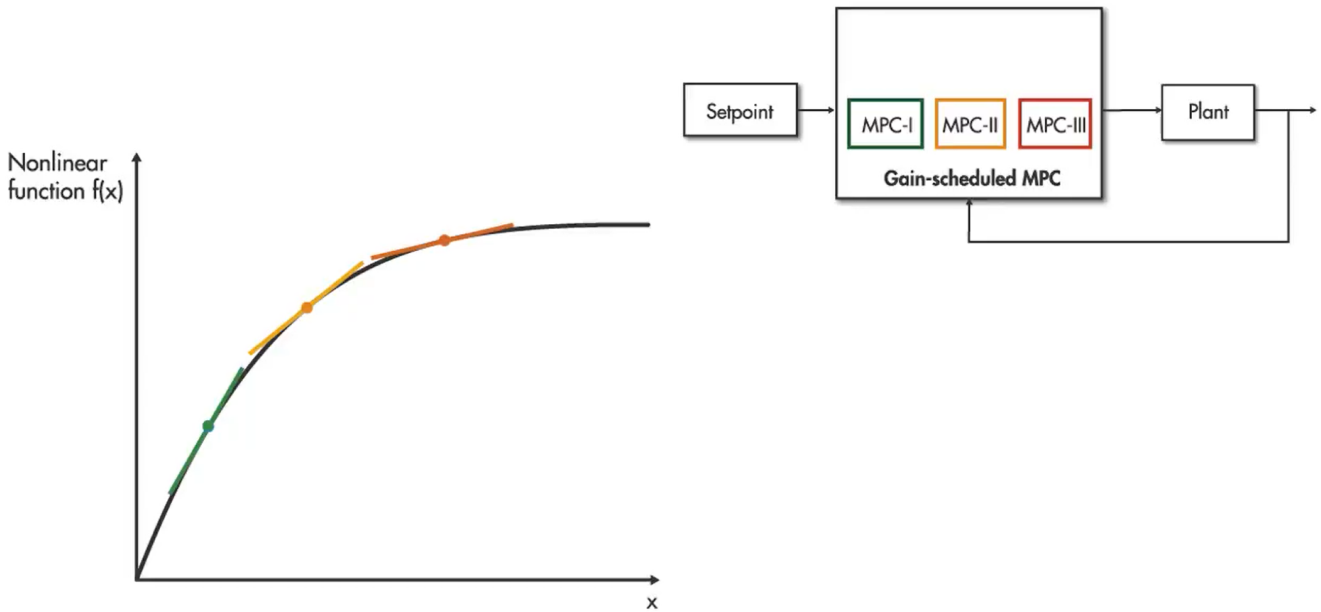


Figure: Example of Gain-Scheduled MPC [9].

Remark

Each controller is **independent** from the other.

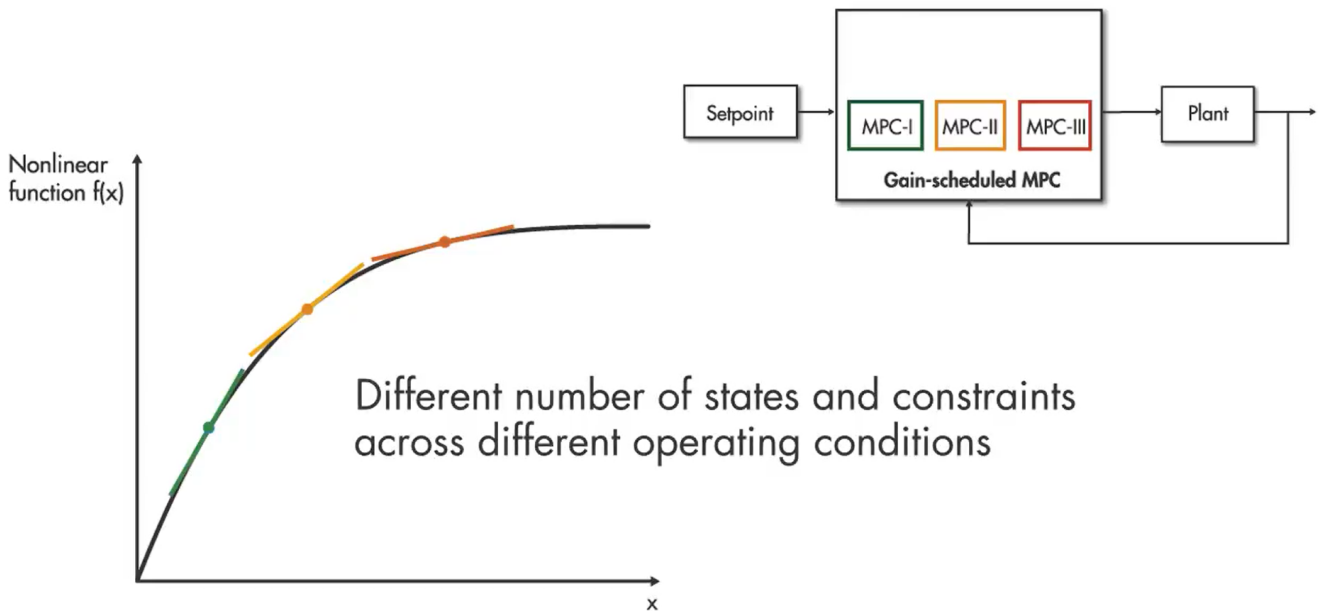


Figure: Example of Gain-Scheduled MPC [9].

For **Nonlinear MPC**, useful if:

- System is **non-linearizable**.
- Existence of **nonlinear** constraints.
- Existence of **nonlinear** cost function.

Nonlinear MPC

Most powerful methods: uses the most accurate representation of the plant $\Rightarrow$ **More accurate predictions**.

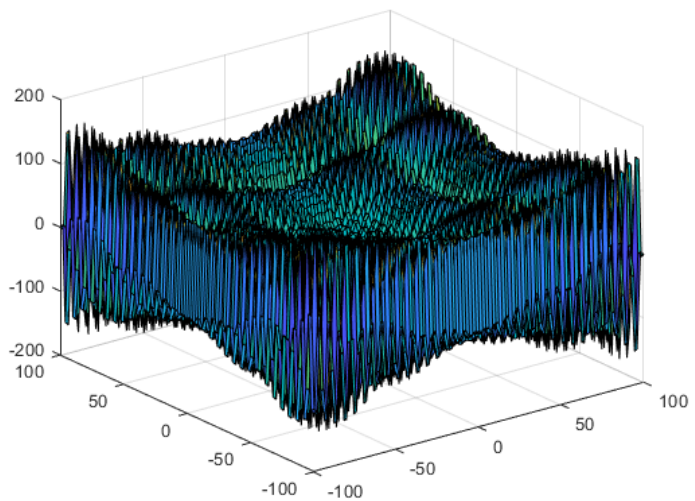


Figure: Example of a non-convex optimization problem.

1 Introduction

2 System Modeling

3 Control of Quadrotors

4 Multi-Flip Maneuvers with Quadrotors

5 Trajectory Optimization

6 Planning

7 References

The physics of a quadrotor flip can be divided into 4 parts:

- **Climb phase**: Maximum vertical acceleration is applied.
- **Multi-flip phase**: It ends when the desired $2n\pi$ are achieved.
- **Descent phase**: Maximum thrust for descent compensation.
- **Re-stabilization**: Altitude regulation to desired value.

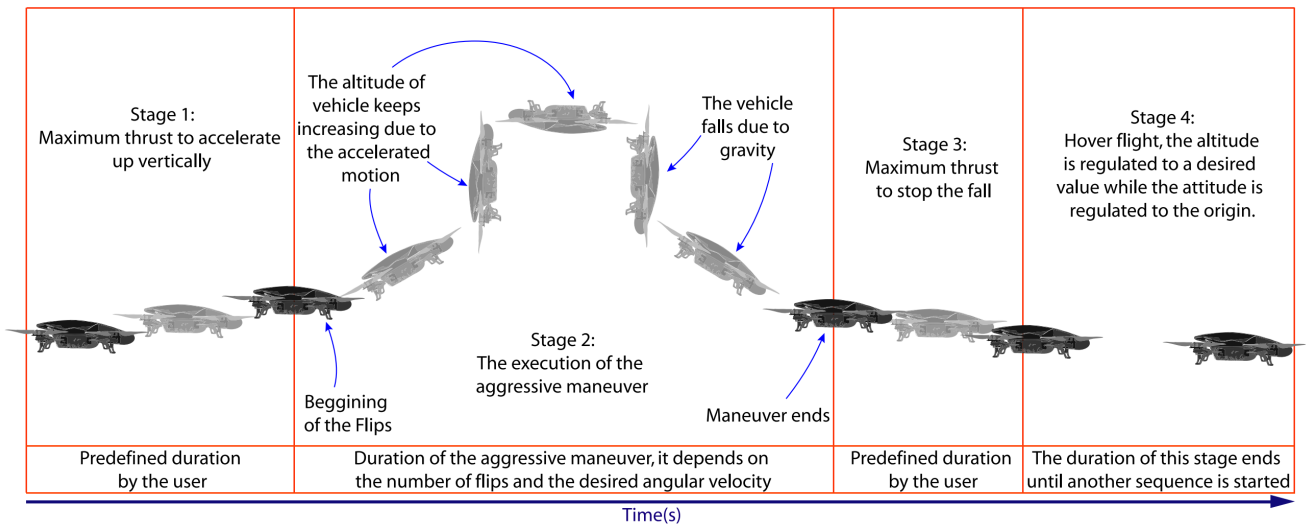


Figure: The phases needed for performing multi-flip maneuvers [11].

The physics of a quadrotor flip can be divided into 4 parts:

- **Climb phase**: Maximum vertical acceleration is applied.
- **Multi-flip phase**: It ends when the desired $2n\pi$ are achieved.
- **Descent phase**: Maximum thrust for descent compensation.
- **Re-stabilization**: Altitude regulation to desired value.

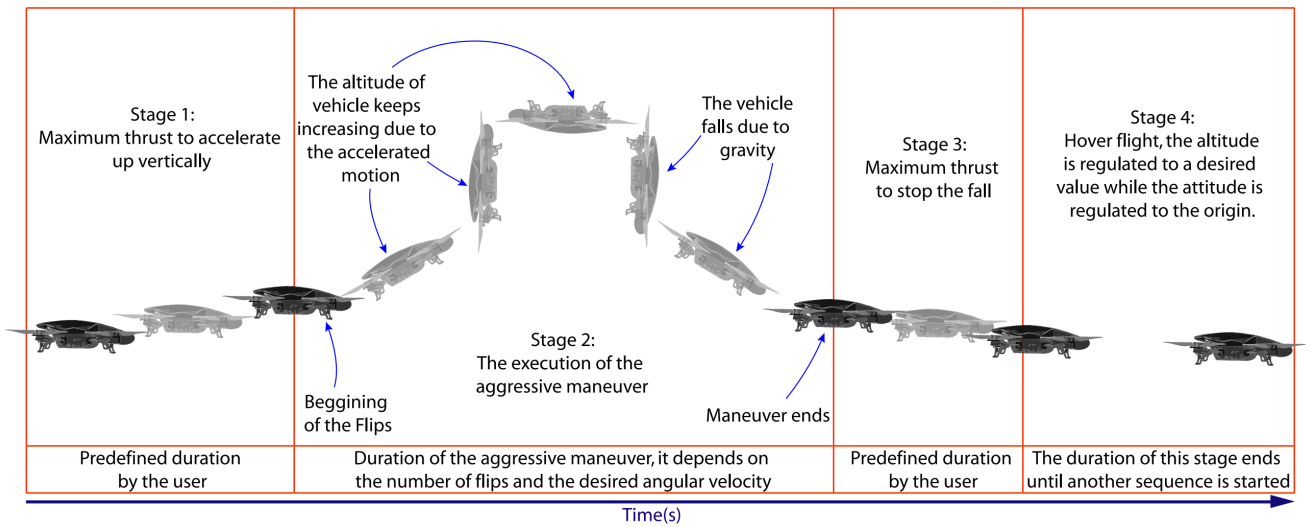


Figure: The phases needed for performing multi-flip maneuvers [11].

The physics of a quadrotor flip can be divided into 4 parts:

- **Climb phase**: Maximum vertical acceleration is applied.
- **Multi-flip phase**: It ends when the desired $2n\pi$ are achieved.
- **Descent phase**: Maximum thrust for descent compensation.
- **Re-stabilization**: Altitude regulation to desired value.

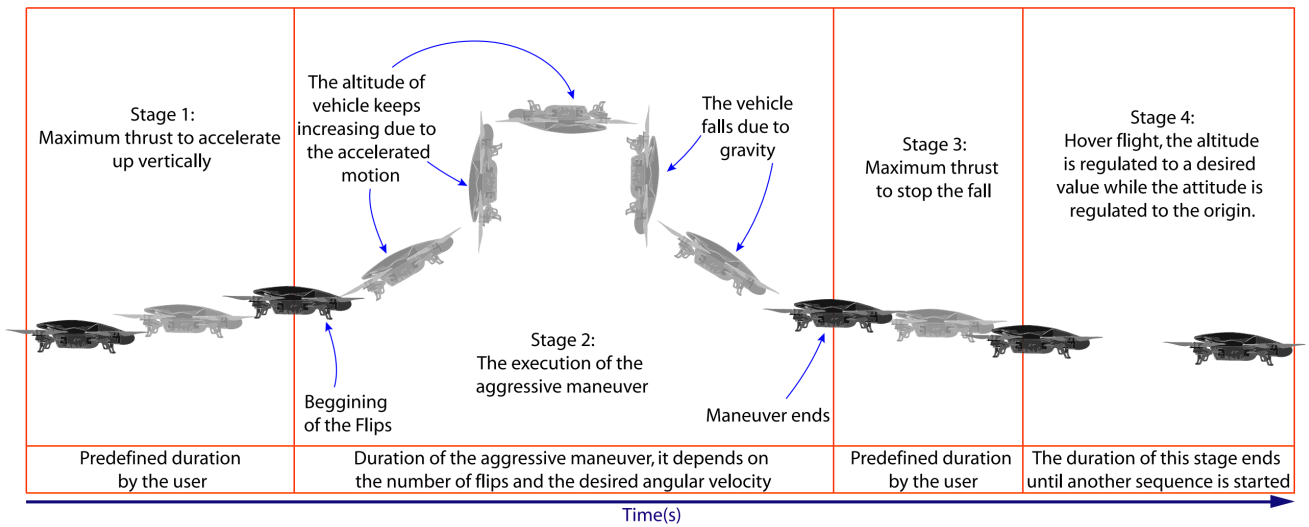


Figure: The phases needed for performing multi-flip maneuvers [11].

The physics of a quadrotor flip can be divided into 4 parts:

- **Climb phase**: Maximum vertical acceleration is applied.
- **Multi-flip phase**: It ends when the desired $2n\pi$ are achieved.
- **Descent phase**: Maximum thrust for descent compensation.
- **Re-stabilization**: Altitude regulation to desired value.

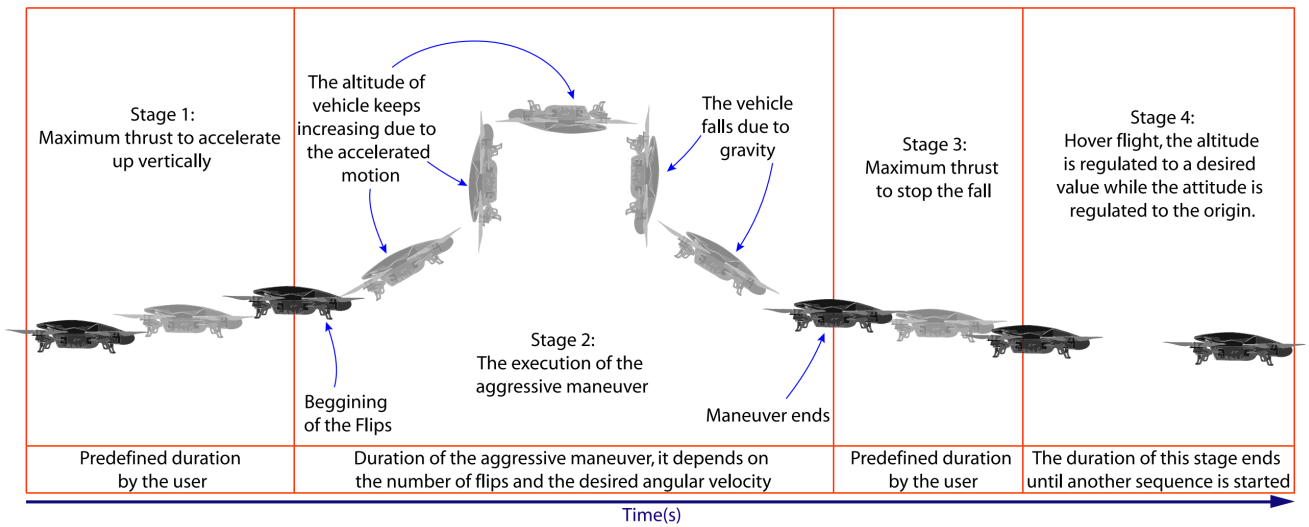


Figure: The phases needed for performing multi-flip maneuvers [11].

In closed-loop attitude control:

- Main focus is on the **attitude part** of the platform.
- **Main idea**: Reference on ω is planned and tracked in closed-loop with a nonlinear control law.

It has the following properties:

1. A flipping angle ϕ_g is defined such that a flip around ***a*** through the UAV's CoG can be defined.
2. Angular speed planning is performed using cubic splines (the desired ω_d is generated).
   - The dynamics of actuation is respected by guaranteeing that ω_{max} is not exceeded.

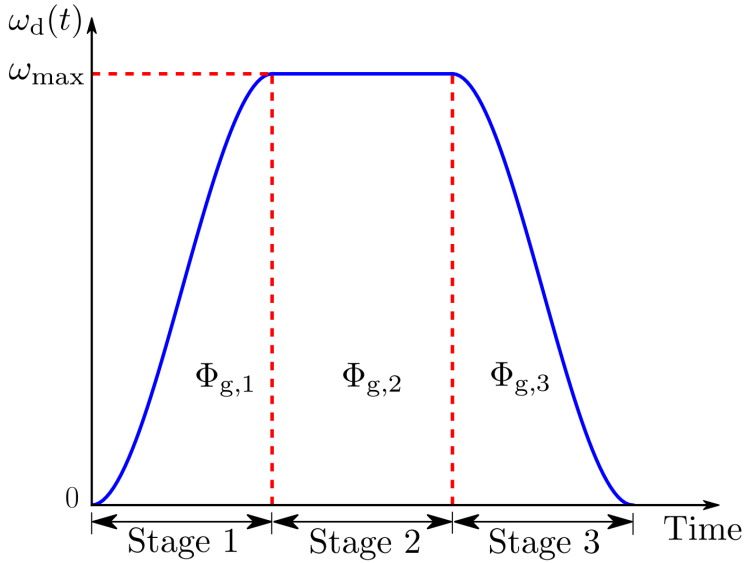
Given the desired amount of rotation accumulated during each stage $\Phi_{g,i}$ the duration of the stages is varied according to the current flipping angle ϕ_g:

$$\omega_d(t) = \begin{cases} \omega_{d,1} & \text{if} \quad \phi_g \leq \Phi_{g,1} \\ \omega_{d,2} & \text{if} \quad \Phi_{g,1} < \phi_g \leq \Phi_{g,1} + \Phi_{g,2} \\ \omega_{d,3} & \text{if} \quad \Phi_{g,1} + \Phi_{g,2} < \phi_g \leq \Phi_{g,1} + \Phi_{g,2} + \Phi_{g,3} \end{cases} \tag{9}$$

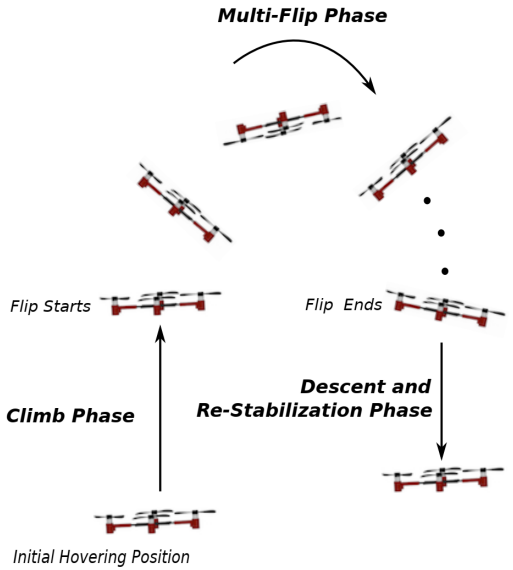
where:

$$\Phi_{g,1} + \Phi_{g,2} + \Phi_{g,3} = 2n\pi \tag{10}$$

in order to carry-out the flip.



(a) Reference of the angular velocity.



(b) Stages of the multi-flip maneuver.

Figure: Reference of the flipping speed and the different flipping phases [2].

❸ Different control approaches can then used to track the angular speed reference.

1 Introduction

2 System Modeling

3 Control of Quadrotors

4 Multi-Flip Maneuvers with Quadrotors

5 Trajectory Optimization

6 Planning

7 References

What is trajectory optimization?

- Set of techniques used to find the desired behavior of a dynamic system.
- It is a class of methods that solves the optimal control problem.

What is trajectory optimization?

- Set of techniques used to find the desired behavior of a dynamic system.
- It is a class of methods that solves the optimal control problem.

Optimal Control: Find the best control to achieve some desired goal.

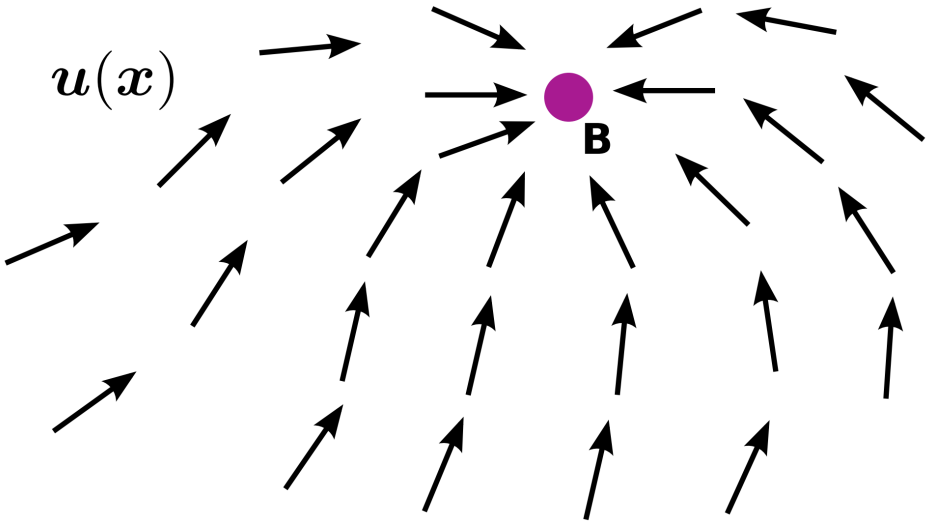


Figure: Closed-loop solution [7].

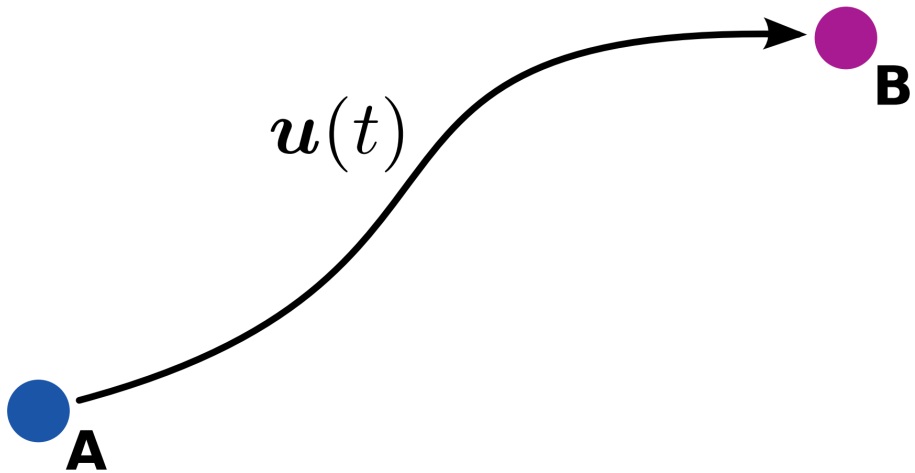


Figure: Open-loop solution [7].

- Global method and solution.
- More difficult to compute.
- Use: lower-dimension problems.

- Local method and solution.
- Less difficult to compute.
- Use: higher-dimensional problems.

Trajectory Optimization Problem

$$\min_{\underbrace{t_0, t_F, \boldsymbol{x}(t), \boldsymbol{u}(t)}_{\text{functions!}}} J\big(t_0, t_F, \boldsymbol{x}(t_0), \boldsymbol{x}(t_F)\big) + \int_{t_0}^{t_F} w\big(\tau, \boldsymbol{x}(\tau), \boldsymbol{u}(\tau)\big)\, d\tau$$

boundary objective function

integral objective function

calculus!

subject to:

$$\boldsymbol{g}\big(t_0, t_F, \boldsymbol{x}(t_0), \boldsymbol{x}(t_F)\big) \leq \mathbf{0}$$

boundary constraints

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\big(t, \boldsymbol{x}(t), \boldsymbol{u}(t)\big)$$

continuous dynamics

$$t_{\text{low}} \leq t_0 < t_F \leq t_{\text{upp}}$$

bound on initial and final time

$$\boldsymbol{h}\big(t, \boldsymbol{x}(t), \boldsymbol{u}(t)\big) \leq \mathbf{0}$$

path constraints

$$\boldsymbol{x}_{0,\text{low}} \leq \boldsymbol{x}(t_0) \leq \boldsymbol{x}_{0,\text{upp}}$$

bound on initial state

$$\boldsymbol{x}_{\text{low}} \leq \boldsymbol{x}(t) \leq \boldsymbol{x}_{\text{upp}}$$

continuous bounds on state

$$\boldsymbol{x}_{F,\text{low}} \leq \boldsymbol{x}(t_F) \leq \boldsymbol{x}_{F,\text{upp}}$$

bound on final state

$$\boldsymbol{u}_{\text{low}} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{\text{upp}}$$

continuous bounds on control

Transcription

trajectory optimization problem

- decision variables are vector ***functions***
- infinite dimensional
- ***differential*** equations

constrained parameter optimization

- decision variables are real ***numbers***
- finite dimensional
- ***algebraic*** equations

$$\min_{t_0, t_F, \boldsymbol{x}(t), \boldsymbol{u}(t)} J\big(t_0, t_F, \boldsymbol{x}(t_0), \boldsymbol{x}(t_F)\big) + \int_{t_0}^{t_F} w\big(\tau, \boldsymbol{x}(\tau), \boldsymbol{u}(\tau)\big)\, d\tau$$

functions! — boundary objective function — integral objective function — calculus!

subject to:

$\boldsymbol{g}\big(t_0, t_F, \boldsymbol{x}(t_0), \boldsymbol{x}(t_F)\big) \leq \mathbf{0}$ — boundary constraints

$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\big(t, \boldsymbol{x}(t), \boldsymbol{u}(t)\big)$ — continuous dynamics

$t_{\text{low}} \leq t_0 < t_F \leq t_{\text{upp}}$ — bound on initial and final time

$\boldsymbol{h}\big(t, \boldsymbol{x}(t), \boldsymbol{u}(t)\big) \leq \mathbf{0}$ — path constraints

$\boldsymbol{x}_{0,\text{low}} \leq \boldsymbol{x}(t_0) \leq \boldsymbol{x}_{0,\text{upp}}$ — bound on initial state

$\boldsymbol{x}_{\text{low}} \leq \boldsymbol{x}(t) \leq \boldsymbol{x}_{\text{upp}}$ — continuous bounds on state

$\boldsymbol{x}_{F,\text{low}} \leq \boldsymbol{x}(t_F) \leq \boldsymbol{x}_{F,\text{upp}}$ — bound on final state

$\boldsymbol{u}_{\text{low}} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{\text{upp}}$ — continuous bounds on control

Non-Linear Program (NLP)

$$\min_{\boldsymbol{z}} f(\boldsymbol{z}) \qquad \text{subject to:}$$

$$\boldsymbol{g}(\boldsymbol{z}) \leq \mathbf{0}$$

$$\boldsymbol{h}(\boldsymbol{z}) = \mathbf{0}$$

$$\boldsymbol{z}_{\text{low}} \leq \boldsymbol{z} \leq \boldsymbol{z}_{\text{upp}}$$

Transcription Methods

Indirect Methods ⟷ **Direct Methods**

- "optimize then discretize"
- more accurate
- harder to pose and solve

- "discretize the optimize"
- less accurate
- easier to pose and solve

Shooting Methods ⟷ **Collocation Methods**

- based on simulation
- better for problems with simple control and no path constraints

- based on function approximation
- better for problems with complicated control and/or path constraints

"h-methods" ⟷ **"p-methods"**

- low-order method
- converge by increasing number of segments

- high-order methods
- converge by increasing method order

1 Introduction

2 System Modeling

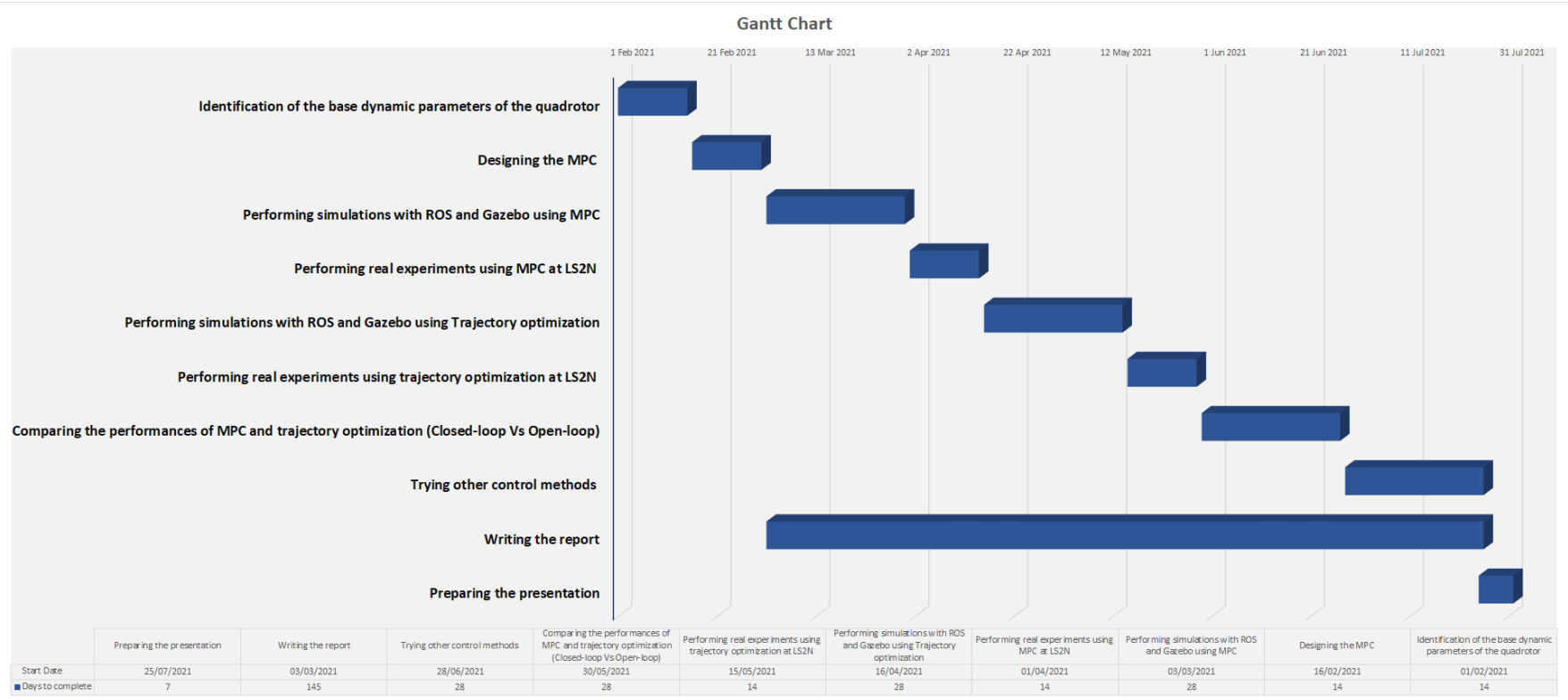
3 Control of Quadrotors

4 Multi-Flip Maneuvers with Quadrotors

5 Trajectory Optimization

6 Planning

7 References



	Preparing the presentation	Writing the report	Trying other control methods	Comparing the performances of MPC and trajectory optimization (Closed-loop Vs Open-loop)	Performing real experiments using trajectory optimization at LS2N	Performing simulations with ROS and Gazebo using Trajectory optimization	Performing real experiments using MPC at LS2N	Performing simulations with ROS and Gazebo using MPC	Designing the MPC	Identification of the base dynamic parameters of the quadrotor
Start Date	25/07/2021	03/03/2021	28/06/2021	30/05/2021	15/05/2021	16/04/2021	01/04/2021	03/03/2021	16/02/2021	01/02/2021
Days to complete	7	145	28	28	14	28	14	28	14	14

Figure: Preliminary plan for the master thesis

1 Introduction

2 System Modeling

3 Control of Quadrotors

4 Multi-Flip Maneuvers with Quadrotors

5 Trajectory Optimization

6 Planning

7 References

- Bouabdallah, S. (2007). “Design and control of quadrotors with application to autonomous flying”. PhD thesis. École Polytechnique Fédérale de Lausanne.
- Chen, Y. and N. O. P´erez-Arancibia (2016). “Generation and real-time implementation of high-speed controlled maneuvers using an autonomous 19-gram quadrotor”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3204–3211.
- Coxworth, B. (2020). *MIT tech uses muscle signals to control a drone*. `https://newatlas.com/drones/muscle-signals-drone-control/#gallery:2`. MIT, Drones.
- Faessler, M., A. Franchi, and D. Scaramuzza (2018). “Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories.”. In: *IEEE Robot. Autom. Lett*, pp. 620–626.

- Fantoni, I. (2016). *Lecture notes in Modelling of Terrestrial and Aerial Vehicles*.
- Jonathan How (2008). *Principles of Optimal Control*. MIT.
- Kelly, M. (2017). "An introduction to trajectory optimization, how to do your own direct collocation". In: *SIAM*, pp. 1–30.
- Lupashin, S. and R. D'Andrea (2012). *Adaptive fast open-loop maneuvers for quadrocopters*. Manhattan, NY: Springer, pp. 89–102.
- MathWorks (2018a). *Adaptive, Gain-Scheduled, and Nonlinear MPC*. `https://www.mathworks.com/videos/understanding-model-predictive-control-part-4-adaptive-gain-scheduled-and-nonlinear-mpc-1530606851674.html`. From the Model Predictive Control Toolbox.


MathWorks (2018b). *How to Run MPC Faster*. https://www.mathworks.com/videos/understanding-model-predictive-control-part-5-how-to-run-mpc-faster-1533108818950.html. From the Model Predictive Control Toolbox.


Oliva-Palomo, F. et al. (2018). "Nonlinear ellipsoid based attitude control for aggressive trajectories in a quadrotor: closed-loop multiflips implementation". In: *Control Engineering Practice*, pp. 150–161.


Wikipedia (2018). *DJI Phantom Quadcopter*. https://en.wikipedia.org/wiki/Quadcopter#/media/File:Quadcopter_camera_drone_in_flight.jpg.