

Aerial Robotic Autonomy

> *Model Predictive flight Control*

Kostas Alexis
Autonomous Robots Lab

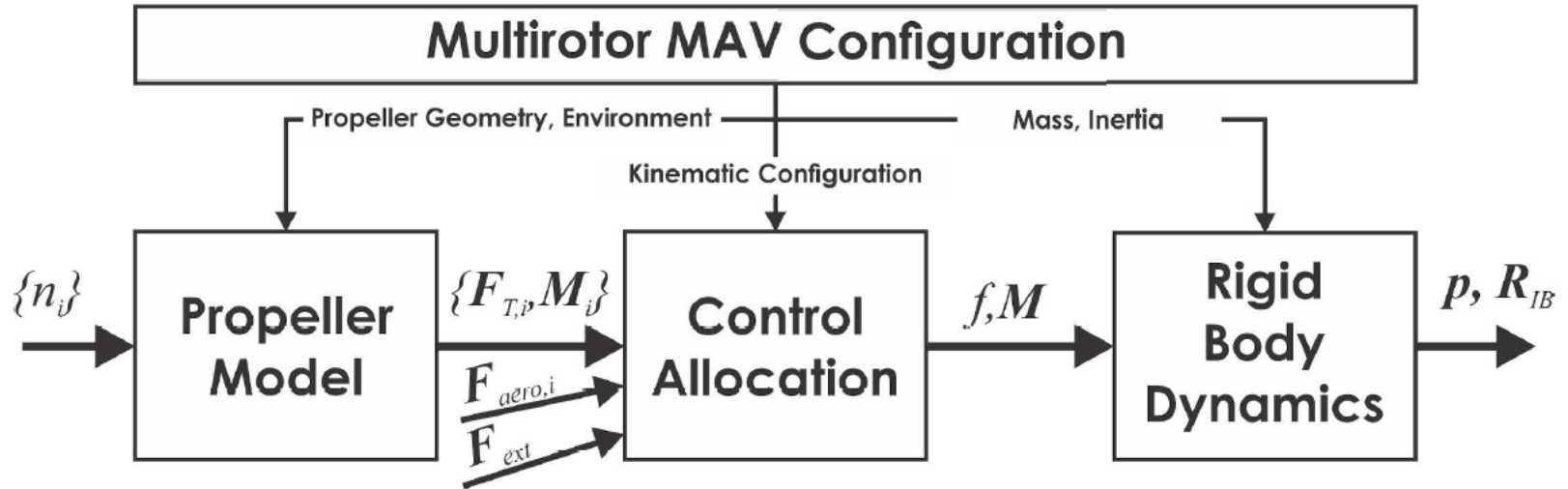
Introduction

Advantages of **Model Predictive Control (MPC)** for Multirotor Aerial Vehicles (and other rotorcrafts):

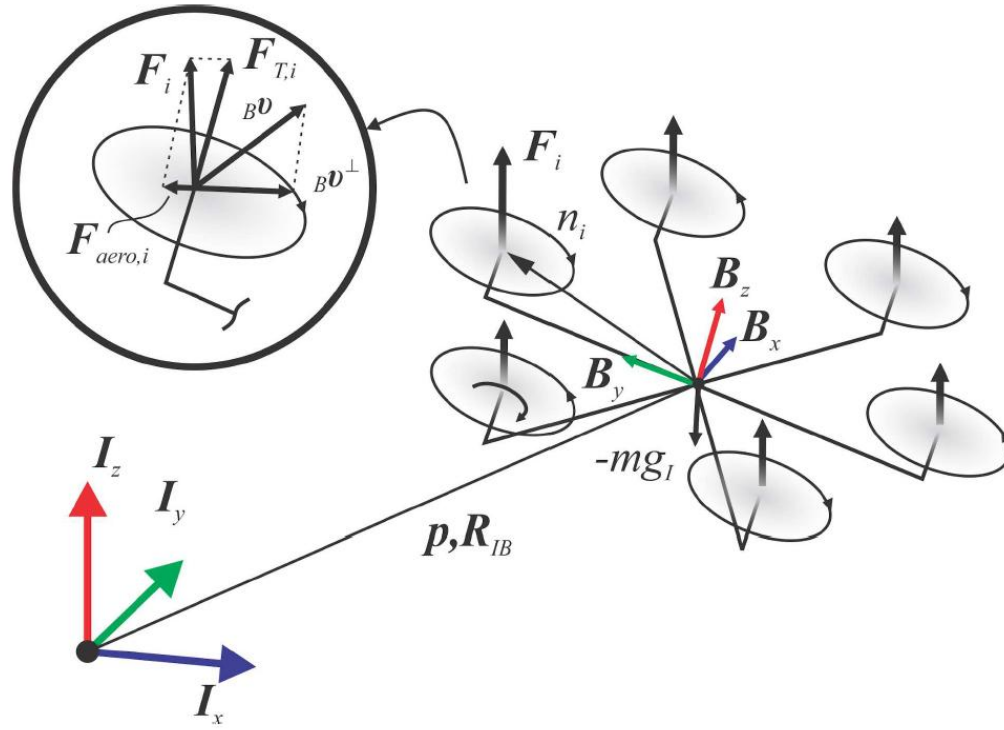
- Model-based approach exploiting knowledge of the system dynamics.
- Feasible (in principle) both for linear and nonlinear systems.
- Supports for hybrid formulations which may manifest in certain conditions (e.g., physical interaction).
- Optimization over a receding horizon can enable improved trajectory tracking, accounting and satisfying input and state constraints, while retaining performance.



MAV Modeling Summary



MAV Modeling Summary



MAV Modeling Summary

For each propeller i the generated thrust and moment take the simplified form (n_i rotor speed, $k_n, k_m > 0$ constants, e_z unit vector in the z direction)

$$\begin{aligned}\mathbf{F}_{T,i} &= k_n n_i^2 \mathbf{e}_z \\ \mathbf{M}_i &= (-1)^{i-1} k_m \mathbf{F}_{T,i}\end{aligned}$$

Lamped effect of phenomena such as blade flapping ($\mathbf{K}_{drag} = \mathbf{diag}(k_D, k_D, 0)$, $k_D > 0$ and $f_{T,i}$ is the z component of the i -th thrust force)

$$\mathbf{F}_{aero,i} = f_{T,i} \mathbf{K}_{drag} \mathbf{R}_{IB}^T \mathbf{v}$$

MAV Modeling Summary

The motion dynamics take the form

$$\begin{aligned}\dot{\mathbf{p}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= \frac{1}{m} \left(\mathbf{R}_{IB} \sum_{i=0}^{N_r} \mathbf{F}_{T,i} - \mathbf{R}_{IB} \sum_{i=0}^{N_r} \mathbf{F}_{aero,i} + \mathbf{F}_{ext} \right) + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \\ \dot{\mathbf{R}}_{IB} &= \mathbf{R}_{IB} [\boldsymbol{\omega} \times] \\ \mathbf{J} \dot{\boldsymbol{\omega}} &= -\boldsymbol{\omega} \times \mathbf{J} + \mathcal{A} \begin{bmatrix} n_1^2 \\ \vdots \\ n_{N_r}^2 \end{bmatrix}\end{aligned}$$

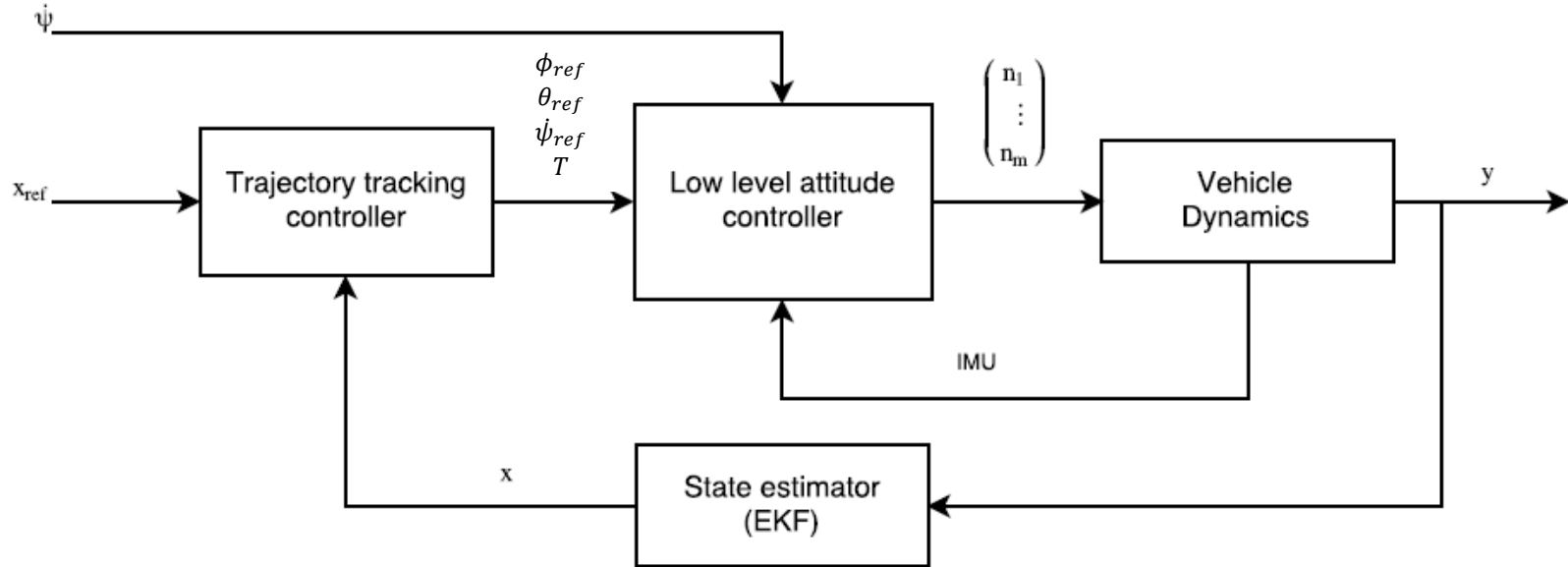
where \mathbf{F}_{ext} represents any external forces acting on the vehicle, and \mathcal{A} is the control allocation matrix, while N_r is the number of propellers.

MAV Modeling Summary

We will focus on position control through MPC, while letting the inner loop of the system to be handled by fixed-gain control.

Applicable low-level controllers span from simple PID implementations, to geometric control on $SO(3)$, to optimization-based controllers (including MPC) or even reinforcement learning-based methods.

MAV Modeling Summary



MAV Modeling Summary

Considering a given controller we can model the closed-loop attitude dynamics through a simple step of system identification.

Despite the fact of these dynamics being at least second order, assuming fine-tuned low-level attitude control we can typically assume a first-order closed-loop structure for the purposes of modeling for position control design:

$$\dot{\phi} = \frac{1}{\tau_{\phi}}(k_{\phi}\phi_{ref} - \phi)$$

$$\dot{\theta} = \frac{1}{\tau_{\theta}}(k_{\theta}\theta_{ref} - \theta)$$

$$\dot{\psi} = \dot{\psi}_{ref}$$

where k_{ϕ}, k_{θ} and $\tau_{\phi}, \tau_{\theta}$ are the dc-gains and time constants of the roll and pitch closed-loop dynamics, respectively. $\phi_{ref}, \theta_{ref}, \psi_{ref}$ are references.

Model Predictive Position Control

- We shall present formulations both for linear and nonlinear MPC.

Linear MPC

Considering the presented dynamics we proceed to linearize for position control without neglecting the closed-loop attitude dynamics. All terms are expressed in the inertial frame

$$\mathbf{x} = [\mathbf{p}^T \ \mathbf{v}^T \ \mathbb{I}\phi \ \mathbb{I}\theta]^T$$
$$\mathbf{u} = [\mathbb{I}\phi_{ref} \ \mathbb{I}\theta_{ref} \ T_{ref}]^T$$

where T_{ref} is the commanded reference thrust.

Linear MPC

The following formula holds for the robot's roll and pitch angles

$$\begin{bmatrix} \phi \\ \theta \end{bmatrix} = \begin{bmatrix} \cos \psi & \sin \psi \\ -\sin \psi & \cos \psi \end{bmatrix} \begin{bmatrix} \mathbb{I}\phi \\ \mathbb{I}\theta \end{bmatrix}$$

Linear MPC

After linearization and discretization the following form is derived (including further terms for the external forces $\mathbf{F}_{ext,k}$ and the disturbance matrix \mathbf{B}_d)

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{B}_d\mathbf{F}_{ext,k}$$

Linear MPC

The Linear MPC repeatedly solves the following Optimal Control Problem in which input constraints are further considered

$$\min_{\mathbf{U}} \sum_{k=0}^{N-1} \left(\|\mathbf{x}_k - \mathbf{x}_{ref,k}\|_{\mathbf{Q}_x}^2 + \|\mathbf{u}_k - \mathbf{u}_{ref,k}\|_{\mathbf{R}_u}^2 \right) + \|\mathbf{x}_N - \mathbf{x}_{ref,N}\|_{\mathbf{P}}^2$$

$$\begin{aligned} \text{s.t.} \quad & \mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{B}_d\mathbf{F}_{ext,k} \\ & \mathbf{F}_{ext,k+1} = \mathbf{F}_{ext,k}, \quad k = 0, \dots, N-1 \\ & \mathbf{u}_k \in \mathbb{U} \\ & \mathbf{x}_0 = \mathbf{x}(t_0), \quad \mathbf{F}_{ext,0} = \mathbf{F}_{ext}(t_0) \end{aligned}$$

where \mathbf{Q}_x , \mathbf{R}_u , \mathbf{P} are positive semi-definite, and $\mathbf{x}_{ref,k}$, $\mathbf{u}_{ref,k}$ are the target state and target control input respectively.

Linear MPC

The input constraints take the form

$$\mathbf{U} = \left\{ \mathbf{u} \in \mathbb{R}^3 \mid \begin{bmatrix} \phi_{\min} \\ \theta_{\min} \\ T_{ref,\min} \end{bmatrix} \leq \mathbf{u} \leq \begin{bmatrix} \phi_{\max} \\ \theta_{\max} \\ T_{ref,\max} \end{bmatrix} \right\}$$

Linear MPC

As per the receding horizon paradigm, MPC applies the first control input \mathbf{u}_0 at each iteration and the whole process is repeated iteratively.

Linear MPC

Note also that we account for non-planarity of the vehicle during position control by the compensating equation

$$\tilde{T}_{ref} = \frac{T_{ref} + g}{\cos \phi \cos \theta}$$

Linear MPC

A disturbance observer may be incorporated to this controller to ensure offset-free tracking.

We do so by augmenting the dynamics with a disturbance vector, while we then apply output feedback

$$\begin{bmatrix} \hat{\mathbf{x}}_{k+1} \\ \hat{\mathbf{F}}_{ext,k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B}_d \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}_k \\ \hat{\mathbf{F}}_{ext,k} \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix} \mathbf{u}_k + \begin{bmatrix} \mathbf{L}_x \\ \mathbf{L}_{F_{ext}} \end{bmatrix} (\mathbf{C}\hat{\mathbf{x}}_k - \mathbf{y}_{m,k})$$

where $\hat{\mathbf{x}}_k, \hat{\mathbf{F}}_{ext,k}, \mathbf{y}_{m,k}$ are the estimated state, external disturbances and measured output. $\mathbf{L}_x, \mathbf{L}_{F_{ext}}$ are the observer gains.

Linear MPC

Considering a stable observer, computing of the steady-state MPC state $\mathbf{x}_{ref,k}$ and control input $\mathbf{u}_{ref,k}$ at time k is through

$$\begin{bmatrix} \mathbf{A} - \mathbf{I} & \mathbf{B} \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{ref,k} \\ \mathbf{u}_{ref,k} \end{bmatrix} = \begin{bmatrix} -\mathbf{B}_d \hat{\mathbf{F}}_{ext,k} \\ \mathbf{r}_k \end{bmatrix}$$

where \mathbf{r}_k is the output vector reference at k time step.

Linear MPC

Coding through CVXGEN

```
1 dimensions
2   m = 3      # dimension of inputs.
3   nd = 3     # dimension of disturbances.
4   nx = 8     # dimension of state vector.
5   T = 18     # horizon - 1.
6 end
7
8 parameters
9   A (nx,nx)  # dynamics matrix.
10  B (nx,m)   # transfer matrix.
11  Bd (nx,nd) # disturbance transfer matrix
12  Q_x (nx,nx) psd # state cost, positive semidefined.
13  P (nx,nx) psd # final state penalty, positive semidefined.
14  R_u (m,m) psd # input penalty, positive semidefined.
15  R_delta (m,m) psd # delta input penalty, positive semidefined.
16  x[0] (nx)  # initial state.
17  d (nd)     # disturbances.
18  u_prev (m) # previous input applied to the system.
19  u_max (m)  # input amplitude limit.
20  u_min (m)  # input amplitude limit.
21  x_ss[t] (nx), t=0..T+1 # reference state.
22  u_ss[t] (m), t=0..T    # reference input.
23 end
24
25 variables
26   x[t] (nx), t=1..T+1 # state.
27   u[t] (m), t=0..T    # input.
28 end
29
30 minimize
```

```
31   quad(x[0]-x_ss[0], Q_x) + quad(u[0]-u_ss[0], R_u) + quad(u[0] -
   u_prev, R_delta) + sum[t=1..T](quad(x[t]-x_ss[t], Q) + quad(
   u[t]-u_ss[t], R_u) + quad(u[t] - u[t-1], R_delta))+quad(x[T
   +1]-x_ss[T+1], P)
32 subject to
33   x[t+1] == A*x[t] + B*u[t] + Bd*d , t=0..T # dynamics
34   u_min <= u[t] <= u_max, t=0..T # input constraint.
35 end
```

Nonlinear MPC

- Directly apply to the position dynamics subject to the identified closed-loop attitude dynamics.

Nonlinear MPC

State

$$\mathbf{x} = [\mathbf{p}^T \mathbf{v}^T \mathbb{I}\phi \mathbb{I}\theta \mathbb{I}\psi]^T$$
$$\mathbf{u} = [\mathbb{I}\phi_{ref} \mathbb{I}\theta_{ref} T_{ref}]^T$$

Nonlinear Optimal Control problem

$$\min_{\mathbf{U}} \int_{t=0}^T \|\mathbf{x}(t) - \mathbf{x}_{ref}(t)\|_{\mathbf{Q}_x}^2 + \|\mathbf{u}(t) - \mathbf{u}_{ref}(t)\|_{\mathbf{R}_u}^2 dt$$
$$+ \|\mathbf{x}(T) - \mathbf{x}_{ref}(T)\|_{\mathbf{P}}^2$$

$$\text{s.t.} \quad \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$
$$\mathbf{u}(t) \in \mathbb{U}$$
$$\mathbf{x}(0) = \mathbf{x}(t_0)$$

Nonlinear MPC

State

$$\mathbf{x} = [\mathbf{p}^T \mathbf{v}^T \ \mathbb{I}\phi \ \mathbb{I}\theta \ \mathbb{I}\psi]^T$$

$$\mathbf{u} = [\mathbb{I}\phi_{ref} \ \mathbb{I}\theta_{ref} \ T_{ref}]^T$$

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \quad \left\{ \begin{array}{l} \dot{\mathbf{p}} = \mathbf{v} \\ \dot{\mathbf{v}} = \frac{1}{m} \left(\mathbf{R}_{IB} \sum_{i=0}^{N_r} \mathbf{F}_{T,i} - \mathbf{R}_{IB} \sum_{i=0}^{N_r} \mathbf{F}_{aero,i} + \mathbf{F}_{ext} \right) + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \\ \dot{\phi} = \frac{1}{\tau_{\phi}} (k_{\phi} \phi_{ref} - \phi) \\ \dot{\theta} = \frac{1}{\tau_{\theta}} (k_{\theta} \theta_{ref} - \theta) \\ \dot{\psi} = \dot{\psi}_{ref} \end{array} \right.$$

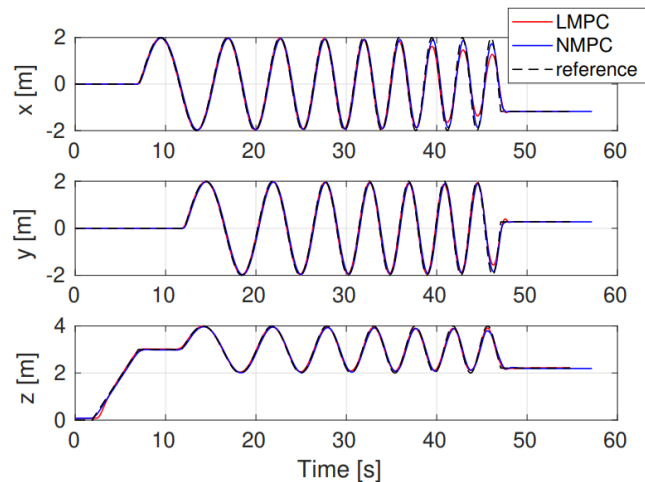
Nonlinear MPC

Coding through ACADO's Matlab interface

```
1 clc;
2 clear all;
3 close all;
4
5 Ts = 0.1; %sampling time
6 EXPORT = 1;
7
8 DifferentialState position(3) velocity(3) roll pitch yaw;
9 Control roll_ref pitch_ref thrust;
10
11 %online data represent data that can be passed to the solver
12   online.
13 OnlineData roll_tau;
14 OnlineData roll_gain;
15 OnlineData pitch_tau;
16 OnlineData pitch_gain;
17 OnlineData yaw_rate_command;
18 OnlineData drag_coefficient(3);
19 OnlineData external_disturbances(3);
20
21 nXD = length(diffStates);
22 nU = length(controls);
23
24 g = [0;0;9.8066];
25
26 %% Differential Equation
27 %define vehicle body z-axis expressed in inertial frame.
28 z_axis = [(cos(yaw)*sin(pitch)*cos(roll)+sin(yaw)*sin(roll));...
29           (sin(yaw)*sin(pitch)*cos(roll)-cos(yaw)*sin(roll));...
30           cos(pitch)*cos(roll)];
31
32 droll = (1/roll_tau)*(roll_gain*roll_ref - roll);
33 dpitch = (1/pitch_tau)*(pitch_gain*pitch_ref - pitch);
34
35 f = dot([position, velocity; roll; pitch; yaw]) == ...
36 [velocity ...;
37  z_axis*thrust - g - diag(drag_coefficient)*velocity +
38  external_disturbances;...
39  droll;...
40  dpitch;...
41  yaw_rate_command];
42
43 h = [position; velocity; roll; pitch; roll_ref; pitch_ref; z_axis
44      (3)*thrust-g(3)];
45
46 hN = [position; velocity];
47
48 %% NMPCexport
49 acadoSet('problemname', 'nmpe_trajectory_tracking');
50
51 N = 20;
52 ocp = acado.OCP( 0.0, N*Ts, N );
53
54 W_mat = eye(length(h));
55 WN_mat = eye(length(hN));
56 W = acado.BMatrix(W_mat);
57 WN = acado.BMatrix(WN_mat);
58
59 ocp.minimizeLSQ( W, h );
60 ocp.minimizeLSQEndTerm( WN, hN );
61 ocp.subjectTo( -deg2rad(45) <= [roll_ref; pitch_ref] <= deg2rad
62               (45));
63 ocp.subjectTo( g(3)/2.0 <= thrust <= g(3)*1.5);
64 ocp.setModel(f);
65
66 mpc = acado.OCPexport( ocp );
67 mpc.set( 'HESSIAN_APPROXIMATION', 'GAUSS_NEWTON' );
68 mpc.set( 'DISCRETIZATION_TYPE', 'MULTIPLE_SHOOTING' );
69
70 mpc.set( 'SPARSE_QP_SOLUTION', 'FULL_CONDENSING_N2' );
71 mpc.set( 'INTEGRATOR_TYPE', 'INT_IRK_GL4' );
72 mpc.set( 'NUM_INTEGRATOR_STEPS', N );
73 mpc.set( 'QP_SOLVER', 'QP_OQPOASES' );
74 mpc.set( 'HOTSTART_QP', 'NO' );
75 mpc.set( 'LEVENBERG_MARQUARDT', 1e-10 );
76 mpc.set( 'LINEAR_ALGEBRA_SOLVER', 'GAUSS_LU' );
77 mpc.set( 'IMPLICIT_INTEGRATOR_NUM_ITS', 4 );
78 mpc.set( 'CG_USE_OPENMP', 'YES' );
79 mpc.set( 'CG_HARDCODE_CONSTRAINT_VALUES', 'NO' );
80 mpc.set( 'CG_USE_VARIABLE_WEIGHTING_MATRIX', 'NO' );
81
82 if EXPORT
83   mpc.exportCode( 'mav_NMPC_trajectory_tracking' );
84   cd mav_NMPC_trajectory_tracking
85   make_acado_solver( '../mav_NMPC_trajectory_tracking' )
86   cd ..
87 end
```

Remarks

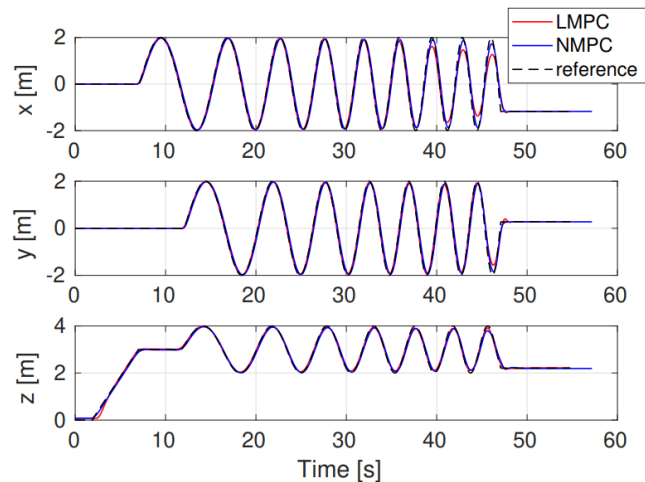
- Nonlinear MPC:
 - Exploit full dynamic model
 - Better tracking performance than Linear MPC but not much in non-aggressive flight



Tracking performance with sinusoidal reference

Remarks

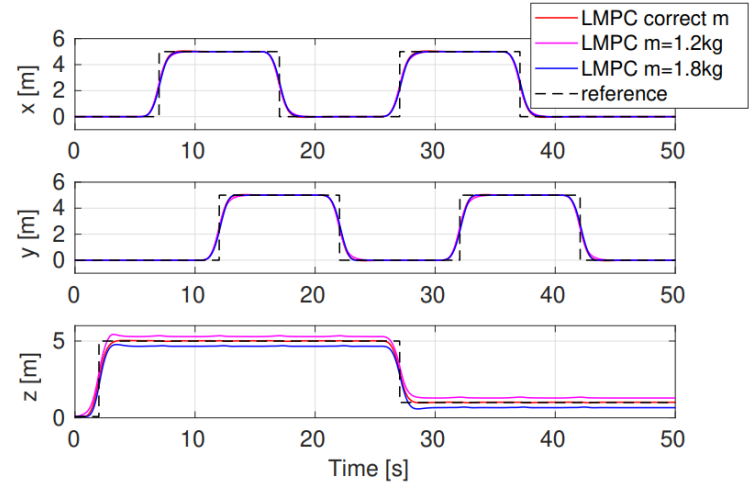
- Nonlinear MPC:
 - Exploit full dynamic model
 - Better tracking performance than Linear MPC but not much in non-aggressive flight



Tracking performance with sinusoidal reference

Remarks

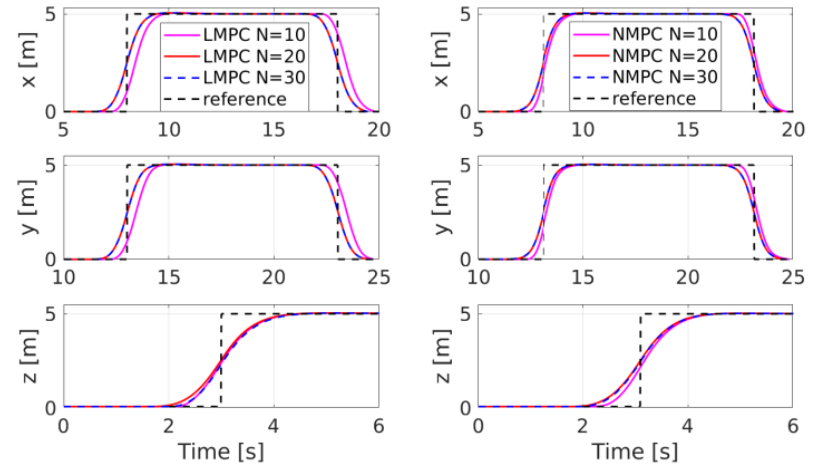
- Nonlinear MPC:
 - Exploit full dynamic model
 - Better tracking performance than Linear MPC but not much in non-aggressive flight
- Disturbance observer is necessary



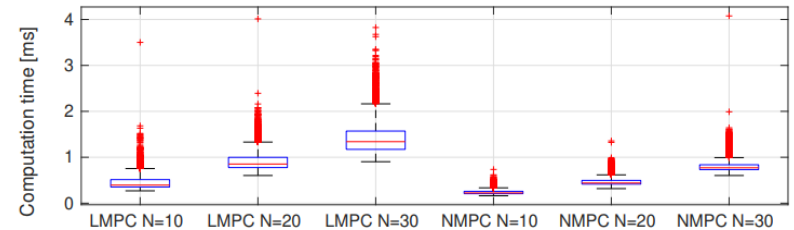
Tracking performance under parameter uncertainty

Remarks

- Nonlinear MPC:
 - Exploit full dynamic model
 - Better tracking performance than Linear MPC but not much in non-aggressive flight
- Disturbance observer is necessary
- Larger prediction horizon:
 - Generally increase tracking performance
 - Larger computation time
 - Tracking performance saturates at some prediction horizon



Tracking performance with different prediction horizon







4x



University of
Zurich^{UZH}



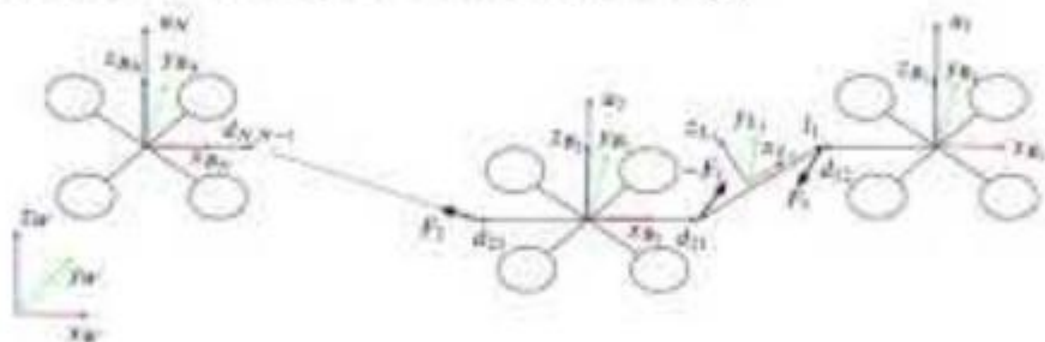
ROBOTICS &
PERCEPTION
GROUP

Performance, Precision, and Payloads:
Adaptive Nonlinear MPC for Quadrotors



Drew Hanover, Philipp Foehn, Sihao Sun, Elia Kaufmann Davide Scaramuzza

Aerial Robotic Chain: Modeling



Assumption 1: the masses of the links are negligible (not a must)

The internal force applied by **link i** to the neighbor ARC-units can be expressed as

$$F_i = f_i(R_1, \Omega_1, \dots, R_N, \Omega_N, R_{L_1}, \Omega_{L_1}, \dots, R_{L_{N-1}}, \Omega_{L_{N-1}}, u_1, \dots, u_N, M_1, \dots, M_N)$$

From that we can derive the nonlinear dynamics of the system

$$\dot{x} = F(x, u_1, \dots, u_N, M_1, \dots, M_N)$$

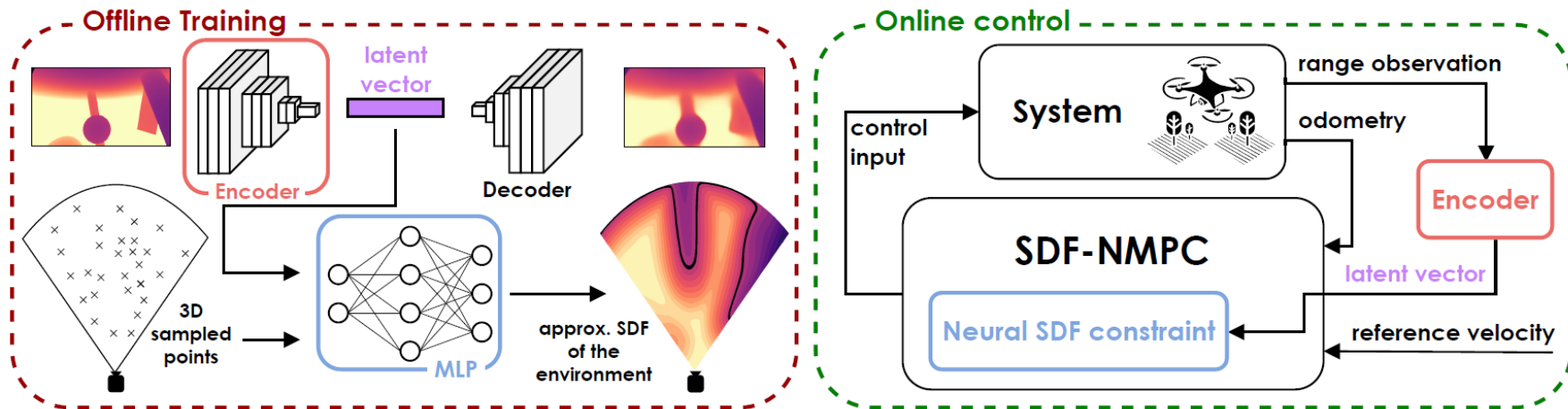
$$\text{with } x = (x_{L_1}, v_{L_1}, R_1, \Omega_1, \dots, R_N, \Omega_N, R_{L_1}, \Omega_{L_1}, \dots, R_{L_{N-1}}, \Omega_{L_{N-1}})^T$$

Link 1's position
and velocity

ARC-units' angles
and angular rates

Links' angles
and angular rates

Neural MPC



Code Examples and Tasks

- https://github.com/ethz-asl/mav_control_rw
- https://github.com/uzh-rpg/rpg_mpc
- <https://cvxgen.com/docs/index.html>
- http://acado.sourceforge.net/doc/html/d4/d26/example_013.html

Find out more

- Kamel, M., Stastny, T., Alexis, K. and Siegwart, R., 2017. Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system. Robot Operating System (ROS) The Complete Reference (Volume 2), pp.3-39.
- Nguyen, H., Kamel, M., Alexis, K. and Siegwart, R., 2021, June. Model predictive control for micro aerial vehicles: A survey. In 2021 European Control Conference (ECC) (pp. 1556-1563). IEEE.
- Steve Brunton, Model Predictive Control (video lecture), <https://youtu.be/YwodGM2eoy4>