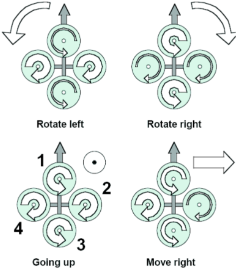
Making Flips with Quadrotors in Constrained Environments

Elie Hatem

Advisors: Dr. Sébastien Briot & Dr. Isabelle Fantoni

19/01/2021

(a) DJI Phantom quadcopter (UAV) [9].



(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.

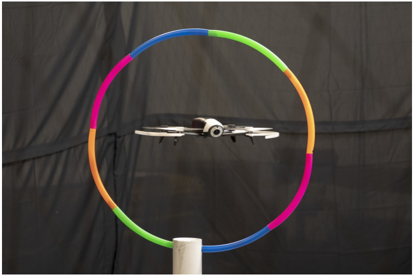
Remark

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.[6]



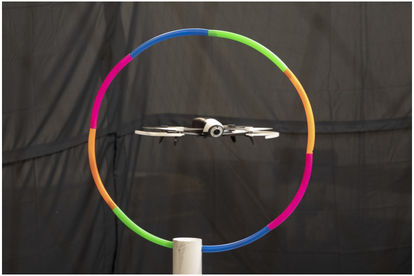
(b) Quadrotor going though a loop [2].

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.[6]



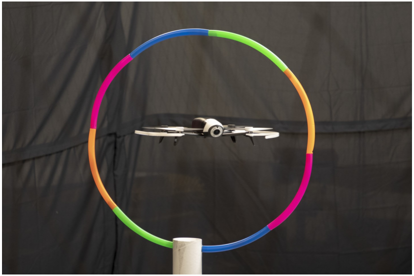
(b) Quadrotor going though a loop [2].

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.[6]



(b) Quadrotor going though a loop [2].

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

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

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

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

which results in [4]:

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

- **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 [4]:

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

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

$$u_1 = u \text{ (thrust)}$$

$$u_2 = \tau_\phi$$

$$u_3 = \tau_\theta$$

$$u_4 = \tau_\psi$$

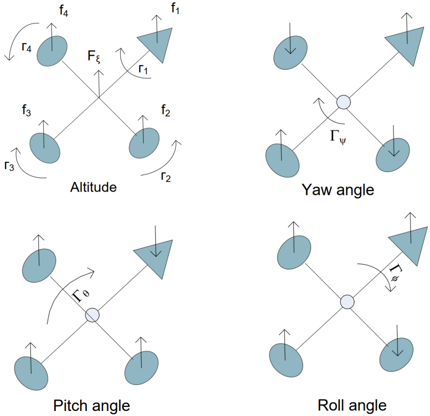


Figure: Quadrotor model with motor torques and Euler angles [4].

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** [4]:
  - **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** [4]:
  - **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** [4]:
  - **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** [4]:
  - **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:

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

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:

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

Contents

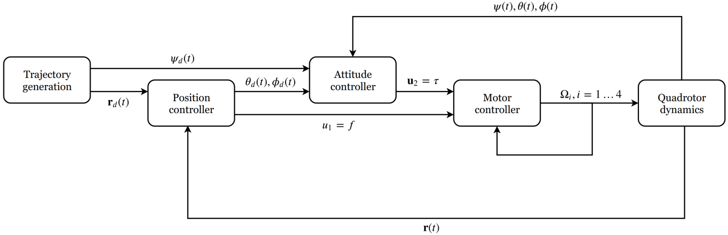


Figure: General control architecture for the position and attitude of a quadrotor [3].

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

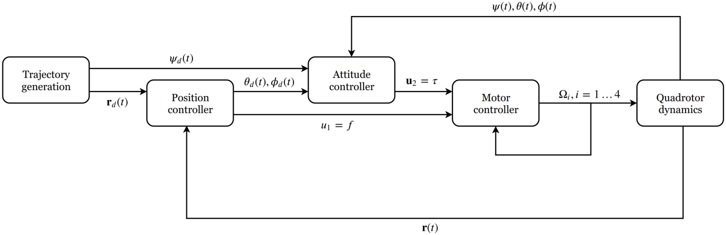


Figure: General control architecture for the position and attitude of a quadrotor [3].

- **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.

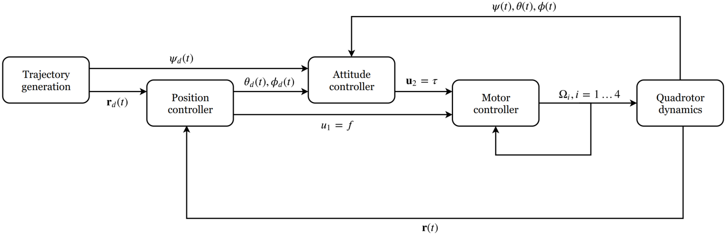


Figure: General control architecture for the position and attitude of a quadrotor [3].

- **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?

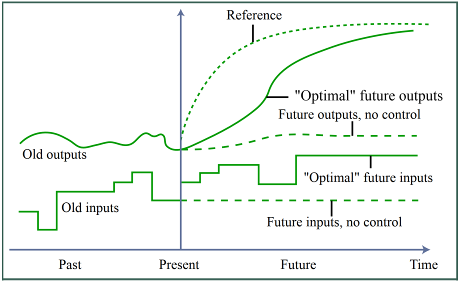


Figure: Basic idea of MPC [5].

- 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?

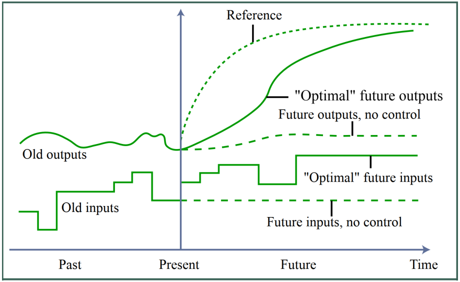


Figure: Basic idea of MPC [5].

- 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?

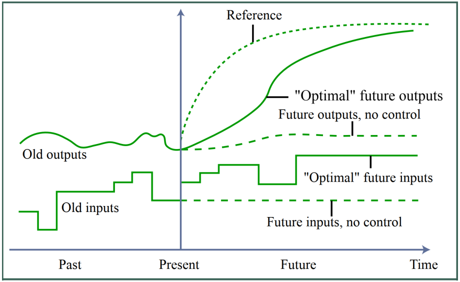


Figure: Basic idea of MPC [5].

- 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.

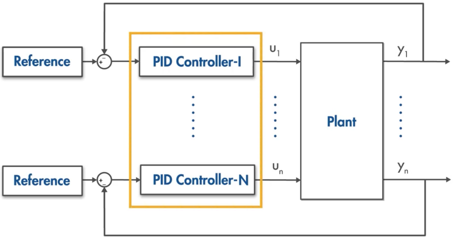


Figure: MIMO system with PIDs [8].

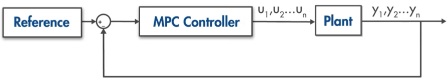


Figure: MIMO system with MPC controller [8].

- 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.

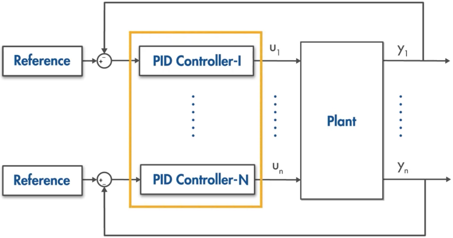


Figure: MIMO system with PIDs [8].

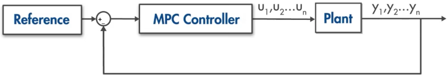


Figure: MIMO system with MPC controller [8].

- 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.

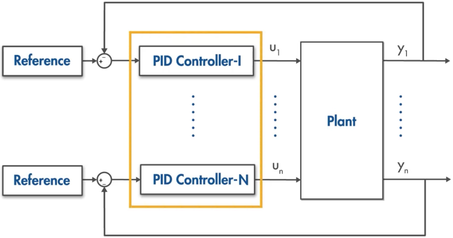


Figure: MIMO system with PIDs [8].

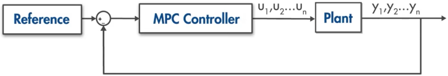


Figure: MIMO system with MPC controller [8].

- 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.

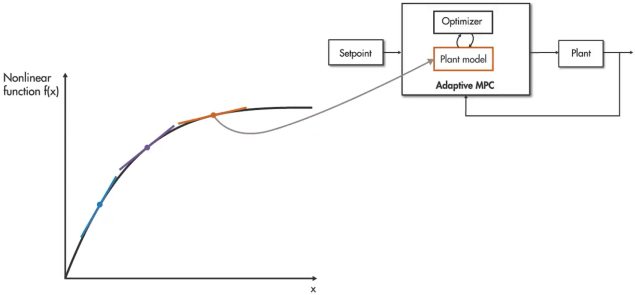


Figure: Example of Adaptive MPC [7].

Remark

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

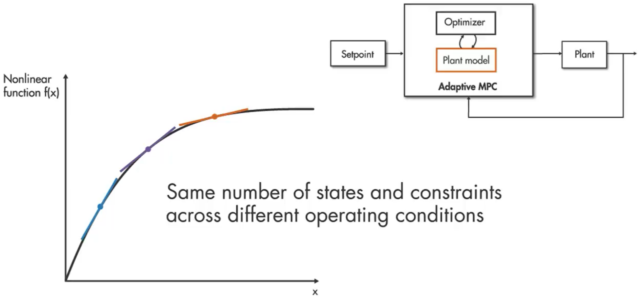


Figure: Example of Adaptive MPC [7].

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.

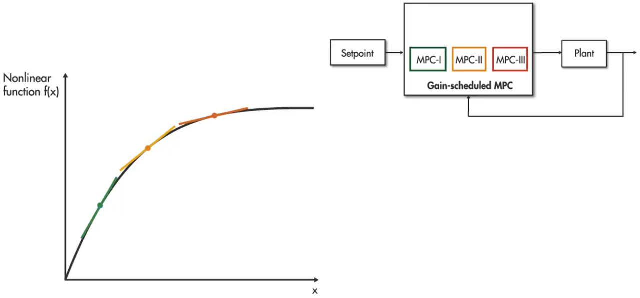


Figure: Example of Gain-Scheduled MPC [7].

Remark

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

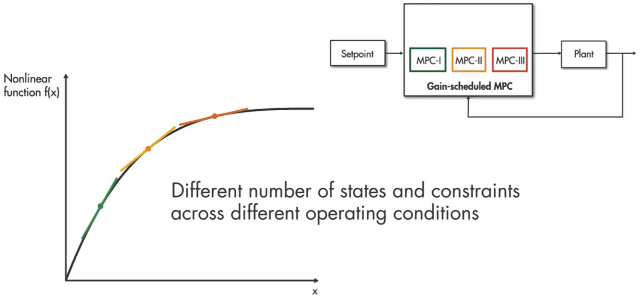


Figure: Example of Gain-Scheduled MPC [7].

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.**

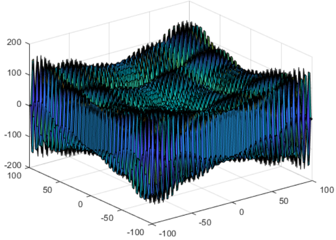


Figure: Example of a non-convex optimization problem.

- Bouabdallah, S. (2007). “Design and control of quadrotors with application to autonomous flying”. PhD thesis. École Polytechnique Fédérale de Lausanne.
- 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.

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.

— (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.

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