Online trajectory generation for mobile robots using model-predictive control

Winter School

Mathematics for Engineering Applications

Politecnico di Bari, January 27-31, 2020

Matthias Gerdts

Institute of Applied Mathematics and Scientific Computing
Department of Aerospace Engineering
Universität der Bundeswehr München (UniBw M)
matthias.gerdts@unibw.de
http://www.unibw.de/ingmathe, http://www.optimal-control.de

Fotos: http://de.wikipedia.org/wiki/München
Magnus Manske (Panorama), Luidger (Theatinerkirche), Kurmis (Chin. Turm), Arad Mojtahedi (Olympiapark), Max-k (Deutsches Museum), Oliver Raupach (Friedensengel), Andreas Praefcke (Nationaltheater)

Universität der Bundeswehr München

Universität der Bundeswehr München
Professur für Ingenieurmathematik

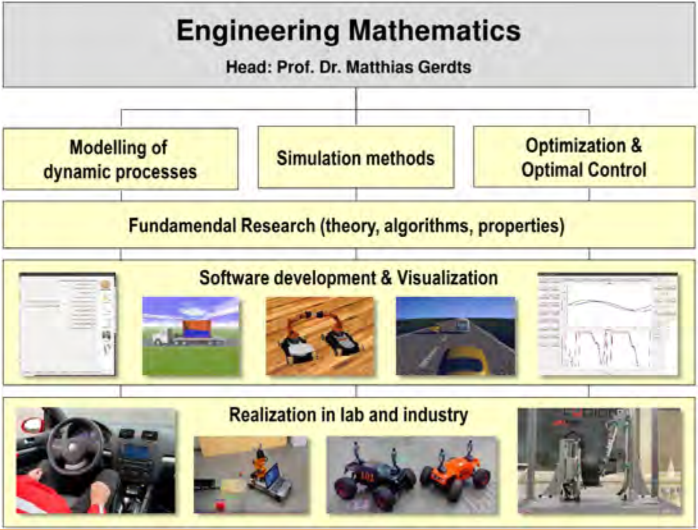
Chair of Engineering Mathematics @ Department of Aerospace Engineering

Prof Dr. Matthias Gerdts
Head of Engineering Mathematics Group
matthias.gerdts@unibw.de
http://www.unibw.de/ingmathe
http://www.optimal-control.de

Curriculum Vitae

1992 – 1997	studies of Mathematics with minor Computer Science, TU Clausthal
1997 – 2001	Phd, TU Clausthal/Uni Bayreuth, "Simulation of test-drives at driving limit"
2001 – 2004	assistant lecturer, Uni Bayreuth
2004 – 2007	Junior Professor (W1) for "Optimal Control", Uni Hamburg
2006	Habilitation, Uni Bayreuth
2007 – 2009	Lecturer for "Mathematical Optimization", University of Birmingham, U.K.
2009 – 2010	Associate Professor (W2) for "Optimal Control", Uni Würzburg
since 2010	Full Professor (W3) for "Engineering Mathematics", UniBw München

Research @ Engineering Mathematics

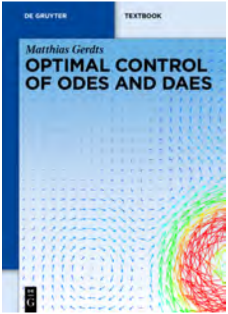


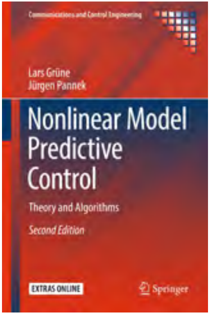
Schedule

Online trajectory generation for mobile robots using model-predictive control

lecture	duration	topic
1 (Mon 09:30-11:00)	90 min	Introduction into Model-Predictive Control (MPC)
2 (Tue 17:00-18:00)	60 min	Numerical Methods and Structure Exploitation
3 (Wed 16:00-17:00)	60 min	Theory of MPC
4 (Fri 11:30-13:00)	90 min	Realtime Approaches and Applications

Literature and Resources

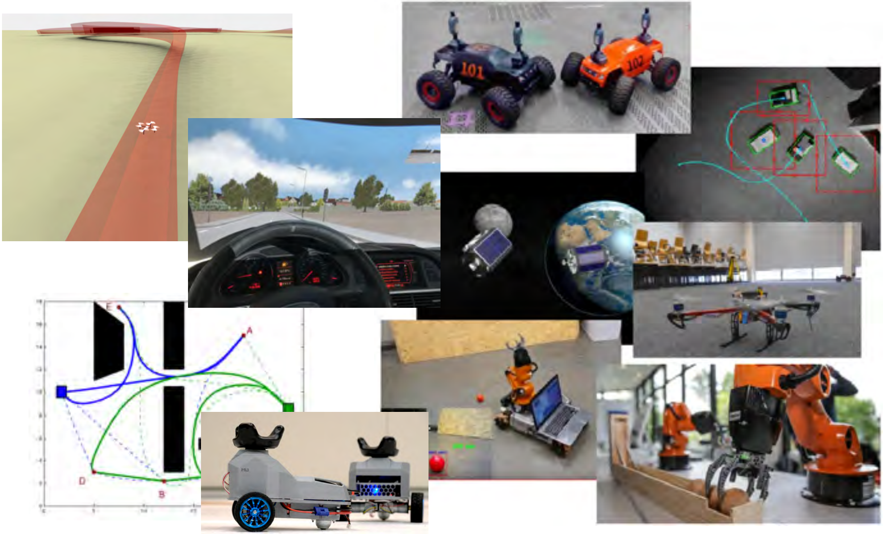




Contents

Contents

Path Planning and Control



Introduction

Basic control task:

Control a dynamic system through control inputs to achieve a desired behavior!

Introduction

Basic control task:

Control a dynamic system through control inputs to achieve a desired behavior!

What is a dynamic system?

Introduction

Basic control task:

Control a dynamic system through control inputs to achieve a desired behavior!

What is a dynamic system?

- A dynamic system is a (technical, biological, economical, ...) system in motion.

Introduction

Basic control task:

> Control a dynamic system through control inputs to achieve a desired behavior!

What is a dynamic system?

- A dynamic system is a (technical, biological, economical, ...) system in motion.
- The state $x(t) \in \mathbb{R}^n$ of the system (e.g. position, velocity, ...) changes with time t.

Introduction

Basic control task:

> Control a dynamic system through control inputs to achieve a desired behavior!

What is a dynamic system?

- A dynamic system is a (technical, biological, economical, ...) system in motion.
- The state $x(t) \in \mathbb{R}^n$ of the system (e.g. position, velocity, ...) changes with time t.
- The system at time t can be influenced by a control input $u(t) \in \mathbb{R}^m$.

Introduction

Dynamic system in continuous time:

$$x(0) = x_0 \qquad \text{(given initial state)}$$

$$x'(t) = F(x(t), u(t)) \qquad (t \geq 0)$$

Introduction

Dynamic system in continuous time:

$$x(0) = x_0 \qquad \text{(given initial state)}$$
$$x'(t) = F(x(t), u(t)) \qquad (t \geq 0)$$

Dynamic system in discrete time:

$$x(0) = x_0 \qquad \text{(given initial state)}$$
$$x(n+1) = f(x(n), u(n)) \qquad (n = 0, 1, 2, \ldots)$$

Note: discrete time $n \triangleq t_n$ and $x(n) \triangleq x(t_n)$

Introduction

Dynamic system in continuous time:

$$\begin{aligned} x(0) &= x_0 && \text{(given initial state)} \\ x'(t) &= F(x(t), u(t)) && (t \geq 0) \end{aligned}$$

Dynamic system in discrete time:

$$\begin{aligned} x(0) &= x_0 && \text{(given initial state)} \\ x(n+1) &= f(x(n), u(n)) && (n = 0, 1, 2, \ldots) \end{aligned}$$

Note: discrete time $n \hat{=} t_n$ and $x(n) \hat{=} x(t_n)$

x : state $\qquad$ u : control

Open-loop Control vs Closed-loop Control

Open-loop control

Apply a **time-dependent** control $u(t)$ (**open-loop control**) to the dynamic system:

$$x'(t) = F(x(t), u(t)), \qquad x(0) = x_0.$$

Open-loop Control vs Closed-loop Control

Open-loop control

Apply a **time-dependent** control $u(t)$ (**open-loop control**) to the dynamic system:

$$x'(t) = F(x(t), u(t)), \qquad x(0) = x_0.$$

$\rightsquigarrow$ an open-loop control cannot react on perturbations in x!

Open-loop Control vs Closed-loop Control

Open-loop control

Apply a **time-dependent** control $u(t)$ (**open-loop control**) to the dynamic system:

$$x'(t) = F(x(t), u(t)), \qquad x(0) = x_0.$$

$\leadsto$ an open-loop control cannot react on perturbations in x!

Closed-loop control / feedback control

Application of a **feedback control law** of type $u(t) = \mu(x(t))$ to the dynamic system yields a **closed-loop system**:

$$x'(t) = F(x(t), \mu(x(t))), \qquad x(0) = x_0.$$

Open-loop Control vs Closed-loop Control

Open-loop control

Apply a **time-dependent** control $u(t)$ (**open-loop control**) to the dynamic system:

$$x'(t) = F(x(t), u(t)), \qquad x(0) = x_0.$$

$\rightsquigarrow$ an open-loop control cannot react on perturbations in x!

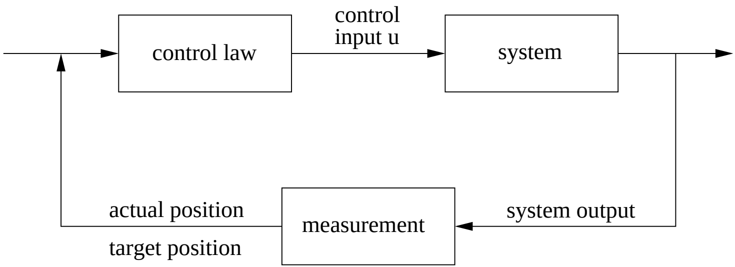
Closed-loop control / feedback control

Application of a **feedback control law** of type $u(t) = \mu(x(t))$ to the dynamic system yields a **closed-loop system**:

$$x'(t) = F(x(t), \mu(x(t))), \qquad x(0) = x_0.$$

$\rightsquigarrow$ a feedback control is able to react on perturbations in x!

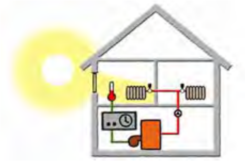
General Feedback-control Scheme



Measurements:

position (GPS), velocity (Hall sensor), acceleration (IMU), pressure, temperature, altitude, . . .

Example: Controlling a Radiator



house/room

radiator

Keep temperature at a given level!

thermostat/controller

Example: Controlling a Radiator

Mathematical model:

$x_1(t_n)$: room temperature at time t_n

Example: Controlling a Radiator

Mathematical model:

$x_1(t_n)$: room temperature at time t_n

$x_2(t_n)$: temperature of radiator at time t_n

Example: Controlling a Radiator

Mathematical model:

$x_1(t_n)$: room temperature at time t_n

$x_2(t_n)$: temperature of radiator at time t_n

$u(t_n)$: thermostat/control

$u > 0$ increases temperature of radiator

$u < 0$ reduces temperature of radiator

Example: Controlling a Radiator

Mathematical model:

$x_1(t_n)$	:	room temperature at time t_n
$x_2(t_n)$	:	temperature of radiator at time t_n
$u(t_n)$	:	thermostat/control
		$u > 0$ increases temperature of radiator
		$u < 0$ reduces temperature of radiator
h	:	time period/step size, $h = t_{n+1} - t_n$

Example: Controlling a Radiator

Mathematical model:

$x_1(t_n)$	:	room temperature at time t_n
$x_2(t_n)$	:	temperature of radiator at time t_n
$u(t_n)$	:	thermostat/control
		$u > 0$ increases temperature of radiator
		$u < 0$ reduces temperature of radiator
h	:	time period/step size, $h = t_{n+1} - t_n$
x_1^*	:	target temperature, $x_1^* = 0$ for simplicity

Example: Controlling a Radiator

Mathematical model:

$x_1(t_n)$	:	room temperature at time t_n
$x_2(t_n)$	:	temperature of radiator at time t_n
$u(t_n)$	:	thermostat/control
		$u > 0$ increases temperature of radiator
		$u < 0$ reduces temperature of radiator
h	:	time period/step size, $h = t_{n+1} - t_n$
x_1^*	:	target temperature, $x_1^* = 0$ for simplicity

Rate of change of temperature in the room:

$$\frac{x_1(t_{n+1}) - x_1(t_n)}{h} = -x_1(t_n) + x_2(t_n)$$

Example: Controlling a Radiator

Mathematical model:

$x_1(t_n)$	:	room temperature at time t_n
$x_2(t_n)$	:	temperature of radiator at time t_n
$u(t_n)$	:	thermostat/control
		$u > 0$ increases temperature of radiator
		$u < 0$ reduces temperature of radiator
h	:	time period/step size, $h = t_{n+1} - t_n$
x_1^*	:	target temperature, $x_1^* = 0$ for simplicity

Rate of change of temperature in the room:

$$\frac{x_1(t_{n+1}) - x_1(t_n)}{h} = -x_1(t_n) + x_2(t_n)$$

Rate of change of temperature of the radiator:

$$\frac{x_2(t_{n+1}) - x_2(t_n)}{h} = u(t_n)$$

Example: Controlling a Radiator

Assumption:

It is possible to measure the room temperature x_1.

Example: Controlling a Radiator

Assumption:

It is possible to measure the room temperature x_1.

A first control law:

- If the room temperature is above the target temperature, then reduce temperature of radiator (cool down).

Example: Controlling a Radiator

Assumption:

It is possible to measure the room temperature x_1.

A first control law:

- If the room temperature is above the target temperature, then reduce temperature of radiator (cool down).
  Mathematically: If $x_1(t_n) > x_1^*$, then choose $u(t_n) < 0$.

Example: Controlling a Radiator

Assumption:

It is possible to measure the room temperature x_1.

A first control law:

- If the room temperature is above the target temperature, then reduce temperature of radiator (cool down).
  Mathematically: If $x_1(t_n) > x_1^*$, then choose $u(t_n) < 0$.
- If the room temperature is below the target temperature, then increase temperature of radiator (heat up).
  Mathematically: If $x_1(t_n) < x_1^*$, then choose $u(t_n) > 0$.

Example: Controlling a Radiator

Assumption:

It is possible to measure the room temperature x_1.

A first control law:

- If the room temperature is above the target temperature, then reduce temperature of radiator (cool down).
  Mathematically: If $x_1(t_n) > x_1^*$, then choose $u(t_n) < 0$.
- If the room temperature is below the target temperature, then increase temperature of radiator (heat up).
  Mathematically: If $x_1(t_n) < x_1^*$, then choose $u(t_n) > 0$.

Feedback control law: (proportional controller, P-controller)

$$\mu(x_1, x_2) = -c \cdot x_1, \qquad c > 0 \text{ constant}$$

Example: Controlling a Radiator

Assumption:

It is possible to measure the room temperature x_1.

A first control law:

- If the room temperature is above the target temperature, then reduce temperature of radiator (cool down).
  Mathematically: If $x_1(t_n) > x_1^*$, then choose $u(t_n) < 0$.
- If the room temperature is below the target temperature, then increase temperature of radiator (heat up).
  Mathematically: If $x_1(t_n) < x_1^*$, then choose $u(t_n) > 0$.

Feedback control law: (proportional controller, P-controller)

$$\mu(x_1, x_2) = -c \cdot x_1, \qquad c > 0 \text{ constant}$$

Closed-loop system:

$$\begin{aligned} x_1(t_{n+1}) &= x_1(t_n) + h \cdot (-x_1(t_n) + x_2(t_n)) \\ x_2(t_{n+1}) &= x_2(t_n) - h \cdot c \cdot x_1(t_n) \end{aligned}$$

Example: Controlling a Radiator

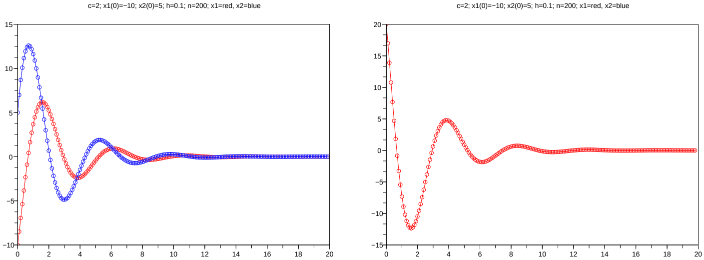
Implementation: (SCILAB, www.scilab.org)

```
function radiator1(x10,x20,h,c,n)
      x1 = zeros(1,n);
      x2 = zeros(1,n);
      u = zeros(1,n-1);
      x1(1) = x10;
      x2(1) = x20;
      for i=1:n-1,
           x1(i+1) = x1(i) + h*(-x1(i)+x2(i));
           u(i) = -c*x1(i);
           x2(i+1) = x2(i) + h*u(i);
      end;
endfunction
```

Example: Controlling a Radiator

Target temperature: $x_1^* = 0$

Control: $u(t_n) = \mu(x_1(t_n), x_2(t_n)) = -c \cdot x_1(t_n)$



red curve: room temperature
blue curve: radiator temperature

Closed-loop system:

$$
\begin{aligned}
x_1(t_{n+1}) &= x_1(t_n) + h \cdot (-x_1(t_n) + x_2(t_n)) \\
x_2(t_{n+1}) &= x_2(t_n) - h \cdot c \cdot x_1(t_n)
\end{aligned}
$$

Example: Controlling a Radiator

What about stability?

Example: Controlling a Radiator

What about stability?

Closed-loop system:

$$\underbrace{\begin{pmatrix} x_1(t_{n+1}) \\ x_2(t_{n+1}) \end{pmatrix}}_{=x(t_{n+1})} = \underbrace{\begin{pmatrix} 1-h & h \\ -h\cdot c & 1 \end{pmatrix}}_{=A} \underbrace{\begin{pmatrix} x_1(t_n) \\ x_2(t_n) \end{pmatrix}}_{=x(t_n)} \qquad (n = 0, 1, 2, \ldots)$$

Example: Controlling a Radiator

What about stability?

Closed-loop system:

$$\underbrace{\begin{pmatrix} x_1(t_{n+1}) \\ x_2(t_{n+1}) \end{pmatrix}}_{=x(t_{n+1})} = \underbrace{\begin{pmatrix} 1-h & h \\ -h \cdot c & 1 \end{pmatrix}}_{=A} \underbrace{\begin{pmatrix} x_1(t_n) \\ x_2(t_n) \end{pmatrix}}_{=x(t_n)} \qquad (n = 0, 1, 2, \ldots)$$

Thus:

$$x(t_n) = A^n x(0) \qquad (n = 0, 1, 2, \ldots)$$

Example: Controlling a Radiator

What about stability?

Closed-loop system:

$$\underbrace{\begin{pmatrix} x_1(t_{n+1}) \\ x_2(t_{n+1}) \end{pmatrix}}_{=x(t_{n+1})} = \underbrace{\begin{pmatrix} 1-h & h \\ -h\cdot c & 1 \end{pmatrix}}_{=A} \underbrace{\begin{pmatrix} x_1(t_n) \\ x_2(t_n) \end{pmatrix}}_{=x(t_n)} \qquad (n = 0,1,2,\ldots)$$

Thus:

$$x(t_n) = A^n x(0) \qquad (n = 0,1,2,\ldots)$$

Eigenvalues ($h > 0$):

$$\boldsymbol{\lambda} = 1 - h\left(1/2 \mp \sqrt{1/4 - c}\right)$$

Example: Controlling a Radiator

What about stability?

Closed-loop system:

$$\underbrace{\begin{pmatrix} x_1(t_{n+1}) \\ x_2(t_{n+1}) \end{pmatrix}}_{=x(t_{n+1})} = \underbrace{\begin{pmatrix} 1-h & h \\ -h \cdot c & 1 \end{pmatrix}}_{=A} \underbrace{\begin{pmatrix} x_1(t_n) \\ x_2(t_n) \end{pmatrix}}_{=x(t_n)} \qquad (n = 0, 1, 2, \ldots)$$

Thus:

$$x(t_n) = A^n x(0) \qquad (n = 0, 1, 2, \ldots)$$

Eigenvalues ($h > 0$):

$$\boldsymbol{\lambda} = 1 - h\left(1/2 \mp \sqrt{1/4 - c}\right)$$

Stability at $x^* = 0$:

- asymptotically stable, if $|\boldsymbol{\lambda}| < 1$ for all eigenvalues of A, i.e. if $h \cdot c < 1$ and $c > 1/4$ or if $0 < c \leq 1/4$

Example: Controlling a Radiator

What about stability?

Closed-loop system:

$$\underbrace{\begin{pmatrix} x_1(t_{n+1}) \\ x_2(t_{n+1}) \end{pmatrix}}_{=x(t_{n+1})} = \underbrace{\begin{pmatrix} 1-h & h \\ -h\cdot c & 1 \end{pmatrix}}_{=A} \underbrace{\begin{pmatrix} x_1(t_n) \\ x_2(t_n) \end{pmatrix}}_{=x(t_n)} \qquad (n = 0, 1, 2, \ldots)$$

Thus:

$$x(t_n) = A^n x(0) \qquad (n = 0, 1, 2, \ldots)$$

Eigenvalues ($h > 0$):

$$\boldsymbol{\lambda} = 1 - h\left(1/2 \mp \sqrt{1/4 - c}\right)$$

Stability at $x^* = 0$:

- asymptotically stable, if $|\boldsymbol{\lambda}| < 1$ for all eigenvalues of A, i.e. if $h \cdot c < 1$ and $c > 1/4$ or if $0 < c \leq 1/4$
- unstable, if $|\boldsymbol{\lambda}| > 1$ for some eigenvalue of A, i.e. if $h \cdot c > 1$ and $c > 1/4$ or if $c < 0$

LEGO Mindstorms

Control task:

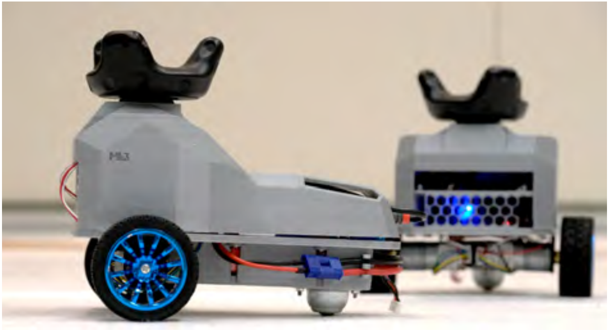
Follow a line!

Idea: proportional controller

$$\mu(x) = C(x - s)$$

s	:	target value (e.g. color of midline)
u	:	control (e.g. motor velocity left and right)
x	:	actual value (e.g. actual color below light sensor)
C	:	constant

GNEP-MPC for Coordination of Interacting Vehicles



Path Planning and Control

- robotics



Path Planning and Control

- robotics



- vehicles, aircrafts, satellites, ...



Path Planning and Control

- robotics



- vehicles, aircrafts, satellites, ...



- driver assistance and automatic driving



Path Planning and Control

- robotics



- vehicles, aircrafts, satellites, ...



- driver assistance and automatic driving



- ...

Path Planning and Control

- robotics



- vehicles, aircrafts, satellites, ...



- driver assistance and automatic driving



- ...

Issues:

- nonlinear dynamics
- uncertainties
- constraints
- optimality (time, energy, comfort,...)
- online control

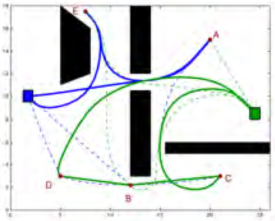
Approach:

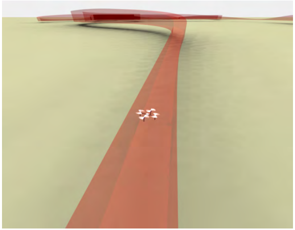
model-predictive control (MPC)

Path Planning and Control

Optimization-based approaches:

- shortest paths (Dijkstra, A^*)
  ⇝ strategic/macroscopic planning
- dynamic programming (Bellman principle, HJB theory) ⇝ feedback control
- optimal control
  ⇝ open loop, reference trajectories
- model-predictive control
  ⇝ feedback control
- ...





Other approaches:

- rapidly exploring random trees
  [Lavalle, S.M.: Rapidly-exploring random trees: A new tool for path planning. In: Computer Science Dept, Iowa State University, Tech. Rep. TR. 1998, S. 98–11.]
- random walks
- ...

Overview on control approaches (not complete)

Controller	Process/ Model	Online Optimization	Constraints obeyed	Complexity in use
PID-Regler	nonlinear	no	no	very low
Ricatti / LQR	linear	no	no	low
flatness	nonlinear	no	no	very low
MPC (linear)	linear	yes	yes	medium
MPC (nonlinear)	nonlinear	yes	yes	high
HJB/dyn. Progr.	nonlinear	yes	yes	medium[1)]
OCP (online)	nonlinear	yes	yes	high
Sensitivity-Update	nonlinear	no	partly	low

MPC is very flexible and individually adaptable, since performance criterion and constraints can be adapted during control.

[1)] if backward phase is computed offline, otherwise very high

Standard Tracking Problem

Given:

reference trajectory $(x_{ref}(n), u_{ref}(n))$, $n = 0, 1, 2, \ldots$, in discrete time $n \mathrel{\hat{=}} t_n$

Tracking Problem

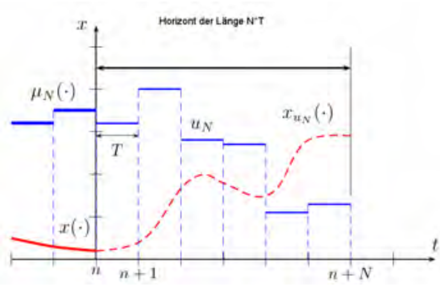
Construct feedback control law $\boldsymbol{\mu} : \mathbb{N} \times X \longrightarrow U$ for the constrained control system in discrete time

$$\begin{aligned} x(n+1) &= f(x(n), u(n)) && (n = 0, 1, 2, \ldots) \\ x(n) &\in X && (n = 0, 1, 2, \ldots) \\ u(n) &\in U && (n = 0, 1, 2, \ldots) \\ x(0) &= x_0 \end{aligned}$$

in order to track the reference trajectory.

($x_0 \in \mathbb{R}^n$ given vector, $X \subset \mathbb{R}^n$, $U \subset \mathbb{R}^m$ given sets)

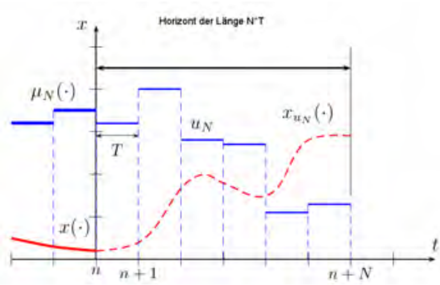
NMPC/Moving Horizon Control/Receding Horizon Control



Scheme:

1. Solve (discretized) optimal control problem on a finite time horizon

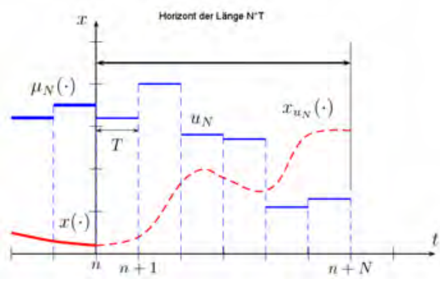
NMPC/Moving Horizon Control/Receding Horizon Control

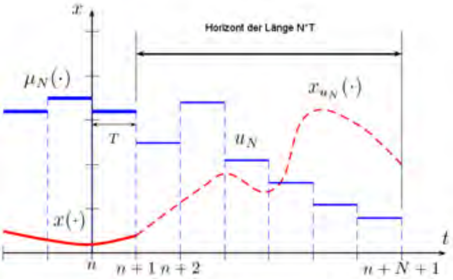


Scheme:

1. Solve (discretized) optimal control problem on a finite time horizon
2. Apply first control

NMPC/Moving Horizon Control/Receding Horizon Control





Scheme:

1. Solve (discretized) optimal control problem on a finite time horizon
2. Apply first control
3. Shift horizon and iterate

Standard Nonlinear Model-Predictive Control (NMPC)

Parameter of standard NMPC:

- preview horizon N

Standard NMPC (moving horizon, receding horizon)

(0) Measure (or predict or estimate) state $x(n)$ at time n.

Standard Nonlinear Model-Predictive Control (NMPC)

Parameter of standard NMPC:

- preview horizon N

Standard NMPC (moving horizon, receding horizon)

(0) Measure (or predict or estimate) state $x(n)$ at time n.

(1) Solve optimal control problem in discrete time on time horizon $[n, n+N]$ with initial value $x(n)$. Let $u^*(n), \dots, u^*(n+N-1)$ be the optimal solution.

Standard Nonlinear Model-Predictive Control (NMPC)

Parameter of standard NMPC:

- preview horizon N

Standard NMPC (moving horizon, receding horizon)

(0) Measure (or predict or estimate) state $x(n)$ at time n.

(1) Solve optimal control problem in discrete time on time horizon $[n, n+N]$ with initial value $x(n)$. Let $u^*(n), \dots, u^*(n+N-1)$ be the optimal solution.

(2) Define the feedback control $\mu_N(n, x(n)) := u^*(n)$ and apply it:

$$x(n+1) = f(x(n), \mu_N(n, x(n)))$$

Standard Nonlinear Model-Predictive Control (NMPC)

Parameter of standard NMPC:

- preview horizon N

Standard NMPC (moving horizon, receding horizon)

(0) Measure (or predict or estimate) state $x(n)$ at time n.

(1) Solve optimal control problem in discrete time on time horizon $[n, n+N]$ with initial value $x(n)$. Let $u^*(n), \ldots, u^*(n+N-1)$ be the optimal solution.

(2) Define the feedback control $\mu_N(n, x(n)) := u^*(n)$ and apply it:

$$x(n+1) = f(x(n), \mu_N(n, x(n)))$$

(3) Set $n \leftarrow n+1$ and go to (0).

Standard Nonlinear Model-Predictive Control (NMPC)

Parameter of standard NMPC:

- preview horizon N

Standard NMPC (moving horizon, receding horizon)

(0) Measure (or predict or estimate) state $x(n)$ at time n.

(1) Solve optimal control problem in discrete time on time horizon $[n, n+N]$ with initial value $x(n)$. Let $u^*(n), \ldots, u^*(n+N-1)$ be the optimal solution.

(2) Define the feedback control $\mu_N(n, x(n)) := u^*(n)$ and apply it:

$$x(n+1) = f(x(n), \mu_N(n, x(n)))$$

(3) Set $n \leftarrow n+1$ and go to (0).

$\rightsquigarrow$ $\mu_N(n, x)$ defines feedback law!

Optimal Control Problem in Discrete Time of Tracking Type

In each step of NMPC solve optimal control problem in discrete time ...

DOCP(n, x_n, N) of Tracking Type

Minimize weighted tracking error

$$\frac{1}{2}\sum_{k=n}^{n+N-1} \|x(k) - x_{ref}(k)\|_{V(k)}^2 + \|u(k) - u_{ref}(k)\|_{W(k)}^2$$

subject to the constraints

$$\begin{aligned} x(k+1) &= f(x(k), u(k)) && (k = n, \ldots, n+N-1) \\ x(k) &\in X && (k = n, \ldots, n+N) \\ u(k) &\in U && (k = n, \ldots, n+N-1) \\ x(n) &= x_n \end{aligned}$$

n: current time, x_n: current state estimation, N: preview horizon, (x_{ref}, u_{ref}): reference trajectory, $\|z\|_M := \left(z^\top M z\right)^{1/2}$: weighted norm

Economic MPC

Economic MPC uses a general objective function (not necessarily of tracking type):

DOCP(n, x_n, N) in Economic MPC

Minimize

$$\varphi(x(n+N)) + \sum_{k=n}^{n+N-1} \ell(x(k), u(k))$$

subject to the constraints

$$\begin{aligned} x(k+1) &= f(x(k), u(k)) && (k = n, \ldots, n+N-1) \\ x(k) &\in X && (k = n, \ldots, n+N) \\ u(k) &\in U && (k = n, \ldots, n+N-1) \\ x(n) &= x_n \end{aligned}$$

n: current time, x_n: current state estimation, φ: terminal costs, ℓ: running costs

Challenges and Modifications of MPC

Problems:

- solving DOCP is expensive and requires time $\rightsquigarrow$ time delay in applying the control
- solving DOCP might fail (numerical issues, infeasibility, ...)

Challenges and Modifications of MPC

Problems:

- solving DOCP is expensive and requires time $\rightsquigarrow$ time delay in applying the control
- solving DOCP might fail (numerical issues, infeasibility, ...)

Approaches and modifications:

- use model to predict future state and start optimization early

Challenges and Modifications of MPC

Problems:

- solving DOCP is expensive and requires time $\rightsquigarrow$ time delay in applying the control
- solving DOCP might fail (numerical issues, infeasibility, ...)

Approaches and modifications:

- use model to predict future state and start optimization early
- multistep NMPC: apply $M \geq 1$ control values $u^*(k), \dots, u^*(k + M - 1)$ without new measurements $\rightsquigarrow$ system is in open-loop for M steps

Challenges and Modifications of MPC

Problems:

- solving DOCP is expensive and requires time $\rightsquigarrow$ time delay in applying the control
- solving DOCP might fail (numerical issues, infeasibility, ...)

Approaches and modifications:

- use model to predict future state and start optimization early
- multistep NMPC: apply $M \geq 1$ control values $u^*(k), \dots, u^*(k+M-1)$ without new measurements $\rightsquigarrow$ system is in open-loop for M steps
- multistep NMPC with re-optimization: re-optimize on remaining intervals $[k+j, k+N]$ with new measurements at $k+1, \dots, k+M-1$.

Challenges and Modifications of MPC

Problems:

- solving DOCP is expensive and requires time $\rightsquigarrow$ time delay in applying the control
- solving DOCP might fail (numerical issues, infeasibility, ...)

Approaches and modifications:

- use model to predict future state and start optimization early
- multistep NMPC: apply $M \geq 1$ control values $u^*(k), \ldots, u^*(k+M-1)$ without new measurements $\rightsquigarrow$ system is in open-loop for M steps
- multistep NMPC with re-optimization: re-optimize on remaining intervals $[k+j, k+N]$ with new measurements at $k+1, \ldots, k+M-1$.
- multistep NMPC with sensitivity updates: Perform a sensitivity update instead of a re-optimization.

Challenges and Modifications of MPC

Problems:

- solving DOCP is expensive and requires time $\rightsquigarrow$ time delay in applying the control
- solving DOCP might fail (numerical issues, infeasibility, ...)

Approaches and modifications:

- use model to predict future state and start optimization early
- multistep NMPC: apply $M \geq 1$ control values $u^*(k), \ldots, u^*(k+M-1)$ without new measurements $\rightsquigarrow$ system is in open-loop for M steps
- multistep NMPC with re-optimization: re-optimize on remaining intervals $[k+j, k+N]$ with new measurements at $k+1, \ldots, k+M-1$.
- multistep NMPC with sensitivity updates: Perform a sensitivity update instead of a re-optimization.
- NMPC with realtime iterations (RTI) and initial value embedding: solve DOCP only approximately and update solutions

  [Diehl, M.: Real-time optimization for large scale nonlinear processes. PhD thesis, University of Heidelberg (2001)]

  [Diehl, M., Findeisen, R., Allgöwer, F., Bock, H.G., Schlöder, J.P.: Nominal stability of the real-time iteration scheme for nonlinear model predictive control. IEE Proc. Control Theory Appl. 152, 296–308 (2005)]

Control and Planning Tasks

Path Tracking Task

Follow a given (optimal) reference trajectory

$$(x_{ref}(n), u_{ref}(n))$$

in discrete time $n \mathrel{\hat{=}} t_n$!

vs

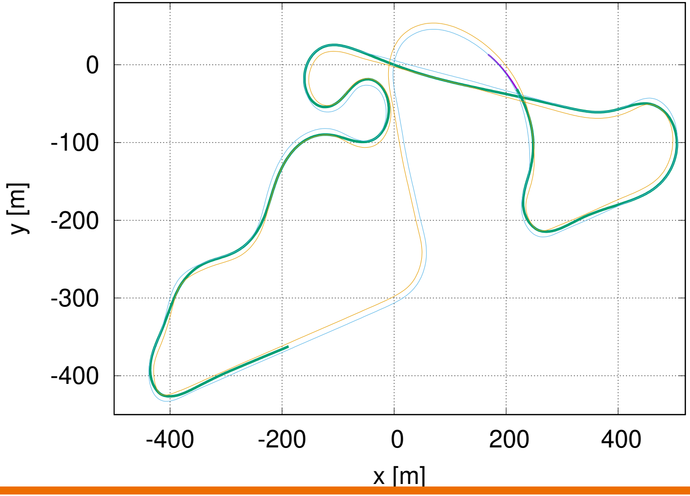
Path Planning Task

Compute a (locally) optimal trajectory

$$(x_{ref}(n), u_{ref}(n))$$

in discrete time $n \mathrel{\hat{=}} t_n$!

Example: Drive along a Track (Course 3 UniBw M)



Theoretical and Practical Issues regarding NMPC

Questions:

Theoretical and Practical Issues regarding NMPC

Questions:

- Under which conditions is the NMPC feedback law μ_N (asymptotically) stable?

Theoretical and Practical Issues regarding NMPC

Questions:

- Under which conditions is the NMPC feedback law μ_N (asymptotically) stable?
- Are there optimality estimates for the feedback law μ_N as $N \to \infty$?
  How good is the feedback law with regard to the minimization of costs?

Theoretical and Practical Issues regarding NMPC

Questions:

- Under which conditions is the NMPC feedback law μ_N (asymptotically) stable?
- Are there optimality estimates for the feedback law μ_N as $N \to \infty$?
  How good is the feedback law with regard to the minimization of costs?
- Under which conditions is the NMPC feedback law μ_N robust w.r.t. perturbations?

Theoretical and Practical Issues regarding NMPC

Questions:

- Under which conditions is the NMPC feedback law μ_N (asymptotically) stable?
- Are there optimality estimates for the feedback law μ_N as $N \to \infty$?
  How good is the feedback law with regard to the minimization of costs?
- Under which conditions is the NMPC feedback law μ_N robust w.r.t. perturbations?
- viability with regard to constraints $x(t) \in X$, $u(t) \in U$

Theoretical and Practical Issues regarding NMPC

Questions:

- Under which conditions is the NMPC feedback law μ_N (asymptotically) stable?
- Are there optimality estimates for the feedback law μ_N as $N \to \infty$?
  How good is the feedback law with regard to the minimization of costs?
- Under which conditions is the NMPC feedback law μ_N robust w.r.t. perturbations?
- viability with regard to constraints $x(t) \in X$, $u(t) \in U$
- realtime capability, i.e. evaluation of control law must not take too much time

Theoretical and Practical Issues regarding NMPC

Questions:

- Under which conditions is the NMPC feedback law μ_N (asymptotically) stable?
- Are there optimality estimates for the feedback law μ_N as $N \to \infty$?
  How good is the feedback law with regard to the minimization of costs?
- Under which conditions is the NMPC feedback law μ_N robust w.r.t. perturbations?
- viability with regard to constraints $x(t) \in X$, $u(t) \in U$
- realtime capability, i.e. evaluation of control law must not take too much time
- implementation details

Theoretical and Practical Issues regarding NMPC

Questions:

- Under which conditions is the NMPC feedback law μ_N (asymptotically) stable?
- Are there optimality estimates for the feedback law μ_N as $N \to \infty$?
  How good is the feedback law with regard to the minimization of costs?
- Under which conditions is the NMPC feedback law μ_N robust w.r.t. perturbations?
- viability with regard to constraints $x(t) \in X$, $u(t) \in U$
- realtime capability, i.e. evaluation of control law must not take too much time
- implementation details

We focus on stability, optimality, and realization of NMPC in this course.

Contents

Numerical Solution of DOCP

The most costly part in NMPC is the solution of DOCP.

DOCP in Economic MPC

Minimize

$$\varphi(x(N)) + \sum_{k=0}^{N-1} \ell(x(k), u(k))$$

subject to the constraints

$$\begin{aligned} x(k+1) &= f(x(k), u(k)) && (k = 0, \dots, N-1) \\ x(k) &\in X && (k = 0, \dots, N) \\ u(k) &\in U && (k = 0, \dots, N-1) \\ x(0) &= x_0 \end{aligned}$$

In the sequel (for simplicity):

$$\begin{aligned} X &:= \{x \in \mathbb{R}^n \mid g(x) \leq 0\} \\ U &:= \{u \in \mathbb{R}^m \mid u_{min} \leq u \leq u_{max}\} \end{aligned}$$

Numerical Solution of DOCP

$$z := (x(0), u(0), \ldots, x(N-1), u(N-1), x(N))^\top$$

Numerical Solution of DOCP

$$z := (x(0), u(0), \ldots, x(N-1), u(N-1), x(N))^\top$$

$$J(z) := \varphi(x(N)) + \sum_{k=0}^{N-1} \ell(x(k), u(k))$$

Numerical Solution of DOCP

$$
\begin{aligned}
z &:= (x(0), u(0), \dots, x(N-1), u(N-1), x(N))^\top \\
J(z) &:= \varphi(x(N)) + \sum_{k=0}^{N-1} \ell(x(k), u(k)) \\
H(z) &:= \begin{pmatrix} x(0) - x_0 \\ f(x(0), u(0)) - x(1) \\ \vdots \\ f(x(N-1), u(N-1)) - x(N) \end{pmatrix},
\end{aligned}
$$

Numerical Solution of DOCP

$$z := (x(0), u(0), \ldots, x(N-1), u(N-1), x(N))^\top$$

$$J(z) := \varphi(x(N)) + \sum_{k=0}^{N-1} \ell(x(k), u(k))$$

$$H(z) := \begin{pmatrix} x(0) - x_0 \\ f(x(0), u(0)) - x(1) \\ \vdots \\ f(x(N-1), u(N-1)) - x(N) \end{pmatrix}, \qquad G(z) := \begin{pmatrix} g(x(0)) \\ \vdots \\ g(x(N)) \\ \hline u(0) - u_{max} \\ \vdots \\ u(N-1) - u_{max} \\ \hline u_{min} - u(0) \\ \vdots \\ u_{min} - u(N-1) \end{pmatrix}$$

Numerical Solution of DOCP

DOCP is of the following type, but with a certain structure.

Nonlinear Optimization Problem (NLO)

$$\text{Minimize} \quad J(z) \quad \text{s.t.} \quad G(z) \leq 0, \quad H(z) = 0$$

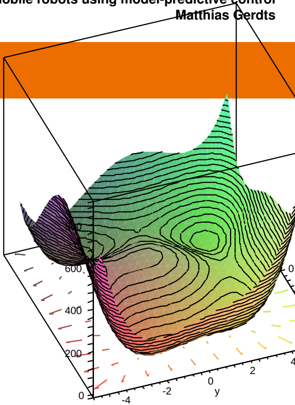
$$z := (x(0), u(0), \ldots, x(N-1), u(N-1), x(N))^\top$$

$$J(z) := \varphi(x(N)) + \sum_{k=0}^{N-1} \ell(x(k), u(k))$$

$$H(z) := \begin{pmatrix} x(0) - x_0 \\ f(x(0), u(0)) - x(1) \\ \vdots \\ f(x(N-1), u(N-1)) - x(N) \end{pmatrix}, \quad G(z) := \begin{pmatrix} g(x(0)) \\ \vdots \\ g(x(N)) \\ \hline u(0) - u_{max} \\ \vdots \\ u(N-1) - u_{max} \\ \hline u_{min} - u(0) \\ \vdots \\ u_{min} - u(N-1) \end{pmatrix}$$

Contents

Optimization and Necessary Conditions



Let

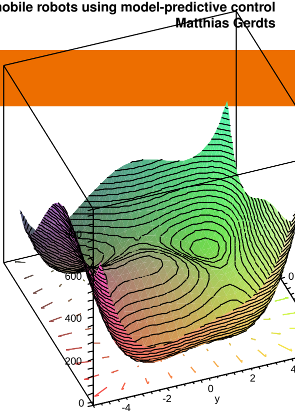
$$
\begin{aligned}
J &: \mathbb{R}^{n_z} \longrightarrow \mathbb{R} \\
H = (H_1, \dots, H_{n_H})^\top &: \mathbb{R}^{n_z} \longrightarrow \mathbb{R}^{n_H} \\
G = (G_1, \dots, G_{n_G})^\top &: \mathbb{R}^{n_z} \longrightarrow \mathbb{R}^{n_G}
\end{aligned}
$$

Nonlinear Optimization Problem (NLO)

$$
\text{Minimize} \quad J(z) \quad \text{s.t.} \quad H(z) = 0, \; G(z) \leq 0
$$

Defintions:

Optimization and Necessary Conditions



Let

$$
\begin{aligned}
J &: \mathbb{R}^{n_z} \longrightarrow \mathbb{R} \\
H = (H_1, \dots, H_{n_H})^\top &: \mathbb{R}^{n_z} \longrightarrow \mathbb{R}^{n_H} \\
G = (G_1, \dots, G_{n_G})^\top &: \mathbb{R}^{n_z} \longrightarrow \mathbb{R}^{n_G}
\end{aligned}
$$

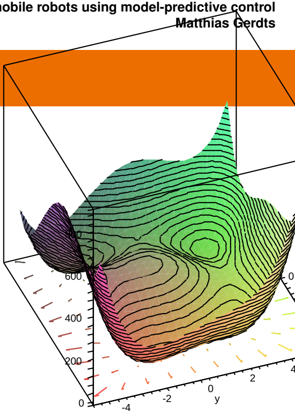
Nonlinear Optimization Problem (NLO)

$$\text{Minimize} \quad J(z) \quad \text{s.t.} \quad H(z) = 0, \; G(z) \leq 0$$

Defintions:

- Feasible set: $\Sigma := \{z \in \mathbb{R}^{n_z} \mid H(z) = 0, \; G(z) \leq 0\}$

Optimization and Necessary Conditions



Let

$$
\begin{aligned}
J &: \mathbb{R}^{n_z} \longrightarrow \mathbb{R} \\
H = (H_1, \ldots, H_{n_H})^\top &: \mathbb{R}^{n_z} \longrightarrow \mathbb{R}^{n_H} \\
G = (G_1, \ldots, G_{n_G})^\top &: \mathbb{R}^{n_z} \longrightarrow \mathbb{R}^{n_G}
\end{aligned}
$$

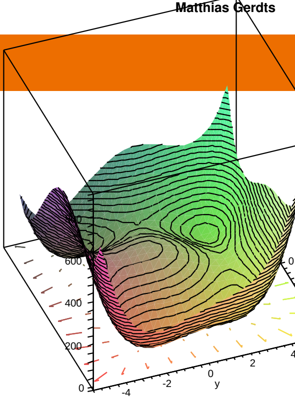
Nonlinear Optimization Problem (NLO)

$$
\text{Minimize} \quad J(z) \quad \text{s.t.} \quad H(z) = 0, \; G(z) \leq 0
$$

Defintions:

- Feasible set: $\Sigma := \{z \in \mathbb{R}^{n_z} \mid H(z) = 0, \; G(z) \leq 0\}$
- Index set of active inequalities: $A(z) := \{i \mid G_i(z) = 0, \; 1 \leq i \leq n_G\}$

Optimization and Necessary Conditions



Let

$$J : \mathbb{R}^{n_z} \longrightarrow \mathbb{R}$$
$$H = (H_1, \ldots, H_{n_H})^\top : \mathbb{R}^{n_z} \longrightarrow \mathbb{R}^{n_H}$$
$$G = (G_1, \ldots, G_{n_G})^\top : \mathbb{R}^{n_z} \longrightarrow \mathbb{R}^{n_G}$$

Nonlinear Optimization Problem (NLO)

$$\text{Minimize} \quad J(z) \quad \text{s.t.} \quad H(z) = 0, \; G(z) \leq 0$$

Defintions:

- Feasible set: $\Sigma := \{z \in \mathbb{R}^{n_z} \mid H(z) = 0, \; G(z) \leq 0\}$
- Index set of active inequalities: $A(z) := \{i \mid G_i(z) = 0, \; 1 \leq i \leq n_G\}$
- $\hat{z} \in \Sigma$ is a local minimum of NLO, iff there exists a ball $B_\epsilon(\hat{z})$ with radius $\epsilon > 0$ around $\hat{z}$ such that

$$J(\hat{z}) \leq J(z) \qquad \forall z \in \Sigma \cap B_\epsilon(\hat{z})$$

Optimization and Necessary Conditions

The following necessary conditions are essential for the further analysis.

Karush-Kuhn-Tucker Conditions (KKT) for NLO

Assumptions:

Optimization and Necessary Conditions

The following necessary conditions are essential for the further analysis.

Karush-Kuhn-Tucker Conditions (KKT) for NLO

Assumptions:

- J, G, H are continuously differentiable.

Optimization and Necessary Conditions

The following necessary conditions are essential for the further analysis.

Karush-Kuhn-Tucker Conditions (KKT) for NLO

Assumptions:

- J, G, H are continuously differentiable.
- $\hat{z}$ is a local minimum of (NLO).

Optimization and Necessary Conditions

The following necessary conditions are essential for the further analysis.

Karush-Kuhn-Tucker Conditions (KKT) for NLO

Assumptions:

- J, G, H are continuously differentiable.
- $\hat{z}$ is a local minimum of (NLO).
- Linear Independence Constraint Qualification (LICQ): The gradients $\nabla H_j(\hat{z})$, $j = 1, \ldots, n_H$, and $\nabla G_j(\hat{z})$, $j \in A(\hat{z})$, are linearly independent.

Optimization and Necessary Conditions

The following necessary conditions are essential for the further analysis.

Karush-Kuhn-Tucker Conditions (KKT) for NLO

Assumptions:

- J, G, H are continuously differentiable.
- $\hat{z}$ is a local minimum of (NLO).
- Linear Independence Constraint Qualification (LICQ): The gradients $\nabla H_j(\hat{z})$, $j = 1, \ldots, n_H$, and $\nabla G_j(\hat{z})$, $j \in A(\hat{z})$, are linearly independent.

Then there exist unique Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^{n_H}$ and $\boldsymbol{\mu} \in \mathbb{R}^{n_G}$ with

Optimization and Necessary Conditions

The following necessary conditions are essential for the further analysis.

Karush-Kuhn-Tucker Conditions (KKT) for NLO

Assumptions:

- J, G, H are continuously differentiable.
- $\hat{z}$ is a local minimum of (NLO).
- Linear Independence Constraint Qualification (LICQ): The gradients $\nabla H_j(\hat{z})$, $j = 1, \ldots, n_H$, and $\nabla G_j(\hat{z})$, $j \in A(\hat{z})$, are linearly independent.

Then there exist unique Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^{n_H}$ and $\boldsymbol{\mu} \in \mathbb{R}^{n_G}$ with

$$0 = \nabla_z L(\hat{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \qquad \text{(stationarity of Lagrange function)}$$

Optimization and Necessary Conditions

The following necessary conditions are essential for the further analysis.

Karush-Kuhn-Tucker Conditions (KKT) for NLO

Assumptions:

- J, G, H are continuously differentiable.
- $\hat{z}$ is a local minimum of (NLO).
- Linear Independence Constraint Qualification (LICQ): The gradients $\nabla H_j(\hat{z})$, $j = 1, \ldots, n_H$, and $\nabla G_j(\hat{z})$, $j \in A(\hat{z})$, are linearly independent.

Then there exist unique Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^{n_H}$ and $\boldsymbol{\mu} \in \mathbb{R}^{n_G}$ with

$$0 = \nabla_z L(\hat{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \qquad \text{(stationarity of Lagrange function)}$$

$$0 \leq \boldsymbol{\mu}, \; \boldsymbol{\mu}^\top G(\hat{z}) = 0 \qquad \text{(complementarity condition)}$$

Optimization and Necessary Conditions

The following necessary conditions are essential for the further analysis.

Karush-Kuhn-Tucker Conditions (KKT) for NLO

Assumptions:

- J, G, H are continuously differentiable.
- $\hat{z}$ is a local minimum of (NLO).
- Linear Independence Constraint Qualification (LICQ): The gradients $\nabla H_j(\hat{z})$, $j = 1, \ldots, n_H$, and $\nabla G_j(\hat{z})$, $j \in A(\hat{z})$, are linearly independent.

Then there exist unique Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^{n_H}$ and $\boldsymbol{\mu} \in \mathbb{R}^{n_G}$ with

$$0 = \nabla_z L(\hat{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \qquad \text{(stationarity of Lagrange function)}$$

$$0 \leq \boldsymbol{\mu}, \;\; \boldsymbol{\mu}^\top G(\hat{z}) = 0 \qquad \text{(complementarity condition)}$$

Lagrange function:

$$L(z, \boldsymbol{\lambda}, \boldsymbol{\mu}) := J(z) + \boldsymbol{\lambda}^\top H(z) + \boldsymbol{\mu}^\top G(z)$$

Optimization and Necessary Conditions

Special cases:

- unconstrained problems: (H and G are not present)

$$0 = \nabla J(\hat{z}) \qquad \text{(stationarity of } J\text{)}$$

Optimization and Necessary Conditions

Special cases:

- unconstrained problems: (H and G are not present)

$$0 = \nabla J(\hat{z}) \qquad \text{(stationarity of } J\text{)}$$

- Equality constrained problems: (G is not present)

$$\begin{aligned} 0 &= \nabla_z L(\hat{z}, \lambda) && \text{(stationarity of } L\text{)} \\ 0 &= H(\hat{z}) && \text{(feasibility)} \end{aligned}$$

Optimization and Necessary Conditions

Special cases:

- unconstrained problems: (H and G are not present)

$$0 = \nabla J(\hat{z}) \qquad \text{(stationarity of } J\text{)}$$

- Equality constrained problems: (G is not present)

$$\begin{aligned} 0 &= \nabla_z L(\hat{z}, \boldsymbol{\lambda}) && \text{(stationarity of } L\text{)} \\ 0 &= H(\hat{z}) && \text{(feasibility)} \end{aligned}$$

$\rightsquigarrow$ nonlinear equations for $\hat{z}$ and $\boldsymbol{\lambda}$, resp., in both cases!

Optimization and Necessary Conditions

Special cases:

- unconstrained problems: (H and G are not present)

$$0 = \nabla J(\hat{z}) \qquad \text{(stationarity of } J\text{)}$$

- Equality constrained problems: (G is not present)

$$\begin{aligned} 0 &= \nabla_z L(\hat{z}, \boldsymbol{\lambda}) && \text{(stationarity of } L\text{)} \\ 0 &= H(\hat{z}) && \text{(feasibility)} \end{aligned}$$

$\rightsquigarrow$ nonlinear equations for $\hat{z}$ and $\boldsymbol{\lambda}$, resp., in both cases!

$\rightsquigarrow$ Apply Newton's method to find a stationary point.

Optimization and Necessary Conditions

Special case:

- Equality constrained linear-quadratic problem:

$$J(z) = \frac{1}{2} z^\top Q z + c^\top z \qquad (Q \in \mathbb{R}^{n_z \times n_z},\ c \in \mathbb{R}^{n_z})$$
$$H(z) = Az - b \qquad (A \in \mathbb{R}^{n_H \times n_z},\ b \in \mathbb{R}^{n_H})$$

Optimization and Necessary Conditions

Special case:

- Equality constrained linear-quadratic problem:

$$J(z) = \frac{1}{2} z^\top Q z + c^\top z \qquad (Q \in \mathbb{R}^{n_z \times n_z}, c \in \mathbb{R}^{n_z})$$

$$H(z) = Az - b \qquad (A \in \mathbb{R}^{n_H \times n_z}, b \in \mathbb{R}^{n_H})$$

KKT conditions:

$$0 = Q\hat{z} + c + A^\top \boldsymbol{\lambda} \qquad \text{(stationarity of } L\text{)}$$

$$0 = Az - b \qquad \text{(feasibility)}$$

Optimization and Necessary Conditions

Special case:

- Equality constrained linear-quadratic problem:

$$J(z) = \frac{1}{2} z^\top Q z + c^\top z \qquad (Q \in \mathbb{R}^{n_z \times n_z}, c \in \mathbb{R}^{n_z})$$

$$H(z) = Az - b \qquad (A \in \mathbb{R}^{n_H \times n_z}, b \in \mathbb{R}^{n_H})$$

KKT conditions:

$$0 = Q\hat{z} + c + A^\top \boldsymbol{\lambda} \qquad \text{(stationarity of } L\text{)}$$

$$0 = Az - b \qquad \text{(feasibility)}$$

That's just a system of linear equations:

$$\begin{pmatrix} Q & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} \hat{z} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} -c \\ b \end{pmatrix}$$

Optimization and Necessary Conditions

Special case:

- Linear-quadratic problem with equality and inequality constraints:

$$
\begin{aligned}
J(z) &= \frac{1}{2} z^\top Q z + c^\top z & \quad & (Q \in \mathbb{R}^{n_z \times n_z}, c \in \mathbb{R}^{n_z}) \\
H(z) &= Az - b & & (A \in \mathbb{R}^{n_H \times n_z}, b \in \mathbb{R}^{n_H}) \\
G(z) &= Bz - d & & (B \in \mathbb{R}^{n_G \times n_z}, d \in \mathbb{R}^{n_G})
\end{aligned}
$$

Optimization and Necessary Conditions

Special case:

- Linear-quadratic problem with equality and inequality constraints:

$$
\begin{aligned}
J(z) &= \frac{1}{2} z^\top Q z + c^\top z && (Q \in \mathbb{R}^{n_z \times n_z}, c \in \mathbb{R}^{n_z}) \\
H(z) &= Az - b && (A \in \mathbb{R}^{n_H \times n_z}, b \in \mathbb{R}^{n_H}) \\
G(z) &= Bz - d && (B \in \mathbb{R}^{n_G \times n_z}, d \in \mathbb{R}^{n_G})
\end{aligned}
$$

KKT conditions:

$$
\begin{aligned}
0 &= Q\hat{z} + c + A^\top \boldsymbol{\lambda} + B^\top \boldsymbol{\mu} && \text{(stationarity of } L) \\
0 &= Az - b && \text{(feasibility, equality constraints)} \\
0 &\geq Bz - d && \text{(feasibility, inequality constraints)} \\
0 &\leq \boldsymbol{\mu}, \quad \boldsymbol{\mu}^\top (Bz - d) = 0 && \text{(complementarity)}
\end{aligned}
$$

Optimization and Necessary Conditions

Special case:

- Linear-quadratic problem with equality and inequality constraints:

$$J(z) = \frac{1}{2} z^\top Q z + c^\top z \qquad (Q \in \mathbb{R}^{n_z \times n_z}, c \in \mathbb{R}^{n_z})$$
$$H(z) = Az - b \qquad (A \in \mathbb{R}^{n_H \times n_z}, b \in \mathbb{R}^{n_H})$$
$$G(z) = Bz - d \qquad (B \in \mathbb{R}^{n_G \times n_z}, d \in \mathbb{R}^{n_G})$$

KKT conditions:

$$0 = Q\hat{z} + c + A^\top \boldsymbol{\lambda} + B^\top \boldsymbol{\mu} \qquad \text{(stationarity of } L\text{)}$$
$$0 = Az - b \qquad \text{(feasibility, equality constraints)}$$
$$0 \geq Bz - d \qquad \text{(feasibility, inequality constraints)}$$
$$0 \leq \boldsymbol{\mu}, \quad \boldsymbol{\mu}^\top (Bz - d) = 0 \qquad \text{(complementarity)}$$

That's not just a system of linear equations anymore!

Contents

Linear MPC w/o Control and State Constraints

For simplicity let $x_{ref} = 0$ and $u_{ref} = 0$.

LMPC-OCP in discrete time

Minimize

$$\frac{1}{2}\sum_{k=0}^{N-1}\left(x(k)^\top V(k)x(k) + u(k)^\top W(k)u(k)\right)$$

subject to the constraints

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) & \quad (k = 0,\ldots,N-1) \\ x(0) &= x_0 & \quad (x_0 \text{ given}) \end{aligned}$$

Linear MPC w/o Control and State Constraints

For simplicity let $x_{ref} = 0$ and $u_{ref} = 0$.

LMPC-OCP in discrete time

Minimize

$$\frac{1}{2}\sum_{k=0}^{N-1}\left(x(k)^\top V(k)x(k) + u(k)^\top W(k)u(k)\right)$$

subject to the constraints

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) & \qquad (k = 0,\ldots,N-1) \\ x(0) &= x_0 & \qquad (x_0 \text{ given}) \end{aligned}$$

Assumptions:

(A1) $W(k)$ symmetric and positive definite for all k

(A2) $V(k)$ symmetric and positive semi-definite for all k

Linear MPC w/o Control and State Constraints

Lagrange function:

$$
\begin{aligned}
L(x,u,\boldsymbol{\lambda},\boldsymbol{\sigma}) \quad := \quad & \frac{1}{2}\sum_{k=0}^{N-1}\Big(x(k)^\top V(k)x(k) + u(k)^\top W(k)u(k)\Big) + \boldsymbol{\sigma}^\top(x(0)-x_0) \\
& + \sum_{k=0}^{N-1}\boldsymbol{\lambda}(k+1)^\top(A(k)x(k)+B(k)u(k)-x(k+1))
\end{aligned}
$$

Linear MPC w/o Control and State Constraints

Lagrange function:

$$L(x,u,\boldsymbol{\lambda},\boldsymbol{\sigma}) := \frac{1}{2}\sum_{k=0}^{N-1}\Big(x(k)^\top V(k)x(k) + u(k)^\top W(k)u(k)\Big) + \boldsymbol{\sigma}^\top(x(0)-x_0) + \sum_{k=0}^{N-1}\boldsymbol{\lambda}(k+1)^\top(A(k)x(k)+B(k)u(k)-x(k+1))$$

Theorem – Optimality

Let (A1)-(A2) hold. Then LMPC-OCP is a convex linear-quadratic optimization problem and the following KKT conditions are necessary and sufficient for global optimality:

Linear MPC w/o Control and State Constraints

Lagrange function:

$$L(x,u,\boldsymbol{\lambda},\boldsymbol{\sigma}) := \frac{1}{2}\sum_{k=0}^{N-1}\left(x(k)^\top V(k)x(k) + u(k)^\top W(k)u(k)\right) + \boldsymbol{\sigma}^\top(x(0)-x_0) + \sum_{k=0}^{N-1}\boldsymbol{\lambda}(k+1)^\top(A(k)x(k)+B(k)u(k)-x(k+1))$$

Theorem – Optimality

Let (A1)-(A2) hold. Then LMPC-OCP is a convex linear-quadratic optimization problem and the following KKT conditions are necessary and sufficient for global optimality:

- optimal control:

$$u(k) = -W(k)^{-1}B(k)^\top\boldsymbol{\lambda}(k+1) \qquad (k = 0,1,\ldots,N-1)$$

Linear MPC w/o Control and State Constraints

Lagrange function:

$$L(x,u,\boldsymbol{\lambda},\boldsymbol{\sigma}) := \frac{1}{2}\sum_{k=0}^{N-1}\left(x(k)^\top V(k)x(k) + u(k)^\top W(k)u(k)\right) + \boldsymbol{\sigma}^\top (x(0)-x_0) + \sum_{k=0}^{N-1} \boldsymbol{\lambda}(k+1)^\top (A(k)x(k)+B(k)u(k)-x(k+1))$$

Theorem – Optimality

Let (A1)-(A2) hold. Then LMPC-OCP is a convex linear-quadratic optimization problem and the following KKT conditions are necessary and sufficient for global optimality:

- optimal control:

$$u(k) = -W(k)^{-1}B(k)^\top \boldsymbol{\lambda}(k+1) \qquad (k=0,1,\ldots,N-1)$$

- Discrete adjoint equation:

$$\boldsymbol{\lambda}(N) = 0$$
$$\boldsymbol{\lambda}(k) = V(k)x(k) + A(k)^\top \boldsymbol{\lambda}(k+1) \qquad (k=N-1,\ldots,1)$$

Linear MPC w/o Control and State Constraints

Proof:

- It is easy to verify that under (A1)-(A2) the Hessian of the objective function is positive semidefinite and since the constraints are linear (and thus convex), LMPC-OCP is convex.

Linear MPC w/o Control and State Constraints

Proof:

- It is easy to verify that under (A1)-(A2) the Hessian of the objective function is positive semidefinite and since the constraints are linear (and thus convex), LMPC-OCP is convex.
- Convexity implies that the first-order necessary Karush-Kuhn-Tucker conditions (KKT conditions) are even sufficient:

$$
\begin{aligned}
0 &= \nabla_{x(0)} L = \boldsymbol{\sigma} + V(0)x(0) + A(0)^\top \boldsymbol{\lambda}(1) \\
0 &= \nabla_{x(k)} L = V(k)x(k) + A(k)^\top \boldsymbol{\lambda}(k+1) - \boldsymbol{\lambda}(k) \qquad (k = 1, \ldots, N-1) \\
0 &= \nabla_{x(N)} L = -\boldsymbol{\lambda}(N) \\
0 &= \nabla_{u(k)} L = W(k)u(k) + B(k)^\top \boldsymbol{\lambda}(k+1) \qquad (k = 0, \ldots, N-1)
\end{aligned}
$$

These yield the assertion. □

Linear MPC w/o Control and State Constraints

KKT conditions and constraints yield a large-scale and sparse linear equation:

$$
\left(\begin{array}{ccccc:cccc}
Q_0 & & & & & E_0^\top & M_0^\top & & \\
 & Q_1 & & & & & E_1^\top & \ddots & \\
 & & \ddots & & & & & \ddots & \\
 & & & Q_{N-1} & & & & \ddots & M_{N-1}^\top \\
 & & & & & & & & E_N^\top \\
\hdashline
E_0 & & & & & & & & \\
M_0 & E_1 & & & & & & & \\
 & \ddots & \ddots & & & & & & \\
 & & M_{N-1} & E_N & & & & &
\end{array}\right)
\left(\begin{array}{c}
z(0) \\ z(1) \\ \vdots \\ z(N-1) \\ z(N) \\ \hdashline -\boldsymbol{\sigma} \\ \boldsymbol{\lambda}(1) \\ \vdots \\ \boldsymbol{\lambda}(N)
\end{array}\right)
=
\left(\begin{array}{c}
0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \hdashline -x_0 \\ 0 \\ \vdots \\ 0
\end{array}\right)
$$

where $z(k) = (x(k), u(k))$, $k = 0, \ldots, N-1$, $z(N) = x(N)$, $S_N = -I$,

$$
Q_k = \begin{pmatrix} V(k) & \\ & W(k) \end{pmatrix}, \quad M_k = \begin{pmatrix} A(k) & B(k) \end{pmatrix}, \quad E_k = \begin{pmatrix} -I & 0 \end{pmatrix} \quad (k = 0, \ldots, N-1)
$$

$\rightsquigarrow$ symmetric, saddle-point structure, direct solution by MA57, PARDISO, SuperLU, ...

Linear MPC w/o Control and State Constraints

What else could be done?

Linear MPC w/o Control and State Constraints

In order to solve the optimality conditions we use the Ansatz:

$$\lambda(k) := P(k)x(k) \qquad \text{for any } x(k),\ k = 1, \ldots, N$$

Linear MPC w/o Control and State Constraints

In order to solve the optimality conditions we use the Ansatz:

$$\lambda(k) := P(k)x(k) \qquad \text{for any } x(k),\ k = 1, \ldots, N$$

Then:

- Since $0 = \lambda(N) = P(N)x(N)$ shall be valid for any $x(N)$, we find $P(N) = 0$.

Linear MPC w/o Control and State Constraints

In order to solve the optimality conditions we use the Ansatz:

$$\lambda(k) := P(k)x(k) \qquad \text{for any } x(k),\, k = 1, \dots, N$$

Then:

- Since $0 = \lambda(N) = P(N)x(N)$ shall be valid for any $x(N)$, we find $P(N) = 0$.
- Optimal control:

$$u(k) := -W(k)^{-1}B(k)^\top P(k+1)x(k+1) \qquad (k = 0, \dots, N-1)$$

Linear MPC w/o Control and State Constraints

In order to solve the optimality conditions we use the Ansatz:

$$\lambda(k) := P(k)x(k) \qquad \text{for any } x(k),\ k = 1, \ldots, N$$

Then:

- Since $0 = \lambda(N) = P(N)x(N)$ shall be valid for any $x(N)$, we find $P(N) = 0$.
- Optimal control:

$$u(k) := -W(k)^{-1}B(k)^\top P(k+1)x(k+1) \qquad (k = 0, \ldots, N-1)$$

- Discrete dynamics:

$$x(k+1) = A(k)x(k) + B(k)u(k)$$

Linear MPC w/o Control and State Constraints

In order to solve the optimality conditions we use the Ansatz:

$$\lambda(k) := P(k)x(k) \qquad \text{for any } x(k),\ k = 1, \dots, N$$

Then:

- Since $0 = \lambda(N) = P(N)x(N)$ shall be valid for any $x(N)$, we find $P(N) = 0$.
- Optimal control:

$$u(k) := -W(k)^{-1}B(k)^\top P(k+1)x(k+1) \qquad (k = 0, \dots, N-1)$$

- Discrete dynamics:

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) \\ &= A(k)x(k) - B(k)W(k)^{-1}B(k)^\top P(k+1)x(k+1) \end{aligned}$$

Linear MPC w/o Control and State Constraints

In order to solve the optimality conditions we use the Ansatz:

$$\lambda(k) := P(k)x(k) \qquad \text{for any } x(k),\ k = 1, \dots, N$$

Then:

- Since $0 = \lambda(N) = P(N)x(N)$ shall be valid for any $x(N)$, we find $P(N) = 0$.
- Optimal control:

$$u(k) := -W(k)^{-1}B(k)^\top P(k+1)x(k+1) \qquad (k = 0, \dots, N-1)$$

- Discrete dynamics:

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) \\ &= A(k)x(k) - B(k)W(k)^{-1}B(k)^\top P(k+1)x(k+1) \end{aligned}$$

Solving for $x(k+1)$ yields

$$x(k+1) = \left(I + B(k)W(k)^{-1}B(k)^\top P(k+1)\right)^{-1} A(k)x(k)$$

Linear MPC w/o Control and State Constraints

Then for $k = 1, \ldots, N-1$:

$$0 = \boldsymbol{\lambda}(k) - P(k)x(k)$$

Linear MPC w/o Control and State Constraints

Then for $k = 1, \ldots, N - 1$:

$$\begin{aligned} 0 &= \boldsymbol{\lambda}(k) - P(k)x(k) \\ &= V(k)x(k) + A(k)^\top \boldsymbol{\lambda}(k+1) - P(k)x(k) \end{aligned}$$

Linear MPC w/o Control and State Constraints

Then for $k = 1, \ldots, N-1$:

$$\begin{aligned} 0 &= \boldsymbol{\lambda}(k) - P(k)x(k) \\ &= V(k)x(k) + A(k)^\top \boldsymbol{\lambda}(k+1) - P(k)x(k) \\ &= V(k)x(k) + A(k)^\top P(k+1)x(k+1) - P(k)x(k) \end{aligned}$$

Linear MPC w/o Control and State Constraints

Then for $k = 1, \ldots, N-1$:

$$
\begin{aligned}
0 &= \boldsymbol{\lambda}(k) - P(k)x(k) \\
&= V(k)x(k) + A(k)^\top \boldsymbol{\lambda}(k+1) - P(k)x(k) \\
&= V(k)x(k) + A(k)^\top P(k+1)x(k+1) - P(k)x(k) \\
&= \left(V(k) + A(k)^\top P(k+1) \left(I + B(k)W(k)^{-1}B(k)^\top P(k+1) \right)^{-1} A(k) - P(k) \right) x(k)
\end{aligned}
$$

Linear MPC w/o Control and State Constraints

Then for $k = 1, \ldots, N-1$:

$$
\begin{aligned}
0 &= \boldsymbol{\lambda}(k) - P(k)x(k) \\
&= V(k)x(k) + A(k)^\top \boldsymbol{\lambda}(k+1) - P(k)x(k) \\
&= V(k)x(k) + A(k)^\top P(k+1)x(k+1) - P(k)x(k) \\
&= \left(V(k) + A(k)^\top P(k+1) \left(I + B(k)W(k)^{-1}B(k)^\top P(k+1) \right)^{-1} A(k) - P(k) \right) x(k)
\end{aligned}
$$

This relation shall hold for any $x(k)$ and thus:

Linear MPC w/o Control and State Constraints

Then for $k = 1, \ldots, N-1$:

$$
\begin{aligned}
0 &= \boldsymbol{\lambda}(k) - P(k)x(k) \\
&= V(k)x(k) + A(k)^\top \boldsymbol{\lambda}(k+1) - P(k)x(k) \\
&= V(k)x(k) + A(k)^\top P(k+1)x(k+1) - P(k)x(k) \\
&= \left(V(k) + A(k)^\top P(k+1) \left(I + B(k)W(k)^{-1}B(k)^\top P(k+1) \right)^{-1} A(k) - P(k) \right) x(k)
\end{aligned}
$$

This relation shall hold for any $x(k)$ and thus:

Discrete Riccati Equation (1st version)

Set $P(N) = 0$ and then solve backwards for $k = N-1, \ldots, 1$:

$$
P(k) = V(k) + A(k)^\top P(k+1) \left(I + B(k)W(k)^{-1}B(k)^\top P(k+1) \right)^{-1} A(K)
$$

Linear MPC w/o Control and State Constraints

Sherman-Morrison-Woodbury Formula

$$(X + UCV)^{-1} = X^{-1} - X^{-1}U(C^{-1} + VX^{-1}U)^{-1}VX^{-1}$$

Linear MPC w/o Control and State Constraints

Sherman-Morrison-Woodbury Formula

$$(X + UCV)^{-1} = X^{-1} - X^{-1}U(C^{-1} + VX^{-1}U)^{-1}VX^{-1}$$

With $X = I$, $U = B(k)$, $C = W(k)^{-1}$, $V = B(k)^\top P(k+1)$ we obtain

$$\begin{aligned}&\left(I + B(k)W(k)^{-1}B(k)^\top P(k+1)\right)^{-1}\\ &\quad = I - B(k)\left(W(k) + B(k)^\top P(k+1)B(k)\right)^{-1} B(k)^\top P(k+1)\end{aligned}$$

Linear MPC w/o Control and State Constraints

By the Sherman-Morrison-Woodbury formula we find:

Discrete Riccati Equation (DRE)

Set $P(N) = 0$ and then solve backwards for $k = N-1, \ldots, 1$:

$$P(k) = V(k) + A(k)^\top \left(P(k+1) - P(k+1)B(k)M(k)^{-1}B(k)^\top P(k+1) \right) A(K)$$

where

$$M(k) := W(k) + B(k)^\top P(k+1)B(k)$$

Linear MPC w/o Control and State Constraints

By the Sherman-Morrison-Woodbury formula we find:

Discrete Riccati Equation (DRE)

Set $P(N) = 0$ and then solve backwards for $k = N-1, \dots, 1$:

$$P(k) = V(k) + A(k)^\top \left(P(k+1) - P(k+1)B(k)M(k)^{-1}B(k)^\top P(k+1) \right) A(K)$$

where

$$M(k) := W(k) + B(k)^\top P(k+1)B(k)$$

Yet another application of the Sherman-Morrison-Woodbury formula yields:

Feedback control law

$$u(k) = -M(k)^{-1}B(k)^\top P(k+1)A(k)x(k) \qquad (k = 0, \dots, N-1)$$

Linear MPC w/o Control and State Constraints

Special case:

$A = A(\cdot)$, $B = B(\cdot)$, $V = V(\cdot)$, $W = W(\cdot)$ are constant matrices.

Taking the limit $N \to \infty$ and assuming $P(\cdot)$ converges to P yields:

Discrete Algebraic Riccati Equation (DARE)

$$P = V + A^\top \left(P - PB \left(W + B^\top PB \right)^{-1} B^\top P \right) A$$

(numerical solution by, e.g., MATLAB solver `idare`)

Linear MPC w/o Control and State Constraints

Option 1: (if matrices A, B, V, W are constant)

LQR controller: Solve DARE and use the feedback control law

$$\mu(k, x(k)) = -Cx(k) \qquad \text{with} \qquad C = (W + B^\top PB)^{-1} B^\top PA$$

Linear MPC w/o Control and State Constraints

Option 1: (if matrices A, B, V, W are constant)

LQR controller: Solve DARE and use the feedback control law

$$\mu(k, x(k)) = -Cx(k) \qquad \text{with} \qquad C = (W + B^\top PB)^{-1} B^\top PA$$

The closed-loop system is

$$\begin{aligned} x(k+1) &= (A - BC)x(k) && (k = 0, 1, 2, \ldots) \\ x(0) &= x_0 && (x_0 \text{ given}) \end{aligned}$$

Linear MPC w/o Control and State Constraints

Option 1: (if matrices A, B, V, W are constant)

LQR controller: Solve DARE and use the feedback control law

$$\mu(k, x(k)) = -Cx(k) \qquad \text{with} \qquad C = (W + B^\top PB)^{-1} B^\top PA$$

The closed-loop system is

$$\begin{aligned} x(k+1) &= (A - BC)x(k) && (k = 0, 1, 2, \ldots) \\ x(0) &= x_0 && (x_0 \text{ given}) \end{aligned}$$

It is asymptotically stable, if $|\lambda| < 1$ for all eigenvalues of $A - BC$.

Linear MPC w/o Control and State Constraints

Option 2:

Linear MPC: Let $P_{k,N}(j)$, $j = 1, \ldots, N$, solve the DRE on the discrete time horizon from k to $k + N$. Use the feedback control law

$$\mu_N(k, x(k)) = -C(k)x(k)$$

with

$$C(k) = \left(W(k) + B(k)^\top P_{k,N}(1)B(k)\right)^{-1} B(k)^\top P_{k,N}(1)A(k)$$

Linear MPC w/o Control and State Constraints

Option 2:

Linear MPC: Let $P_{k,N}(j)$, $j = 1, \ldots, N$, solve the DRE on the discrete time horizon from k to $k + N$. Use the feedback control law

$$\mu_N(k, x(k)) = -C(k)x(k)$$

with

$$C(k) = \left(W(k) + B(k)^\top P_{k,N}(1)B(k)\right)^{-1} B(k)^\top P_{k,N}(1)A(k)$$

The closed-loop system is

$$x(k+1) = (A(k) - B(k)C(k))x(k) \qquad (k = 0, 1, 2, \ldots)$$

$$x(0) = x_0 \qquad (x_0 \text{ given})$$

Linear MPC w/o Control and State Constraints

Option 2:

Linear MPC: Let $P_{k,N}(j)$, $j = 1, \dots, N$, solve the DRE on the discrete time horizon from k to $k + N$. Use the feedback control law

$$\mu_N(k, x(k)) = -C(k)x(k)$$

with

$$C(k) = \left(W(k) + B(k)^\top P_{k,N}(1)B(k)\right)^{-1} B(k)^\top P_{k,N}(1)A(k)$$

The closed-loop system is

$$\begin{aligned} x(k+1) &= (A(k) - B(k)C(k))x(k) && (k = 0, 1, 2, \dots) \\ x(0) &= x_0 && (x_0 \text{ given}) \end{aligned}$$

Note: In the LMPC context the LMPC-OCP has to be considered on the discrete time interval from k to $k + N$ in step k of the LMPC control.

Contents

Linear MPC with Control Constraints

For simplicity let $x_{ref} = 0$ and $u_{ref} = 0$ and assume constant matrices $A = A(\cdot)$, $B = B(\cdot)$, $V = V(\cdot)$, $W = W(\cdot)$.

LMPC-OCP in discrete time

Minimize

$$J(x,u) = \frac{1}{2}\sum_{k=0}^{N-1}\left(x(k)^\top V x(k) + u(k)^\top W u(k)\right)$$

subject to the constraints

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) && (k = 0,\dots,N-1) \\ u(k) &\in U := [u_{min}, u_{max}] && (k = 0,\dots,N-1) \\ x(0) &= x_0 && (x_0 \text{ given}) \end{aligned}$$

Linear MPC with Control Constraints

For simplicity let $x_{ref} = 0$ and $u_{ref} = 0$ and assume constant matrices $A = A(\cdot)$, $B = B(\cdot)$, $V = V(\cdot)$, $W = W(\cdot)$.

LMPC-OCP in discrete time

Minimize

$$J(x,u) = \frac{1}{2}\sum_{k=0}^{N-1}\Big(x(k)^\top V x(k) + u(k)^\top W u(k)\Big)$$

subject to the constraints

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) && (k = 0,\ldots,N-1) \\ u(k) &\in U := [u_{min}, u_{max}] && (k = 0,\ldots,N-1) \\ x(0) &= x_0 && (x_0 \text{ given}) \end{aligned}$$

Assumptions:

(A1) W symmetric and positive definite

(A2) V symmetric and positive semi-definite

Linear MPC with Control Constraints

Solving the dynamics in discrete time recursively (= shooting idea) yields:

$$x(1) = Ax_0 + Bu(0)$$

Linear MPC with Control Constraints

Solving the dynamics in discrete time recursively (= shooting idea) yields:

$$x(1) = Ax_0 + Bu(0)$$

$$x(2) = Ax(1) + Bu(1)$$

Linear MPC with Control Constraints

Solving the dynamics in discrete time recursively (= shooting idea) yields:

$$x(1) = Ax_0 + Bu(0)$$

$$x(2) = Ax(1) + Bu(1) = A^2 x_0 + ABu(0) + Bu(1)$$

Linear MPC with Control Constraints

Solving the dynamics in discrete time recursively (= shooting idea) yields:

$$x(1) = Ax_0 + Bu(0)$$

$$x(2) = Ax(1) + Bu(1) = A^2 x_0 + ABu(0) + Bu(1)$$

$$x(3) = Ax(2) + Bu(2)$$

Linear MPC with Control Constraints

Solving the dynamics in discrete time recursively (= shooting idea) yields:

$$x(1) = Ax_0 + Bu(0)$$

$$x(2) = Ax(1) + Bu(1) = A^2 x_0 + ABu(0) + Bu(1)$$

$$x(3) = Ax(2) + Bu(2) = A^3 x_0 + A^2 Bu(0) + ABu(1) + Bu(2)$$

Linear MPC with Control Constraints

Solving the dynamics in discrete time recursively (= shooting idea) yields:

$$
\begin{aligned}
x(1) &= Ax_0 + Bu(0) \\
x(2) &= Ax(1) + Bu(1) = A^2 x_0 + ABu(0) + Bu(1) \\
x(3) &= Ax(2) + Bu(2) = A^3 x_0 + A^2 Bu(0) + ABu(1) + Bu(2) \\
&\vdots
\end{aligned}
$$

Linear MPC with Control Constraints

Solving the dynamics in discrete time recursively (= shooting idea) yields:

$$
\begin{aligned}
x(1) &= Ax_0 + Bu(0)\\
x(2) &= Ax(1) + Bu(1) = A^2 x_0 + ABu(0) + Bu(1)\\
x(3) &= Ax(2) + Bu(2) = A^3 x_0 + A^2 Bu(0) + ABu(1) + Bu(2)\\
&\vdots\\
x(k) &= A^k x_0 + \sum_{j=0}^{k-1} A^{k-1-j} Bu(j) \qquad (k = 0, \ldots, N)
\end{aligned}
$$

Linear MPC with Control Constraints

Solving the dynamics in discrete time recursively (= shooting idea) yields:

$$
\begin{aligned}
x(1) &= Ax_0 + Bu(0) \\
x(2) &= Ax(1) + Bu(1) = A^2 x_0 + ABu(0) + Bu(1) \\
x(3) &= Ax(2) + Bu(2) = A^3 x_0 + A^2 Bu(0) + ABu(1) + Bu(2) \\
&\vdots \\
x(k) &= A^k x_0 + \sum_{j=0}^{k-1} A^{k-1-j} Bu(j) \qquad (k = 0, \ldots, N)
\end{aligned}
$$

In vector notation:

$$
\underbrace{\begin{pmatrix} x(0) \\ x(1) \\ \vdots \\ x(N) \end{pmatrix}}_{=:X_N} = \underbrace{\begin{pmatrix} 0 & 0 & \cdots & 0 \\ B & 0 & \cdots & 0 \\ AB & B & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{pmatrix}}_{=:K_N} \underbrace{\begin{pmatrix} u(0) \\ u(1) \\ \vdots \\ u(N-1) \end{pmatrix}}_{=:U_{N-1}} + \underbrace{\begin{pmatrix} x_0 \\ Ax_0 \\ \vdots \\ A^N x_0 \end{pmatrix}}_{=:c_N}
$$

Linear MPC with Control Constraints

Introduction into the objective function yields a reduced objective function:

$$J(X_{N-1}, U_{N-1}) = \frac{1}{2} \sum_{k=0}^{N-1} \Big(x(k)^\top V x(k) + u(k)^\top W u(k) \Big)$$

Linear MPC with Control Constraints

Introduction into the objective function yields a reduced objective function:

$$\begin{aligned} J(X_{N-1}, U_{N-1}) &= \frac{1}{2} \sum_{k=0}^{N-1} \left(x(k)^\top V x(k) + u(k)^\top W u(k) \right) \\ &= \frac{1}{2} \left(X_{N-1}^\top \mathcal{V}_{N-1} X_{N-1} + U_{N-1}^\top \mathcal{W}_{N-1} U_{N-1} \right) \end{aligned}$$

with

$$\mathcal{V}_{N-1} := \text{diag}(\underbrace{V, \ldots, V}_{N \text{ times}}), \qquad \mathcal{W}_{N-1} := \text{diag}(\underbrace{W, \ldots, W}_{N \text{ times}})$$

Linear MPC with Control Constraints

Introduction into the objective function yields a reduced objective function:

$$
\begin{aligned}
J(X_{N-1}, U_{N-1}) &= \frac{1}{2}\sum_{k=0}^{N-1}\left(x(k)^\top Vx(k) + u(k)^\top Wu(k)\right)\\
&= \frac{1}{2}\left(X_{N-1}^\top \mathcal{V}_{N-1} X_{N-1} + U_{N-1}^\top \mathcal{W}_{N-1} U_{N-1}\right)\\
&= \frac{1}{2}\Big((K_{N-1}U_{N-1} + c_{N-1})^\top \mathcal{V}_{N-1}(K_{N-1}U_{N-1} + c_{N-1})\\
&\qquad + U_{N-1}^\top \mathcal{W}_{N-1} U_{N-1}\Big)
\end{aligned}
$$

with

$$
\mathcal{V}_{N-1} := \text{diag}(\underbrace{V, \ldots, V}_{N \text{ times}}), \qquad \mathcal{W}_{N-1} := \text{diag}(\underbrace{W, \ldots, W}_{N \text{ times}})
$$

Linear MPC with Control Constraints

Introduction into the objective function yields a reduced objective function:

$$\begin{aligned}
J(X_{N-1}, U_{N-1}) &= \frac{1}{2}\sum_{k=0}^{N-1}\left(x(k)^\top V x(k) + u(k)^\top W u(k)\right) \\
&= \frac{1}{2}\left(X_{N-1}^\top \mathcal{V}_{N-1} X_{N-1} + U_{N-1}^\top \mathcal{W}_{N-1} U_{N-1}\right) \\
&= \frac{1}{2}\Big((K_{N-1}U_{N-1} + c_{N-1})^\top \mathcal{V}_{N-1}(K_{N-1}U_{N-1} + c_{N-1}) \\
&\qquad + U_{N-1}^\top \mathcal{W}_{N-1} U_{N-1}\Big) \\
&= \frac{1}{2}U_{N-1}^\top\left(\mathcal{W}_{N-1} + K_{N-1}^\top \mathcal{V}_{N-1} K_{N-1}\right)U_{N-1} \\
&\qquad + c_{N-1}^\top \mathcal{V}_{N-1} K_{N-1} U_{N-1} + \frac{1}{2}c_{N-1}^\top \mathcal{V}_{N-1} c_{N-1}
\end{aligned}$$

with

$$\mathcal{V}_{N-1} := \text{diag}(\underbrace{V,\ldots,V}_{N \text{ times}}), \qquad \mathcal{W}_{N-1} := \text{diag}(\underbrace{W,\ldots,W}_{N \text{ times}})$$

Linear MPC with Control Constraints

Reduced LMPC-OCP

Minimize

$$J_R(U_{N-1}) := \frac{1}{2} U_{N-1}^\top \left(\mathcal{W}_{N-1} + K_{N-1}^\top \mathcal{V}_{N-1} K_{N-1} \right) U_{N-1} + c_{N-1}^\top \mathcal{V}_{N-1} K_{N-1} U_{N-1}$$

subject to the constraints

$$U_{N-1} \in U^N = [u_{min}, u_{max}]^N$$

Note:

- $U_{N-1} = (u(0), u(1), \ldots, u(N-1))^\top$
- Gradient:

$$\nabla J_R(U_{N-1}) = \left(\mathcal{W}_{N-1} + K_{N-1}^\top \mathcal{V}_{N-1} K_{N-1} \right) U_{N-1} + K_{N-1}^\top \mathcal{V}_{N-1} c_{N-1}$$

Linear MPC with Control Constraints

A necessary (and in this case sufficient) condition is:

$$\nabla J_R(\hat{U}_{N-1})^\top (u - \hat{U}_{N-1}) \geq 0 \qquad \forall u \in U^N$$

Linear MPC with Control Constraints

A necessary (and in this case sufficient) condition is:

$$\nabla J_R(\hat{U}_{N-1})^\top (u - \hat{U}_{N-1}) \geq 0 \qquad \forall u \in U^N$$

or equivalently (for convex control sets)

$$\hat{U}_{N-1} = \mathrm{proj}_{U^N}\left(\hat{U}_{N-1} - \boldsymbol{\nu}\nabla J_R(\hat{U}_{N-1})\right) \qquad (\boldsymbol{\nu} > 0 \text{ arbitrary})$$

Linear MPC with Control Constraints

A necessary (and in this case sufficient) condition is:

$$\nabla J_R(\hat{U}_{N-1})^\top (u - \hat{U}_{N-1}) \geq 0 \qquad \forall u \in U^N$$

or equivalently (for convex control sets)

$$\hat{U}_{N-1} = \text{proj}_{U^N}\left(\hat{U}_{N-1} - \nu \nabla J_R(\hat{U}_{N-1})\right) \qquad (\nu > 0 \text{ arbitrary})$$

$\rightsquigarrow$ stopping criterion in the following algorithm!

Linear MPC with Control Constraints

A necessary (and in this case sufficient) condition is:

$$\nabla J_R(\hat{U}_{N-1})^\top (u - \hat{U}_{N-1}) \geq 0 \qquad \forall u \in U^N$$

or equivalently (for convex control sets)

$$\hat{U}_{N-1} = \text{proj}_{U^N}\left(\hat{U}_{N-1} - \nu \nabla J_R(\hat{U}_{N-1})\right) \qquad (\nu > 0 \text{ arbitrary})$$

$\rightsquigarrow$ stopping criterion in the following algorithm!

Projection onto the control set $U = [u_{min}, u_{max}]$ (scalar case):

$$\text{proj}_U(u) = \max\{u_{min}, \min\{u_{max}, u\}\} = \begin{cases} u_{min}, & \text{if } u \leq u_{min} \\ u, & \text{if } u_{min} < u < u_{max} \\ u_{max}, & \text{if } u \geq u_{max} \end{cases}$$

Linear MPC with Control Constraints

A necessary (and in this case sufficient) condition is:

$$\nabla J_R(\hat{U}_{N-1})^\top (u - \hat{U}_{N-1}) \geq 0 \qquad \forall u \in U^N$$

or equivalently (for convex control sets)

$$\hat{U}_{N-1} = \mathrm{proj}_{U^N}\left(\hat{U}_{N-1} - \nu \nabla J_R(\hat{U}_{N-1})\right) \qquad (\nu > 0 \text{ arbitrary})$$

$\rightsquigarrow$ stopping criterion in the following algorithm!

Projection onto the control set $U = [u_{min}, u_{max}]$ (scalar case):

$$\mathrm{proj}_U(u) = \max\{u_{min}, \min\{u_{max}, u\}\} = \begin{cases} u_{min}, & \text{if } u \leq u_{min} \\ u, & \text{if } u_{min} < u < u_{max} \\ u_{max}, & \text{if } u \geq u_{max} \end{cases}$$

Note:

- The operators max, min, proj, when applied to a vector, are applied component-wise.

Projected Gradient Method for Reduced LMPC-OCP

Apply, e.g., a first order method (quick iterations, but linear convergence rate):

Projected gradient method

(0) Choose $U_{N-1}^{(0)} \in U^N$, $tol > 0$, and set $k := 0$.

Projected Gradient Method for Reduced LMPC-OCP

Apply, e.g., a first order method (quick iterations, but linear convergence rate):

Projected gradient method

(0) Choose $U_{N-1}^{(0)} \in U^N$, $tol > 0$, and set $k := 0$.

(1) If $\| U_{N-1}^{(k)} - \text{proj}_{U^N} \left(U_{N-1}^{(k)} - \nabla J_R(U_{N-1}^{(k)}) \right) \| \leq tol$, STOP.

Projected Gradient Method for Reduced LMPC-OCP

Apply, e.g., a first order method (quick iterations, but linear convergence rate):

Projected gradient method

(0) Choose $U_{N-1}^{(0)} \in U^N$, $tol > 0$, and set $k := 0$.

(1) If $\| U_{N-1}^{(k)} - \text{proj}_{U^N}\left(U_{N-1}^{(k)} - \nabla J_R(U_{N-1}^{(k)}) \right) \| \leq tol$, STOP.

(2) Set

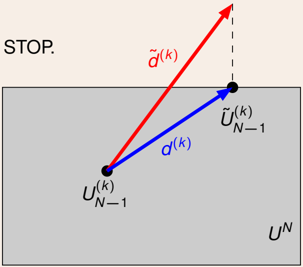
$$\tilde{d}^{(k)} := -\nabla J_R(U_{N-1}^{(k)}),$$

compute

$$\tilde{U}_{N-1}^{(k)} := \text{proj}_{U^N}\left(U_{N-1}^{(k)} + \tilde{d}^{(k)} \right)$$

and

$$d^{(k)} := \tilde{U}_{N-1}^{(k)} - U_{N-1}^{(k)}.$$



Projected Gradient Method for Reduced LMPC-OCP

Apply, e.g., a first order method (quick iterations, but linear convergence rate):

Projected gradient method

(0) Choose $U_{N-1}^{(0)} \in U^N$, $tol > 0$, and set $k := 0$.

(1) If $\| U_{N-1}^{(k)} - \text{proj}_{U^N}\left(U_{N-1}^{(k)} - \nabla J_R(U_{N-1}^{(k)})\right) \| \leq tol$, STOP.

(2) Set

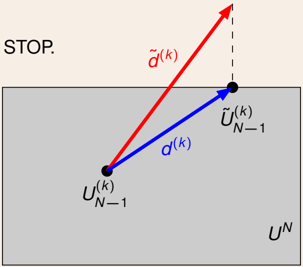
$$\tilde{d}^{(k)} := -\nabla J_R(U_{N-1}^{(k)}),$$

compute

$$\tilde{U}_{N-1}^{(k)} := \text{proj}_{U^N}\left(U_{N-1}^{(k)} + \tilde{d}^{(k)}\right)$$

and

$$d^{(k)} := \tilde{U}_{N-1}^{(k)} - U_{N-1}^{(k)}.$$



(3) Find step-size $\alpha_k \in (0, 1]$ that minimizes $J_R(U_{N-1}^{(k)} + \alpha d^{(k)})$ w.r.t. $\alpha > 0$.

Projected Gradient Method for Reduced LMPC-OCP

Apply, e.g., a first order method (quick iterations, but linear convergence rate):

Projected gradient method

(0) Choose $U_{N-1}^{(0)} \in \mathcal{U}^N$, $tol > 0$, and set $k := 0$.

(1) If $\| U_{N-1}^{(k)} - \text{proj}_{\mathcal{U}^N}\left(U_{N-1}^{(k)} - \nabla J_R(U_{N-1}^{(k)}) \right) \| \leq tol$, STOP.

(2) Set

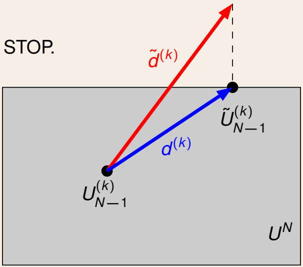
$$\tilde{d}^{(k)} := -\nabla J_R(U_{N-1}^{(k)}),$$

compute

$$\tilde{U}_{N-1}^{(k)} := \text{proj}_{\mathcal{U}^N}\left(U_{N-1}^{(k)} + \tilde{d}^{(k)} \right)$$

and

$$d^{(k)} := \tilde{U}_{N-1}^{(k)} - U_{N-1}^{(k)}.$$



(3) Find step-size $\alpha_k \in (0, 1]$ that minimizes $J_R(U_{N-1}^{(k)} + \alpha d^{(k)})$ w.r.t. $\alpha > 0$.

(4) Set $U_{N-1}^{(k+1)} := U_{N-1}^{(k)} + \alpha_k d^{(k)}$, $k \leftarrow k + 1$, and go to (1).

Projected Gradient Method – Example

Example

Minimize

$$\frac{h}{2}\sum_{k=0}^{N-1}\left(x(t_k)^2 + u(t_k)^2\right)$$

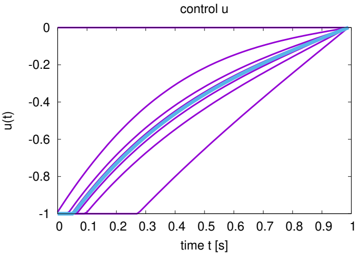
subject to the constraints

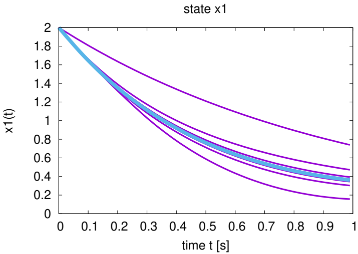
$$\begin{aligned} x(t_{k+1}) &= x(t_k) + h(-x(t_k) + \sqrt{3}u(t_k)),\ x(0) = 2 \qquad && (k = 0, \ldots, N-1) \\ u(t_k) &\in [-1, 0] && (k = 0, \ldots, N-1) \end{aligned}$$

Output of projected gradient method: ($u^{(0)} = 0$, $N = 100$, $\beta = 0.9$, $\sigma = 0.1$, $h = 1/N$, $t_k = kh$)

```
  K     ALPHA              OBJ                optimality         direct. deriv.
------------------------------------------------------------------------------
  0     0.00000000E+00     0.87037219E+00     0.10000000E+01     -0.56844172E+00
  1     0.10000000E+01     0.69730450E+00     0.53499282E+00     -0.12068778E+00
  2     0.10000000E+01     0.66899329E+00     0.24771541E+00     -0.34974402E-01
  3     0.10000000E+01     0.66069894E+00     0.13601372E+00     -0.10017036E-01
  4     0.10000000E+01     0.65839539E+00     0.72270190E-01     -0.29381895E-02
  5     0.10000000E+01     0.65771320E+00     0.39308112E-01     -0.85411090E-03
  6     0.10000000E+01     0.65751687E+00     0.21130142E-01     -0.24957026E-03
...
 22     0.10000000E+01     0.65743560E+00     0.11102230E-05     -0.63847553E-12
 23     0.10000000E+01     0.65743560E+00     0.62172489E-06     -0.18697552E-12
```

Projected Gradient Method – Example





Contents

NMPC with Constraints

How to handle control and state constraints, e.g.,

$$g(x(k)) \leq 0 \qquad (k = 0, \ldots, N)$$
$$u(k) \in [u_{min}, u_{max}] \qquad (k = 0, \ldots, N-1)$$

in NMPC?

$\rightsquigarrow$ direct solution of KKT conditions not possible anymore!

$\rightsquigarrow$ complementarity conditions cause trouble!

$\rightsquigarrow$ need for more general approach!

Solving Nonlinear Optimization Problems

Optimization frameworks: (many options!)

- nonlinear problems: sequential-quadratic programming (SQP), interior-point methods (IP), penalty or multiplier-penalty methods, semi-smooth Newton methods, ...
- linear quadratic problems: active-set methods, IP, semi-smooth Newton, ADMM, OSQP, ...

Globalization: (convergence from arbitrary points)

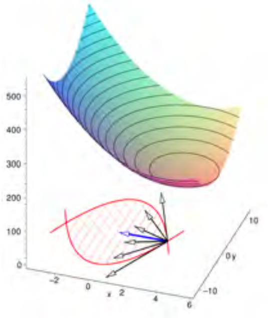
- line-search / trust-region methods / filter methods

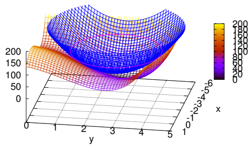
Peripheral problems:

- sparsity / large scales / rank deficiencies / regularization

Comprehensive textbook:

[1] J. Nocedal and S. J. Wright.
Numerical optimization.
2nd ed. New York, NY: Springer, 2006.

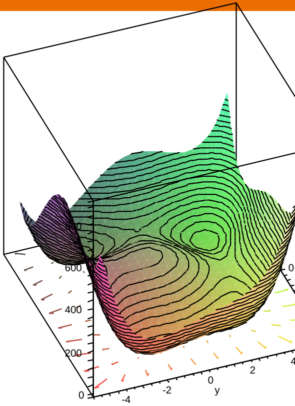




Solving Nonlinear Optimization Problems

Let

$$J : \mathbb{R}^{n_z} \longrightarrow \mathbb{R}$$
$$H = (H_1, \ldots, H_{n_H})^\top : \mathbb{R}^{n_z} \longrightarrow \mathbb{R}^{n_H}$$
$$G = (G_1, \ldots, G_{n_G})^\top : \mathbb{R}^{n_z} \longrightarrow \mathbb{R}^{n_G}$$



Nonlinear Optimization Problem (NLO)

$$\text{Minimize} \quad J(z) \quad \text{s.t.} \quad H(z) = 0,\; G(z) \leq 0$$

Contents

Interior-Point Method

Idea: Add slack variables to inequality constraints and a barrier term to objective function!

Barrier Problem BP(η)

Minimize

$$J(z) - \eta \sum_{i=1}^{n_G} \ln(s_i) \qquad (\eta > 0)$$

subject to the constraints

$$H(z) = 0$$
$$G(z) + s = 0$$

Approach:

- Solve BP(η_k) for a null sequence $\eta_k \downarrow 0$!
- The reduction of η_k and the iterative solution of BP(η_k) are intertwined.

Interior-Point Method

Lagrange function for BP(η):

$$L_\eta(z, s, \lambda, \mu) := J(z) - \eta \sum_{i=1}^{n_G} \ln(s_i) + \lambda^\top H(z) + \mu^\top (G(z) + s)$$

KKT Condition for Barrier Problem BP(η)

Let $(\hat{z}, \hat{s}) = (z(\eta), s(\eta))$ be a local minimizer of BP(η). Then, subject to LICQ:

$$\begin{aligned} \nabla_z L_\eta(\hat{z}, \hat{s}, \lambda, \mu) &= 0 \\ s_i \mu_i &= \eta \qquad (i = 1, \ldots, n_G) \\ H(\hat{z}) &= 0 \\ G(\hat{z}) + \hat{s} &= 0 \end{aligned}$$

$\rightsquigarrow$ nonlinear equation system; apply Newton's method

Note: Perturbation of KKT conditions of NLO.

Interior-Point Method

Lagrange-Newton Method for BP(η)

Newton step:

$$\begin{bmatrix} \nabla^2_{zz} L_{\eta,k} & H_k'^\top & G_k'^\top \\ H_k' & 0 & 0 \\ G_k' & 0 & -R_k^{-1} S_k \end{bmatrix} \begin{bmatrix} d_z \\ d_\lambda \\ d_\mu \end{bmatrix} = - \begin{bmatrix} \nabla_z L_{\eta,k} \\ H_k \\ \eta R_k^{-1} e + G_k \end{bmatrix}$$

and

$$d_s = -G_k' d_z - \left(G_k + s^{(k)} \right)$$

Update:

$$\begin{aligned} z^{(k+1)} &= z^{(k)} + \alpha_k d_z & (\alpha_k > 0 \text{ suitable}) \\ s^{(k+1)} &= s^{(k)} + \alpha_k d_s \\ \lambda^{(k+1)} &= \lambda^{(k)} + \alpha_k d_\lambda \\ \mu^{(k+1)} &= \mu^{(k)} + \beta_k d_\mu & (\beta_k > 0 \text{ suitable}) \end{aligned}$$

Notation:

- index k denotes evaluation at current iterate, e.g., $H_k' = H'(z^{(k)})$
- $R := \operatorname{diag}(\mu_1, \ldots, \mu_{n_G})$, $S := \operatorname{diag}(s_1, \ldots, s_{n_G})$

Interior-Point Method

More details (choice of step-sizes, adaption of barrier parameter,...):

[J. Nocedal and S. J. Wright. Numerical optimization. 2nd ed. New York, NY: Springer, 2nd ed. edition, 2006.]

[F. E. Curtis, O. Schenk, and A. Wächter. An interior-point algorithm for large-scale nonlinear optimization with inexact step computations. SIAM Journal of Scientific Computing, 32(6):3447–3475, 2010.]

[A. Wächter and L. T. Biegler. Line search filter methods for nonlinear programming: Local convergence. SIAM Journal on Optimization, 16(1):32–48, 2005.]

[A. Wächter and L. T. Biegler. Line search filter methods for nonlinear programming: Motivation and global convergence. SIAM Journal on Optimization, 16(1):1–31, 2005]

Contents

Semi-Smooth Newton Method

How to deal with the complementarity conditions

$$\mu_i \geq 0, \qquad G_i(z) \leq 0, \qquad \mu_i G_i(x) = 0 \qquad (i = 1, \ldots, n_G)$$

in the KKT conditions?

Idea: Use so-called nonlinear complementarity (NCP) functions!

NCP function

$\phi : \mathbb{R}^2 \longrightarrow \mathbb{R}$ is called an NCP function, iff

$$\phi(a, b) = 0 \qquad \Longleftrightarrow \qquad a \geq 0,\ b \geq 0,\ a \cdot b = 0$$

Semi-Smooth Newton Method

Examples of NCP functions:

- Min-Function:

$$\phi_{min}(a, b) := \min\{a, b\}$$

- Fischer-Burmeister-Function:

$$\phi_{FB}(a, b) := \sqrt{a^2 + b^2} - a - b$$

- ...

$\rightsquigarrow$ not differentiable, but Lipschitz-continuous

Semi-Smooth Newton Method

Application of NCP function to complementarity conditions in KKT conditions yields:

$$0 = \phi_{FB}(\boldsymbol{\mu}_i, -G_i(z)) \qquad (i = 1, \ldots, n_G)$$

KKT conditions are equivalent with the nonlinear equation system:

$$0 = F(z, \boldsymbol{\lambda}, \boldsymbol{\mu}) := \begin{pmatrix} \nabla_z L(z, \boldsymbol{\lambda}, \boldsymbol{\mu}) \\ H(z) \\ \phi_{FB}(\boldsymbol{\mu}, -G(z)) \end{pmatrix}$$

It remains to find a zero of F. We use a generalized version of Newton's method, since F is not differentiable.

Semi-Smooth Newton Method

Let $w := (z, \boldsymbol{\lambda}, \boldsymbol{\mu})$.

Bouligand-Differential, Clarke's Generalized Jacobian

(a) B(ouligand)-differential:

$$\partial_B F(w) \quad := \quad \left\{ V \,\middle|\, V = \lim_{\substack{w_i \in D_F \\ w_i \to w}} F'(w_i) \right\}$$

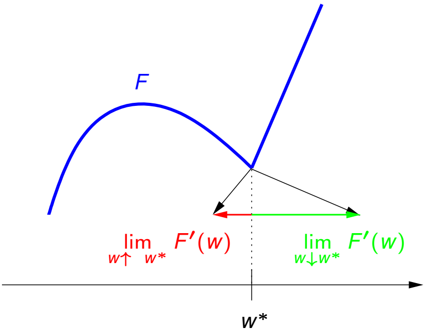
(D_F is the set of points at which F' exists)

(b) Clarke's Generalized Jacobian:

$$\partial F(w) := \text{conv}\,(\partial_B F(w))$$

(conv is the convex hull)

Semi-Smooth Newton Method



Semi-Smooth Newton Method

Example

Fischer-Burmeister-Function:

$$\phi_{FB}(a,b) := \sqrt{a^2+b^2} - a - b$$

B-differential:

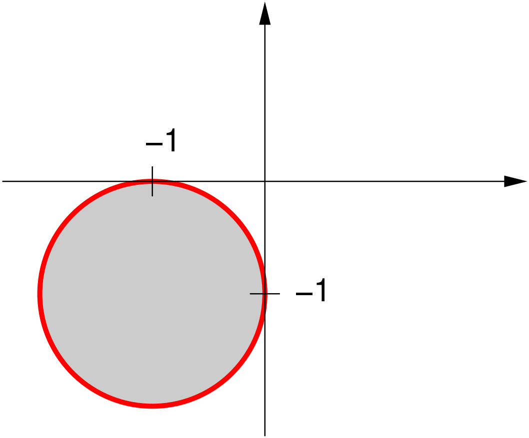
$$\partial_B \phi_{FB}(a,b) = \begin{cases} \left\{\left(\frac{a}{\sqrt{a^2+b^2}} - 1, \frac{b}{\sqrt{a^2+b^2}} - 1\right)\right\}, & \text{if } (a,b) \neq (0,0), \\ \left\{(s,r) \mid (r+1)^2 + (s+1)^2 = 1\right\}, & \text{if } (a,b) = (0,0). \end{cases}$$

Clarke's generalized Jacobian:

$$\partial \phi_{FB}(a,b) = \begin{cases} \left\{\left(\frac{a}{\sqrt{a^2+b^2}} - 1, \frac{b}{\sqrt{a^2+b^2}} - 1\right)\right\}, & \text{if } (a,b) \neq (0,0), \\ \left\{(s,r) \mid (r+1)^2 + (s+1)^2 \leq 1\right\}, & \text{if } (a,b) = (0,0). \end{cases}$$

Semi-Smooth Newton Method

B-differential $\partial_B \phi_{FB}$ (red circle) and Clarke's generalized Jacobian $\partial \phi_{FB}$ (shaded disc):



Semi-Smooth Newton Method

Semi-Smooth Newton Method

(0) Choose $w^{(0)} = (z^{(0)}, \boldsymbol{\lambda}^{(0)}, \boldsymbol{\mu}^{(0)})^\top$ and set $k := 0$.

(1) If $F(w^{(k)}) = 0$, STOP.

(2) Compute search direction $d^{(k)}$ from linear equation

$$V_k d = -F(w^{(k)})$$

with arbitrary $V_k \in \partial F(w^{(k)})$.

(3) Set $w^{(k+1)} = w^{(k)} + d^{(k)}$, set $k \leftarrow k + 1$, and go to (1).

Under standard assumptions the algorithm has the same nice convergence properties as the classic Newton method, i.e. locally quadratic convergence.

Semi-Smooth Newton Method

$$\partial F(z,\boldsymbol{\lambda},\boldsymbol{\mu}) \subset \begin{pmatrix} \nabla_{zz} L(z,\boldsymbol{\lambda},\boldsymbol{\mu}) & H'(z)^\top & G'(z)^\top \\ H'(z) & 0 & 0 \\ -R(\boldsymbol{\mu},z)G'(z) & 0 & S(\boldsymbol{\mu},z) \end{pmatrix}$$

with

$$R(\boldsymbol{\mu},z) = \operatorname{diag}\left(R_1(\boldsymbol{\mu}_1,z),\ldots,R_{n_G}(\boldsymbol{\mu}_{n_G},z)\right)$$
$$S(\boldsymbol{\mu},z) = \operatorname{diag}\left(S_1(\boldsymbol{\mu}_1,z),\ldots,S_{n_G}(\boldsymbol{\mu}_{n_G},z)\right)$$

and

$$(S_i(\boldsymbol{\mu}_i,z),R_i(\boldsymbol{\mu}_i,z)) \in \partial\phi_{FB}(\boldsymbol{\mu}_i,-G_i(z)) \qquad (i=1,\ldots,N_G)$$

Semi-Smooth Newton Method

Example

$$\text{Minimize} \quad f(x_1, x_2) = (x_1 - 2)^2 + (x_2 - 3)^2 \quad \text{s.t.} \quad \begin{cases} x_2 + \frac{1}{2}x_1 - \frac{1}{2} & = & 0 \\ x_2 + 2x_1^2 - 2 & \leq & 0 \\ x_1^2 - x_2 - 1 & \leq & 0 \end{cases}$$

Lagrange function: ($x = (x_1, x_2)^\top$, $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)^\top$)

$$L(x, \boldsymbol{\lambda}, \boldsymbol{\mu}) = (x_1 - 2)^2 + (x_2 - 3)^2 + \boldsymbol{\lambda}\left(x_2 + \frac{1}{2}x_1 - \frac{1}{2}\right) + \boldsymbol{\mu}_1\left(x_2 + 2x_1^2 - 2\right) + \boldsymbol{\mu}_2\left(x_1^2 - x_2 - 1\right)$$

KKT conditions as non-smooth equation using Fischer-Burmeister function:

$$0 = F(w) = \begin{pmatrix} 2(x_1 - 2) + \frac{1}{2}\boldsymbol{\lambda} + (4\boldsymbol{\mu}_1 + 2\boldsymbol{\mu}_2)x_1 \\ 2(x_2 - 3) + \boldsymbol{\lambda} + \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 \\ x_2 + \frac{1}{2}x_1 - \frac{1}{2} \\ \phi_{FB}\left(\boldsymbol{\mu}_1, -\left(x_2 + 2x_1^2 - 2\right)\right) \\ \phi_{FB}\left(\boldsymbol{\mu}_2, -\left(x_1^2 - x_2 - 1\right)\right) \end{pmatrix}$$

Semi-Smooth Newton Method

Example

Init: $(x_1, x_2) = (5, -1)$, $\boldsymbol{\mu} = (0, 0)^\top$, $\boldsymbol{\lambda} = 0$

```
----- NSNEWTON VERSION 1.0 (C) Matthias Gerdts, University of Hamburg, 2006 -----
       NUMBER OF VARIABLES               :          2
       NUMBER OF EQUALITY CONSTRAINTS    :          1
       NUMBER OF INEQUALITY CONSTRAINTS  :          2
       OPTIMALITY TOLERANCE              :  0.100E-09
       LINE SEARCH PARAMETER             :  SIGMA=  0.100E-01 BETA=  0.900E+00
       DESCENT PARAMETER                 :    RHO=  0.100E-01
       MAXIMUM NUMBER OF ITERATIONS      :        100
       ROUNDOFF TOLERANCE                :  0.222E-15
---------------------------------------------------------------------------------
  ITER     ALPHA          NB           GL           KKT          |D|
---------------------------------------------------------------------------------
    0    0.0000E+00   0.1065E+03   0.1000E+02   0.1069E+03   0.1112E+02   ns
    1    0.1000E+01   0.2573E+02   0.3126E+01   0.2591E+02   0.3570E+01   ns
    2    0.1000E+01   0.5656E+01   0.3613E+00   0.5668E+01   0.1675E+01   ns
    3    0.1000E+01   0.9059E+00   0.1487E+00   0.9180E+00   0.2537E+00   ns
    4    0.1000E+01   0.3193E+00   0.3267E-01   0.3210E+00   0.2528E+00   ns
    5    0.1000E+01   0.1705E+00   0.4643E-01   0.1767E+00   0.3670E+00   ns
    6    0.1000E+01   0.2410E-01   0.1261E+00   0.1284E+00   0.4789E-01   ns
    7    0.1000E+01   0.1877E-03   0.2919E-02   0.2925E-02   0.2046E-02   ns
    8    0.1000E+01   0.1340E-07   0.1079E-05   0.1079E-05   0.6293E-06   ns
    9    0.1000E+01   0.7457E-16   0.2251E-13   0.2251E-13   0.6293E-06   ns
---------------------------------------------------------------------------------
 OBJ =          0.9800000000000001E+01
 VARIABLE   VALUE
  1          0.6000000000000090E+00
  2          0.1999999999999955E+00
 CONSTRAINT VALUE                          LAMBDA
  1          0.0000000000000000E+00        0.5600000000000009E+01
  2         -0.1079999999999983E+01       -0.3005334439095832E-16
  3         -0.8399999999999848E+00       -0.6824801378326157E-16
```

Contents

Structure Exploitation and Realtime Approaches in MPC

Core requirement of MPC:

Realtime capability, i.e. evaluation of control law must not take too much time!

How to achieve this?

- structure exploitation in linear algebra
- reduce number of online optimizations using multi-step MPC or sensitivity updates
- re-use of previous solutions as initial guess for next MPC step
- faster hardware
- use of efficient compilers and code generators
- ...

Contents

Linear MPC Revisited

The linear case revisited ...

LMPC-OCP in discrete time

Minimize

$$\frac{1}{2}\sum_{k=0}^{N-1}\left(x(k)^\top V(k)x(k) + u(k)^\top W(k)u(k)\right)$$

subject to the constraints

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) && (k = 0,\ldots,N-1) \\ x(0) &= x_0 && (x_0 \text{ given}) \end{aligned}$$

Assumptions:

(A1) $W(k)$ symmetric and positive definite for all k

(A2) $V(k)$ symmetric and positive semi-definite for all k

Linear MPC Revisited

We already noticed that the KKT conditions and constraints yield a large-scale and sparse linear equation:

$$
\left(\begin{array}{cccc|cccc}
Q_0 & & & & E_0^\top & M_0^\top & & \\
 & Q_1 & & & & E_1^\top & & \\
 & & \ddots & & & & \ddots & \\
 & & & Q_{N-1} & & & \ddots & M_{N-1}^\top \\
 & & & & & & & E_N^\top \\
\hline
E_0 & & & & & & & \\
M_0 & E_1 & & & & & & \\
 & \ddots & \ddots & & & & & \\
 & & M_{N-1} & E_N & & & &
\end{array}\right)
\left(\begin{array}{c}
z(0) \\ z(1) \\ \vdots \\ z(N-1) \\ z(N) \\ \hline -\boldsymbol{\sigma} \\ \boldsymbol{\lambda}(1) \\ \vdots \\ \boldsymbol{\lambda}(N)
\end{array}\right)
=
\left(\begin{array}{c}
0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \hline -x_0 \\ 0 \\ \vdots \\ 0
\end{array}\right)
$$

where $z(k) = (x(k), u(k))$, $k = 0, \ldots, N-1$, $z(N) = x(N)$, $S_N = -I$,

$$
Q_k = \begin{pmatrix} V(k) & \\ & W(k) \end{pmatrix}, \quad M_k = \begin{pmatrix} A(k) & B(k) \end{pmatrix}, \quad E_k = \begin{pmatrix} -I & 0 \end{pmatrix} \quad (k = 0, \ldots, N-1)
$$

Linear MPC Revisited – Structure Exploitation

Re-arranging the matrix by column and row permutations yield:

$$
\begin{pmatrix}
 & E_0 & & & & & & & & \\
E_0^\top & Q_0 & M_0^\top & & & & & & & \\
 & M_0 & & E_1 & & & & & & \\
 & & E_1^\top & Q_1 & M_1^\top & & & & & \\
 & & & M_1 & & E_2 & & & & \\
 & & & \ddots & \ddots & \ddots & & & & \\
 & & & & \ddots & \ddots & E_{N-1} & & & \\
 & & & & & E_{N-1}^\top & Q_{N-1} & M_{N-1}^\top & & \\
 & & & & & & M_{N-1} & & E_N & \\
 & & & & & & & E_N^\top & &
\end{pmatrix}
\begin{pmatrix}
-\sigma \\ z(0) \\ \lambda(1) \\ z(1) \\ \lambda(2) \\ \vdots \\ z(N-1) \\ \lambda(N) \\ z(N)
\end{pmatrix}
=
\begin{pmatrix}
-x_0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0
\end{pmatrix}
$$

- ⇝ banded symmetric matrix, bandwidth depends only on number of states and controls
- ⇝ computational effort for LU factorization depends linearly on the preview horizon N!
- ⇝ LU factorization by, e.g., LAPACK or INTEL MKS routines DGBTRF, DGBTRS

Structure Exploitation in NMPC

DOCP is of the following type, but with a certain structure.

Nonlinear Optimization Problem (NLO)

$$\text{Minimize} \quad J(z) \quad \text{s.t.} \quad G(z) \leq 0, \quad H(z) = 0$$

$$z := (x(0), u(0), \ldots, x(N-1), u(N-1), x(N))^\top$$

$$J(z) := \varphi(x(N)) + \sum_{k=0}^{N-1} \ell(x(k), u(k))$$

$$H(z) := \begin{pmatrix} f(x(0), u(0)) - x(1) \\ \vdots \\ f(x(N-1), u(N-1)) - x(N) \end{pmatrix}, \qquad G(z) := \begin{pmatrix} g(x(0)) \\ \vdots \\ g(x(N)) \\ \hline u(0) - u_{max} \\ \vdots \\ u(N-1) - u_{max} \\ \hline u_{min} - u(0) \\ \vdots \\ u_{min} - u(N-1) \end{pmatrix}$$

Structure Exploitation in NMPC

Interior-point methods require to solve symmetric linear equations with saddle point structure:

$$\begin{pmatrix} Q & A^\top & B^\top \\ A & 0 & 0 \\ B & 0 & -R^{-1}S \end{pmatrix}$$

Semi-smooth Newton methods require to solve non-symmetric linear equations of type

$$\begin{pmatrix} Q & A^\top & B^\top \\ A & 0 & 0 \\ -RB & 0 & S \end{pmatrix}$$

Remarks:

- In iteration k: $Q := \nabla^2_{zz} L(z^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\mu}^{(k)})$, $A := H'(z^{(k)})$, $B := G'(z^{(k)})$
- S, R: diagonal matrices with the following properties:
  - positive definite for IP
  - positive semidefinite for semi-smooth Newton (with Fischer-Burmeister function)
- active-set SQP methods (not discussed here) yield IP structure with $S = 0$ and only active constraints in B

Structure Exploitation in NMPC

Hessian of the Lagrangian:

$$Q = \begin{bmatrix} Q_0 & & & & \\ & Q_1 & & & \\ & & \ddots & & \\ & & & Q_{N-1} & \\ & & & & Q_N \end{bmatrix}$$

Structure Exploitation in NMPC

Jacobians of the Constraints:

$$A = \begin{bmatrix} E_0 & & & \\ \hdashline M_0 & E_1 & & \\ & \ddots & \ddots & \\ & & M_{N-1} & E_N \end{bmatrix}, \qquad \begin{aligned} M_k &= \begin{pmatrix} f'_x(x(k),u(k)) & f'_u(x(k),u(k)) \end{pmatrix} \\ E_k &= \begin{pmatrix} -I & 0 \end{pmatrix} \quad (k = 0,\ldots,N-1) \\ E_N &= -I \end{aligned}$$

$$B = \begin{bmatrix} C_0 & & & \\ & C_1 & & \\ & & \ddots & \\ & & & C_N \end{bmatrix}, \qquad \begin{aligned} C_k &= \begin{pmatrix} g'(x(k)) & 0 \\ 0 & I \\ 0 & -I \end{pmatrix} \quad (k = 0,\ldots,N-1) \\ C_N &= g'(x(N)) \end{aligned}$$

Structure Exploitation in NMPC

KKT Matrix after column and row permutations: (Interior-Point method, $D_k = -R_k^{-1} S_k$)

$$
\left(\begin{array}{c|ccc|ccc|c|ccc|cc}
 & E_0 & & & & & & & & & & & \\
\hline
E_0^\top & Q_0 & C_0^\top & M_0^\top & & & & & & & & & \\
 & C_0 & D_0 & & & & & & & & & & \\
 & M_0 & & & E_1 & & & & & & & & \\
\hline
 & & & E_1^\top & Q_1 & C_1^\top & M_1^\top & & & & & & \\
 & & & & C_1 & D_1 & & & & & & & \\
 & & & & M_1 & & & E_2 & & & & & \\
\hline
 & & & & & & \ddots & \ddots \quad \ddots & & & & & \\
 & & & & & & & \ddots \quad \ddots & E_{N-1} & & & & \\
\hline
 & & & & & & & E_{N-1}^\top & Q_{N-1} & C_{N-1}^\top & M_{N-1}^\top & & \\
 & & & & & & & & C_{N-1} & D_{N-1} & & & \\
 & & & & & & & & M_{N-1} & & & E_N & \\
\hline
 & & & & & & & & & & E_N^\top & Q_N & C_N^\top \\
 & & & & & & & & & & & C_N & D_N
\end{array}\right)
$$

Structure Exploitation in NMPC

KKT Matrix after column and row permutations: (semi-smooth Newton method)

$$
\begin{pmatrix}
 & E_0 & & & & & & & & & & & & \\
E_0^\top & Q_0 & C_0^\top & M_0^\top & & & & & & & & & & \\
 & -R_0 C_0 & S_0 & & & & & & & & & & & \\
 & M_0 & & & E_1 & & & & & & & & & \\
 & & & E_1^\top & Q_1 & C_1^\top & M_1^\top & & & & & & & \\
 & & & & -R_1 C_1 & S_1 & & & & & & & & \\
 & & & & M_1 & & & E_2 & & & & & & \\
 & & & & & & \ddots & \ddots & \ddots & & & & & \\
 & & & & & & & \ddots & \ddots & & & & & \\
 & & & & & & & & & E_{N-1} & & & & \\
 & & & & & & & & E_{N-1}^\top & Q_{N-1} & C_{N-1}^\top & M_{N-1}^\top & & \\
 & & & & & & & & & -R_{N-1} C_{N-1} & S_{N-1} & & & \\
 & & & & & & & & & M_{N-1} & & & E_N & \\
 & & & & & & & & & & & E_N^\top & Q_N & C_N^\top \\
 & & & & & & & & & & & & -R_N C_N & S_N
\end{pmatrix}
$$

Structure Exploitation in NMPC

Observations:

- similar structure with banded matrix, bandwidth depends only on number of states, controls, and constraints
- symmetric system for Interior-Point method
- non-symmetric system for semi-smooth Newton method

Numerical solution:

- computational effort for LU factorization depends linearly on the preview horizon N
- LU factorization for both systems by, e.g., LAPACK or INTEL MKS routines DGBTRF, DGBTRS

Composed Linear Equation Systems

If terminal conditions or parameters are included, systems of the following type arise:

$$\begin{bmatrix} \Gamma & V^\top \\ V & \Lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} \qquad (\Gamma \text{ large scale, banded, } \Lambda \text{ low dimensional})$$

Solution procedure:

(1) Compute LU decompostion of block diagonal matrix Γ by LAPACK with OpenBLAS (or INTEL MKL).

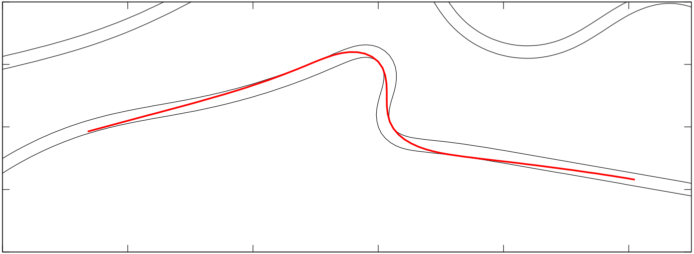
(2) Solve low dimensional system

$$\left(\Lambda - V\Gamma^{-1}V^\top\right) y = \boldsymbol{\beta} - V\Gamma^{-1}\boldsymbol{\alpha}$$

(3) Solve large dimensional system $\Gamma x = \boldsymbol{\alpha} - V^\top y$.

[A. Huber, M. Gerdts, E. Bertolazzi: Structure Exploitation in an Interior-Point Method for Fully Discretized, State Constrained Optimal Control Problems, Vietnam Journal of Mathematics, Vol. 46(4), pp. 1089–1113, 2018.]

Path Generation, Test 1



Solution with a horizon of 500 [m] and 80 gridpoints needs 0.022 [s] to optimize.

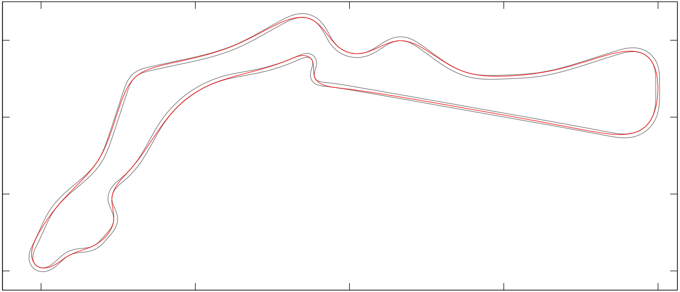
Path Generation, Test 1, parallel $\Gamma^{-1} V^\top$

Grid points	LAPACK KKT		HSL/MA57	
	T_{ges}	T_{lin}	T_{ges}	T_{lin}
10	0.002600 s	0.001125 s	0.003664 s	0.002169 s
100	0.022445 s	0.008535 s	0.035134 s	0.020237 s
1000	0.152356 s	0.080296 s	0.555060 s	0.479449 s
10000	1.592431 s	0.754386 s	10.160251 s	9.498326 s
100000	10.875577 s	6.980800 s	427.733114 s	423.264320 s
150000	17.158301 s	10.833331 s	931.503742 s	924.729321 s

Table: Test of linear solvers

```
export OMP_PROC_BIND=TRUE
export OMP_WAIT_POLICY=PASSIVE
```

Path Generation Full Lap



Optimal control problem with free final time and boundary conditions.

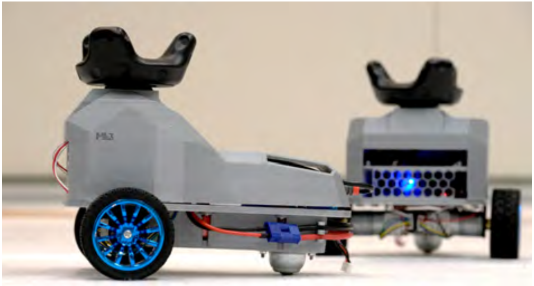
Solution for a lap with 600 gridpoints needs 0.26 s to optimize.

Contents

Path Planning for Mobile Robot

Parameter influence:

- location of obstacles
- distribution of weight
- surface excitations
- voltage fluctuation
- initial state ($\rightsquigarrow$ MPC)
- ...

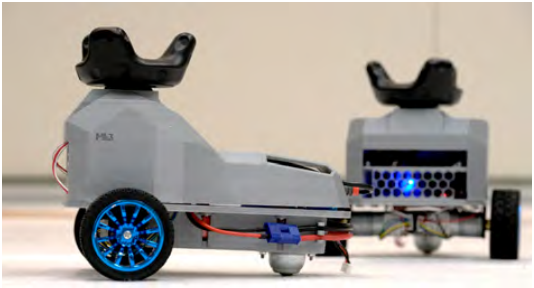


Path Planning for Mobile Robot

Parameter influence:

- location of obstacles
- distribution of weight
- surface excitations
- voltage fluctuation
- initial state ($\rightsquigarrow$ MPC)
- ...

Questions:



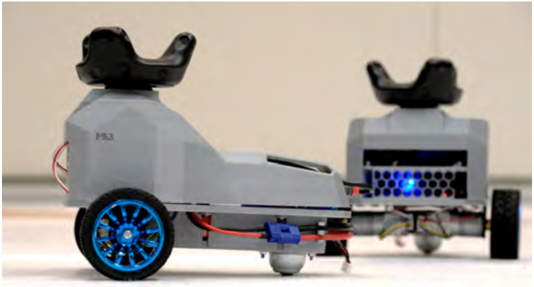
Path Planning for Mobile Robot

Parameter influence:

- location of obstacles
- distribution of weight
- surface excitations
- voltage fluctuation
- initial state ($\rightsquigarrow$ MPC)
- ...

Questions:

- How does the optimal solution change, if parameters change?



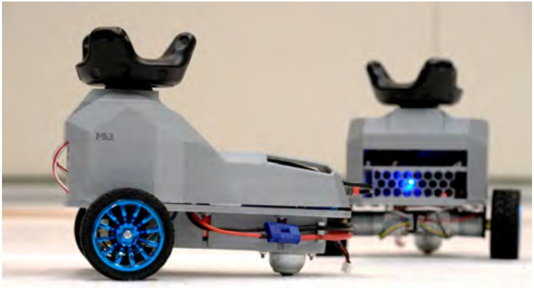
Path Planning for Mobile Robot

Parameter influence:

- location of obstacles
- distribution of weight
- surface excitations
- voltage fluctuation
- initial state ($\rightsquigarrow$ MPC)
- ...

Questions:

- How does the optimal solution change, if parameters change?
- How can sensitivities be computed?



Linear MPC Revisited – Parameter Dependence

KKT system in linear MPC w/o control and state constraints:

$$
\left(\begin{array}{c:cc:cc:c:cc:c}
 & E_0 & & & & & & & \\ \hdashline
E_0^\top & Q_0 & M_0^\top & & & & & & \\
 & M_0 & & E_1 & & & & & \\ \hdashline
 & & E_1^\top & Q_1 & M_1^\top & & & & \\
 & & & M_1 & & E_2 & & & \\ \hdashline
 & & & \ddots & \ddots & \ddots & & & \\
 & & & & \ddots & \ddots & E_{N-1} & & \\ \hdashline
 & & & & & E_{N-1}^\top & Q_{N-1} & M_{N-1}^\top & \\
 & & & & & & M_{N-1} & & E_N \\ \hdashline
 & & & & & & & E_N^\top &
\end{array}\right)
\left(\begin{array}{c}
-\boldsymbol{\sigma} \\ \hdashline
z(0) \\ \boldsymbol{\lambda}(1) \\ \hdashline
z(1) \\ \boldsymbol{\lambda}(2) \\ \hdashline
\vdots \\ \hdashline
z(N-1) \\ \boldsymbol{\lambda}(N) \\ \hdashline
z(N)
\end{array}\right)
=
\left(\begin{array}{c}
-x_0 \\ \hdashline
0 \\ 0 \\ \hdashline
0 \\ 0 \\ \hdashline
\vdots \\ \hdashline
0 \\ 0 \\ \hdashline
0
\end{array}\right)
$$

$\rightsquigarrow$ solution depends linearly on x_0

$\rightsquigarrow$ all matrices constant $\Rightarrow$ only one LU factorization needed for all LMPC steps

Parametric Optimization

Consider the unconstrained optimization problem:

$$\min_{z \in \mathbb{R}^n} \quad J(z, p)$$

Parametric Optimization

Consider the unconstrained optimization problem:

$$\min_{z \in \mathbb{R}^n} \quad J(z, p)$$

Necessary condition: (at $\hat{z}$ and p_0)

$$0 = \nabla_z J(\hat{z}, p_0)$$

Parametric Optimization

Consider the unconstrained optimization problem:

$$\min_{z \in \mathbb{R}^n} \quad J(z, p)$$

Necessary condition: (at $\hat{z}$ and p_0)

$$0 = \nabla_z J(\hat{z}, p_0)$$

Application of implicit function theorem:

If $\nabla^2_{zz} J(\hat{z}, p_0)$ is positive definite, then we can solve for z as a function of p, i.e.

$$z = z(p) \qquad \text{satisfies} \qquad \nabla_z J(z(p), p) = 0 \quad \forall p \in B_\delta(p_0).$$

Parametric Optimization

Consider the unconstrained optimization problem:

$$\min_{z \in \mathbb{R}^n} \quad J(z, p)$$

Necessary condition: (at $\hat{z}$ and p_0)

$$0 = \nabla_z J(\hat{z}, p_0)$$

Application of implicit function theorem:

If $\nabla^2_{zz} J(\hat{z}, p_0)$ is positive definite, then we can solve for z as a function of p, i.e.

$$z = z(p) \qquad \text{satisfies} \qquad \nabla_z J(z(p), p) = 0 \quad \forall p \in B_\delta(p_0).$$

Moreover,

$$\nabla^2_{zz} J(\hat{z}, p_0) \cdot z'(p_0) = -\nabla_{zp} J(z(p_0), p_0)$$

Parametric Optimization

Consider the unconstrained optimization problem:

$$\min_{z \in \mathbb{R}^n} \quad J(z, p)$$

Necessary condition: (at $\hat{z}$ and p_0)

$$0 = \nabla_z J(\hat{z}, p_0)$$

Application of implicit function theorem:

If $\nabla^2_{zz} J(\hat{z}, p_0)$ is positive definite, then we can solve for z as a function of p, i.e.

$$z = z(p) \qquad \text{satisfies} \qquad \nabla_z J(z(p), p) = 0 \quad \forall p \in B_\delta(p_0).$$

Moreover,

$$\nabla^2_{zz} J(\hat{z}, p_0) \cdot \underbrace{z'(p_0)}_{\textit{sensitivity matrix}} = -\nabla_{zp} J(z(p_0), p_0)$$

Parametric Optimization

Real-time updates

Taylor approximation:

For a perturbed parameter $p \neq p_0$ compute

$$\tilde{z}(p) := z(p_0) + z'(p_0)(p - p_0)$$

and use $\tilde{z}(p)$ as an approximation of $z(p)$.

⇝ very quick, only matrix-vector multiplication
⇝ $z'(p_0)$ can be computed offline and stored in a database for different parameters
⇝ only valid for small perturbations[1]

[1] see [C. Buckner, M. Gerdts, R. Lampariello: Neighborhood estimation in sensitivity-based update rules for real-time optimal control. 2020 European Control Conference, to appear, 2020]

Parametric Optimization

NLP(p)

$$
\begin{aligned}
\text{Minimize} \quad & J(z, p) \\
\text{s.t.} \quad & g_i(z, p) = 0, \quad i = 1, \ldots, n_E, \\
& g_i(z, p) \leq 0, \quad i = n_E + 1, \ldots, n_g
\end{aligned}
$$

Notation:

- $J, g_i : \mathbb{R}^{n_z} \times \mathbb{R}^{n_p} \longrightarrow \mathbb{R}$, $i = 1, \ldots, n_g$, twice continuously differentiable
- parameter $p \in \mathbb{R}^{n_p}$ (no optimization variable!)
- Active inequalities: $I(z, p) := \{i \in \{n_E + 1, \ldots, n_g\} \mid g_i(z, p) = 0\}$
- Active set: $A(z, p) := \{1, \ldots, n_E\} \cup I(z, p)$

Parametric Optimization

Definition

z^* strongly regular local minimum of $NLP(p_0)$ iff

(a) z^* is feasible.

Parametric Optimization

Definition

z^* strongly regular local minimum of $NLP(p_0)$ iff

(a) z^* is feasible.

(b) z^* fulfills the linear independence constraint qualification (LICQ).

Parametric Optimization

Definition

z^* strongly regular local minimum of $NLP(p_0)$ iff

(a) z^* is feasible.

(b) z^* fulfills the linear independence constraint qualification (LICQ).

(c) The KKT conditions hold at $(z^*, \boldsymbol{\lambda}^*)$ with Lagrange multiplier $\boldsymbol{\lambda}^*$.

Parametric Optimization

Definition

z^* strongly regular local minimum of $NLP(p_0)$ iff

(a) z^* is feasible.

(b) z^* fulfills the linear independence constraint qualification (LICQ).

(c) The KKT conditions hold at $(z^*, \boldsymbol{\lambda}^*)$ with Lagrange multiplier $\boldsymbol{\lambda}^*$.

(d) The strict complementarity condition holds:

$$\boldsymbol{\lambda}_i^* - g_i(z^*, p_0) > 0 \quad \text{for all } i = n_E + 1, \ldots, n_g.$$

Parametric Optimization

Definition

z^* strongly regular local minimum of $NLP(p_0)$ iff

(a) z^* is feasible.

(b) z^* fulfills the linear independence constraint qualification (LICQ).

(c) The KKT conditions hold at $(z^*, \boldsymbol{\lambda}^*)$ with Lagrange multiplier $\boldsymbol{\lambda}^*$.

(d) The strict complementarity condition holds:

$$\boldsymbol{\lambda}_i^* - g_i(z^*, p_0) > 0 \quad \text{for all } i = n_E + 1, \ldots, n_g.$$

(e) Second-order sufficient condition:

$$L''_{zz}(z^*, \boldsymbol{\lambda}^*, p_0)(d, d) > 0$$

for all $0 \neq d \in T_C(z^*, p_0)$ with the critical cone

$$T_C(z,p) := \left\{ d \in \mathbb{R}^{n_z} \;\middle|\; \begin{array}{lcl} g'_{i,z}(z,p)(d) & \leq & 0,\ i \in I(z,p), \boldsymbol{\lambda}_i = 0, \\ g'_{i,z}(z,p)(d) & = & 0,\ i \in I(z,p), \boldsymbol{\lambda}_i > 0, \\ g'_{i,z}(z,p)(d) & = & 0,\ i \in \{1,\ldots,n_E\} \end{array} \right\}.$$

Parametric Optimization

Sensitivity Theorem [Fiacco'83]

Let z^* be a strongly regular local minimum of $NLP(p_0)$ for nominal parameter p_0.

Then there exist neighborhoods $B_\epsilon(p_0)$ and $B_\delta(z^*, \boldsymbol{\lambda}^*)$ with:

Parametric Optimization

Sensitivity Theorem [Fiacco'83]

Let z^* be a strongly regular local minimum of $NLP(p_0)$ for nominal parameter p_0.

Then there exist neighborhoods $B_\epsilon(p_0)$ and $B_\delta(z^*, \lambda^*)$ with:

- $NLP(p)$ has unique strongly regular local minimum

$$(z(p), \lambda(p)) \in B_\delta(z^*, \lambda^*)$$

  for each $p \in B_\epsilon(p_0)$.

Parametric Optimization

Sensitivity Theorem [Fiacco'83]

Let z^* be a strongly regular local minimum of $NLP(p_0)$ for nominal parameter p_0.

Then there exist neighborhoods $B_\epsilon(p_0)$ and $B_\delta(z^*, \boldsymbol{\lambda}^*)$ with:

- $NLP(p)$ has unique strongly regular local minimum

$$(z(p), \boldsymbol{\lambda}(p)) \in B_\delta(z^*, \boldsymbol{\lambda}^*)$$

  for each $p \in B_\epsilon(p_0)$.
- $z(p)$, $\boldsymbol{\lambda}(p)$ are continuously differentiable w.r.t. p.

Parametric Optimization

Sensitivity Theorem [Fiacco'83]

Let z^* be a strongly regular local minimum of $NLP(p_0)$ for nominal parameter p_0.

Then there exist neighborhoods $B_\epsilon(p_0)$ and $B_\delta(z^*, \lambda^*)$ with:

- $NLP(p)$ has unique strongly regular local minimum

$$(z(p), \lambda(p)) \in B_\delta(z^*, \lambda^*)$$

  for each $p \in B_\epsilon(p_0)$.
- $z(p)$, $\lambda(p)$ are continuously differentiable w.r.t. p.
- The active set remains unchanged for each $p \in B_\epsilon(p_0)$:

$$A(z^*, p_0) = A(z(p), p).$$

Parametric Optimization

Sensitivity Theorem [Fiacco'83]

Let z^* be a strongly regular local minimum of $NLP(p_0)$ for nominal parameter p_0.

Then there exist neighborhoods $B_\epsilon(p_0)$ and $B_\delta(z^*, \boldsymbol{\lambda}^*)$ with:

- $NLP(p)$ has unique strongly regular local minimum

$$(z(p), \boldsymbol{\lambda}(p)) \in B_\delta(z^*, \boldsymbol{\lambda}^*)$$

  for each $p \in B_\epsilon(p_0)$.
- $z(p)$, $\boldsymbol{\lambda}(p)$ are continuously differentiable w.r.t. p.
- The active set remains unchanged for each $p \in B_\epsilon(p_0)$:

$$A(z^*, p_0) = A(z(p), p).$$

Proof: implicit function theorem + stability of LICQ, critical cone and second-order sufficient condition

Computing Neighboring Solutions in Realtime

Real-Time Approximation for NLP(p)

(0) Let a nominal parameter p_0 be given.

Computing Neighboring Solutions in Realtime

Real-Time Approximation for NLP(p)

(0) Let a nominal parameter p_0 be given.

(1) Offline computation: Solve NLP(p_0) with solution $z_0 = z(p_0)$ and the linear equation

$$\begin{pmatrix} \nabla^2_{zz} L & \nabla_z g_{A(z_0,p_0)} \\ (\nabla_z g_{A(z_0,p_0)})^\top & 0 \end{pmatrix} \begin{pmatrix} z'(p_0) \\ \lambda'(p_0) \end{pmatrix} = -\begin{pmatrix} \nabla^2_{zp} L \\ (\nabla_p g_{A(z_0,p_0)})^\top \end{pmatrix}$$

Computing Neighboring Solutions in Realtime

Real-Time Approximation for NLP(p)

(0) Let a nominal parameter p_0 be given.

(1) Offline computation: Solve NLP(p_0) with solution $z_0 = z(p_0)$ and the linear equation

$$\begin{pmatrix} \nabla^2_{zz} L & \nabla_z g_{A(z_0,p_0)} \\ (\nabla_z g_{A(z_0,p_0)})^\top & 0 \end{pmatrix} \begin{pmatrix} z'(p_0) \\ \lambda'(p_0) \end{pmatrix} = - \begin{pmatrix} \nabla^2_{zp} L \\ (\nabla_p g_{A(z_0,p_0)})^\top \end{pmatrix}$$

(2) Taylor approximation (real-time approximation): For a perturbed parameter $p \neq \hat{p}$ compute

$$\tilde{z}(p) := z(p_0) + z'(p_0)(p - p_0)$$

and use $\tilde{z}(p)$ as an approximation of $z(p)$.

Computing Neighboring Solutions in Realtime

Real-Time Approximation for NLP(p)

(0) Let a nominal parameter p_0 be given.

(1) Offline computation: Solve NLP(p_0) with solution $z_0 = z(p_0)$ and the linear equation

$$\begin{pmatrix} \nabla^2_{zz} L & \nabla_z g_{A(z_0,p_0)} \\ (\nabla_z g_{A(z_0,p_0)})^\top & 0 \end{pmatrix} \begin{pmatrix} z'(p_0) \\ \lambda'(p_0) \end{pmatrix} = - \begin{pmatrix} \nabla^2_{zp} L \\ (\nabla_p g_{A(z_0,p_0)})^\top \end{pmatrix}$$

(2) Taylor approximation (real-time approximation): For a perturbed parameter $p \neq \hat{p}$ compute

$$\tilde{z}(p) := z(p_0) + z'(p_0)(p - p_0)$$

and use $\tilde{z}(p)$ as an approximation of $z(p)$.

Limitations:

Computing Neighboring Solutions in Realtime

Real-Time Approximation for NLP(p)

(0) Let a nominal parameter p_0 be given.

(1) Offline computation: Solve NLP(p_0) with solution $z_0 = z(p_0)$ and the linear equation

$$\begin{pmatrix} \nabla^2_{zz} L & \nabla_z g_{A(z_0,p_0)} \\ (\nabla_z g_{A(z_0,p_0)})^\top & 0 \end{pmatrix} \begin{pmatrix} z'(p_0) \\ \lambda'(p_0) \end{pmatrix} = - \begin{pmatrix} \nabla^2_{zp} L \\ (\nabla_p g_{A(z_0,p_0)})^\top \end{pmatrix}$$

(2) Taylor approximation (real-time approximation): For a perturbed parameter $p \neq \hat{p}$ compute

$$\tilde{z}(p) := z(p_0) + z'(p_0)(p - p_0)$$

and use $\tilde{z}(p)$ as an approximation of $z(p)$.

Limitations:

- approximation error $\|z(p) - \tilde{z}(p)\| = o(\|p - p_0\|)$

Computing Neighboring Solutions in Realtime

Real-Time Approximation for NLP(p)

(0) Let a nominal parameter p_0 be given.

(1) Offline computation: Solve NLP(p_0) with solution $z_0 = z(p_0)$ and the linear equation

$$\begin{pmatrix} \nabla^2_{zz} L & \nabla_z g_{A(z_0,p_0)} \\ (\nabla_z g_{A(z_0,p_0)})^\top & 0 \end{pmatrix} \begin{pmatrix} z'(p_0) \\ \lambda'(p_0) \end{pmatrix} = - \begin{pmatrix} \nabla^2_{zp} L \\ (\nabla_p g_{A(z_0,p_0)})^\top \end{pmatrix}$$

(2) Taylor approximation (real-time approximation): For a perturbed parameter $p \neq \hat{p}$ compute

$$\tilde{z}(p) := z(p_0) + z'(p_0)(p - p_0)$$

and use $\tilde{z}(p)$ as an approximation of $z(p)$.

Limitations:

- approximation error $\|z(p) - \tilde{z}(p)\| = o(\|p - p_0\|)$
- constraint violation due to linearization (projection required if feasibility is an issue)

Computing Neighboring Solutions in Realtime

Real-Time Approximation for NLP(p)

(0) Let a nominal parameter p_0 be given.

(1) Offline computation: Solve NLP(p_0) with solution $z_0 = z(p_0)$ and the linear equation

$$\begin{pmatrix} \nabla^2_{zz} L & \nabla_z g_{A(z_0,p_0)} \\ (\nabla_z g_{A(z_0,p_0)})^\top & 0 \end{pmatrix} \begin{pmatrix} z'(p_0) \\ \lambda'(p_0) \end{pmatrix} = - \begin{pmatrix} \nabla^2_{zp} L \\ (\nabla_p g_{A(z_0,p_0)})^\top \end{pmatrix}$$

(2) Taylor approximation (real-time approximation): For a perturbed parameter $p \neq \hat{p}$ compute

$$\tilde{z}(p) := z(p_0) + z'(p_0)(p - p_0)$$

and use $\tilde{z}(p)$ as an approximation of $z(p)$.

Limitations:

- approximation error $\|z(p) - \tilde{z}(p)\| = o(\|p - p_0\|)$
- constraint violation due to linearization (projection required if feasibility is an issue)
- linearization only justified in neighborhood $B_\epsilon(p_0)$, same active set

Corrector Iteration for Feasibility [Büskens]

NLP(p)

$$\text{Minimize} \quad J(z, p) \quad \text{s.t.} \quad g(z, p) = 0$$

Corrector Iteration for Feasibility [Büskens]

NLP(p)

$$\text{Minimize} \quad J(z, p) \quad \text{s.t.} \quad g(z, p) = 0$$

Realtime approximation:

$$\tilde{z}^{[0]}(p) := \tilde{z}(p) = z(p_0) + \frac{dz}{dp}(p_0)(p - p_0)$$

Corrector Iteration for Feasibility [Büskens]

NLP(p)

$$\text{Minimize} \quad J(z, p) \quad \text{s.t.} \quad g(z, p) = 0$$

Realtime approximation:

$$\tilde{z}^{[0]}(p) := \tilde{z}(p) = z(p_0) + \frac{dz}{dp}(p_0)(p - p_0)$$

Introducing $\tilde{z}(p)$ into constraints yields constraint violation

$$\varepsilon^{[0]} := g(\tilde{z}(p), p)$$

Corrector Iteration for Feasibility [Büskens]

NLP(p)

$$\text{Minimize} \qquad J(z, p) \qquad \text{s.t.} \qquad g(z, p) = 0$$

Realtime approximation:

$$\tilde{z}^{[0]}(p) := \tilde{z}(p) = z(p_0) + \frac{dz}{dp}(p_0)(p - p_0)$$

Introducing $\tilde{z}(p)$ into constraints yields constraint violation

$$\varepsilon^{[0]} := g(\tilde{z}(p), p)$$

Idea of corrector iteration:

- perform sensitivity analysis w.r.t. ε for nominal parameter $\varepsilon_0 = 0$

Corrector Iteration for Feasibility [Büskens]

NLP(p)

$$\text{Minimize} \quad J(z,p) \quad \text{s.t.} \quad g(z,p) = 0$$

Realtime approximation:

$$\tilde{z}^{[0]}(p) := \tilde{z}(p) = z(p_0) + \frac{dz}{dp}(p_0)(p - p_0)$$

Introducing $\tilde{z}(p)$ into constraints yields constraint violation

$$\varepsilon^{[0]} := g(\tilde{z}(p), p)$$

Idea of corrector iteration:

- perform sensitivity analysis w.r.t. ε for nominal parameter $\varepsilon_0 = 0$
- apply correction (fixed point iteration):

$$\tilde{z}^{[i+1]}(p) := \tilde{z}^{[i]}(p) - \frac{dz}{d\varepsilon}(p_0, \varepsilon_0)\varepsilon^{[i]}, \qquad \varepsilon^{[i]} := g(\tilde{z}^{[i]}(p), p)$$

(note the minus sign!)

Full Car Model

BMW 1800/2000 [von Heydenaber'80]

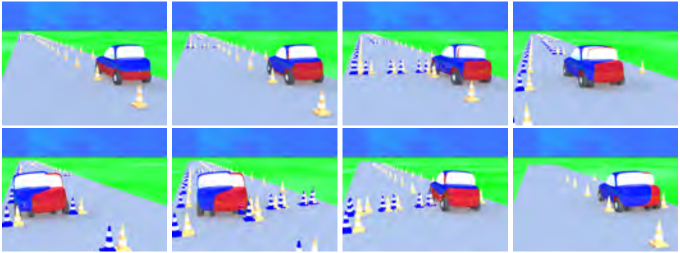
$$\begin{aligned} \dot{x}(t) &= f(x(t), y(t), u(t)) \\ 0 &= g(x(t), y(t), u(t)) \end{aligned}$$

Notation

state	:	$x(t) \in \mathbb{R}^{37}$, $y(t) \in \mathbb{R}^4$
control	:	$u(t)$ (steering angle velocity)
DAE	:	index 1, semi-explicit, g'_y non-sing.,
		piecewise defined dynamics
		(wheel: ground contact yes/np)

Example: Testdrive

$t_0 = 0$ [s]



$t_f = 6.79784$ [s]

parameters: height of center of gravity and offset of obstacle

Example: Emergency Landing Maneuver I

Parameters in emergency landing maneuver:

- p_1 with nominal value $\hat{p}_1 = 0$ models uncertainties in the air density $\rho(h) = \rho_0 \exp(-\beta h)$ through

$$\rho_0 = 1.249512(1 + p_1).$$

- p_2 with nominal value $\hat{p}_2 = 0$ models uncertainties in the initial altitude $h(0)$ according to

$$h(0) = 33900 + 10^4 p_2.$$

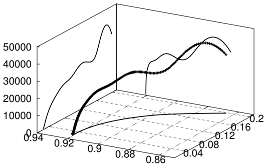
  Herein, the initial altitude is assumed to be free with the restriction $h(0) \geq 33900$.

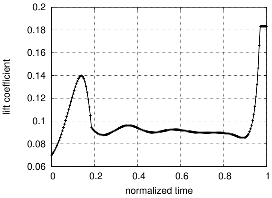
- p_3 with nominal value $\hat{p}_3 = 0$ models uncertainties in the terminal altitude $h(t_f)$ according to

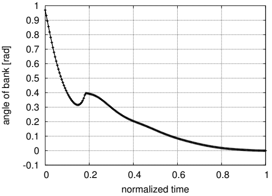
$$h(t_f) = 500 - p_3.$$

Example: Emergency Landing Maneuver II

Nominal solution for $N = 201$: flight path (left), lift coeff. (middle), bank angle (right)







Example: Emergency Landing Maneuver III

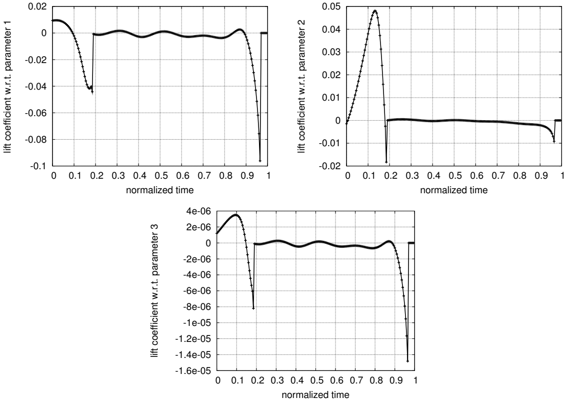
The dynamic pressure constraint is active in the approximate normalized time interval $[0.185, 0.19]$.

Sensitivities of free final time t_f with nominal value $t_f \approx 726.57$:

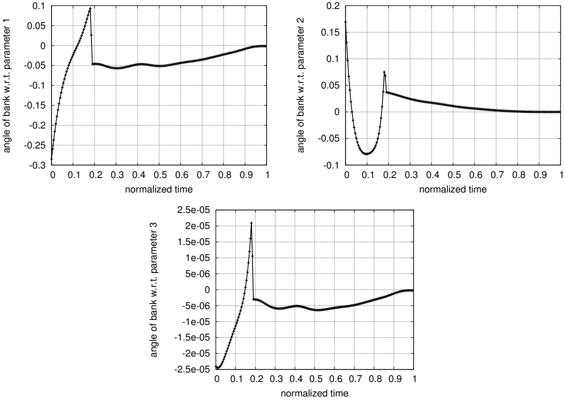
$$\frac{dt_f}{dp_1}(\hat{p}) \approx 92.54, \quad \frac{dt_f}{dp_2}(\hat{p}) \approx 9.164, \quad \frac{dt_f}{dp_3}(\hat{p}) \approx 0.014.$$

Example: Emergency Landing Maneuver IV

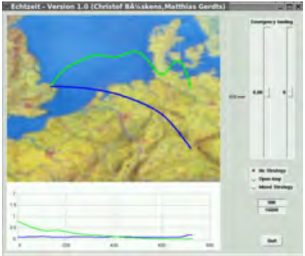
Sensitivities of the nominal controls C_L (top) and μ (bottom) with respect to p_1 (left), p_2 (middle), and p_3 (right):



Example: Emergency Landing Maneuver V



Emergency Landing Manoeuvre in Realtime

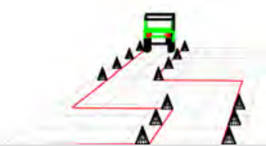


- Scenario: propulsion system breakdown
- Goal: maximization of range w.r.t. current position
- Controls: lift coefficient, angle of bank
- no thrust available; no fuel consumption (constant mass)

Collision Avoidance and Sensitivity I

Single lane change (avoidance maneuver):

Sensitivity analysis of maneuver w.r.t. initial yaw angle (nominal value $p_1^* = 0$) and obstacle motion (nominal value $p_2^* = 0$).



Obstacle position at time t: ($v_{obs} = 100$ [km/h], $\psi_{obs} = 170$ [°], d = initial distance)

$$x_{obs}(t) = d + t\, p_2 v_{obs} \cos\psi_{obs}, \qquad y_{obs}(t) = 3.5 + t\, p_2 v_{obs} \sin\psi_{obs}$$

Constraint: (b = width of car)

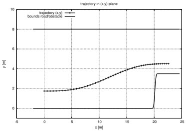
$$s(x(t), x_{obs}(t), y_{obs}(t)) + \frac{b}{2} \leq y(t) \leq 8 - \frac{b}{2}$$

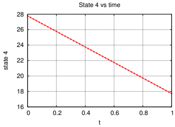
with

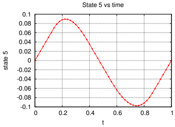
$$s(x,d,h) := \begin{cases} 0, & \text{if } x < d, \\ 4h(x-d)^3, & \text{if } d \leq x < d + 0.5, \\ 4h(x-(d+1))^3 + h, & \text{if } d + 0.5 \leq x < d + 1, \\ h, & \text{if } x \geq d + 1. \end{cases}$$

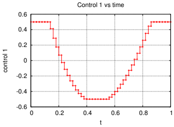
Collision Avoidance and Sensitivity II

Nominal solution for $p_1^* = p_2^* = 0$: ($N = 51$, $t_f \approx 1.00541$ [s], $d \approx 19.62075$ [m])

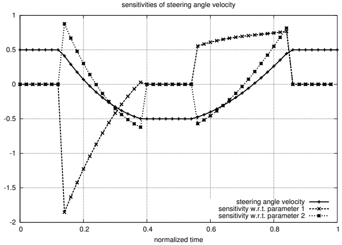








Sensitivity of steering angle velocity w.r.t. p_1, p_2:



Sensitivity of final time t_f w.r.t. p_1, p_2 :

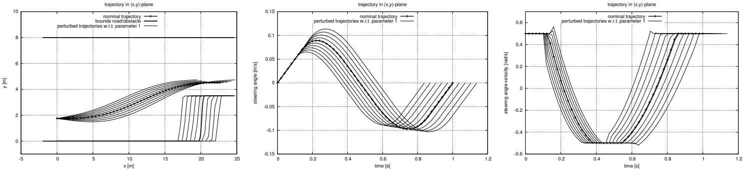
$$\frac{\partial t_f}{\partial p_1} \approx -1.66018, \qquad \frac{\partial t_f}{\partial p_2} \approx 0.50118$$

Sensitivity of minimal distance d w.r.t. p_1, p_2 :

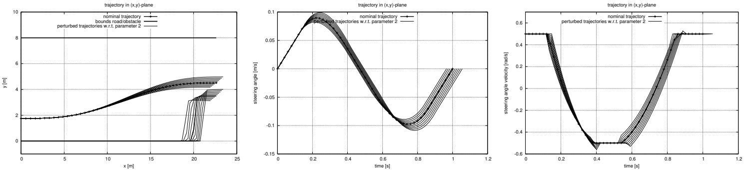
$$\frac{\partial d}{\partial p_1} \approx -28.95949, \qquad \frac{\partial d}{\partial p_2} \approx 35.66225$$

Collision Avoidance and Sensitivity III

Predicted optimal solutions for perturbations $p_1 \in [-0.1, 0.1]$:



Predicted optimal solutions for perturbations $p_2 \in [-0.1, 0.1]$:



References

[1] A. V. Fiacco.
Introduction to Sensitivity and Stability Analysis in Nonlinear Programming, volume 165 of *Mathematics in Science and Engineering*.
Academic Press, New York, 1983.

[2] B. Bank, J. Guddat, D. Klatte, B. Kummer, and K. Tammer.
Non-linear parametric optimization.
Birkhäuser, Basel, 1983.

[3] J. F. Bonnans and A. Shapiro.
Perturbation Analysis of Optimization Problems.
Springer Series in Operations Research. Springer, New York, 2000.

[4] K. Malanowski and H. Maurer.
Sensitivity analysis for parametric control problems with control-state constraints.
Computational Optimization and Applications, 5(3):253–283, 1996.

[5] K. Malanowski and H. Maurer.
Sensitivity analysis for optimal control problems subject to higher order state constraints.
Annals of Operations Research, 101:43–73, 2001.

[6] H. Maurer and D. Augustin.
Sensitivity Analysis and Real-Time Control of Parametric Optimal Control Problems Using Boundary Value Methods.
In M. Grötschel, S. O. Krumke, and J. Rambau, editors, *Online Optimization of Large Scale Systems*, pages 17–55. Springer, 2001.

[7] C. Büskens.
Real-Time Solutions for Perturbed Optimal Control Problems by a Mixed Open- and Closed-Loop Strategy.
In M. Grötschel, S. O. Krumke, and J. Rambau, editors, *Online Optimization of Large Scale Systems*, pages 105–116. Springer, 2001.

[8] C. Büskens and H. Maurer.
Sensitivity Analysis and Real-Time Control of Parametric Optimal Control Problems Using Nonlinear Programming Methods.
In M. Grötschel, S. O. Krumke, and J. Rambau, editors, *Online Optimization of Large Scale Systems*, pages 56–68. Springer, 2001.

[9] H. J. Pesch.
Numerical computation of neighboring optimum feedback control schemes in real-time.
Applied Mathematics and Optimization, 5:231–252, 1979.

[10] H. J. Pesch.
Real-time computation of feedback controls for constrained optimal control problems. i, ii.
Optimal Control Applications and Methods, 10(2):129–145, 147–171, 1989.

Contents

M-multistep NMPC with re-optimization

M-multistep NMPC with re-optimization

(0) Input: preview horizon N, reference trajectory $(x_r(\cdot), u_r(\cdot))$, weight matrices V and W, control horizon $M \leq N$. Set $k = 0$.

M-multistep NMPC with re-optimization

M-multistep NMPC with re-optimization

(0) Input: preview horizon N, reference trajectory $(x_r(\cdot), u_r(\cdot))$, weight matrices V and W, control horizon $M \leq N$. Set $k = 0$.

(1) For $j = 0, \ldots, M-1$ do

M-multistep NMPC with re-optimization

M-multistep NMPC with re-optimization

(0) Input: preview horizon N, reference trajectory $(x_r(\cdot), u_r(\cdot))$, weight matrices V and W, control horizon $M \leq N$. Set $k = 0$.

(1) For $j = 0, \dots, M - 1$ do

  (1a) Measure state $x(k + j) \in X$ at time $k + j$.

M-multistep NMPC with re-optimization

M-multistep NMPC with re-optimization

(0) Input: preview horizon N, reference trajectory $(x_r(\cdot), u_r(\cdot))$, weight matrices V and W, control horizon $M \leq N$. Set $k = 0$.

(1) For $j = 0, \ldots, M - 1$ do

   (1a) Measure state $x(k + j) \in X$ at time $k + j$.

   (1b) Solve OCP$(k + j, x(k + j), N - j)$ on time horizon $[k + j, k + N]$. Let $\hat{u}(k + j), \ldots, \hat{u}(k + N - 1)$ be the optimal control.

M-multistep NMPC with re-optimization

M-multistep NMPC with re-optimization

(0) Input: preview horizon N, reference trajectory $(x_r(\cdot), u_r(\cdot))$, weight matrices V and W, control horizon $M \leq N$. Set $k = 0$.

(1) For $j = 0, \dots, M-1$ do

(1a) Measure state $x(k+j) \in X$ at time $k+j$.

(1b) Solve OCP$(k+j, x(k+j), N-j)$ on time horizon $[k+j, k+N]$. Let $\hat{u}(k+j), \dots, \hat{u}(k+N-1)$ be the optimal control.

(1c) Define the feedback control

$$\boldsymbol{\mu}_{N,M}(k+j, x(k+j)) := \hat{u}(k+j)$$

and apply it

$$x(k+j+1) = f(x(k+j), \boldsymbol{\mu}_{N,M}(k+j, x(k+j))).$$

M-multistep NMPC with re-optimization

M-multistep NMPC with re-optimization

(0) Input: preview horizon N, reference trajectory $(x_r(\cdot), u_r(\cdot))$, weight matrices V and W, control horizon $M \leq N$. Set $k = 0$.

(1) For $j = 0, \ldots, M-1$ do

(1a) Measure state $x(k+j) \in X$ at time $k+j$.

(1b) Solve OCP$(k+j, x(k+j), N-j)$ on time horizon $[k+j, k+N]$. Let $\hat{u}(k+j), \ldots, \hat{u}(k+N-1)$ be the optimal control.

(1c) Define the feedback control

$$\mu_{N,M}(k+j, x(k+j)) := \hat{u}(k+j)$$

and apply it

$$x(k+j+1) = f(x(k+j), \mu_{N,M}(k+j, x(k+j))).$$

(2) Set $k \leftarrow k+M$ and go to (1).

M-multistep NMPC with re-optimization

M-multistep NMPC with re-optimization

(0) Input: preview horizon N, reference trajectory $(x_r(\cdot), u_r(\cdot))$, weight matrices V and W, control horizon $M \leq N$. Set $k = 0$.

(1) For $j = 0, \ldots, M-1$ do

(1a) Measure state $x(k+j) \in X$ at time $k+j$.

(1b) Solve OCP$(k+j, x(k+j), N-j)$ on time horizon $[k+j, k+N]$. Let $\hat{u}(k+j), \ldots, \hat{u}(k+N-1)$ be the optimal control.

(1c) Define the feedback control

$$\mu_{N,M}(k+j, x(k+j)) := \hat{u}(k+j)$$

and apply it

$$x(k+j+1) = f(x(k+j), \mu_{N,M}(k+j, x(k+j))).$$

(2) Set $k \leftarrow k + M$ and go to (1).

Re-optimization is done on a reduced time horizon! Good initial guess available!

M-multistep NMPC with Sensitivity Update

M-multistep NMPC with Sensitivity Update

(0) Input: N, $M \leq N$, reference $(x_r(\cdot), u_r(\cdot))$, matrices V and W. Set $k = 0$.

M-multistep NMPC with Sensitivity Update

M-multistep NMPC with Sensitivity Update

(0) Input: N, $M \leq N$, reference $(x_r(\cdot), u_r(\cdot))$, matrices V and W. Set $k = 0$.

(1) Measure $x(k) \in X$ and solve OCP$(k, x(k), N)$. Solution: $(\hat{x}(\cdot), \hat{u}(\cdot))$.

M-multistep NMPC with Sensitivity Update

M-multistep NMPC with Sensitivity Update

(0) Input: N, $M \leq N$, reference $(x_r(\cdot), u_r(\cdot))$, matrices V and W. Set $k = 0$.

(1) Measure $x(k) \in X$ and solve OCP$(k, x(k), N)$. Solution: $(\hat{x}(\cdot), \hat{u}(\cdot))$.

(2) Perform in parallel:

M-multistep NMPC with Sensitivity Update

M-multistep NMPC with Sensitivity Update

(0) Input: N, $M \leq N$, reference $(x_r(\cdot), u_r(\cdot))$, matrices V and W. Set $k = 0$.

(1) Measure $x(k) \in X$ and solve OCP$(k, x(k), N)$. Solution: $(\hat{x}(\cdot), \hat{u}(\cdot))$.

(2) Perform in parallel:

   (2a) Apply feedback control $\boldsymbol{\mu}_{N,M}(k, x(k)) := \hat{u}(k)$:

$$x(k+1) = f(x(k), \boldsymbol{\mu}_{N,M}(k, x(k))).$$

M-multistep NMPC with Sensitivity Update

M-multistep NMPC with Sensitivity Update

(0) Input: N, $M \leq N$, reference $(x_r(\cdot), u_r(\cdot))$, matrices V and W. Set $k = 0$.

(1) Measure $x(k) \in X$ and solve OCP$(k, x(k), N)$. Solution: $(\hat{x}(\cdot), \hat{u}(\cdot))$.

(2) Perform in parallel:

(2a) Apply feedback control $\mu_{N,M}(k, x(k)) := \hat{u}(k)$:

$$x(k+1) = f(x(k), \mu_{N,M}(k, x(k))).$$

(2b) For $j = 1, \ldots, M-1$ compute sensitivities $u_j^*(k+\ell)(\hat{p}_j)$, $\ell = j, \ldots, N-1$, of OCP$(k+j, \hat{x}(k+j), N-j)$ w.r.t. $\hat{p}_j := \hat{x}(k+j)$. Let

$$S_j := u_j^*(k+j)'(\hat{p}_j) \qquad \text{(= sensitivity of } \hat{u}(k+j) \text{ w.r.t. } \hat{p}_j\text{)}$$

M-multistep NMPC with Sensitivity Update

M-multistep NMPC with Sensitivity Update

(0) Input: N, $M \leq N$, reference $(x_r(\cdot), u_r(\cdot))$, matrices V and W. Set $k = 0$.

(1) Measure $x(k) \in X$ and solve OCP$(k, x(k), N)$. Solution: $(\hat{x}(\cdot), \hat{u}(\cdot))$.

(2) Perform in parallel:

(2a) Apply feedback control $\boldsymbol{\mu}_{N,M}(k, x(k)) := \hat{u}(k)$:

$$x(k+1) = f(x(k), \boldsymbol{\mu}_{N,M}(k, x(k))).$$

(2b) For $j = 1, \ldots, M-1$ compute sensitivities $u_j^*(k+\ell)(\hat{p}_j)$, $\ell = j, \ldots, N-1$, of OCP$(k+j, \hat{x}(k+j), N-j)$ w.r.t. $\hat{p}_j := \hat{x}(k+j)$. Let

$$S_j := u_j^*(k+j)'(\hat{p}_j) \qquad (= \text{sensitivity of } \hat{u}(k+j) \text{ w.r.t. } \hat{p}_j)$$

(3) For $j = 1, \ldots, M-1$ do

M-multistep NMPC with Sensitivity Update

M-multistep NMPC with Sensitivity Update

(0) Input: N, $M \leq N$, reference $(x_r(\cdot), u_r(\cdot))$, matrices V and W. Set $k = 0$.

(1) Measure $x(k) \in X$ and solve OCP$(k, x(k), N)$. Solution: $(\hat{x}(\cdot), \hat{u}(\cdot))$.

(2) Perform in parallel:

(2a) Apply feedback control $\boldsymbol{\mu}_{N,M}(k, x(k)) := \hat{u}(k)$:

$$x(k+1) = f(x(k), \boldsymbol{\mu}_{N,M}(k, x(k))).$$

(2b) For $j = 1, \ldots, M-1$ compute sensitivities $u_j^*(k+\ell)(\hat{p}_j)$, $\ell = j, \ldots, N-1$, of OCP$(k+j, \hat{x}(k+j), N-j)$ w.r.t. $\hat{p}_j := \hat{x}(k+j)$. Let

$$S_j := u_j^*(k+j)'(\hat{p}_j) \qquad (= \text{sensitivity of } \hat{u}(k+j) \text{ w.r.t. } \hat{p}_j)$$

(3) For $j = 1, \ldots, M-1$ do

(3a) Measure state $x(k+j) \in X$ at time $k+j$.

M-multistep NMPC with Sensitivity Update

M-multistep NMPC with Sensitivity Update

(0) Input: N, $M \leq N$, reference $(x_r(\cdot), u_r(\cdot))$, matrices V and W. Set $k = 0$.

(1) Measure $x(k) \in X$ and solve OCP$(k, x(k), N)$. Solution: $(\hat{x}(\cdot), \hat{u}(\cdot))$.

(2) Perform in parallel:

(2a) Apply feedback control $\boldsymbol{\mu}_{N,M}(k, x(k)) := \hat{u}(k)$:

$$x(k+1) = f(x(k), \boldsymbol{\mu}_{N,M}(k, x(k))).$$

(2b) For $j = 1, \ldots, M-1$ compute sensitivities $u_j^*(k+\ell)(\hat{p}_j)$, $\ell = j, \ldots, N-1$, of OCP$(k+j, \hat{x}(k+j), N-j)$ w.r.t. $\hat{p}_j := \hat{x}(k+j)$. Let

$$S_j := u_j^*(k+j)'(\hat{p}_j) \qquad (= \text{sensitivity of } \hat{u}(k+j) \text{ w.r.t. } \hat{p}_j)$$

(3) For $j = 1, \ldots, M-1$ do

(3a) Measure state $x(k+j) \in X$ at time $k+j$.
(3b) Define the feedback control

$$\boldsymbol{\mu}_{N,M}(k+j, x(k+j)) := \hat{u}(k+j) + S_j \cdot (x(k+j) - \hat{x}(k+j))$$

and apply it

$$x(k+j+1) = f(x(k+j), \boldsymbol{\mu}_{N,M}(k+j, x(k+j))).$$

M-multistep NMPC with Sensitivity Update

M-multistep NMPC with Sensitivity Update

(0) Input: N, $M \leq N$, reference $(x_r(\cdot), u_r(\cdot))$, matrices V and W. Set $k = 0$.

(1) Measure $x(k) \in X$ and solve OCP$(k, x(k), N)$. Solution: $(\hat{x}(\cdot), \hat{u}(\cdot))$.

(2) Perform in parallel:

(2a) Apply feedback control $\boldsymbol{\mu}_{N,M}(k, x(k)) := \hat{u}(k)$:

$$x(k+1) = f(x(k), \boldsymbol{\mu}_{N,M}(k, x(k))).$$

(2b) For $j = 1, \ldots, M-1$ compute sensitivities $u_j^*(k+\ell)(\hat{p}_j), \ell = j, \ldots, N-1$, of OCP$(k+j, \hat{x}(k+j), N-j)$ w.r.t. $\hat{p}_j := \hat{x}(k+j)$. Let

$$S_j := u_j^*(k+j)'(\hat{p}_j) \qquad (= \text{sensitivity of } \hat{u}(k+j) \text{ w.r.t. } \hat{p}_j)$$

(3) For $j = 1, \ldots, M-1$ do

(3a) Measure state $x(k+j) \in X$ at time $k+j$.
(3b) Define the feedback control

$$\boldsymbol{\mu}_{N,M}(k+j, x(k+j)) := \hat{u}(k+j) + S_j \cdot (x(k+j) - \hat{x}(k+j))$$

and apply it

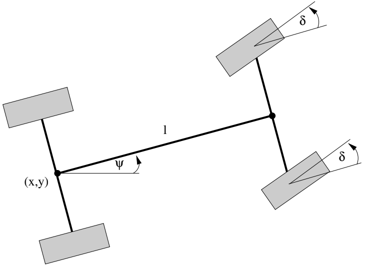
$$x(k+j+1) = f(x(k+j), \boldsymbol{\mu}_{N,M}(k+j, x(k+j))).$$

(4) Set $k \leftarrow k + M$ and go to (1).

Example: Tracking a Raceline

Kinematic car model

$$
\begin{aligned}
x'(t) &= v(t)\cos\psi(t), & x(0) &= x_0,\\
y'(t) &= v(t)\sin\psi(t), & y(0) &= y_0,\\
\psi'(t) &= \frac{v(t)}{\ell}\tan\delta(t), & \psi(0) &= \psi_0,\\
v'(t) &= u_1(t), & v(0) &= v_0,\\
\delta'(t) &= u_2(t), & \delta(0) &= \delta_0.
\end{aligned}
$$

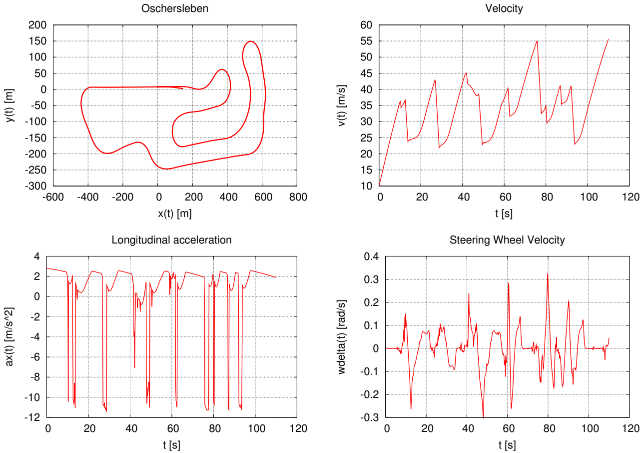


Notation:

δ	steering angle
v	velocity
ψ	yaw angle
ℓ	distance front to rear axle
(x, y)	reference point

Example: Tracking a Raceline

Reference trajectory (racetrack in Oschersleben):



Example: Tracking a Raceline

Objective function:

$$\int_0^T \alpha_1 \left\| \begin{pmatrix} x(t) - x_r(t) \\ y(t) - y_r(t) \end{pmatrix} \right\|^2 + \alpha_2 (v(t) - v_r(t))^2 + \alpha_3 \left\| \begin{pmatrix} u_1(t) - u_{1,r}(t) \\ u_2(t) - u_{2,r}(t) \end{pmatrix} \right\|^2 dt$$

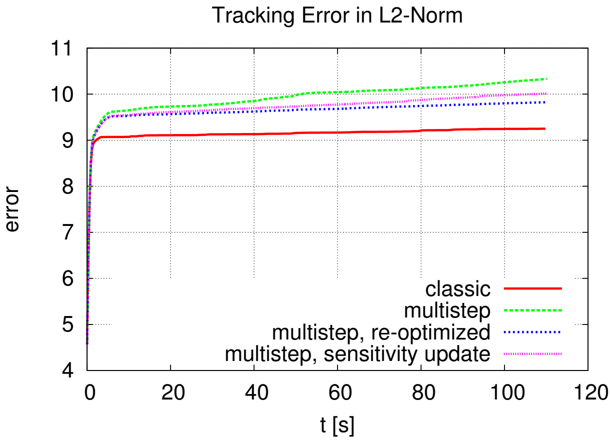
$$(\alpha_1 = 1, \alpha_2 = 10^{-1}, \alpha_3 = 10^{-3})$$

Parameters:

- $(x_0, y_0, \psi_0, v_0, \delta_0) = (0\ [m], 0\ [m], 0\ [rad], 10\ [m/s], 0\ [rad])$
- preview horizon of $T = 3\ [s]$, $N = 11$ grid points (i.e. a step-size of $h = 0.3\ [s]$)
- control horizon $M = 3$
- pertubations of position and velocity by equally distributed noise in the range $[-0.05, 0.05]$; initial perturbation in y-position of 8.3 $[m]$
- control bounds $u_1 \in [-12, 3]\ [m/s^2]$ and $u_2 \in [-0.5, 0.5]\ [rad/s]$
- state constraints $v \in [0, 60]\ [m/s]$ and $\delta \in [-0.5, 0.5]\ [rad]$
- $\ell = 4\ [m]$

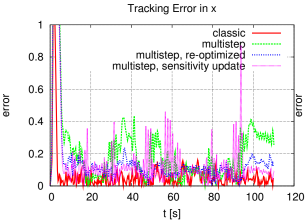
Example: Tracking a Raceline

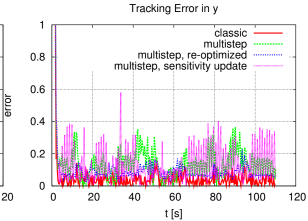
Tracking error $\|(x - x_r, y - y_r, v - v_r)\|_{L_2((0,t_f))}$: (initial y-deviation of 8.3 [m])

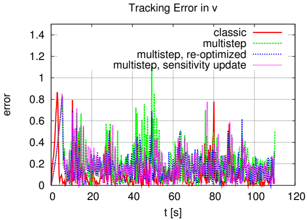


Example: Tracking a Raceline

Errors in the (x,y)-position and the velocity:







All schemes are able to track the reference solution at a high precision. Recall that

Contents

References

Very many contributions on MPC (linear or nonlinear) exist.

The state-of-the-art of MPC with a rigorous mathematical analysis can be found in:

[1] L. Grüne, J. Pannek.
Nonlinear Model Predictive Control – Theory and Algorithms.
2nd Edition, Springer, 2017

[2] J. B. Rawlings, D. Q. Mayne, M. Diehl.
Model Predictive Control: Theory, Computation, and Design.
2nd Edition, Nob Hill Publishing, Madison, 2018.

NMPC Algorithm Revisited

NMPC Algorithm (Basic Version)

Init: $n = 0$, $N \in \mathbb{N}$.

(0) Measure $x(n) \in X$.

(1) Set $x_0 = x(n)$ and solve $DOCP(x_0, N)$:

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{k=0}^{N-1} \ell(x(k), u(k)) \\
\text{s.t.} \quad & x(0) = x_0, \\
& x(k+1) = f(x(k), u(k)) && (k = 0, \dots, N-1) \\
& x(k) \in X && (k = 0, \dots, N) \\
& u(k) \in U && (k = 0, \dots, N-1)
\end{aligned}
$$

Let $u^*(\cdot)$ denote the optimal control.

(2) Set $\mu_N(n, x(n)) = u^*(0) \in U$. Set $n \leftarrow n + 1$ and go to (0).

Note: The discrete time is denoted by n. $DOCP(x_0, N)$ does not explicitly depend on n and thus can be considered on the discrete time interval from 0 to N instead of n to $n + N$.

NMPC Algorithm Revisited

General assumptions:

- An equilibrium solution $(x^*, u^*) \in X \times U$ with $x^* = f(x^*, u^*)$ exists.
- $\ell(x^*, u^*) = 0$ and $\ell(x, u) > 0$ for all $x \in X$, $u \in U$, $x \neq x^*$.
  $\rightsquigarrow$ satisfied for, e.g., tracking costs
- Viability: For each $x \in X$ there exists $u \in U$ with $f(x, u) \in X$.
  $\rightsquigarrow$ this is a crucial assumption! Not always satisfied!
- Existence of minimizer: There exists an optimal solution $u^*(\cdot)$ for $DOCP(x_0, N)$ for every $x_0 \in X$ and $N \in \mathbb{N}$.

Some Definitions

On finite time horizons ...

- Let $x_u(k, x_0)$, $k = 0, 1, 2, \ldots$, denote the solution of the dynamic system for given control sequence $u = u(\cdot)$ and initial value x_0.
- For $N \in \mathbb{N}$ and control input $u(k)$, $k = 0, \ldots, N-1$, finite horizon costs:

$$J_N(x_0, u(\cdot)) := \sum_{k=0}^{N-1} \ell(x_u(k, x_0), u(k))$$

- Finite horizon value function:

$$V_N(x_0) := \inf_{u(\cdot) \in U^N} J_N(x_0, u(\cdot))$$

with feasible control set

$$U^N := \{u : \{0, \ldots, N-1\} \longrightarrow U \mid x_u(k+1, x_0) \in X \ \forall k = 0, \ldots, N-1\}$$

Some Definitions

Similarly on infinite time horizons ...

- For control sequence $u(k)$, $k = 0, 1, 2, \ldots$, infinite horizon costs:

$$J_\infty(x_0, u(\cdot)) := \sum_{k=0}^{\infty} \ell(x_u(k, x_0), u(k))$$

- Infinite horizon value function:

$$V_\infty(x_0) := \inf_{u(\cdot)\in U^\infty} J_\infty(x_0, u(\cdot))$$

with feasible control set

$$U^\infty := \{u : \mathbb{N}_0 \longrightarrow U \mid x_u(k+1, x_0) \in X \ \forall k \in \mathbb{N}_0\}$$

Clear for non-negative stage costs ℓ: $V_{N-1}(x) \leq V_N(x) \leq V_\infty(x)$ for all $x \in X$, $N \in \mathbb{N}$.

Some Definitions

Likewise for the feedback law ...

- For a feedback law $\mu : \mathbb{N}_0 \times X \longrightarrow U$ let

$$x_\mu(k, x_0) \qquad (k = 0, 1, \ldots)$$

denote the solution of the closed-loop system

$$\begin{aligned} x(0) &= x_0 \\ x(k+1) &= f(x(k), \mu(k, x(k))) \qquad (k = 0, 1, 2, \ldots) \end{aligned}$$

- For $N \in \mathbb{N}$ and feedback law μ, finite horizon closed-loop costs:

$$J_N^{cl}(x_0, \mu) := \sum_{k=0}^{N-1} \ell(x_\mu(k, x_0), \mu(k, x_\mu(k, x_0)))$$

- For feedback law μ, infinite horizon closed-loop costs:

$$J_\infty^{cl}(x_0, \mu) = \sum_{k=0}^{\infty} \ell(x_\mu(k, x_0), \mu(k, x_\mu(k, x_0)))$$

Motivation of NMPC

Ultimate goal:

Find optimal feedback law $\mu : \mathbb{N}_0 \times X \longrightarrow U$ such that $J_\infty^{cl}(x_0, \mu) = V_\infty(x_0)$!

This is usually computationalle intractable!

Idea:

Approximate $J_\infty(x_0, u(\cdot))$ by $J_N(x_0, u(\cdot))$ and $V_\infty(x_0)$ by $V_N(x_0)$.

Questions:

- What is the relation between $V_\infty(x_0)$ and $V_N(x_0)$?
- How good is $J_\infty^{cl}(x_0, \mu_N)$ compared to $V_\infty(x_0)$?
- Under which conditions is the NMPC feedback law μ_N (asymptotically) stable?

Asymptotic Stability

- A function $\rho : [0, \infty) \longrightarrow [0, \infty)$ is a $\mathcal{K}$-function, if it is continuous, strictly increasing, and $\rho(0) = 0$.
- A function $\beta : [0, \infty) \times [0, \infty) \longrightarrow [0, \infty)$ is a $\mathcal{KL}$-function, if it is continuous and
  - $\beta(r, \cdot)$ is decreasing for every $r \geq 0$,
  - $\lim_{t \to \infty} \beta(r, t) = 0$ for every $r \geq 0$,
  - $\beta(\cdot, t)$ is a $\mathcal{K}$-function for every $t \geq 0$.

Definition (Asymptotic Stability)

An equilibrium $x^* \in X$ is *asymptotically stable for the closed loop system*, if there exists a $\mathcal{KL}$-function β with

$$\|x_\mu(k, x_0) - x^*\| \leq \beta(\|x_0 - x^*\|, k).$$

We say: The feedback law μ asymptotically stabilizes x^*.

Asymptotic Stability

How to ensure asymptotic stability?

- Option 1: Add terminal constraint $x(k+N) = x^*$ in NMPC! (x^*: equilibrium state)
- Option 2: Add terminal cost in objective function of NMPC!
- Option 3: Choose sufficiently large preview horizon N!

Contents

NMPC Algorithm with Terminal Constraint

NMPC Algorithm with Terminal Constraint

Init: $n = 0$, $N \in \mathbb{N}$.

(0) Measure $x(n) \in X$.

(1) Set $x_0 = x(n)$ and solve $DOCP_{TC}(x_0, N)$:

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{k=0}^{N-1} \ell(x(k), u(k)) \\
\text{s.t.} \quad & x(0) = x_0, \\
& x(k+1) = f(x(k), u(k)) && (k = 0, \ldots, N-1) \\
& x(k) \in X && (k = 0, \ldots, N) \\
& u(k) \in U && (k = 0, \ldots, N-1) \\
& x(k+N) = x^*
\end{aligned}
$$

Let $u^*(\cdot)$ denote the optimal control.

(2) Set $\mu_N(n, x(n)) = u^*(0) \in U$. Set $n \leftarrow n+1$ and go to (0).

NMPC Algorithm with Terminal Constraint

Stability Theorem with Terminal Constraint [Grüne/Pannek, Thm. 5.5]

Assume:

- $x^* \in X$ is an equilibrium point, i.e. there exists $u^* \in U$ with $x^* = f(x^*, u^*)$.
- $\ell(x^*, u^*) = 0$, $\ell(x, u) \geq 0$ for all $(x, u) \in X \times U$
- Let $\mathcal{K}_\infty$-functions $\alpha_1, \alpha_2, \alpha_3$ exist with

$$\begin{aligned} \alpha_1(\|x - x^*\|) &\leq V_N(x) \leq \alpha_2(\|x - x^*\|) \\ \alpha_3(\|x - x^*\|) &\leq \ell(x, u) \end{aligned} \qquad \forall x \in X, u \in U$$

Then μ_N stabilizes x^* on X. Moreover, we have

$$J_\infty^{cl}(x, \mu_N) \leq V_N(x) \qquad \forall x \in X.$$

Big problem: Existence of NMPC solutions not guaranteed with terminal constraint!

Contents

NMPC Algorithm with Terminal Cost Term

NMPC Algorithm with Terminal Cost Term

Init: $n = 0$, $N \in \mathbb{N}$.

(0) Measure $x(n) \in X$.

(1) Set $x_0 = x(n)$ and solve $DOCP_{TCT}(x_0, N)$:

$$\begin{aligned} \text{Minimize} \quad & \sum_{k=0}^{N-1} \ell(x(k), u(k)) + F(x(N)) \\ \text{s.t.} \quad & x(0) = x_0, \\ & x(k+1) = f(x(k), u(k)) && (k = 0, \ldots, N-1) \\ & x(k) \in X && (k = 0, \ldots, N) \\ & u(k) \in U && (k = 0, \ldots, N-1) \end{aligned}$$

Let $u^*(\cdot)$ denote the optimal control.

(2) Set $\mu_N(n, x(n)) = u^*(0) \in U$. Set $n \leftarrow n + 1$ and go to (0).

NMPC Algorithm with Terminal Cost Term

Stability Theorem with Terminal Cost [Grüne/Pannek, Thm. 5.13]

Assume:

- x^* is an equilibrium point, i.e. there exists $u^* \in U$ with $x^* = f(x^*, u^*)$.
- Lyapunov terminal cost: $F : X \longrightarrow [0, \infty)$ and for each $x \in X$ there exists $u \in U$ with $F(f(x, u)) + \ell(x, u) \leq F(x)$.
- Let $\mathcal{K}_\infty$-functions $\alpha_1, \alpha_2, \alpha_3$ exist with

$$\begin{aligned} \alpha_1(\|x - x^*\|) &\leq V_N(x) \leq \alpha_2(\|x - x^*\|) \\ \alpha_3(\|x - x^*\|) &\leq \ell(x, u) \end{aligned} \qquad \forall x \in X, u \in U$$

Then μ_N stabilizes x^* on X. Moreover, we have

$$J_\infty^{cl}(x, \mu_N) \leq V_N(x) \qquad \forall x \in X.$$

Contents

Nonlinear MPC without Terminal Conditions

Asymptotic Stability and Suboptimality Estimate [Grüne/Pannek, Thm. 4.11, Palma'2015, Prop. 2.1.3]

Let $V : X \longrightarrow [0, \infty)$ and $\mu : \mathbb{N}_0 \times X \longrightarrow X$ satisfy

$$V(x) \geq \alpha \ell(x, \mu(n, x)) + V(f(x, \mu(n, x)))$$

for some $\alpha \in (0, 1]$, all $n \in \mathbb{N}_0$, and all $x \in X$.

Then:

$$V_\infty(x) \leq J_\infty^{cl}(x, \mu) \leq V(x)/\alpha \qquad \forall x \in X.$$

If moreover there exist $\mathcal{K}_\infty$-functions $\alpha_1, \alpha_2, \alpha_3$ with

$$\alpha_1(\|x - x^*\|) \leq V(x) \leq \alpha_2(\|x - x^*\|) \quad \text{and} \quad \alpha_3(\|x - x^*\|) \leq \ell(x, u) \qquad \forall x \in X,$$

then μ asymptotically stabilizes x^* on X.

$\alpha \in (0, 1]$ is called index of suboptimality. It measures how well μ approximates the minimizer of J_∞.

NMPC without Terminal Conditions

Proof: Let $x \in X$ and $x_\mu(0) = x$. Viability yields $x_\mu(n) \in X$ for $n \in \mathbb{N}_0$. Moreover,

$$\begin{aligned} \alpha \ell(x_\mu(n), \mu(n, x_\mu(n))) &\leq V(x_\mu(n)) - V(f(x_\mu(n), \mu(n, x_\mu(n)))) \\ &= V(x_\mu(n)) - V(x_\mu(n+1)) \end{aligned}$$

Summing over n yields for arbitrary $K \in \mathbb{N}$ owing to the non-negativity of V:

$$\begin{aligned} \alpha \sum_{n=0}^{K-1} \ell(x_\mu(n), \mu(n, x_\mu(n))) &\leq V(x_\mu(0)) - V(x_\mu(n+K)) \\ &\leq V(x). \end{aligned}$$

The term on the left is monotonically non-decreasing and bounded by $V(x)$ for every K. For $K \to \infty$ it converges to $\alpha J_\infty^{cl}(x, \mu)$, which yields the first assertion.

For the second part one has to show that $V(x_\mu(n))$ is strictly decreasing as $n \to \infty$. The assertion then follows because $V(x_\mu(n)) \geq \alpha_1(\|V(x_\mu(n)) - x^*\|)$. □

Nonlinear MPC without Terminal Conditions

Assumptions:

(A1) Let there exist $\mathcal{K}_\infty$-functions α_3, α_4 with

$$\alpha_3(\|x - x^*\|) \leq \ell^*(x) \leq \alpha_4(\|x - x^*\|) \qquad \forall x \in X,$$

where $\ell^*(x) = \inf_{u \in U} \ell(x, u)$.

(A2) Let there exist a $\mathcal{K}_\infty$-function B_K such that

$$V_K(x) \leq B_K(\ell^*(x)) \qquad \forall x \in X, K \in \mathbb{N}.$$

(A3) Let $\alpha \in (0, 1]$ solve the optimization problem

$$\min_{\lambda_0,\ldots,\lambda_{N-1},\nu} \alpha = \frac{1}{\lambda_0}\left(\sum_{n=0}^{N-1} \lambda_n - \nu\right)$$

$$s.t. \sum_{n=k}^{N-1} \lambda_n \leq B_{N-k}(\lambda_k), \quad k = 0, \ldots, N-2,$$

$$\nu \leq \sum_{n=0}^{j-1} \lambda_{n+1} + B_{N-j}(\lambda_{j+1}), \qquad j = 0, \ldots, N-2,$$

$$\lambda_0 > 0, \lambda_1, \ldots, \lambda_{N-1}, \nu > 0.$$

Nonlinear MPC without Terminal Conditions

It can be shown that V_N of the NMPC satisfies the assumptions of the previous theorem under suitable assumptions.

Stability Theorem without Terminal Conditions [Grüne/Pannek, Thm. 6.20, 6.24, Palma'2015, Theorem 2.1.8]

- If (A2) and (A3) hold, then we have

$$V_N(x) \geq \alpha \ell(x, \mu_N(n, x)) + V_N(f(x, \mu_N(n, x))) \qquad \forall x \in X, n \in \mathbb{N}_0.$$

- If (A1), (A2), A(3) hold, then μ_N asymptotically stabilizes x^* on X and

$$V_\infty(x) \leq J_\infty^{cl}(x, \mu_N) \leq V_N(x)/\alpha \leq V_\infty(x)/\alpha \qquad \forall x \in X.$$

- If (A1), (A2) with linear $B_K(r) = \gamma_K r$ with $\gamma_\infty = \sup_{k \in \mathbb{N}} \gamma_k < \infty$ hold, then μ_N asymptotically stabilizes x^* on X, if N is sufficiently large. Moreover, for every $C > 1$ there exists $N_C > 0$ with

$$V_\infty(x) \leq J_\infty^{cl}(x, \mu_N) \leq C V_N(x) \leq C V_\infty(x) \qquad \forall x \in X, N \geq N_C.$$

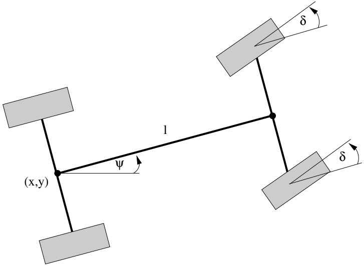
Contents

Contents

Example: NMPC on Narrow Road

Kinematic car model

$$
\begin{aligned}
x'(t) &= v(t)\cos\psi(t), & x(0) &= x_0,\\
y'(t) &= v(t)\sin\psi(t), & y(0) &= y_0,\\
\psi'(t) &= \frac{v(t)}{\ell}\tan\delta(t), & \psi(0) &= \psi_0,\\
v'(t) &= u_1(t), & v(0) &= v_0,\\
\delta'(t) &= u_2(t), & \delta(0) &= \delta_0.
\end{aligned}
$$



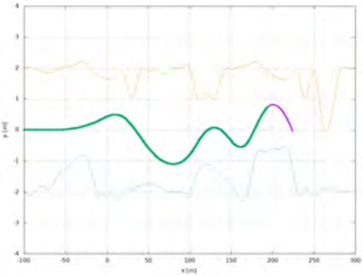
Notation:

δ	steering angle
v	velocity
ψ	yaw angle
ℓ	distance front to rear axle
(x, y)	reference point

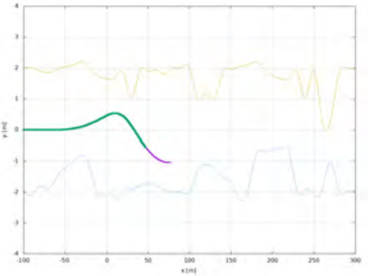
Example: NMPC on Narrow Road

state constraints on 4 corners of car, minimization of control effort:

narrow road

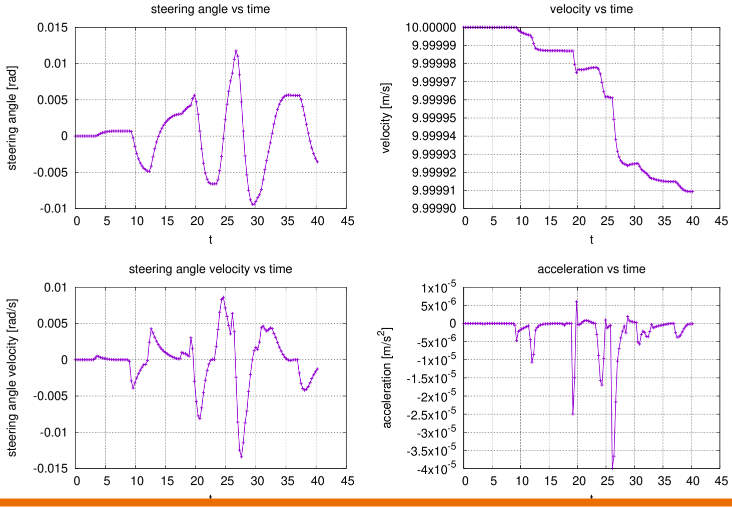


too narrow road

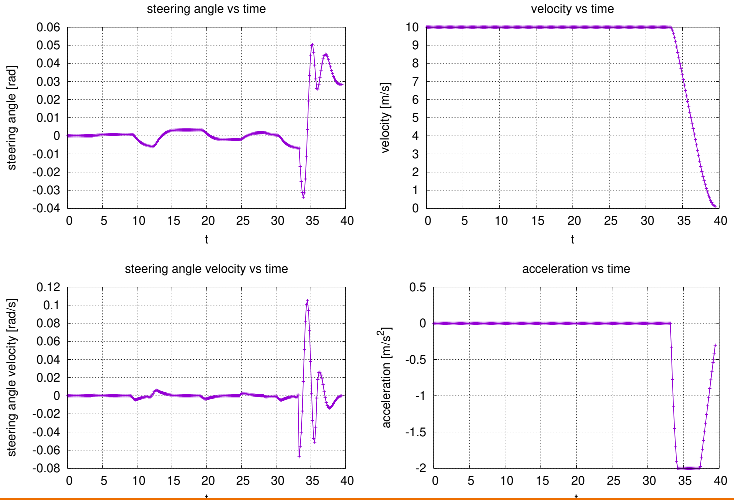


width 1.8 [m], $\ell = 4$ [m], $N = 10$, $M = 1$, preview 3 [s], $u_1 \in [-2, 2]$, $u_2 \in [-0.5, 0.5]$, $|\delta| \leq 45^\circ$

Example: NMPC on Narrow Road



Example: NMPC on Too Narrow Road

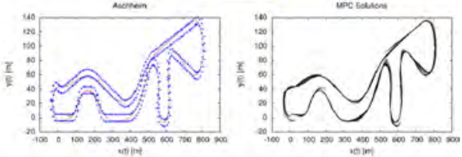


Contents

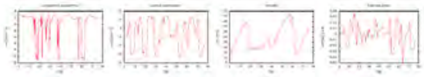
Research @ Engineering Mathematics

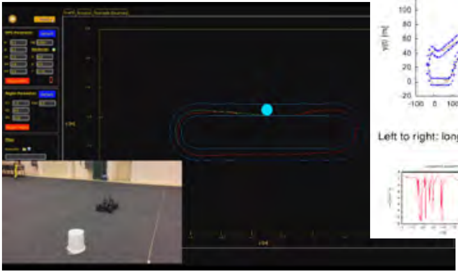
Application: Automatic Driving

- Modelling of an “optimal” driver (time minimal, fuel efficient)
- Consideration of track bounds and obstacles
- Online optimization



Left to right: longitudinal acceleration, lateral acceleration, velocity, slip angle



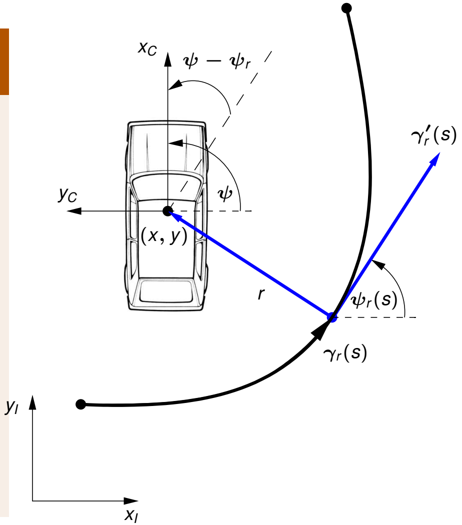


Nonlinear Kinematic Model

Motion in (s,r)-system along a reference curve

Given:

- reference curve $\gamma_r = (x_r, y_r)^\top$
- curvature κ_r



Nonlinear Kinematic Model

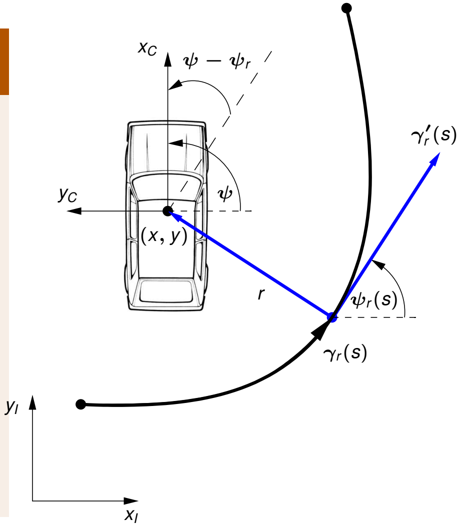
Motion in (s,r)-system along a reference curve

Given:

- reference curve $\gamma_r = (x_r, y_r)^\top$
- curvature κ_r

Motion in moving reference system aligned with γ_r:

$$\begin{aligned} s' &= \frac{v\cos(\psi - \psi_r)}{1 - r \cdot \kappa_r(s)} \\ r' &= v\sin(\psi - \psi_r) \\ \psi' &= v \cdot \kappa \\ \kappa' &= u \\ \psi_r' &= \kappa_r(s) \cdot s' \end{aligned}$$



Decoupling

Decoupling ...

Decoupling

Decoupling ...

Path Planning (yields parametrized curve w.r.t. arclength)

Minimize

$$-\alpha_1 s(L) + \alpha_2 \int_0^L \kappa(\ell)^2 d\ell + \alpha_3 \int_0^L u(\ell)^2 d\ell$$

s.t. dynamics with $v(t) \equiv 1$, initial conditions, and control/state constraints

$$(r, u, \kappa) \in [-r_{max}, r_{max}] \times [-u_{max}, u_{max}] \times [-\kappa_{max}, \kappa_{max}]$$

Decoupling

Decoupling ...

Path Planning (yields parametrized curve w.r.t. arclength)

Minimize

$$-\alpha_1 s(L) + \alpha_2 \int_0^L \kappa(\ell)^2 d\ell + \alpha_3 \int_0^L u(\ell)^2 d\ell$$

s.t. dynamics with $v(t) \equiv 1$, initial conditions, and control/state constraints

$$(r, u, \kappa) \in [-r_{max}, r_{max}] \times [-u_{max}, u_{max}] \times [-\kappa_{max}, \kappa_{max}]$$

and

Velocity Profile Generation

Find velocity profile $v(\ell)$ for $\ell \in [0, L]$ on computed path.

Decoupling

Decoupling ...

Path Planning (yields parametrized curve w.r.t. arclength)

Minimize

$$-\alpha_1 s(L) + \alpha_2 \int_0^L \kappa(\ell)^2 d\ell + \alpha_3 \int_0^L u(\ell)^2 d\ell$$

s.t. dynamics with $v(t) \equiv 1$, initial conditions, and control/state constraints

$$(r, u, \kappa) \in [-r_{max}, r_{max}] \times [-u_{max}, u_{max}] \times [-\kappa_{max}, \kappa_{max}]$$

and

Velocity Profile Generation

Find velocity profile $v(\ell)$ for $\ell \in [0, L]$ on computed path.

... increases robustness and flexibility.

Velocity Profile Generation

Given:

- path with curvature $\kappa(\ell)$, arclength parametrization $\ell \in [0, L]$

Multiobjective optimal control problem

Minimize

$$\alpha_1 \underbrace{\int_0^L \frac{1}{v(\ell)} d\ell}_{\text{final time}} + \alpha_2 \underbrace{\int_0^L u(\ell)^2 d\ell}_{\text{control effort}} + \alpha_3 \underbrace{a_{max}}_{\text{max. lateral acceleration}}$$

s.t.

$$v'(\ell) = \frac{u(\ell)}{v(\ell)} - c_0 - c_1 v(\ell) \qquad \text{(dynamics with friction and drag)}$$

$$|\kappa(\ell)| v(\ell)^2 \leq a_{max} \qquad \text{(lateral acceleration)}$$

$$u(\ell) \in [u_{min}, u_{max}] \qquad \text{(longitudinal acceleration)}$$

$$v(0) = v_0, \; v(L) = v_L$$

Velocity Profile Generation

Approaches:

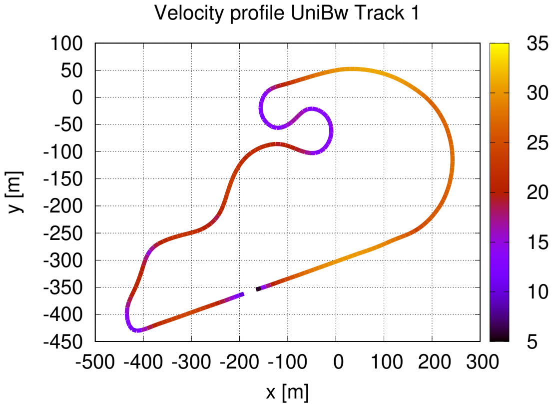
- semi-analytical solution for minimum-time problem

  [E. Bertolazzi, M. Frego: Semianalytical minimum-time solution for the optimal control of a vehicle subject to limited acceleration. Optim Control Appl Meth. 39, pp. 774–791, 2018]

- direct discretization methods

- dynamic programming (quick and robust)

Velocity Profile Generation



$(\alpha_1 = 0.1, \alpha_2 = 0.01, \alpha_3 = 0, [u_{min}, u_{max}] = [-10, 3.5], a_{max} = 5, L = 2100, c_0 = 0.001, c_1 = 0.0015, v_0 = 5, N = 201)$

Linear Kinematic Model for Tracking Tasks

Linearization at $\psi \approx \psi_r$ and $r \approx 0$:

$$r' = v(\psi - \psi_r), \quad \psi' = v \cdot \kappa, \quad \kappa' = u, \quad \psi_r' = v \cdot \kappa_r(s(t)), \quad s(t) = \int_0^t v(\tau)d\tau$$

Linear model (given velocity profile)

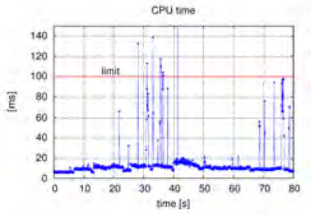
$$\underbrace{\begin{pmatrix} r \\ \psi \\ \kappa \\ \psi_r \end{pmatrix}'}_{=x'} = \underbrace{\begin{pmatrix} 0 & v & 0 & -v \\ 0 & 0 & v & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}}_{=A} \underbrace{\begin{pmatrix} r \\ \psi \\ \kappa \\ \psi_r \end{pmatrix}}_{=x} + \underbrace{\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}}_{=B} u + \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ v \cdot \kappa_r \end{pmatrix}}_{=d}$$

Observed states:

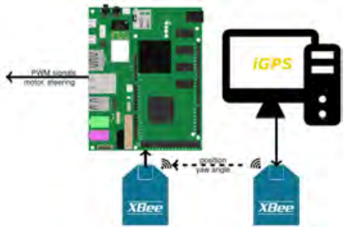
$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}}_{=y} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}}_{=C} \underbrace{\begin{pmatrix} r \\ \psi \\ \kappa \\ \psi_r \end{pmatrix}}_{=x}$$

Realization on Scale Cars

CPU times: ($N = 18$)



Control architecture:



Realization on Scale Cars

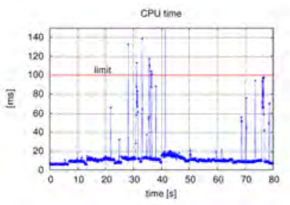
Details:

- simple car models
- curvilinear coordinates instead of Cartesian coordinates
- projection procedure from Cartesian measurements to curvilinear coordinates
- preview steering controller to track the NMPC solutions; PID controller for velocity tracking
- Hall sensors for velocity measurements
- time delays in iGPS system 100...200 ms
- multithreaded control architecture

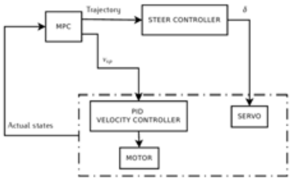
To be improved:

- Kalman filtering of position and velocity measurements
- sensor fusion (acceleration and gyro sensor, Hall sensor, iGPS)
- ...

CPU times: ($N = 18$)



Control architecture:

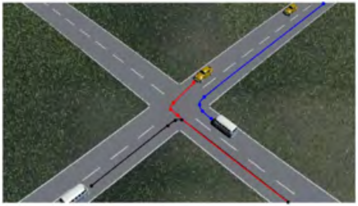


NMPC and Online-Optimization

Research @ Engineering Mathematics

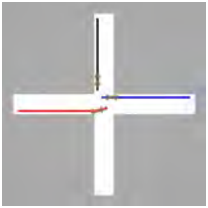
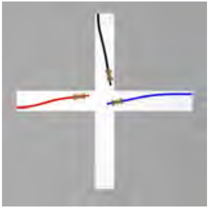
Application: Cooperative Automatic Driving

- Multiple vehicles communicate and exchange information with regard to positions etc.
- Individual goals of the vehicles, e.g. consumption, comfort, time.
- Consideration of constraints, e.g. road bounds, collision avoidance, velocity restrictions.
- Implementation with model-predictive control using state dependent hierarchies of the vehicles or generalized Nash equilibria

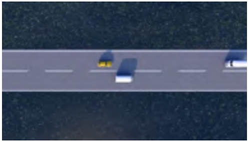


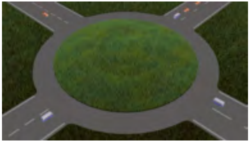
Car-to-car Communication & MPC

Simulation results: Nash equilibria / hierarchic control

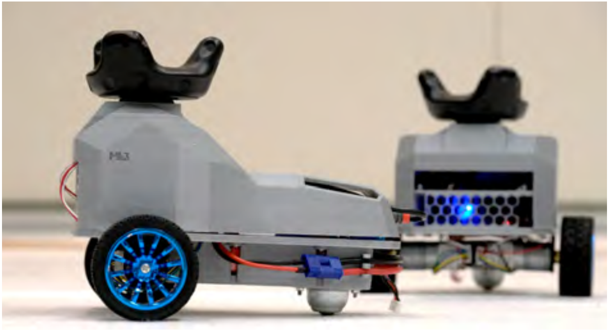




Takeover/crossing/roundabout:





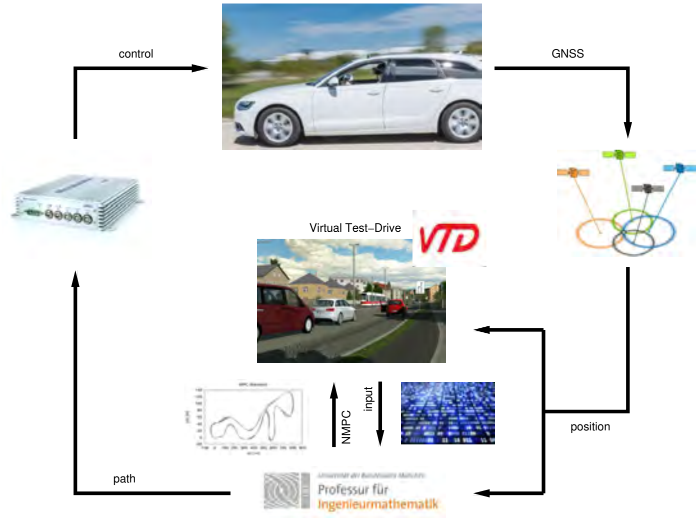
GNEP-MPC for Coordination of Interacting Vehicles



Automated Interconnected Vehicle-in-the-Loop (AN-VIL) @ Engineering Mathematics

- platform combining
  virtual reality & real driving & automated driving
- two experimental Audi A6 equipped with VTD, IMU, D-GPS
- versatile and safe tool in automated driving, cooperative driving, and human-machine interaction

Concept


control
GNSS
Virtual Test–Drive
VTD
NMPC
input
position
path
Professur für
Ingenieurmathematik

Testing Area @ UniBw M

Testing Area @ UniBw M

Research

Automated Driving

Cooperative Driving

Human-Machine-Interaction

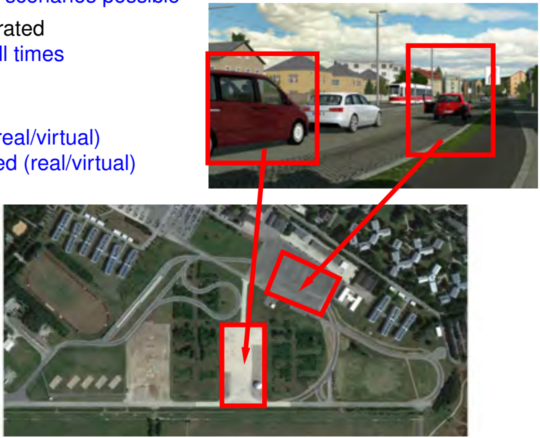
- path planning and tracking
- MPC / online optimization
- obstacle avoidance

- distributed control
- hierarchies vs Nash equilibria
- obstacle avoidance

- many user studies performed by Prof. Färber and Prof. Nitsch, LRT-11
- identification of comfort criteria

Vision

- driving in the same virtual scenario ...
  $\rightsquigarrow$ virtually dangerous scenarios possible
- ... but physically separated
  $\rightsquigarrow$ physically safe at all times
- interactions
  human – human
  human – automated (real/virtual)
  automated – automated (real/virtual)



Contents

Path Planning of a UAV

Motion in a flight corridor

- reference ground curve

$$\gamma_r(s) = \begin{pmatrix} x_r(s) \\ y_r(s) \end{pmatrix},$$

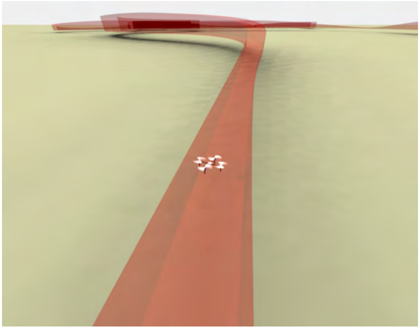
curvature κ_r, curve parameter s

- altitude bounds

$$z_{min}(s) \leq z(s) \leq z_{max}(s)$$

- width bounds

$$r_{min}(s) \leq r(s) \leq r_{max}(s)$$



[M. Burger, M. Gerdts: DAE Aspects in Vehicle Dynamics and Mobile Robotics, in Applications of Differential-Algebraic Equations: Examples and Benchmarks, Differential-Algebraic Equations Forum DAE-F, Eds. S. Campbell, A. Ilchmann, V. Mehrmann, T. Reis, Springer, pp. 37–80, 2019.]

Path Planning of a UAV

Motion in a flight corridor

$$
\begin{aligned}
s' &= \frac{v_{xy} \cdot \cos(\psi - \psi_r)}{1 - r \cdot \kappa_r(s)} \\
r' &= v_{xy} \cdot \sin(\psi - \psi_r) \\
z' &= v_z \\
m \cdot v_x' &= u_1 \cdot \cos\phi \cdot \sin\kappa - D_x \\
m \cdot v_y' &= -u_1 \cdot \sin\phi - D_y \\
m \cdot v_z' &= u_1 \cdot \cos(\phi) \cdot \cos(\kappa) - m \cdot g - D_z \\
\phi' &= \frac{u_2 - \phi}{\delta} \\
\kappa' &= \frac{u_3 - \kappa}{\delta}
\end{aligned}
$$

Notation:

- (s, r, z)=position in curvilinear coordinates
- u_1 = thrust
- u_2 = commanded roll angle
- u_3 = commanded pitch angle
- $v_{xy} = \sqrt{v_x^2 + v_y^2}$,
  $\psi = \arctan(v_y / v_x)$
- δ = delay factor

Path Planning of a UAV

Objective: (to be minimized)

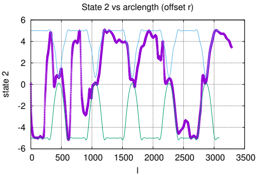
$$\underbrace{\int_0^L \frac{1}{v(\ell)}\, d\ell}_{\text{flight time}} + \underbrace{\int_0^L u_1(\ell)^2 + u_2(\ell)^2 + u_3(\ell)^2\, d\ell}_{\text{control effort}}$$

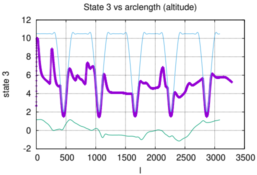
State and control constraints:

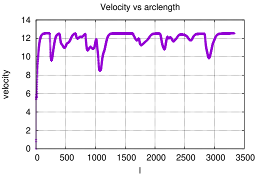
$z_{min}(s(\ell)) \leq z(\ell) \leq z_{max}(s(\ell))$	(altitude)
$r_{min}(s(\ell)) \leq r(\ell) \leq r_{max}(s(\ell))$	(offset)
$v_{min} \leq v \leq v_{max}$	(velocity)
$\lvert\phi\rvert \leq \phi_{max},\ \lvert\kappa\rvert \leq \kappa_{max}$	(angles)
$u_i \in [u_{i,min}, u_{i,max}],\ i = 1, 2, 3$	(controls)

NMPC Results Quadrocopter

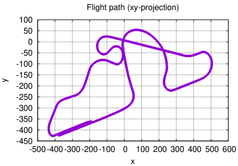
States: (offset r, altitude, velocity)

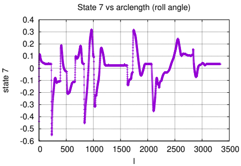


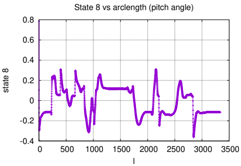




States: (xy-path, roll and pitch angle)

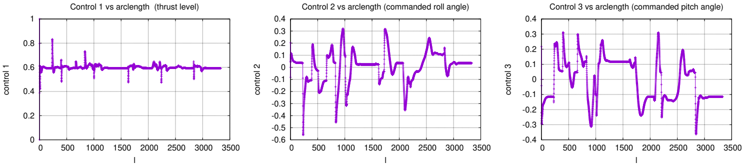






NMPC Results Quadrocopter

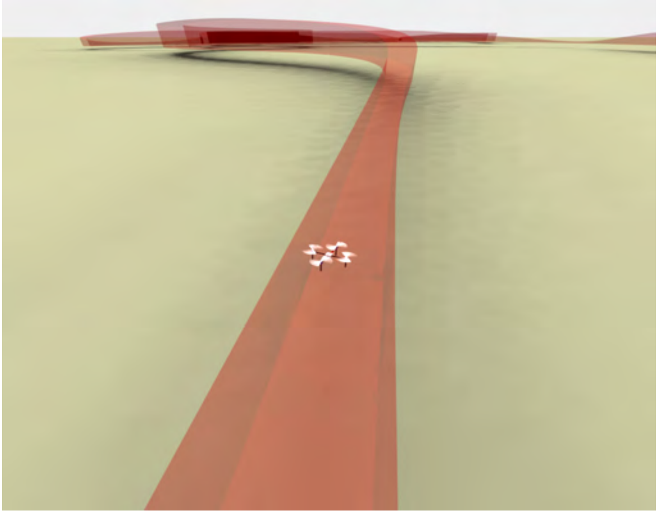
Controls: (thrust level, commanded roll and pitch)



$m = 3$ [kg], $\delta = 0.1$, $v_{max} = 15$ [m/s], $\phi_{max} = \kappa_{max} = 45^{\circ}$, $L = 20$ [m], $N = 30$, $T_{max} = 50$ [N]

- total flight time: 284.59 [s]
- CPU time: 460.015 [s] for 5001 OCPs
- CPU time per OCP: 0.09 [s]

NMPC Results Quadrocopter

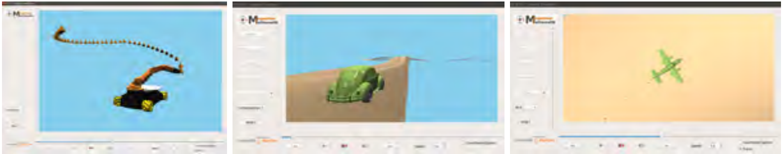


Contents

NMPC Results youBot

Contents

ROCS – Realtime Optimization and Control Software



(written in Qt3D/C++, export of animations)

One versatile tool for ...

- ... visualization of cars, robot, aircrafts, etc.
  $\rightsquigarrow$ visualization of precomputed trajectories using data files
- ... simulation
  $\rightsquigarrow$ online simulation with mathematical models
- ... online path planning and control
  $\rightsquigarrow$ optimal control and feedback control

Software OCPID-DAE1

OCPID-DAE1 – Optimal Control and Parameter IDentifaction with Differential-Algebraic-Equations of index 1

- direct multiple shooting discretization
- SQP method (non-monotone linesearch, filter, BFGS update, primal active-set QP solver)
- various integrators (Runge-Kutta, BDF methods, linearized Runge-Kutta methods)
- various control approximations (B-splines of order k)
- gradients by sensitivity differential equation
- sensitivity analysis and adjoint estimation
- extensions to adjoint gradient computation and mixed-integer optimal control problems
- parameter identification

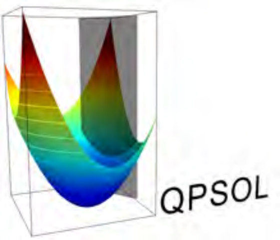
www.optimal-control.de

QP Solver

Methods:

- primal-dual interior point solver with Mehrotra predictor corrector step based on

  [E. M. Gertz, S. J. Wright: Object-oriented software for quadratic programming, ACM Transactions on Mathematical Software (TOMS), Volume 29 (1), 2003.]
- globalized nonsmooth Newton method



Linear algebra:

- MA57, MA48 (http://www.hsl.rl.ac.uk)
- SuperLU (http://crd.lbl.gov/∼xiaoye/SuperLU/)
- iterative solvers (CGNE,CGNR,CGS,BICGSTAB)

Features:

- automatic scaling (QP data, KKT system)
- iterative refinement for direct solvers
- constraint regularization mode
- warm start option for IP method

More Resources

Optimal control software:

- CasADI, ACADO: M. Diehl et al.; http://casadi.org; http://sourceforge.net/p/acado/
- NUDOCCCS: C. Büskens, University of Bremen
- SOCS: J. Betts, The Boeing Company, Seattle; http://www.boeing.com/boeing/phantom/socs/
- DIRCOL: O. von Stryk, TU Darmstadt; http://www.sim.informatik.tu-darmstadt.de/res/sw/dircol
- MUSCOD-II: H.G. Bock et al., IWR Heidelberg; http://www.iwr.uni-heidelberg.de/∼agbock/RESEARCH/muscod.php
- MISER: K.L. Teo et al., Curtin University, Perth; http://school.maths.uwa.edu.au/ les/miser/
- PSOPT: http://www.psopt.org/
- FALCON.m: https://www.fsd.lrg.tum.de/software/falcon-m/
- GPOPS-II: http://www.gpops2.com/
- ...

Optimization software:

- WORHP (sparse large-scale problems): C. Büskens/M. Gerdts, https://www.worhp.de
- NPSOL (dense problems), SNOPT (sparse large-scale problems): Stanford Business Software; http://www.sbsi-sol-optimize.com
- KNITRO (sparse large-scale problems): Ziena Optimization; http://www.ziena.com/knitro.htm
- IPOPT (sparse large-scale problems): A. Wächter: https://projects.coin-or.org/Ipopt
- filterSQP: R. Fletcher, S. Leyffer; http://www.mcs.anl.gov/ leyffer/solvers.html
- ooQP: M. Gertz, S. Wright; http://pages.cs.wisc.edu/ swright/ooqp/
- qpOASES: H.J. Ferreau, A. Potschka, C. Kirches; http://homes.esat.kuleuven.be/ optec/software/qpOASES/
- OSQP: B. Stellato, G. Banjac, P. Goulart, A. Bemporad, S. Boyd; https://osqp.org/
- ...

Links:

- Decision Tree for Optimization Software; http://plato.la.asu.edu/guide.html
- CUTEr: large collection of optimization test problems; http://www.cuter.rl.ac.uk/
- COPS: large-scale optimization test problems; http://www.mcs.anl.gov/∼more/cops/
- MINTOC: testcases for mixed-integer optimal control; http://mintoc.de/
- ...

Thanks for your Attention!

Questions?

Further information:

matthias.gerdts@unibw.de
www.unibw.de/ingmathe
www.optimal-control.de

Fotos: http://de.wikipedia.org/wiki/München
Magnus Manske (Panorama), Luidger (Theatinerkirche), Kurmis (Chin. Turm), Arad Mojtahedi (Olympiapark), Max-k (Deutsches Museum), Oliver Raupach (Friedensengel), Andreas Praefcke (Nationaltheater)