Fast Model Predictive Contol and Moving Horizon Estimation for Tethered Planes

Mario Zanon

Sébastien Gros, Moritz Diehl

Outline

Model Predictive Control

$$\begin{aligned}
\min_{x,u} \quad & \|x(t_k+t_h) - x^r(t_k+t_h)\|_P + \int_{t_k}^{t_k+t_h} \|x(t) - x^r(t)\|_Q \\
& \qquad + \|u(t) - u^r(t)\|_R dt \\
\text{s.t.} \quad & x(t_k) - \hat{x}(t_k) = 0, \\
& f(\dot{x}(t), x(t), z(t), u(t)) = 0, \\
& h(x(t), z(t), u(t)) \geq 0, \\
& x(t_k+t_h) \in \mathbb{X}_f,
\end{aligned} \tag{1}$$

where $\|w\|_S = w^T S w$.

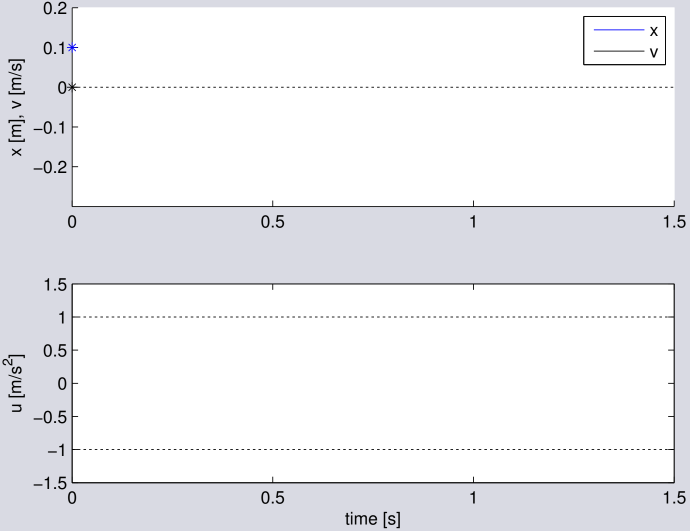
- At each sampling time t_k:
  - get the (estimated) initial state $\hat{x}(t_k)$
  - solve the OCP (1)
  - apply the control $u^*(t_k)$, solution of (1)

A simple illustrative example

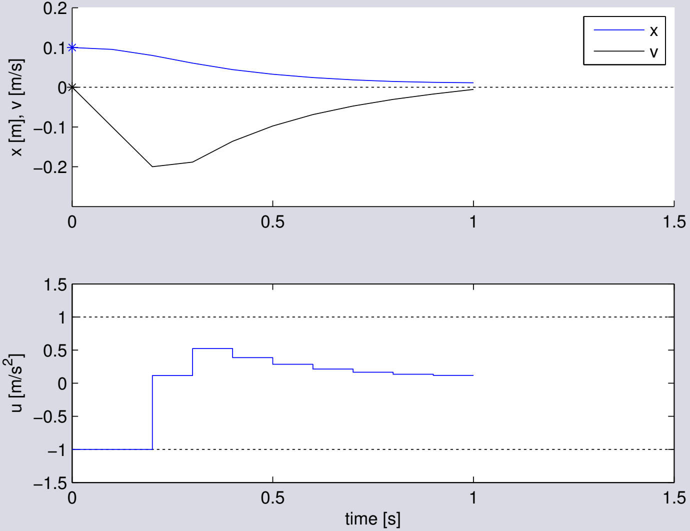
$$\dot{x} = v$$
$$\dot{v} = u$$

- Very simple system, input bounds, solution in the μs timescale
- Without constraints $\Rightarrow$ LQR $\equiv$ MPC
- I am lazy $\Rightarrow$ I used ACADO Code Generation
- Generated code called in Matlab with a mex
- Purpose:
  - illustrate how MPC works
  - advertise ACADO Code Generation *http://www.acadotoolkit.org*

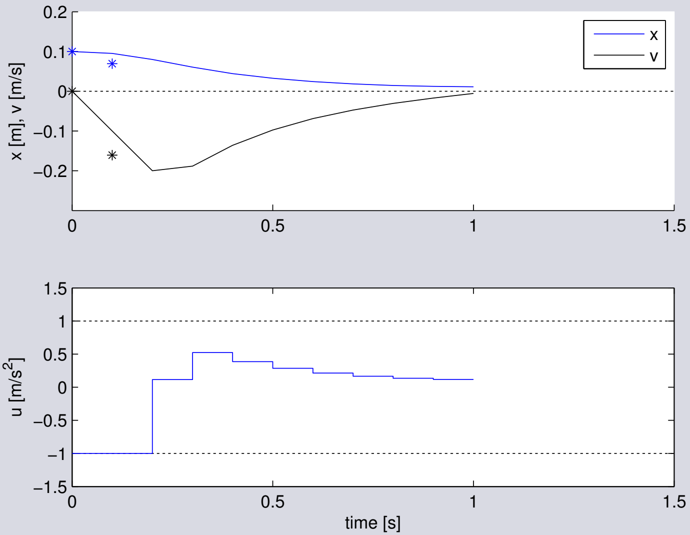
A simple illustrative example



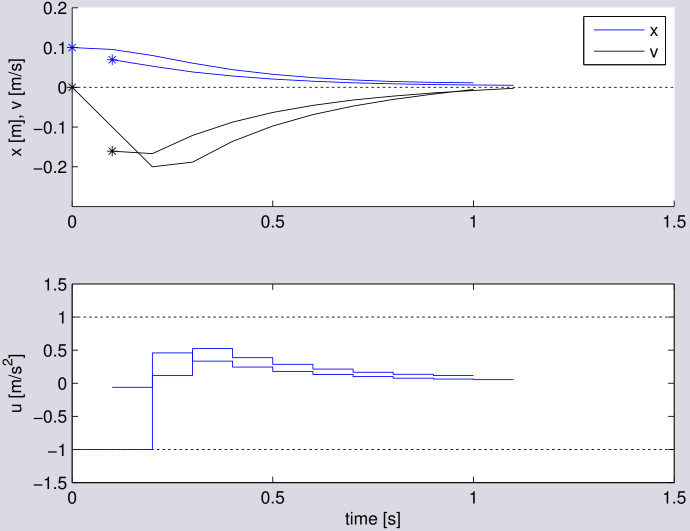
A simple illustrative example



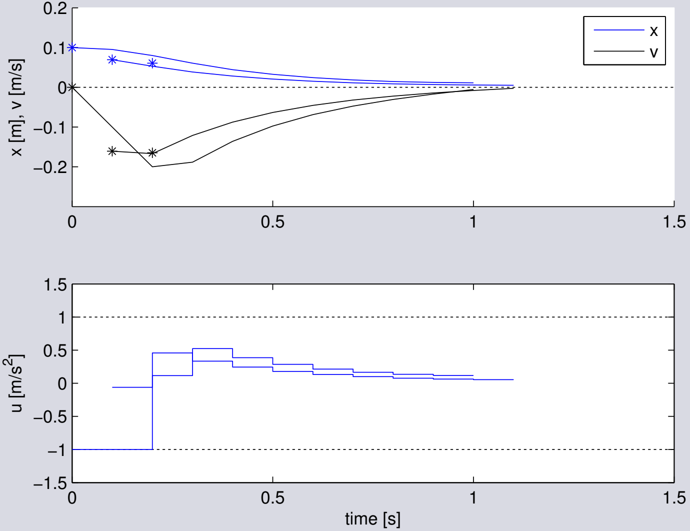
A simple illustrative example



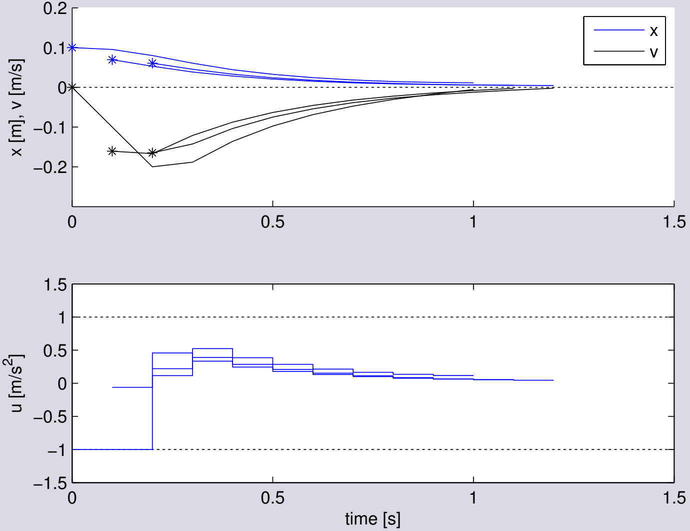
A simple illustrative example



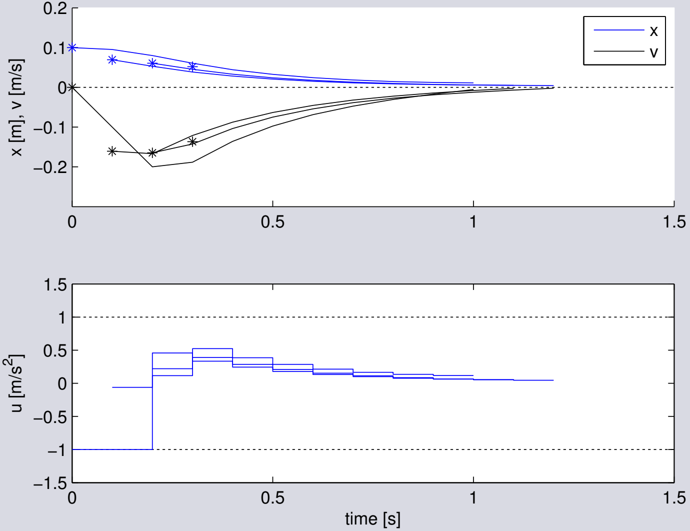
A simple illustrative example



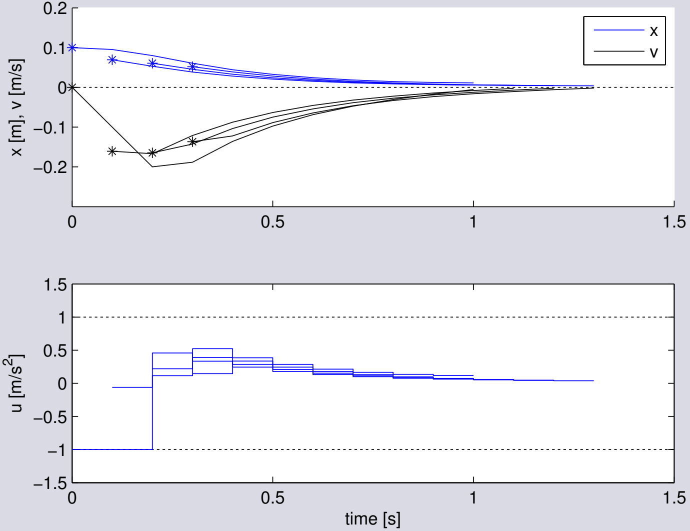
A simple illustrative example



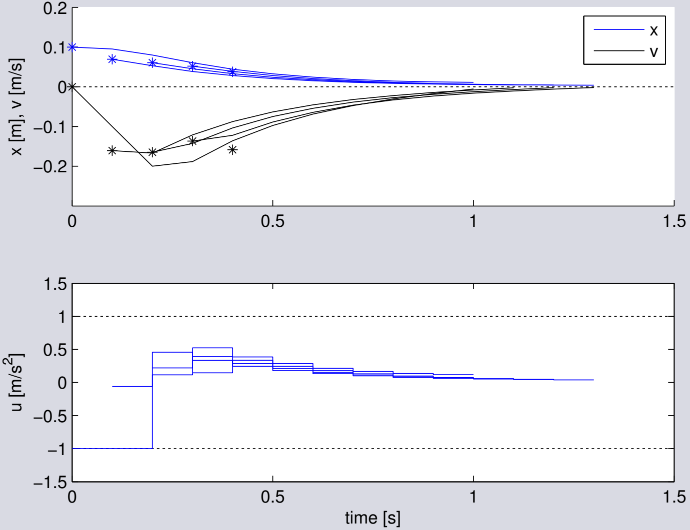
A simple illustrative example



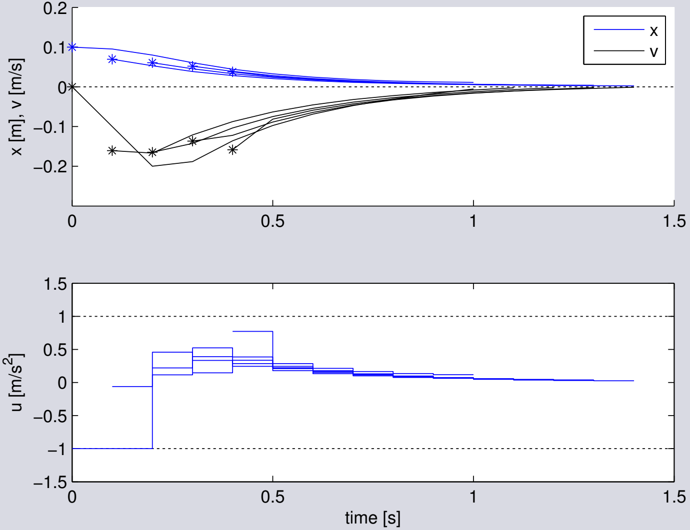
A simple illustrative example



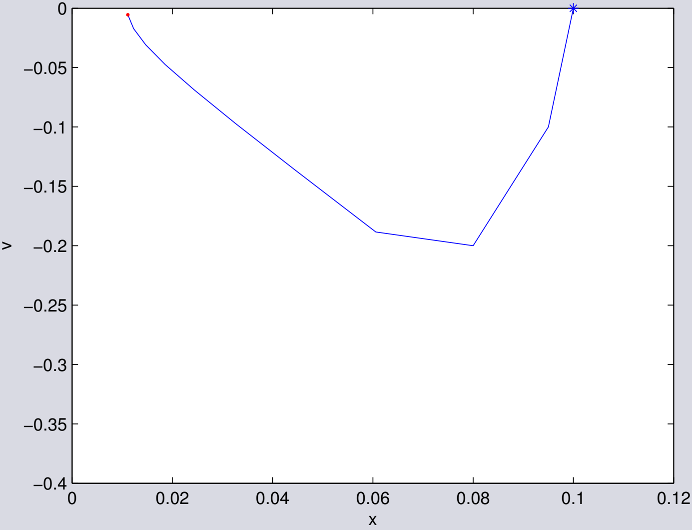
A simple illustrative example



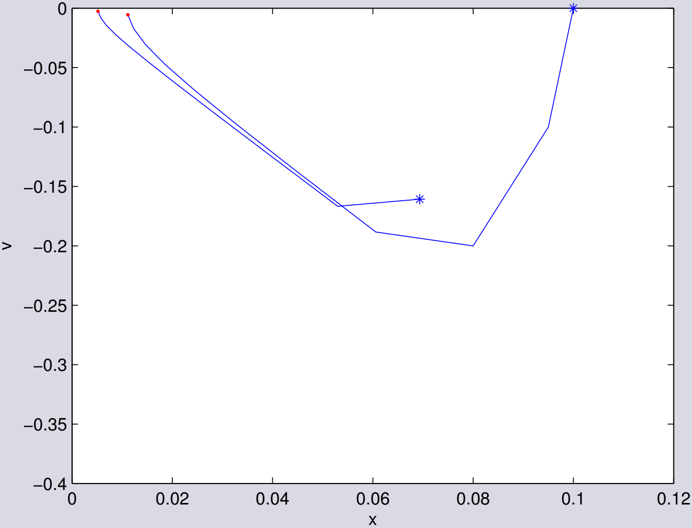
A simple illustrative example



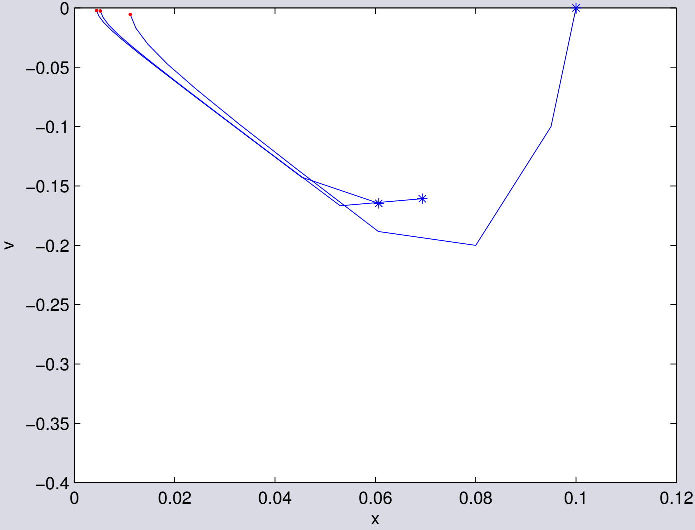
The phase diagram



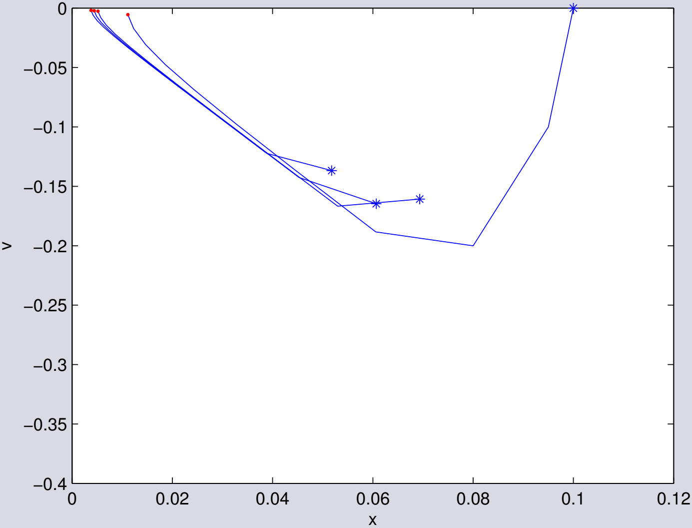
The phase diagram



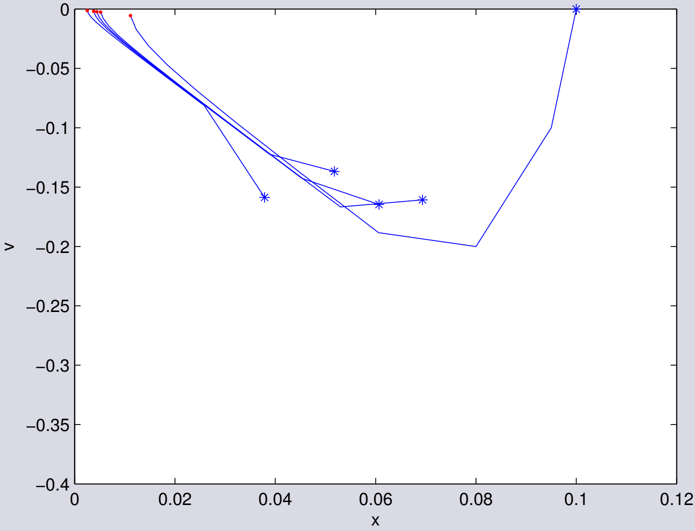
The phase diagram



The phase diagram



The phase diagram



Moving Horizon Estimation

$$\begin{aligned}
\min_{x,u} \quad & \|x(t_k - t_h) - \hat{x}(t_k - t_h)\|_{\Sigma_s^{-1}} + \int_{t_k - t_h}^{t_k} \|y(t) - y^m(t)\|_{\Sigma_y^{-1}} \\
& \qquad + \|u(t) - u^m(t)\|_{\Sigma_u^{-1}} dt \\
\text{s.t.} \quad & f(\dot{x}(t), x(t), z(t), u(t)) = 0, \\
& y(t) - g(x(t), z(t), u(t)) = 0, \\
& h(x(t), z(t), u(t)) \geq 0, \\
& c(x(t_k)) = 0, \qquad (2)
\end{aligned}$$

where $\|w\|_S = w^T S w$.

- At each sampling time t_k:
  - get the measured output y^m and control u^m
  - solve the OCP (2)
  - output the estimated state $\hat{x}(t_k) = x^*(t_k)$, solution of (2)

Problem formulation: NLP

$$\begin{aligned} \min_x \ & f(x) \\ \text{s.t.} \ & g(x) = 0 \\ & h(x) \geq 0 \end{aligned} \tag{3}$$

Newton type algorithm

Given an initial guess x_0, keep iterating:

1. determine a (descent) direction p_k
2. determine a step length α_k
3. compute the step: $x_{k+1} = x_k + \alpha_k p_k$
4. check for convergence and return the solution

Let's look at the simpler problem

$$\min_x f(x)$$

First order necessary condition for optimality: $\nabla f(x^*) = 0$

Let's look at the simpler problem

$$\min_x f(x)$$

First order necessary condition for optimality: $\nabla f(x^*) = 0$

Newton's Method

Linearize $\nabla f(x) = 0$ to obtain:

$$\nabla f(x_k) + \nabla^2 f(x_k) p_k = 0$$

$$\Updownarrow$$

$$p_k = -\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)$$

Let's look at the simpler problem

$$\min_x f(x)$$

First order necessary condition for optimality: $\nabla f(x^*) = 0$

Newton's Method

Linearize $\nabla f(x) = 0$ to obtain:

$$\nabla f(x_k) + \nabla^2 f(x_k) p_k = 0$$

$$\Updownarrow$$

$$p_k = -\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)$$

Newton Type Algorithms

Replace $\nabla^2 f(x_k)$ with a suitable approximation

Some Hessian Approximations

1. Steepest descent:

$$\nabla^2 f(x_k) \approx \mathbb{I}$$

Convergence rate: linear (bad)

Some Hessian Approximations

1. Steepest descent:

$$\nabla^2 f(x_k) \approx \mathbb{I}$$

Convergence rate: linear (bad)

2. Gauss-Newton:

$$\nabla^2 f(x_k) \approx \nabla F(x_k)^T \nabla F(x_k), \qquad f(x) = \|F(x)\|_2^2$$

Convergence rate: linear (good)

Some Hessian Approximations

1. Steepest descent:

$$\nabla^2 f(x_k) \approx \mathbb{I}$$

Convergence rate: linear (bad)

2. Gauss-Newton:

$$\nabla^2 f(x_k) \approx \nabla F(x_k)^T \nabla F(x_k), \qquad f(x) = \|F(x)\|_2^2$$

Convergence rate: linear (good)

3. BFGS update:

$$\nabla^2 f(x_k) \approx B_k, \qquad B_{k+1} = B_k + \frac{B_k s s^T B_k}{s^T B_k s} + \frac{y y^t}{s^T y}$$

Convergence rate: superlinear

Why does Gauss-Newton perform so well?

The exact Hessian is given by:

$$\nabla_x^2 \mathcal{L} = \nabla^2 f - \sum \lambda_i \nabla^2 g_i - \sum \mu_i \nabla^2 h_i$$

with

$$\nabla^2 f = J^T J + \sum F_i \nabla^2 F_i$$

and $J = \nabla F^T$.

In Gauss-Newton: $\nabla_x^2 \mathcal{L} \approx J^T J$.

When does it perform particularly well?

- $\|F\|$ small: good fit
- $\nabla^2 F_i$ small: F is nearly linear
- $\|\lambda\|$ and $\|\mu\|$ small (true when $\|F\|$ is small)

Least Squares NLP

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \frac{1}{2} \| F(x) \|_2^2 \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \geq 0 \end{aligned} \tag{4}$$

Linearize (4) at x_k (inside the norm)

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \frac{1}{2} \| F(x_k) + J(x_k)\,(x - x_k) \|_2^2 \\ \text{s.t.} \quad & g(x_k) + \nabla g(x_k)^T\,(x - x_k) = 0 \\ & h(x_k) + \nabla h(x_k)^T\,(x - x_k) \geq 0 \end{aligned} \tag{5}$$

where $J(x_k) = \nabla F(x_k)^T$

Let's rewrite (5)

$$
\begin{aligned}
\min_{\Delta x \in \mathbb{R}^n} \quad & \frac{1}{2}\Delta x^T J^T J \Delta x + F^T J \Delta x + \frac{1}{2} F^T F \\
\text{s.t.} \quad & g + \nabla g^T \Delta x = 0 \\
& h + \nabla h^T \Delta x \geq 0
\end{aligned} \tag{6}
$$

where $\Delta x = (x - x_k)$ and $f = f(x_k)$.

In the absence of inequality constraints

The solution of (6) is equivalent to the solution of:

$$
\begin{bmatrix} J^T J & \nabla g \\ \nabla g^T & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ -\lambda \end{bmatrix} = - \begin{bmatrix} J^T F \\ g \end{bmatrix} \tag{7}
$$

True for every Newton type method, $J^T J$ replaced by B_k.

In the general case

Problem (6) is a QP. Two solution approaches are possible:

1. Active set methods
2. Interior point methods

1. Active set

Given a feasible initial guess x_0 with corresponding active set $\mathcal{A}_0 = \{i | h_i(x_0) = 0\}$, iterate:

1. solve equation (7)
2. update the active set (only linear algebra, no matrix update)
3. if no active set change: solution found

2. Interior point

Modify the NLP to get a Barrier Problem:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \geq 0 \end{aligned} \quad \Rightarrow \quad \begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) - \tau \sum_{i=1}^{q} log\left(h_i(x)\right) \\ \text{s.t.} \quad & g(x) = 0 \\ & h_i(x)\mu_i - \tau = 0 \end{aligned} \tag{8}$$

Given an initial guess x_0, start with a big $\tau \gg 0$, choose $\beta \in (0, 1)$ and iterate:

1. solve (8)
2. update $\tau = \beta\tau$
3. check for convergence

Which method to choose?

- IP methods:
  - have guarantees on maximum runtime
  - can directly solve NLPs (no SQP)
  - perform well especially for large NLPs
- Active set methods:
  - can be warm started
  - perform extremely well if the initial guess is good
  - particularly suited for homotopies (no need to go back to the central path)

Proposed scheme to solve the MPC problem

1. MPC: iteratively solve an OCP (usually to track a reference)

Proposed scheme to solve the MPC problem

1. MPC: iteratively solve an OCP (usually to track a reference)
2. OCP: translated into a (LSQ) NLP (direct multiple shooting)

Proposed scheme to solve the MPC problem

1. MPC: iteratively solve an OCP (usually to track a reference)
2. OCP: translated into a (LSQ) NLP (direct multiple shooting)
3. NLP: SQP method with Gauss-Newton approximation

Proposed scheme to solve the MPC problem

1. MPC: iteratively solve an OCP (usually to track a reference)
2. OCP: translated into a (LSQ) NLP (direct multiple shooting)
3. NLP: SQP method with Gauss-Newton approximation
4. QP: active set strategy, warm-start, RTI

Proposed scheme to solve the MPC problem

1. MPC: iteratively solve an OCP (usually to track a reference)
2. OCP: translated into a (LSQ) NLP (direct multiple shooting)
3. NLP: SQP method with Gauss-Newton approximation
4. QP: active set strategy, warm-start, RTI
5. Code generation

Proposed scheme to solve the MPC problem

1. MPC: iteratively solve an OCP (usually to track a reference)
2. OCP: translated into a (LSQ) NLP (direct multiple shooting)
3. NLP: SQP method with Gauss-Newton approximation
4. QP: active set strategy, warm-start, RTI
5. Code generation

Not covered in this talk

- Discretization method (single-multiple shooting, collocation):
  - need for an integrator
  - sensitivity computation (AD)
- Detailed description of the QP solution strategy
- Code generation

Real Time Iterations

At each sampling time perform:

- integration and sensitivities: compute the QP matrices
- condensing: large and sparse QP reduced to small and dense
- QP solution (qpOASES)

Real Time Iterations

At each sampling time perform:

- integration and sensitivities: compute the QP matrices
- condensing: large and sparse QP reduced to small and dense
- QP solution (qpOASES)

Most computations can be prepared without prior knowledge of the initial state $\hat{x}_0$:

- Preparation phase: integrate the system, condense, set up the QP
- Feedback phase: solve the QP and immediately apply u_0

Real Time Iterations

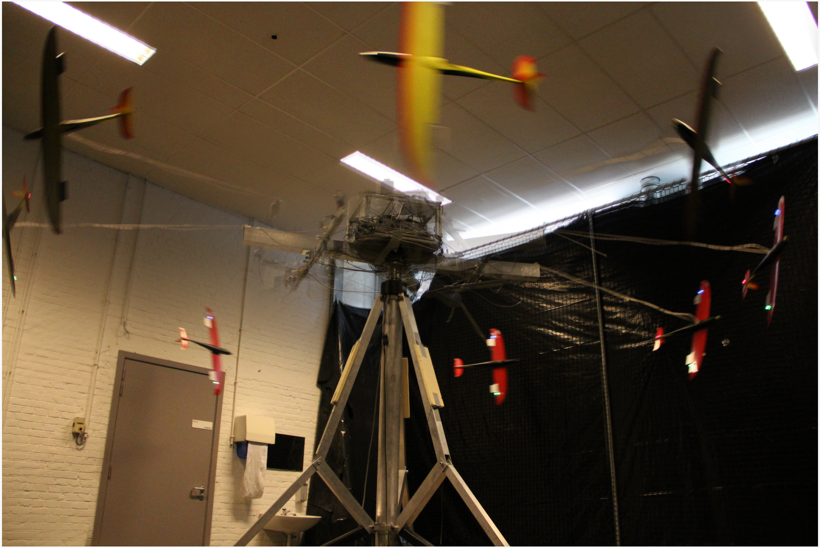
At each sampling time perform:

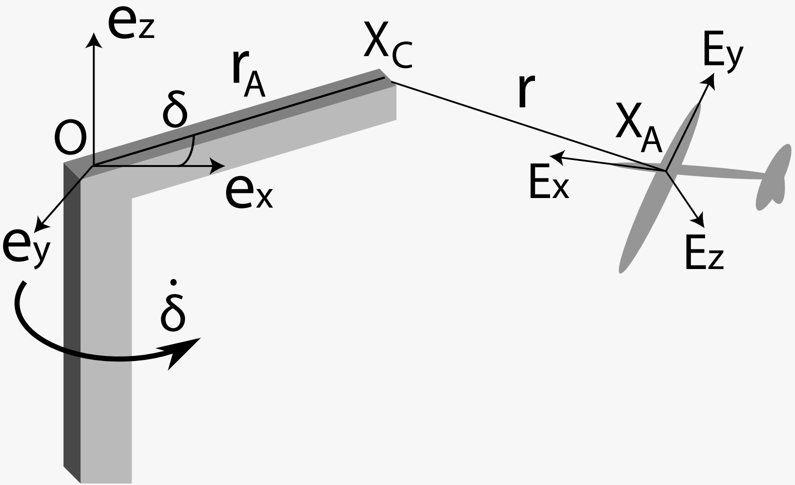
- integration and sensitivities: compute the QP matrices
- condensing: large and sparse QP reduced to small and dense
- QP solution (qpOASES)

Most computations can be prepared without prior knowledge of the initial state $\hat{x}_0$:

- Preparation phase: integrate the system, condense, set up the QP
- Feedback phase: solve the QP and immediately apply u_0

- being fast is often more important than being accurate
- converge while the system dynamics evolve




ez
rA
XC
r
Ey
δ
O
XA
Ex
ex
ey
Ez
δ̇

Model features

- Full rotational model, 22 states, 3 controls
- Tether: constraint
- Rotation: full parametrization of the rotation matrix

Model equations: index 1 DAE (after index reduction)

$$\ddot{\delta} = u_\delta,$$

$$\dot{R} = R\Omega,$$

$$J\dot{\omega} = T_A - \omega \times J\omega$$

$$\begin{bmatrix} m \cdot I_3 & -X \\ -X^T & 0 \end{bmatrix} \begin{bmatrix} \ddot{X} \\ z \end{bmatrix} = \begin{bmatrix} F - \mathcal{V}_X - \dot{m}\dot{X} \\ \dot{X}^T\dot{X} \end{bmatrix},$$

$$\left(X^T X - r^2\right)_{t=t_0} = 0, \quad \left(X^T \dot{X}\right)_{t=t_0} = 0,$$

MPC problem formulation

$$\begin{aligned} \min_{x,u} \quad & \|x_N - x_N^r\|_P + \sum_i \|x_i - x_i^r\|_Q + \|u_i - u_i^r\|_R \\ \text{s.t.} \quad & x_0 - \hat{x}_k = 0, \\ & f(x_{i+1}, x_i, z_i, u_i) = 0, \\ & h(x_i, z_i, u_i) \geq 0, \end{aligned}$$

- terminal cost: LQR (stabilize the invariants)
- no terminal constraint
- path constraints: $-1 \leq C_L \leq 1$, $z \geq 0$ (not yet included)
- possible slack reformulation to increase feasibility

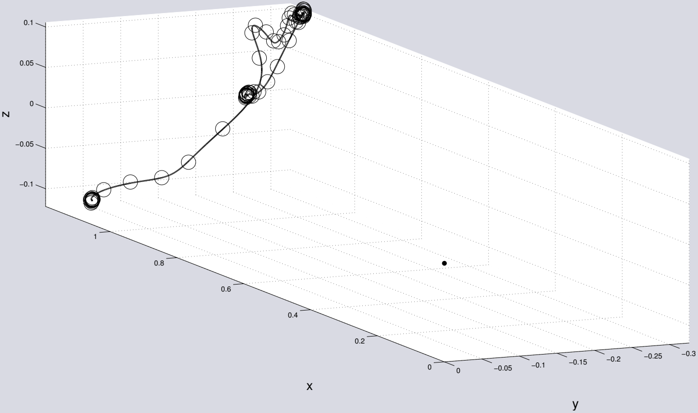
MHE problem formulation

$$\begin{aligned}
\min_{x,u} \quad & \|x_0 - \hat{x}_0\|_{\Sigma_x^{-1}} + \sum_i \|y_i - y_i^m\|_{\Sigma_y^{-1}} + \|u_i - u_i^m\|_{\Sigma_u^{-1}} \\
\text{s.t.} \quad & f(x_{i+1}, x_i, z_i, u_i) = 0, \\
& y_i - g(x_i, z_i, u_i) = 0, \\
& c(x_N) = 0,
\end{aligned}$$

- terminal constraint: enforce the invariants
- measurements:
  - IMU $(\ddot{X}, \omega)$
  - 2 cameras (3 LED position)
  - encoder (δ)
- no path constraints (rotation matrix)
- no arrival cost (not implemented yet, available for free)

Simulation results and computational times

	MHE	MPC
Preparation phase	7 ms	5 ms
Feedback phase	1 ms	0.2 ms



Experimental results

Movie

Thank you for your attention!