Simulation and Optimal Control using CasADi and acados

Katrin Baumgärtner

Systems Control and Optimization Laboratory (syscop)

workshop @ SPP2364 Doktorand:innenseminar

November 2023

universität freiburg

Intro

- Who has experience with `python`?

Intro

- Who has experience with `python`?
- Who has experience with `CasADi`?

Intro

- Who has experience with `python`?
- Who has experience with `CasADi`?

- Who models their system in terms of an ordinary differential equation (ODE)?
- Who models their system in terms of an differential algebraic equation (DAE)?
- Who models their systems using a neural network?

Intro

- Who has experience with `python`?
- Who has experience with `CasADi`?

- Who models their system in terms of an ordinary differential equation (ODE)?
- Who models their system in terms of an differential algebraic equation (DAE)?
- Who models their systems using a neural network?

- Who has installed the provided `docker`?

Workshop Outline

- Part 1: Nonlinear Optimization
- Part 2: Direct Optimal Control

Workshop Outline

- Part 1: Nonlinear Optimization using `CasADi`
- Part 2: Direct Optimal Control

Workshop Outline

- Part 1: Nonlinear Optimization using `CasADi`
- Part 2: Direct Optimal Control using `CasADi` and `acados`

Workshop Outline

- Part 1: Nonlinear Optimization using `CasADi`
- Part 2: Direct Optimal Control using `CasADi` and `acados`

Most of the theory part of this talk is based on slides by Armin Nurkanović.

Outline

Part 1: Nonlinear Optimization

1. Basic definitions
2. Conditions of optimality
3. Nonlinear programming algorithms
4. Nonlinear optimization with `CasADi`

Part 2: Direct Optimal Control

Outline

Part 1: Nonlinear Optimization

1. Basic definitions
2. Conditions of optimality
3. Nonlinear programming algorithms
4. Nonlinear optimization with `CasADi`

Part 2: Direct Optimal Control

What is an optimization problem?

Minimize (or maximize) an objective function $F(w)$ depending on decision variables w subject to equality and/or inequality constraints.

What is an optimization problem?

Minimize (or maximize) an objective function $F(w)$ depending on decision variables w subject to equality and/or inequality constraints.

An optimization problem

$$\min_{w} \; F(w) \tag{1a}$$
$$\text{s.t.} \quad G(w) = 0 \tag{1b}$$
$$H(w) \geq 0 \tag{1c}$$

Terminology

- w - decision variable
- F: objective/cost function
- G, H: equality and inequality constraint functions

What is an optimization problem?

Minimize (or maximize) an objective function $F(w)$ depending on decision variables w subject to equality and/or inequality constraints.

An optimization problem

$$\min_{w} \; F(w) \tag{1a}$$

$$\text{s.t.} \quad G(w) = 0 \tag{1b}$$

$$H(w) \geq 0 \tag{1c}$$

Terminology

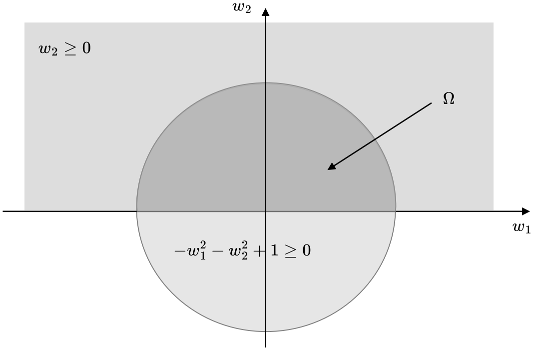
- w - decision variable
- F: objective/cost function
- G, H: equality and inequality constraint functions

- Only in few special cases a closed form solution exist
- Use an iterative algorithm to find solution

Basic definitions: the feasible set

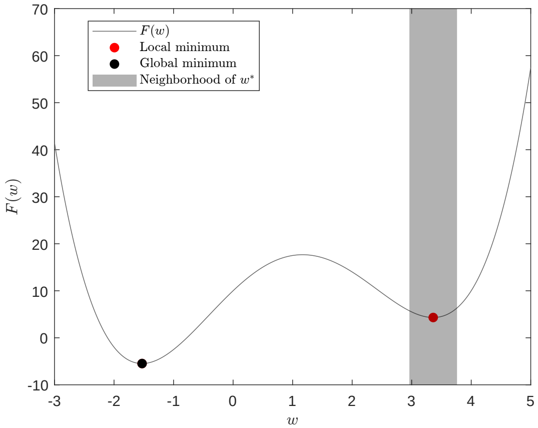
Definition

The feasible set of the optimization problem (1) is defined as $\Omega = \{w \in \mathbb{R}^n \mid G(w) = 0, H(w) \geq 0\}$. A point $w \in \Omega$ is is called a feasible point.



The feasible set is the intersection of the two grey areas (halfspace and circle)

Basic definitions: local and global minimizer



The value $F(w^*)$ at a local/global minimizer w^* is called local/global minimum.

Convex optimization problems

A convex optimization problem

$$\begin{aligned} \min_{w} \quad & F(w) \\ \text{s.t.} \quad & G(w) = 0 \\ & H(w) \geq 0 \end{aligned}$$

An optimization problem is convex if the objective function F is convex and the feasible set Ω is convex.

- Example: convex objective and linear equalities and linear inequalities.

- A locally optimal solution is globally optimal!
- First order conditions are necessary and sufficient (we come back to this)

Some classifications of optimization problems

Optimization problems can be:

- unconstrained ($\Omega = \mathbb{R}^n$) or constrained ($\Omega \subset \mathbb{R}^n$)
- convex or nonconvex
- linear or nonlinear
- finite or infinite dimensional

Some classifications of optimization problems

Optimization problems can be:

- unconstrained ($\Omega = \mathbb{R}^n$) or constrained ($\Omega \subset \mathbb{R}^n$)
- convex or nonconvex
- linear or nonlinear
- finite or infinite dimensional

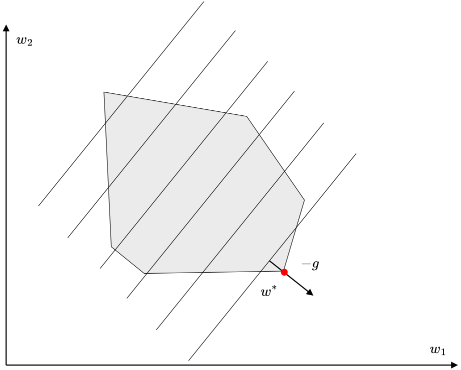
Three important **classes** of optimization problems:

- Linear Program (LP)
- Quadratic Program (QP)
- Nonlinear Program (NLP)

Class 1: Linear Programming (LP)

Linear program

$$\min_{w} \; g^\top w$$
$$\text{s.t.} \quad Aw - b = 0$$
$$\quad\quad Cw - d \geq 0$$

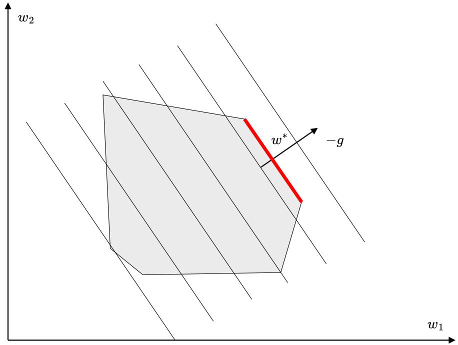


- convex optimization problem
- 1947: simplex method by Dantzig, 1984: polynomial time interior-point method by Karmarkar
- if not unbounded, the solution is always at edge or vertex of the feasible set
- today very mature and reliable

Class 1: Linear Programming (LP)

Linear program

$$\begin{aligned} \min_{w}\ & g^\top w \\ \text{s.t.}\quad & Aw - b = 0 \\ & Cw - d \geq 0 \end{aligned}$$

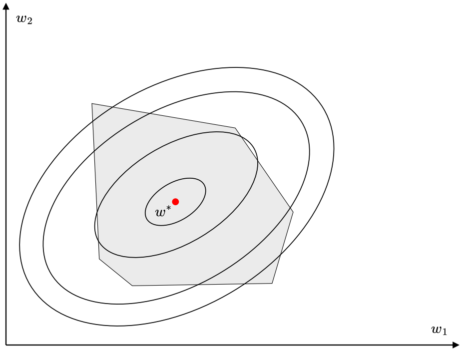


- convex optimization problem
- 1947: simplex method by Dantzig, 1984: polynomial time interior-point method by Karmarkar
- if not unbounded, the solution is always at edge or vertex of the feasible set
- today very mature and reliable

Class 2: Quadratic Programming (QP)

Quadratic program

$$\begin{aligned} \min_{w} \quad & \frac{1}{2} w^\top Q w + g^\top w \\ \text{s.t.} \quad & Aw - b = 0 \\ & Cw - d \geq 0 \end{aligned}$$

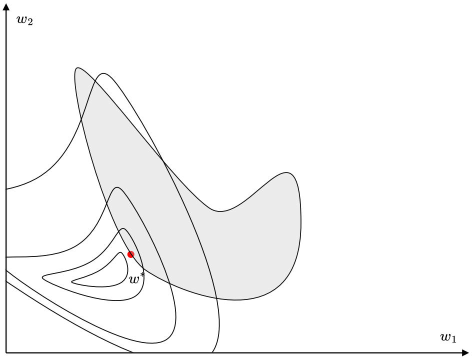


- depending on Q, can be convex and nonconvex
- solved online in linear model predictive control (linear system model + linear constraints + quadratic cost)
- many good solvers: `Gurobi`, `OSQP`, `HPIPM`, `qpOASES`, `OOQP`, ...
- subsproblems in nonlinear optimization

Class 3: Nonlinear Program (NLP)

Nonlinear programming problem

$$\begin{aligned} \min_{w} \quad & F(w) \\ \text{s.t.} \quad & G(w) = 0 \\ & H(w) \geq 0 \end{aligned}$$



- can be convex and nonconvex
- solved with iterative Newton-type algorithms
- solved in nonlinear model predictive control

Classify your control problem

- Linear Program (LP)
- Quadratic Program (QP)
- Nonlinear Program (NLP)

Direct optimal control methods solve Nonlinear Programs (NLP)

Continuous time OCP

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_c(x(t),u(t))\,\mathrm{d}t + E(x(T)) \\
\text{s.t.} \quad & x(0) = \bar{x}_0 \\
& \dot{x}(t) = f_c(x(t),u(t)) \\
& 0 \geq h(x(t),u(t)),\ t \in [0,T] \\
& 0 \geq r(x(T))
\end{aligned}
$$

Direct methods (like direct collocation, multiple shooting) first discretize, then optimize.

Direct optimal control methods solve Nonlinear Programs (NLP)

Continuous time OCP

$$\begin{aligned} \min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_c(x(t),u(t))\,\mathrm{d}t + E(x(T)) \\ \text{s.t.} \quad & x(0) = \bar{x}_0 \\ & \dot{x}(t) = f_c(x(t),u(t)) \\ & 0 \geq h(x(t),u(t)),\ t \in [0,T] \\ & 0 \geq r(x(T)) \end{aligned}$$

Direct methods (like direct collocation, multiple shooting) first discretize, then optimize.

Discrete time OCP (an NLP)

$$\begin{aligned} \min_{x,u} \quad & \sum_{k=0}^{N-1} \ell(x_k,u_k) + E(x_N) \\ \text{s.t.} \quad & x_0 = \bar{x}_0 \\ & x_{k+1} = f(x_k,u_k),\ k = 0,\dots,N-1 \\ & 0 \geq h(x_k,u_k),\ k = 0,\dots,N-1 \\ & 0 \geq r(x_N) \end{aligned}$$

Variables $x = (x_0,\dots,x_N)$ and $u = (u_0,\dots,u_{N-1})$ can be summarized in vector $w = (x,u) \in \mathbb{R}^n$.

Direct optimal control methods solve Nonlinear Programs (NLP)

Discrete time OCP (an NLP)

$$
\begin{aligned}
\min_{x,u} \quad & \textstyle\sum_{k=0}^{N-1} \ell(x_k, u_k) + E(x_N) \\
\text{s.t.} \quad & x_0 = \bar{x}_0 \\
& x_{k+1} = f(x_k, u_k), \; k = 0, \dots, N-1 \\
& 0 \geq h(x_k, u_k), \; k = 0, \dots, N-1 \\
& 0 \geq r(x_N)
\end{aligned}
$$

Variables $x = (x_0, \dots, x_N)$ and $u = (u_0, \dots, u_{N-1})$ can be summarized in vector $w = (x, u) \in \mathbb{R}^n$.

Nonlinear MPC solves Nonlinear Programs (NLP)

Discrete time NMPC Problem (an NLP)

$$
\begin{aligned}
\min_{x,u} \; & \sum_{k=0}^{N-1} \ell(x_k, u_k) + E(x_N) \\
\text{s.t.} \quad & x_0 = \bar{x}_0 \\
& x_{k+1} = f(x_k, u_k) \\
& 0 \geq h(x_k, u_k), \; k = 0, \dots, N-1 \\
& 0 \geq r(x_N)
\end{aligned}
$$

Variables $x = (x_0, \dots, x_N)$ and $u = (u_0, \dots, u_{N-1})$ can be summarized in vector $w = (x, u) \in \mathbb{R}^n$.

Nonlinear Program (NLP)

$$
\begin{aligned}
\min_{w \in \mathbb{R}^n} \; & F(w) \\
\text{s.t.} \; & G(w) = 0 \\
& H(w) \geq 0
\end{aligned}
$$

Outline

Part 1: Nonlinear Optimization

1. Basic definitions
2. Conditions of optimality
3. Nonlinear programming algorithms
4. Nonlinear optimization with `CasADi`

Part 2: Direct Optimal Control

Algebraic characterization of **unconstrained** local optima

Consider the unconstrained problem: $\min_{w \in \mathbb{R}^n} \quad F(w)$

First-Order **Necessary** Condition of Optimality (FONC)

w^* local optimum $\quad \Rightarrow \quad \nabla F(w^*) = 0, \; w^*$ stationary point

Second-Order **Necessary** Condition of Optimality (SONC)

w^* local optimum $\quad \Rightarrow \quad \nabla^2 F(w^*) \succeq 0$

Algebraic characterization of **unconstrained** local optima

Consider the unconstrained problem: $\min_{w \in \mathbb{R}^n} \quad F(w)$

First-Order **Necessary** Condition of Optimality (FONC)

w^* local optimum $\Rightarrow$ $\nabla F(w^*) = 0$, w^* stationary point

Second-Order **Necessary** Condition of Optimality (SONC)

w^* local optimum $\Rightarrow$ $\nabla^2 F(w^*) \succeq 0$

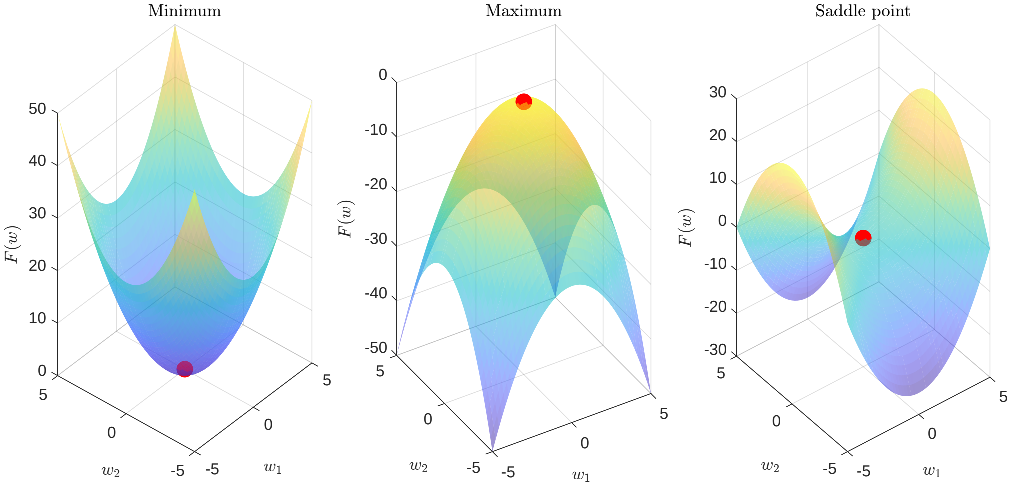
Second-Order **Sufficient** Conditions of Optimality (SOSC)

$\nabla F(w^*) = 0$ and $\nabla^2 F(w^*) \succ 0$ $\Rightarrow$ x^* strict local minimum

$\nabla F(w^*) = 0$ and $\nabla^2 F(w^*) \prec 0$ $\Rightarrow$ x^* strict local maximum

No conclusion can be drawn in the case $\nabla^2 F(w^*)$ is indefinite!

Type of stationary points



A stationary point can be a minimum, maximum or a saddle point

FONC for equality constraints

Nonlinear Program (NLP)

$$\min_{w \in \mathbb{R}^n} \; F(w)$$
$$\text{s.t. } G(w) = 0$$

$\mathcal{L}(w, \lambda) = F(w) - \lambda^\top G(w)$ is the Lagrangian

FONC for equality constraints

Nonlinear Program (NLP)

$$\min_{w \in \mathbb{R}^n} F(w)$$
$$\text{s.t. } G(w) = 0$$

$\mathcal{L}(w, \lambda) = F(w) - \lambda^\top G(w)$ is the Lagrangian

Definition (LICQ)

A point w satisfies Linear Independence Constraint Qualification LICQ if and only if $\nabla G(w)$ is full column rank

FONC for equality constraints

Nonlinear Program (NLP)

$$\min_{w \in \mathbb{R}^n} \; F(w)$$
$$\text{s.t.} \; G(w) = 0$$

$\mathcal{L}(w, \lambda) = F(w) - \lambda^\top G(w)$ is the Lagrangian

Definition (LICQ)

A point w satisfies Linear Independence Constraint Qualification LICQ if and only if $\nabla G\left(w\right)$ is full column rank

First-order Necessary Conditions

Let F, G in $\mathcal{C}^1$. If w^* is a (local) minimizer, **and** w^* satisfies LICQ, then there is a unique vector λ such that:

$$\nabla_w \mathcal{L}(w^*, \lambda^*) = \nabla F(w^*) - \nabla G(w^*)\lambda = 0 \qquad \text{Dual feasibility}$$
$$\nabla_\lambda \mathcal{L}(w^*, \lambda^*) = G(w^*) = 0 \qquad \text{Primal feasibility}$$

The KKT conditions

Nonlinear Program (NLP)

$$\min_{w\in\mathbb{R}^n} \; F(w)$$
$$\text{s.t. } G(w) = 0$$
$$H(w) \geq 0$$

$$\mathcal{L}(w,\lambda) = F(w) - \lambda^\top G(w) - \mu^\top H(w)$$

The KKT conditions

Nonlinear Program (NLP)

$$\min_{w \in \mathbb{R}^n} \; F(w)$$
$$\text{s.t.} \; G(w) = 0$$
$$H(w) \geq 0$$

$$\mathcal{L}(w, \lambda) = F(w) - \lambda^\top G(w) - \mu^\top H(w)$$

Definition (LICQ)

A point w satisfies LICQ if and only if

$$\left[\nabla G\left(w\right), \quad \nabla H_{\mathcal{A}}\left(w\right) \right]$$

is full column rank

Active set $\mathcal{A} = \{ i \mid H_i(w) = 0 \}$

The KKT conditions

Nonlinear Program (NLP)

$$\min_{w \in \mathbb{R}^n} \; F(w)$$
$$\text{s.t. } G(w) = 0$$
$$H(w) \geq 0$$

$$\mathcal{L}(w, \lambda) = F(w) - \lambda^\top G(w) - \mu^\top H(w)$$

Definition (LICQ)

A point w satisfies LICQ if and only if

$$[\nabla G(w), \quad \nabla H_{\mathcal{A}}(w)]$$

is full column rank

Active set $\mathcal{A} = \{i \mid H_i(w) = 0\}$

Theorem (KKT conditions)

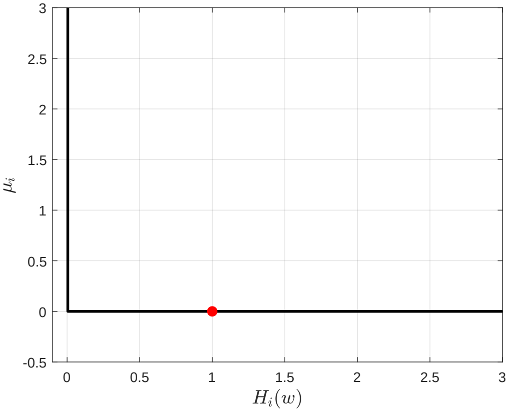
Let F, G, H be $\mathcal{C}^1$. If w^* is a (local) minimizer ***and*** *satisfies LICQ, then there are unique vectors λ^* and μ^* such that (w^*, λ^*, μ^*) satisfies:*

$$\nabla_w \mathcal{L}(w^*, \mu^*, \lambda^*) = 0, \quad \mu^* \geq 0, \qquad \textit{Dual feasibility}$$
$$G(w^*) = 0, \quad H(w^*) \geq 0 \qquad \textit{Primal feasibility}$$
$$\mu_i^* H_i(w^*) = 0, \quad \forall i \qquad \textit{Complementary slackness}$$

The complementary slackness condition

Active constraints:

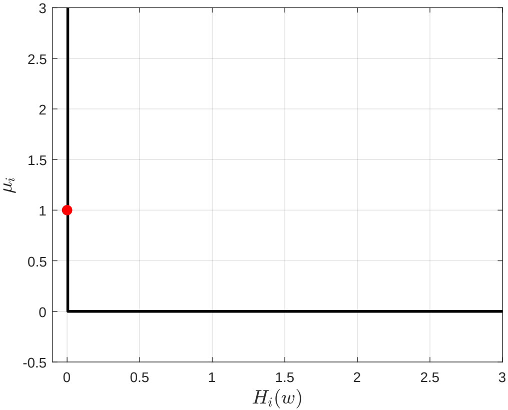
- $H_i(w^*) > 0$ then $\mu_i^* = 0$, and H_i is inactive



The complementary slackness condition

Active constraints:

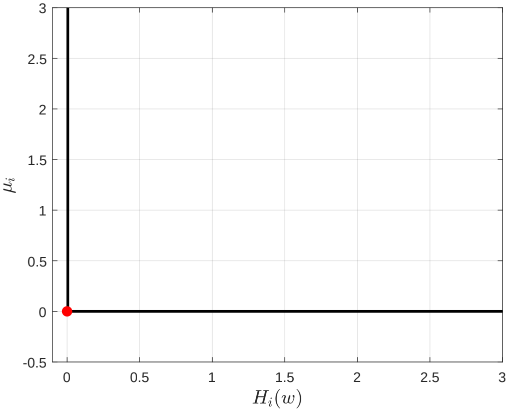
- $H_i(w^*) > 0$ then $\mu_i^* = 0$, and H_i is inactive
- $\mu_i^* > 0$ and $H_i(w) = 0$ then $H_i(w)$ is strictly active



The complementary slackness condition

Active constraints:

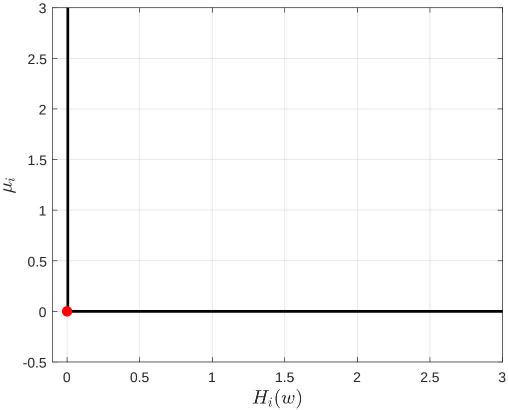
- $H_i(w^*) > 0$ then $\mu_i^* = 0$, and H_i is inactive
- $\mu_i^* > 0$ and $H_i(w) = 0$ then $H_i(w)$ is strictly active
- $\mu_i^* = 0$ and $H_i(w) = 0$ then then $H_i(w)$ is weakly active



The complementary slackness condition

Active constraints:

- $H_i(w^*) > 0$ then $\mu_i^* = 0$, and H_i is inactive
- $\mu_i^* > 0$ and $H_i(w) = 0$ then $H_i(w)$ is strictly active
- $\mu_i^* = 0$ and $H_i(w) = 0$ then then $H_i(w)$ is weakly active
- We define the **active set** $\mathbb{A}^*$ as the set of indices i of the active constraints



Summary of optimality conditions

Optimality conditions for NLP with equality and/or inequality constraints:

- **First-Order Necessary Conditions**: A regular local optimum of a (differentiable) NLP is a KKT point

Summary of optimality conditions

Optimality conditions for NLP with equality and/or inequality constraints:

- **First-Order Necessary Conditions**: A regular local optimum of a (differentiable) NLP is a KKT point
- **Second-Order Sufficient Conditions** require positivity of the Hessian in all critical feasible directions

Summary of optimality conditions

Optimality conditions for NLP with equality and/or inequality constraints:

- **First-Order Necessary Conditions**: A regular local optimum of a (differentiable) NLP is a KKT point
- **Second-Order Sufficient Conditions** require positivity of the Hessian in all critical feasible directions

Nonconvex problem $\Rightarrow$ minimum is not necessarily global.
But some nonconvex problems have a unique minimum

Summary of optimality conditions

Optimality conditions for NLP with equality and/or inequality constraints:

- **First-Order Necessary Conditions**: A regular local optimum of a (differentiable) NLP is a KKT point
- **Second-Order Sufficient Conditions** require positivity of the Hessian in all critical feasible directions

Nonconvex problem ⇒ minimum is not necessarily global.
But some nonconvex problems have a unique minimum

Some important practical consequences...

- A KKT point **may not** be a local (global) optimum
  ... the lack of equivalence results from a lack of regularity and/or SOSC
- A local (global) optimum **may not** be a KKT point
  ... due to violation of constraint qualifications, e.g. LICQ violated.

Outline

Part 1: Nonlinear Optimization

1. Basic definitions
2. Conditions of optimality
3. Nonlinear programming algorithms
4. Nonlinear optimization with `CasADi`

Part 2: Direct Optimal Control

Newton's method

To solve a nonlinear system, solve a sequence of linear systems

Root-finding problem. Find x such that $F(x) = 0$.

Linearization of F at linearization point $\bar{w}$

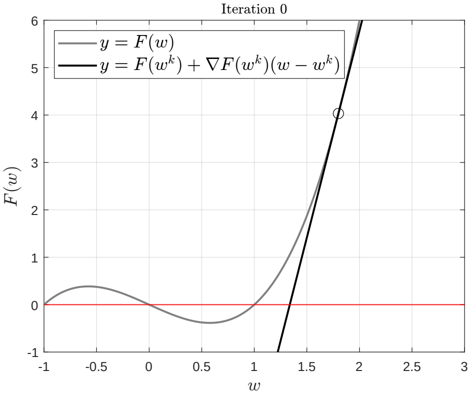
equals

First-order Taylor series at $\bar{w}$

equals

$$F_{\mathrm{L}}(w; \bar{w}) := F(\bar{w}) + \frac{\partial F}{\partial w}(\bar{w}) \quad (w - \bar{w})$$

(for continuously differentiable $F : \mathbb{R}^n \to \mathbb{R}^n$)



Newton's method

To solve a nonlinear system, solve a sequence of linear systems

Root-finding problem. Find x such that $F(x) = 0$.

Linearization of F at linearization point $\bar{w}$

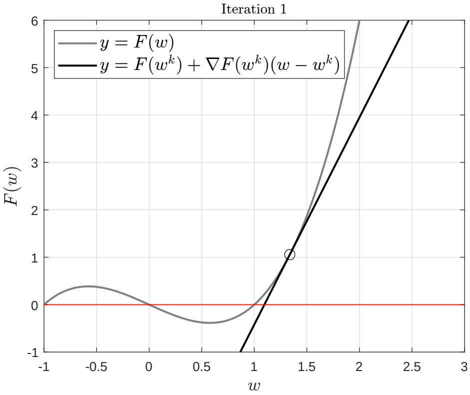
equals

First-order Taylor series at $\bar{w}$

equals

$$F_{\mathrm{L}}(w; \bar{w}) := F(\bar{w}) + \nabla_w F(\bar{w})^\top (w - \bar{w})$$

(for continuously differentiable $F : \mathbb{R}^n \to \mathbb{R}^n$)



Newton's method

To solve a nonlinear system, solve a sequence of linear systems

Root-finding problem. Find x such that $F(x) = 0$.

Linearization of F at linearization point $\bar{w}$

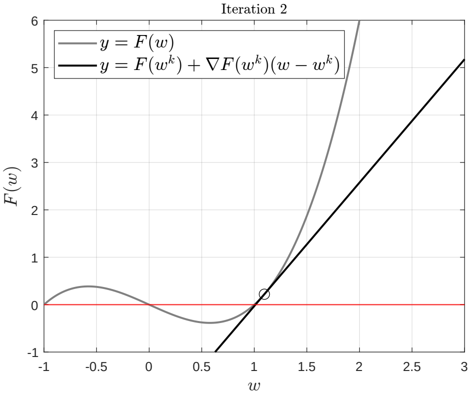
equals

First-order Taylor series at $\bar{w}$

equals

$$F_{\mathrm{L}}(w; \bar{w}) := F(\bar{w}) + \nabla_w F(\bar{w})^\top (w - \bar{w})$$

(for continuously differentiable $F : \mathbb{R}^n \to \mathbb{R}^n$)



Newton's method

To solve a nonlinear system, solve a sequence of linear systems

Root-finding problem. Find x such that $F(x) = 0$.

Linearization of F at linearization point $\bar{w}$

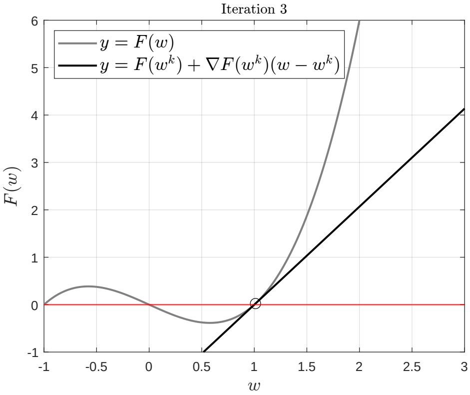
equals

First-order Taylor series at $\bar{w}$

equals

$$F_{\mathrm{L}}(w; \bar{w}) := F(\bar{w}) + \nabla_w F(\bar{w})^\top (w - \bar{w})$$

(for continuously differentiable $F : \mathbb{R}^n \to \mathbb{R}^n$)



General Nonlinear Program (NLP)

In direct methods, we have to solve the discretized optimal control problem, which is a Nonlinear Program (NLP)

General Nonlinear Program (NLP)

$$\min_{w} F(w) \quad \text{s.t.} \quad \begin{cases} G(w) & = & 0 \\ H(w) & \geq & 0 \end{cases}$$

We first treat the case without inequalities

NLP only with equality constraints

$$\min_{w} F(w) \quad \text{s.t.} \quad G(w) = 0$$

Lagrange function and optimality conditions

Lagrange function

$$\mathcal{L}(w, \lambda) = F(w) - \lambda^T G(w)$$

Then for an optimal solution w^* exist multipliers λ^* such that

Nonlinear root-finding problem

$$\begin{aligned} \nabla_w \mathcal{L}(w^*, \lambda^*) &= 0 \\ G(w^*) &= 0 \end{aligned}$$

Newton's Method on optimality conditions

How to solve nonlinear equations

$$\begin{aligned} \nabla_w \mathcal{L}(w^*, \lambda^*) &= 0 \\ G(w^*) &= 0 \quad ? \end{aligned}$$

Linearize!

$$\begin{array}{rrrcl} \nabla_w \mathcal{L}(w^k, \lambda^k) & +\nabla_w^2 \mathcal{L}(w^k, \lambda^k)\Delta w & -\nabla_w G(w^k)\Delta\lambda & = & 0 \\ G(w^k) & +\nabla_w G(w^k)^T \Delta w & & = & 0 \end{array}$$

This is equivalent, due to $\nabla \mathcal{L}(w^k, \lambda^k) = \nabla F(w^k) - \nabla G(w^k)\lambda^k$ with the shorthand $\lambda^+ = \lambda^k + \Delta\lambda$ to

$$\begin{array}{rrrcl} \nabla_w F(w^k) & +\nabla_w^2 \mathcal{L}(w^k, \lambda^k)\Delta w & -\nabla_w G(w^k)\lambda^+ & = & 0 \\ G(w^k) & +\nabla_w G(w^k)^T \Delta w & & = & 0 \end{array}$$

Newton Step = Solution to a Quadratic Program

Conditions

$$\begin{array}{rrrcl} \nabla_w F(w^k) & +\nabla_w^2 \mathcal{L}(w^k, \lambda^k)\Delta w & -\nabla_w G(w^k)\lambda^+ & = & 0 \\ G(w^k) & +\nabla_w G(w^k)^T \Delta w & & = & 0 \end{array}$$

are optimality conditions of a quadratic program (QP), namely:

Quadratic program

$$\begin{array}{ll} \min\limits_{\Delta w} & \nabla F(w^k)^T \Delta w + \frac{1}{2} \Delta w^T A^k \Delta w \\ \text{s.t.} & G(w^k) + \nabla G(w^k)^T \Delta w \quad = \quad 0, \end{array}$$

with

$$A^k = \nabla_w^2 \mathcal{L}(w^k, \lambda^k)$$

Newton's method

The full step Newton's Method iterates by solving in each iteration the quadratic program (QP)

$$\begin{aligned} \min_{\Delta w} \quad & \nabla F(w^k)^T \Delta w + \frac{1}{2} \Delta w^T A^k \Delta w \\ \text{s.t.} \quad & G(w^k) + \nabla G(w^k)^T \Delta w \quad = \quad 0, \end{aligned}$$

with $A^k = \nabla^2_w \mathcal{L}(w^k, \lambda^k)$. As solution, we obtain the step Δw^k and the new multiplier λ^+_{QP}.

New iterate

$$\begin{aligned} w^{k+1} &= w^k + \Delta w^k \\ \lambda^{k+1} &= \lambda^k + \Delta \lambda^k = \lambda^+_{\mathrm{QP}} \end{aligned}$$

This Newton's method is also called Sequential Quadratic Programming (SQP) for equality constrained optimization (with *exact Hessian* and *full steps*)

NLP with Inequalities

Regard again NLP with both, equalities and inequalities:

NLP with equality and inequality constraints

$$\min_{w} F(w) \ \text{s.t.} \ \begin{cases} G(w) & = & 0 \\ H(w) & \geq & 0 \end{cases}$$

Lagrangian function for NLP with equality and inequality constraints

$$\mathcal{L}(w, \lambda, \mu) = F(w) - \lambda^T G(w) - \mu^T H(w)$$

Optimality conditions with inequalities

Theorem (Karush-Kuhn-Tucker (KKT) conditions)

Let F, G, H be $\mathcal{C}^2$. If w^* is a (local) minimizer and satisfies LICQ, then there are unique vectors λ^* and μ^* such that (w^*, λ^*, μ^*) satisfies:

$$\begin{aligned} \nabla_w \mathcal{L}(w^*, \mu^*, \lambda^*) &= 0 \\ G(w^*) &= 0 \\ H(w^*) &\geq 0 \\ \mu^* &\geq 0 \\ H(w^*)^\top \mu^* &= 0 \end{aligned}$$

- These contain nonsmooth conditions (the last three) which are called *complementarity conditions*
- This system cannot be solved by Newton's Method. But still with SQP...

Sequential Quadratic Programming (SQP)

By Linearizing all functions within the KKT Conditions, and setting $\lambda^+ = \lambda^k + \Delta\lambda$ and $\mu^+ = \mu^k + \Delta\mu$, we obtain the KKT conditions of a Quadratic Program (QP).

QP with inequality constraints

$$\min_{\Delta w} \quad \nabla F(w^k)^T \Delta w + \frac{1}{2}\Delta w^T A^k \Delta w$$

$$\text{s.t.} \quad \begin{cases} G(w^k) + \nabla G(w^k)^T \Delta w & = & 0 \\ H(w^k) + \nabla H(w^k)^T \Delta w & \geq & 0 \end{cases}$$

with

$$A^k = \nabla_w^2 \mathcal{L}(w^k, \lambda^k, \mu^k)$$

and its solution delivers

$$\Delta w^k, \quad \lambda_{\text{QP}}^+, \quad \mu_{\text{QP}}^+$$

Constrained Gauss-Newton Method

In special case of least squares objectives

Least squares objective function

$$F(w) = \frac{1}{2}\|R(w)\|_2^2$$

can approximate Hessian $\nabla_w^2 \mathcal{L}(w^k, \lambda^k, \mu^k)$ by much cheaper

$$A^k = \nabla R(w) \nabla R(w)^T.$$

Need no multipliers to compute A^k! QP= linear least squares:

Gauss-Newton QP

$$\begin{aligned}
\min_{\Delta w} \quad & \frac{1}{2}\|R(w^k) + \nabla R(w^k)^T \Delta w\|_2^2 \\
\text{s.t.} \quad & G(w^k) + \nabla G(w^k)^T \Delta w = 0 \\
& H(w^k) + \nabla H(w^k)^T \Delta w \geq 0
\end{aligned}$$

Convergence: linear (better if $\|R(w^*)\|$ small)

Interior point methods

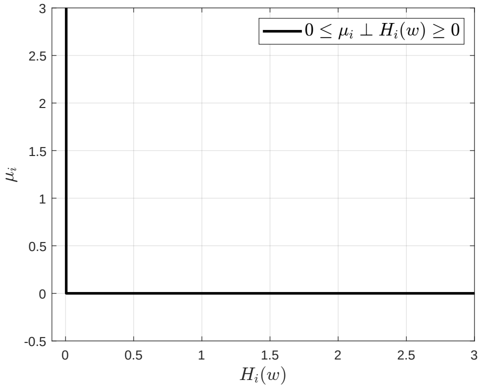
NLP with inequalites

$$\min_{w} \; F(w)$$
$$\text{s.t.} \quad H(w) \geq 0$$

KKT conditions

$$\nabla F(w) - \nabla H(w)^\top \mu = 0$$
$$0 \leq \mu \perp H(w) \geq 0$$

Main difficulty: inequality conditions introduce nonsmoothness in the KKT conditions



The barrier problem

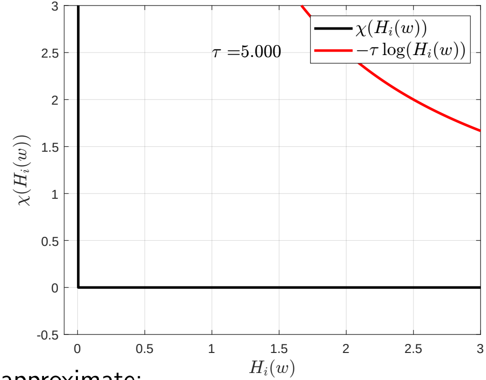
NLP with inequalites

$$\min_{w} \; F(w)$$
$$\text{s.t.} \quad H(w) \geq 0$$

Barrier problem

$$\min_{w} \; F(w) - \tau \sum_{i=1}^{m} \log(H_i(w)) =: F_\tau(w)$$

Main idea: put inequality constraint into objective



approximate:

$$\chi(H_i(w)) = \begin{cases} 0 & \text{if } H_i(w) \geq 0 \\ \infty & \text{if } H_i(w) < 0 \end{cases}$$

The barrier problem

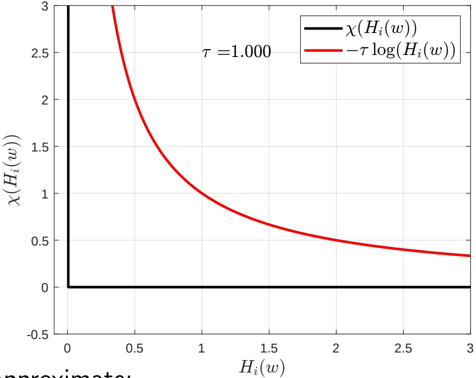
NLP with inequalites

$$\min_{w} \; F(w)$$
$$\text{s.t.} \quad H(w) \geq 0$$

Barrier problem

$$\min_{w} \; F(w) - \tau \sum_{i=1}^{m} \log(H_i(w)) =: F_\tau(w)$$

Main idea: put inequality constraint into objective



approximate:

$$\chi(H_i(w)) = \begin{cases} 0 & \text{if } H_i(w) \geq 0 \\ \infty & \text{if } H_i(w) < 0 \end{cases}$$

The barrier problem

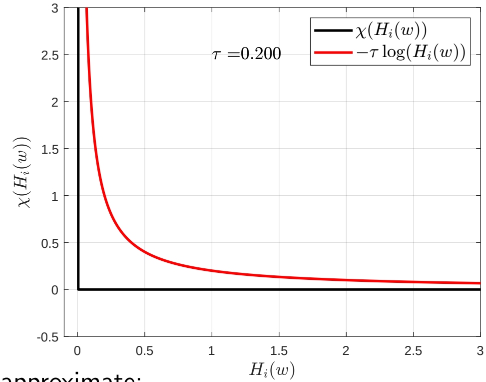
NLP with inequalites

$$\min_{w} \; F(w)$$
$$\text{s.t.} \quad H(w) \geq 0$$

Barrier problem

$$\min_{w} \; F(w) - \tau \sum_{i=1}^{m} \log(H_i(w)) =: F_\tau(w)$$

Main idea: put inequality constraint into objective



approximate:

$$\chi(H_i(w)) = \begin{cases} 0 & \text{if } H_i(w) \geq 0 \\ \infty & \text{if } H_i(w) < 0 \end{cases}$$

The barrier problem

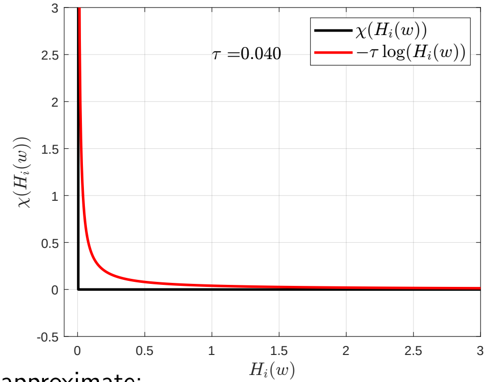
NLP with inequalites

$$\min_{w} \; F(w)$$
$$\text{s.t.} \quad H(w) \geq 0$$

Barrier problem

$$\min_{w} \; F(w) - \tau \sum_{i=1}^{m} \log(H_i(w)) =: F_\tau(w)$$

Main idea: put inequality constraint into objective



approximate:

$$\chi(H_i(w)) = \begin{cases} 0 & \text{if } H_i(w) \geq 0 \\ \infty & \text{if } H_i(w) < 0 \end{cases}$$

The barrier problem

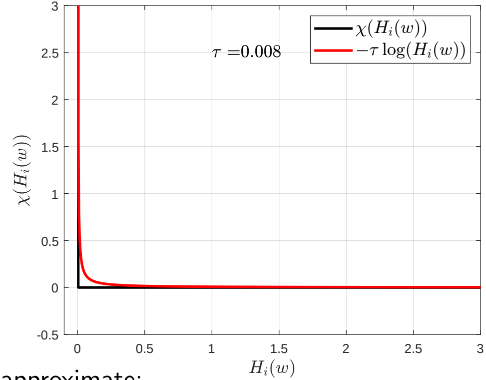
NLP with inequalites

$$\min_{w} \; F(w)$$
$$\text{s.t.} \quad H(w) \geq 0$$

Barrier problem

$$\min_{w} \; F(w) - \tau \sum_{i=1}^{m} \log(H_i(w)) =: F_\tau(w)$$

Main idea: put inequality constraint into objective



approximate:

$$\chi(H_i(w)) = \begin{cases} 0 & \text{if } H_i(w) \geq 0 \\ \infty & \text{if } H_i(w) < 0 \end{cases}$$

The barrier problem

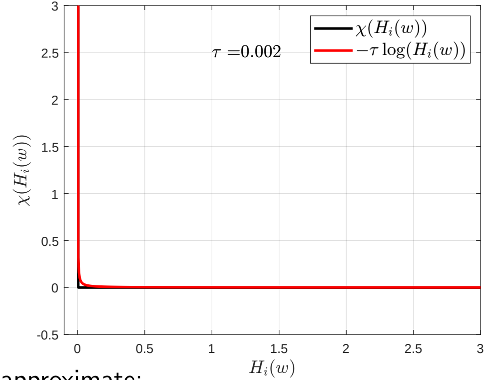
NLP with inequalites

$$\begin{aligned} \min_{w} \ & F(w) \\ \text{s.t.} \ & H(w) \geq 0 \end{aligned}$$

Barrier problem

$$\min_{w} \ F(w) - \tau \sum_{i=1}^{m} \log(H_i(w)) =: F_\tau(w)$$

Main idea: put inequality constraint into objective



approximate:

$$\chi(H_i(w)) = \begin{cases} 0 & \text{if } H_i(w) \geq 0 \\ \infty & \text{if } H_i(w) < 0 \end{cases}$$

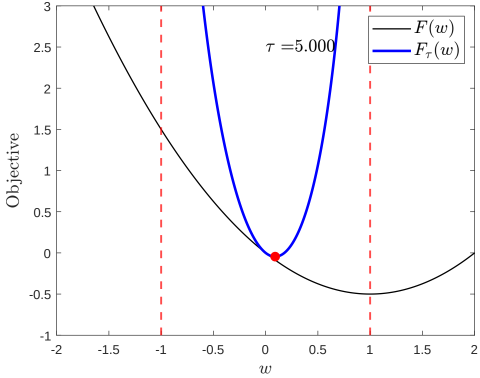
An example of the barrier problem

Example NLP

$$\min_{w} \; 0.5w^2 - 2w$$
$$\text{s.t.} \quad -1 \leq w \leq 1$$

Barrier problem

$$\min_{w} \; 0.5w^2 - 2 - \tau \log(w+1) - \tau \log(1-w)$$



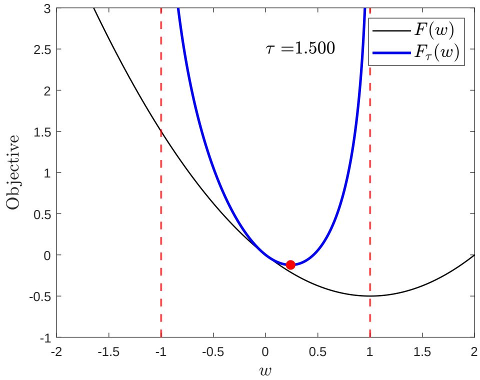
An example of the barrier problem

Example NLP

$$\min_{w} \; 0.5w^2 - 2w$$
$$\text{s.t.} \quad -1 \leq w \leq 1$$

Barrier problem

$$\min_{w} \; 0.5w^2 - 2 - \tau \log(w+1) - \tau \log(1-w)$$



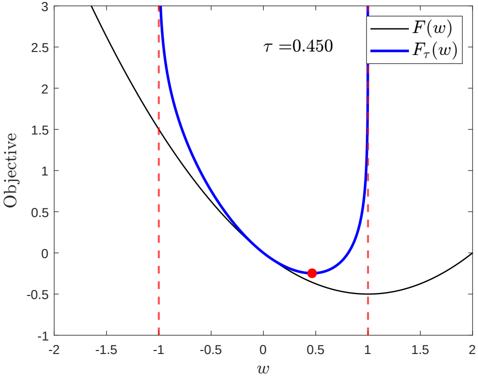
An example of the barrier problem

Example NLP

$$\min_{w} \; 0.5w^2 - 2w$$
$$\text{s.t.} \quad -1 \le w \le 1$$

Barrier problem

$$\min_{w} \; 0.5w^2 - 2 - \tau \log(w+1) - \tau \log(1-w)$$



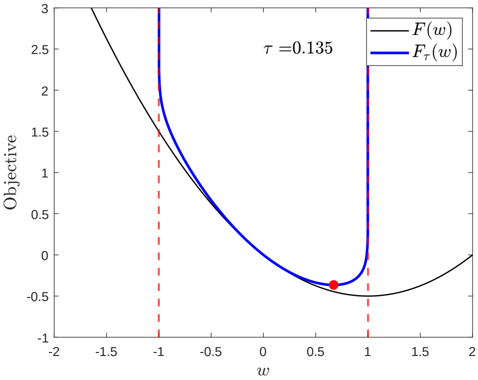
An example of the barrier problem

Example NLP

$$\begin{aligned} \min_{w} \ & 0.5w^2 - 2w \\ \text{s.t.} \quad & -1 \leq w \leq 1 \end{aligned}$$

Barrier problem

$$\min_{w} \ 0.5w^2 - 2 - \tau \log(w+1) - \tau \log(1-w)$$



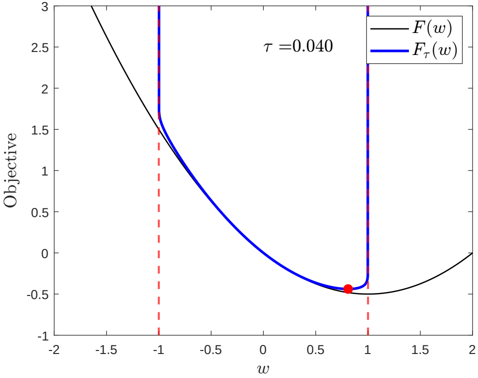
An example of the barrier problem

Example NLP

$$\min_{w} \; 0.5w^2 - 2w$$
$$\text{s.t.} \quad -1 \leq w \leq 1$$

Barrier problem

$$\min_{w} \; 0.5w^2 - 2 - \tau \log(w+1) - \tau \log(1-w)$$



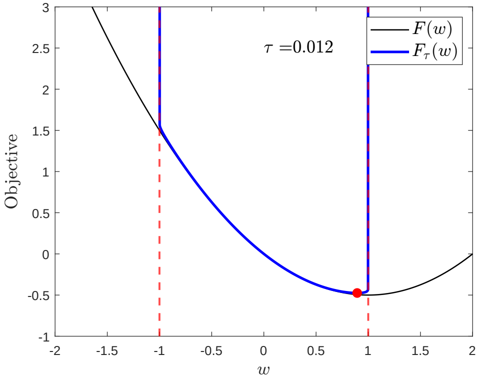
An example of the barrier problem

Example NLP

$$\min_{w} \; 0.5w^2 - 2w$$
$$\text{s.t.} \quad -1 \leq w \leq 1$$

Barrier problem

$$\min_{w} \; 0.5w^2 - 2 - \tau \log(w+1) - \tau \log(1-w)$$



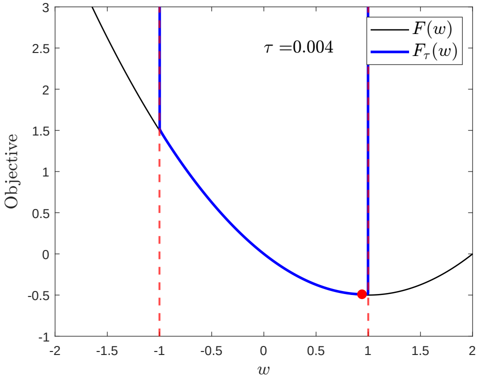
An example of the barrier problem

Example NLP

$$\min_{w} \; 0.5w^2 - 2w$$
$$\text{s.t.} \quad -1 \leq w \leq 1$$

Barrier problem

$$\min_{w} \; 0.5w^2 - 2 - \tau \log(w+1) - \tau \log(1-w)$$



Summary Newton-type optimization

- Newton type optimization solves the necessary optimality conditions
- Newton's method linearizes the nonlinear system in each iteration
- for constraints: requires Lagrangian function
- for equality constraints: KKT conditions are smooth, can apply Newton's method directly
- for inequality constraints: KKT conditions are non-smooth
  $\rightarrow$ Sequential Quadratic Programming (SQP)
- QP subproblem might be solved via an interior point solver, active set solver, ADMM, etc.

Outline

Part 1: Nonlinear Optimization

1. Basic definitions
2. Conditions of optimality
3. Nonlinear programming algorithms
4. Nonlinear optimization with `CasADi`

Part 2: Direct Optimal Control

CasADi[1] is an open-source tool for nonlinear optimization and algorithmic differentiation.

https://web.casadi.org/

CasADi provides

- algorithmic differentiation on user-defined symbolic expressions
- standardized interfaces to a variety of numerical routines:
  - simulation and nonlinear constrained optimization
  - solution of linear and nonlinear equations
- CasADi can be used from C++, python, Octave or MATLAB.

[1] Joel A. E. Andersson, Joris Gillis, Greg Horn, James B. Rawlings and Moritz Diehl: *CasADi – A software framework for nonlinear optimization and optimal control*; Mathematical Programming Computation (2019).

CasADi

`CasADi`[1] is an open-source tool for nonlinear optimization and algorithmic differentiation.

`https://web.casadi.org/`

`CasADi` provides

- algorithmic differentiation on user-defined symbolic expressions
- standardized interfaces to a variety of numerical routines:
  - simulation and nonlinear constrained optimization → Interior point solver `IPOPT`
  - solution of linear and nonlinear equations
- `CasADi` can be used from `C++`, `python`, `Octave` or `MATLAB`.

[1] Joel A. E. Andersson, Joris Gillis, Greg Horn, James B. Rawlings and Moritz Diehl: *CasADi – A software framework for nonlinear optimization and optimal control*; Mathematical Programming Computation (2019).

Exercise Session – Part 1

1. Read the docs! `https://web.casadi.org/docs`
   - What is the difference between a `CasADi` expression and a `CasADi` function?
   - How do you compute a derivative using `CasADi`?
2. Work on the exercise sheet.
   - How to formulate a constrained nonlinear optimization problem with `CasADi`? How to solve the NLP with the solver `IPOPT`?

Ordinary differential equations and controlled dynamical system

Let:

- $t \in \mathbb{R}$ be the time
- $x(t) \in \mathbb{R}^{n_x}$ the differential states and $\dot{x}(t) = \frac{\mathrm{d}x(t)}{\mathrm{d}t}$
- $u(t) \in \mathbb{R}^{n_u}$ a given control function

Ordinary differential equations and controlled dynamical system

Let:

- $t \in \mathbb{R}$ be the time
- $x(t) \in \mathbb{R}^{n_x}$ the differential states and $\dot{x}(t) = \frac{\mathrm{d}x(t)}{\mathrm{d}t}$
- $u(t) \in \mathbb{R}^{n_u}$ a given control function

Ordinary differential equations

- Let $F : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ be a function such that the Jacobian $\frac{\partial F}{\partial \dot{x}}(\cdot)$ is invertible. The system of equations:

$$F(t, \dot{x}(t), x(t), u(t)) = 0,$$

is called an Ordinary Differential Equation (ODE).

Ordinary differential equations and controlled dynamical system

Let:

- $t \in \mathbb{R}$ be the time
- $x(t) \in \mathbb{R}^{n_x}$ the differential states and $\dot{x}(t) = \frac{\mathrm{d}x(t)}{\mathrm{d}t}$
- $u(t) \in \mathbb{R}^{n_u}$ a given control function

Ordinary differential equations

- Let $F : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ be a function such that the Jacobian $\frac{\partial F}{\partial \dot{x}}(\cdot)$ is invertible. The system of equations:

$$F(t, \dot{x}(t), x(t), u(t)) = 0,$$

  is called an Ordinary Differential Equation (ODE).
- Given a function $f : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ then a system of equations:

$$\dot{x}(t) = f(t, x(t), u(t)) \tag{2}$$

  is called an explicit ODE.

ODE Example: harmonic oscillator

Mass m with spring constant k and friction coefficient c:

$$\begin{aligned}
\dot{x}_1(t) &= x_2(t) \\
\dot{x}_2(t) &= -\frac{k}{m}(x_2(t) - u(t)) \quad - \frac{\beta}{m} x_1(t)
\end{aligned}$$

ODE Example: harmonic oscillator

Mass m with spring constant k and friction coefficient c:

$$\begin{aligned}\dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -\frac{k}{m}(x_2(t) - u(t)) \quad - \frac{\beta}{m} x_1(t)\end{aligned}$$

- state $x(t) \in \mathbb{R}^2$
- position of mass $x_1(t)$ $\longleftarrow$ measured
- velocity of mass $x_2(t)$
- control action: spring position $u(t) \in \mathbb{R}$ $\longleftarrow$ manipulated

ODE Example: harmonic oscillator

Mass m with spring constant k and friction coefficient c:

$$\begin{aligned}
\dot{x}_1(t) &= x_2(t) \\
\dot{x}_2(t) &= -\frac{k}{m}(x_2(t) - u(t)) \quad - \frac{\beta}{m} x_1(t)
\end{aligned}$$

- state $x(t) \in \mathbb{R}^2$
- position of mass $x_1(t)$ $\longleftarrow$ measured
- velocity of mass $x_2(t)$
- control action: spring position $u(t) \in \mathbb{R}$ $\longleftarrow$ manipulated

As explicit ODE: $\dot{x} = f(x,u)$ with

$$f(x,u) = \begin{bmatrix} x_2 \\ -\frac{k}{m}(x_2 - u) - \frac{c}{m} x_1 \end{bmatrix}$$

As implicit ODE: $0 = F(\dot{x}, x, u)$ with

$$F(\dot{x}, x, u) = \begin{bmatrix} x_2 - \dot{x}_1 \\ -\frac{k}{m}(x_2 - u) - \frac{\beta}{m} x_1 - \dot{x}_2 \end{bmatrix}$$

Differential algebraic equations

Let:

- $x(t) \in \mathbb{R}^{n_x}$ the differential states with $\dot{x}(t) = \frac{\mathrm{d}x(t)}{\mathrm{d}t}$
- $z(t) \in \mathbb{R}^{n_z}$ the algebraic states
- $u(t) \in \mathbb{R}^{n_u}$ a given control function

Differential algebraic equations

Let:

- $x(t) \in \mathbb{R}^{n_x}$ the differential states with $\dot{x}(t) = \frac{\mathrm{d}x(t)}{\mathrm{d}t}$
- $z(t) \in \mathbb{R}^{n_z}$ the algebraic states
- $u(t) \in \mathbb{R}^{n_u}$ a given control function

Differential algebraic equations

- Let $F : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ be a function such that the matrix $[\frac{\partial F}{\partial \dot{x}} \; \frac{\partial F}{\partial z}]$ is invertible (*index one*). The system of equations:

$$F(t, \dot{x}(t), x(t), z(t), u(t)) = 0,$$

is called an fully implicit Differential Algebraic Equation (DAE).

Differential algebraic equations

Let:

- $x(t) \in \mathbb{R}^{n_x}$ the differential states with $\dot{x}(t) = \frac{\mathrm{d}x(t)}{\mathrm{d}t}$
- $z(t) \in \mathbb{R}^{n_z}$ the algebraic states
- $u(t) \in \mathbb{R}^{n_u}$ a given control function

Differential algebraic equations

- Let $F : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ be a function such that the matrix $[\frac{\partial F}{\partial \dot{x}} \;\; \frac{\partial F}{\partial z}]$ is invertible (*index one*). The system of equations:

$$F(t, \dot{x}(t), x(t), z(t), u(t)) = 0,$$

  is called an fully implicit Differential Algebraic Equation (DAE).

- Let $f : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ and $g : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_z}$ with $\frac{\partial g}{\partial z}$ invertible. The system of equations:

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

  is called a semi-explicit DAE.

Basic definitions of numerical simulation

- IVPs have only in special cases a closed form solution
- Instead, compute numerically a solution approximation $\tilde{x}(t)$ that approximately satisfies:

$$\begin{aligned}\dot{\tilde{x}}(t) &\approx f(t, \tilde{x}(t), u(t)), \quad t \in [0, T]\\ \tilde{x}(0) &= x(0) = x_0\end{aligned}$$

Basic definitions of numerical simulation

- IVPs have only in special cases a closed form solution
- Instead, compute numerically a solution approximation $\tilde{x}(t)$ that approximately satisfies:

$$\begin{aligned} \dot{\tilde{x}}(t) &\approx f(t, \tilde{x}(t), u(t)), \quad t \in [0, T] \\ \tilde{x}(0) &= x(0) = x_0 \end{aligned}$$

- Recursively generate solution approximation $x_n := \tilde{x}(t_n) \approx x(t_n)$ at N discrete time points $0 = t_0 < t_1 < \ldots < t_N = T$
- Integration interval $[0, T]$ split into subintervals $[t_n, t_{n+1}]$ where $h = t_{n+1} - t_n$

Single step numerical simulation as discrete time system

Single step abstract integration method

ODE.

$$x_{n+1} = \phi(x_n, u_n)$$

where ϕ computes the next state based on current state and input.

DAE.

$$\begin{bmatrix} x_{n+1} \\ z_n \end{bmatrix} = \phi(x_n, u_n)$$

where ϕ computes the next state and algebraic variables based on the current state and input.

Single step numerical simulation as discrete time system

Single step abstract integration method

ODE.

$$x_{n+1} = \phi(x_n, u_n)$$

where ϕ computes the next state based on current state and input.

DAE.

$$\begin{bmatrix} x_{n+1} \\ z_n \end{bmatrix} = \phi(x_n, u_n)$$

where ϕ computes the next state and algebraic variables based on the current state and input.

Simplest Example: Explicit Euler

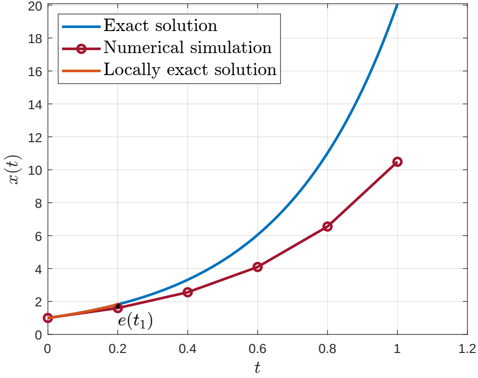
$$x_{n+1} = x_n + hf(x_n, u_n).$$

Integration error

Local and global error

- Local integration error at t_{n+1}:

$$e(t_{n+1}) = \|x(t_{n+1}) - \phi(x(t_n), u_0)\|.$$

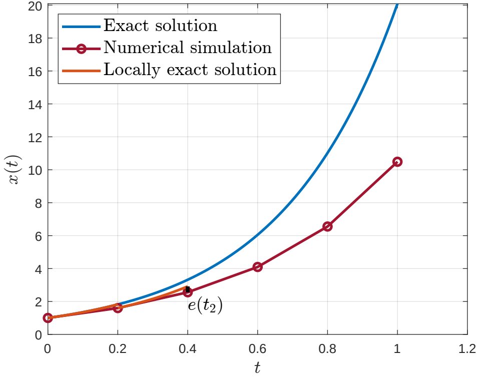


Integration error

Local and global error

- Local integration error at t_{n+1}:

$$e(t_{n+1}) = \|x(t_{n+1}) - \phi(x(t_n), u_0)\|.$$

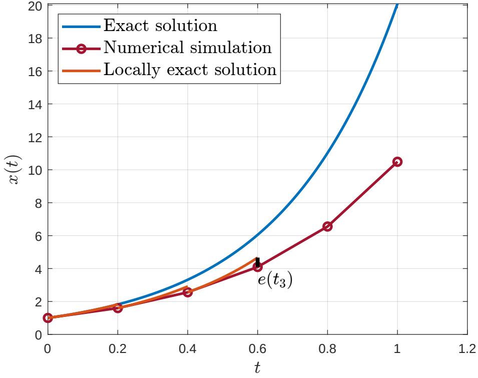


Integration error

Local and global error

- Local integration error at t_{n+1}:

$$e(t_{n+1}) = \|x(t_{n+1}) - \phi(x(t_n), u_0)\|.$$

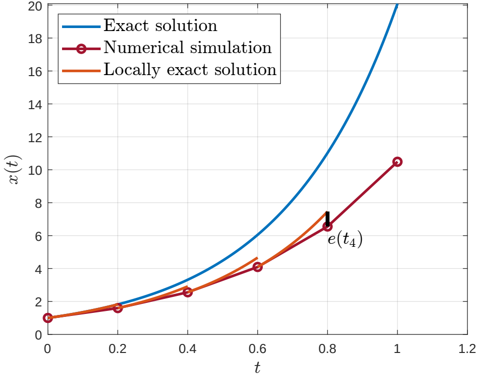


Integration error

Local and global error

- Local integration error at t_{n+1}:

$$e(t_{n+1}) = \|x(t_{n+1}) - \phi(x(t_n), u_0)\|.$$



Integration error

Local and global error

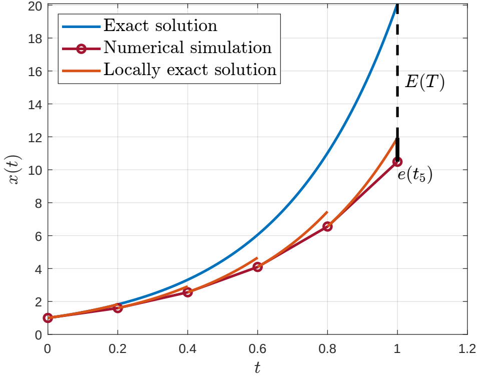
- Local integration error at t_{n+1}:

$$e(t_{n+1}) = \|x(t_{n+1}) - \phi(x(t_n), u_0)\|.$$

- Global integration error at $t = T$:

$$E(T) = \|x(T) - x_N\|.$$

- Global error - accumulation of local errors



Convergence and integrator order

Integrator convergence and accuracy

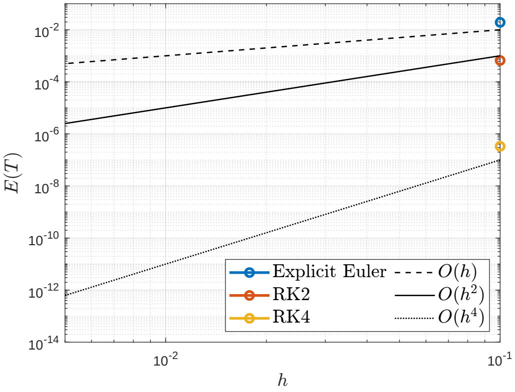
- Convergence

$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq C h^{p+1} = O(h^{p+1}), C > 0$$

- Higher order p:
  - less, but more expensive steps for same accuracy
  - in total fewer r.h.s. evaluations for same accuracy



Convergence and integrator order

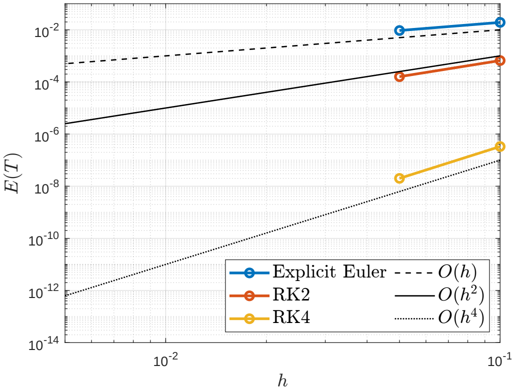
Integrator convergence and accuracy

- Convergence

$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq Ch^{p+1} = O(h^{p+1}), C > 0$$



- Higher order p:
  - less, but more expensive steps for same accuracy
  - in total fewer r.h.s. evaluations for same accuracy

Convergence and integrator order

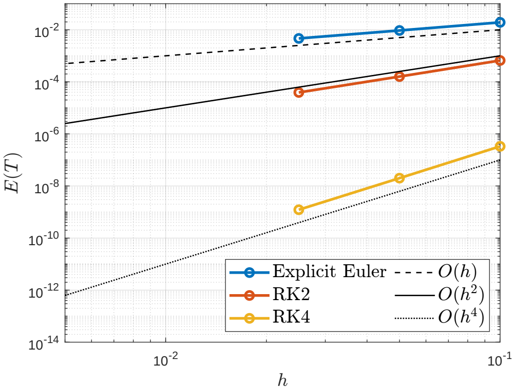
Integrator convergence and accuracy

- Convergence

$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq Ch^{p+1} = O(h^{p+1}), C > 0$$



- Higher order p:
  - less, but more expensive steps for same accuracy
  - in total fewer r.h.s. evaluations for same accuracy

Convergence and integrator order

Integrator convergence and accuracy

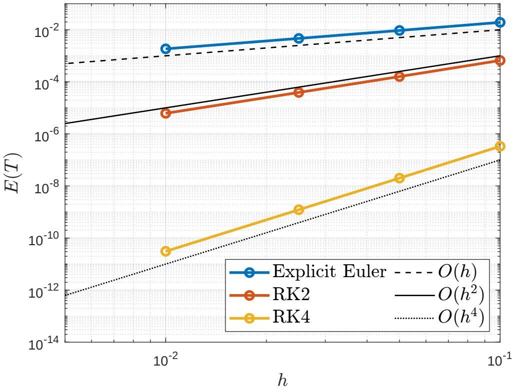
- Convergence

$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq Ch^{p+1} = O(h^{p+1}), C > 0$$

- Higher order p:
  - less, but more expensive steps for same accuracy
  - in total fewer r.h.s. evaluations for same accuracy



Convergence and integrator order

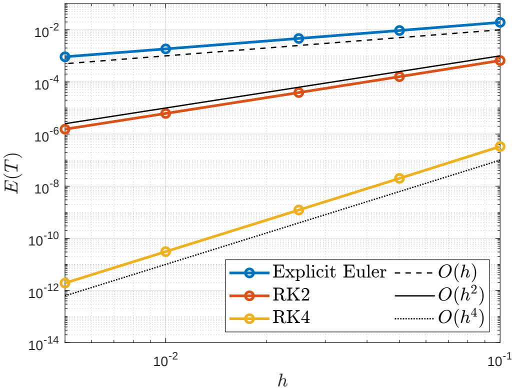
Integrator convergence and accuracy

- Convergence

$$\lim_{h\to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h\to 0} e(t_i) \leq Ch^{p+1} = O(h^{p+1}), C > 0$$



- Higher order p:
  - less, but more expensive steps for same accuracy
  - in total fewer r.h.s. evaluations for same accuracy

Convergence and integrator order

Integrator convergence and accuracy

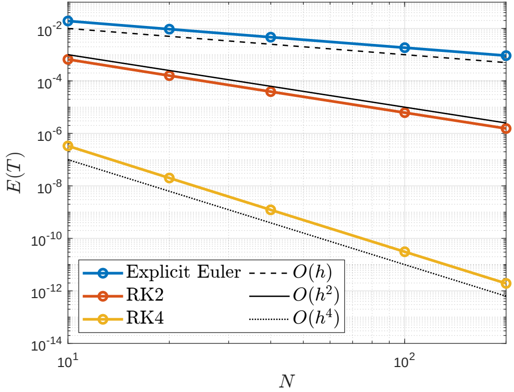
- Convergence

$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq C h^{p+1} = O(h^{p+1}), C > 0$$

- Higher order p:
  - less, but more expensive steps for same accuracy
  - in total fewer r.h.s. evaluations for same accuracy



Stability and convergence

Integrator convergence and accuracy

- Convergence

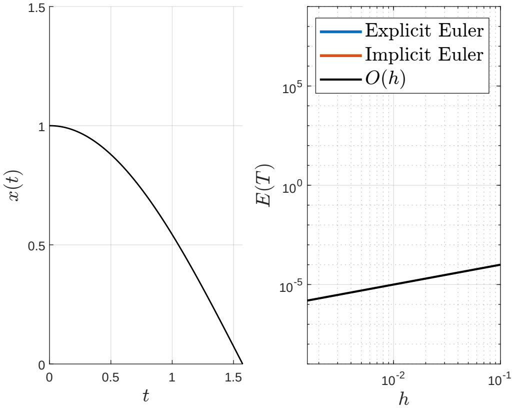
$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq C h^{p+1} = O(h^{p+1}), C > 0$$

- Stability: damping of errors, does it work for $h \gg 0$?

- If integrator is unstable, it does not converge and has $p = 0$, unless h very small



$$\dot{x}(t) = -300(x(t) - \cos(t)), \;\; t \in [0, 2]$$
$$x(0) = 1$$

Stability and convergence

Integrator convergence and accuracy

- Convergence

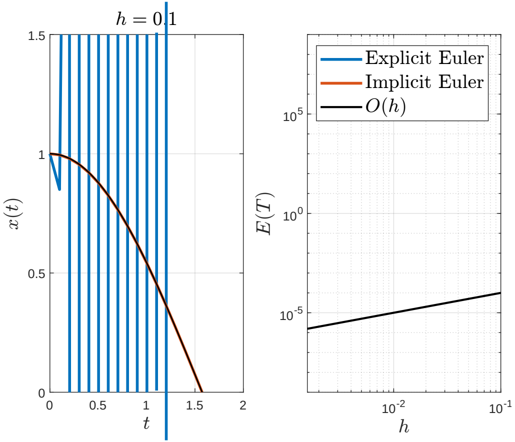
$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq Ch^{p+1} = O(h^{p+1}), C > 0$$

- Stability: damping of errors, does it work for $h \gg 0$?

- If integrator is unstable, it does not converge and has $p = 0$, unless h very small



$$\dot{x}(t) = -300(x(t) - \cos(t)), \;\; t \in [0, 2]$$
$$x(0) = 1$$

Stability and convergence

Integrator convergence and accuracy

- Convergence

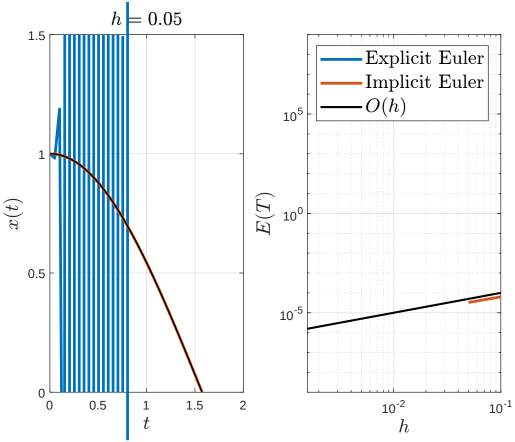
$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq C h^{p+1} = O(h^{p+1}), C > 0$$

- Stability: damping of errors, does it work for $h \gg 0$?

- If integrator is unstable, it does not converge and has $p = 0$, unless h very small



$$\dot{x}(t) = -300(x(t) - \cos(t)),\ \ t \in [0, 2]$$
$$x(0) = 1$$

Stability and convergence

Integrator convergence and accuracy

- Convergence

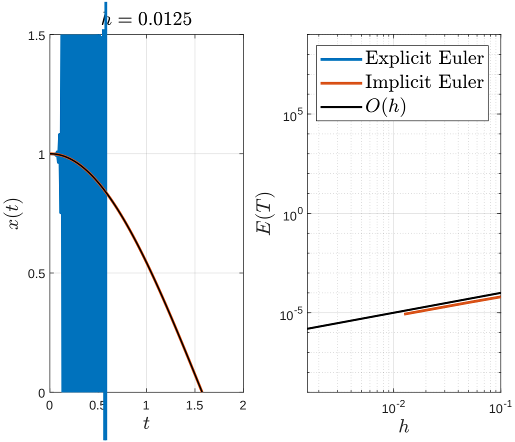
$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq C h^{p+1} = O(h^{p+1}), C > 0$$

- Stability: damping of errors, does it work for $h \gg 0$?

- If integrator is unstable, it does not converge and has $p = 0$, unless h very small



$$\dot{x}(t) = -300(x(t) - \cos(t)), \ t \in [0, 2]$$
$$x(0) = 1$$

Stability and convergence

Integrator convergence and accuracy

- Convergence

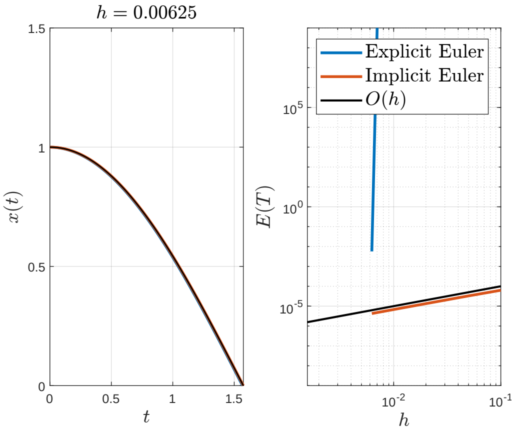
$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq C h^{p+1} = O(h^{p+1}), C > 0$$

- Stability: damping of errors, does it work for $h \gg 0$?

- If integrator is unstable, it does not converge and has $p = 0$, unless h very small



$$\dot{x}(t) = -300(x(t) - \cos(t)), \ t \in [0, 2]$$

$$x(0) = 1$$

Stability and convergence

Integrator convergence and accuracy

- Convergence

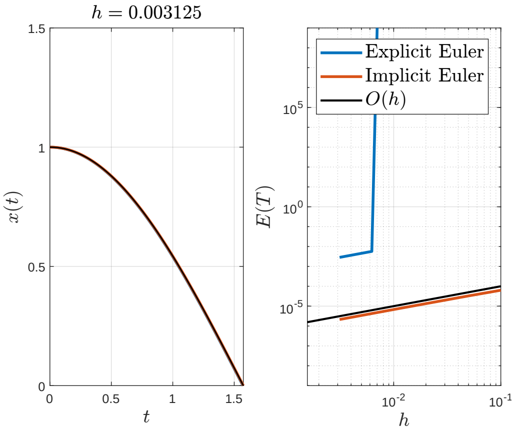
$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq C h^{p+1} = O(h^{p+1}), C > 0$$

- Stability: damping of errors, does it work for $h \gg 0$?

- If integrator is unstable, it does not converge and has $p = 0$, unless h very small



$$\dot{x}(t) = -300(x(t) - \cos(t)), \ t \in [0, 2]$$
$$x(0) = 1$$

Stability and convergence

Integrator convergence and accuracy

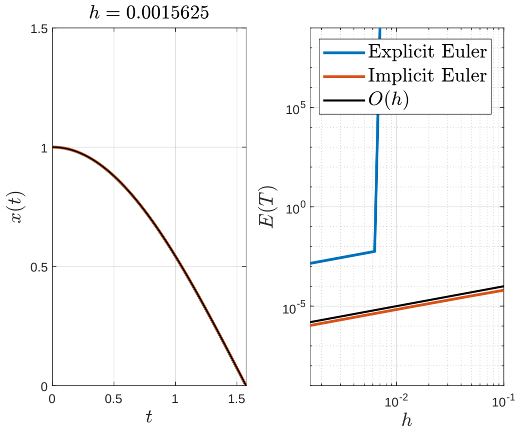
- Convergence

$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq C h^{p+1} = O(h^{p+1}), C > 0$$

- Stability: damping of errors, does it work for $h \gg 0$?



- If integrator is unstable, it does not converge and has $p = 0$, unless h very small

$$\dot{x}(t) = -300(x(t) - \cos(t)), \;\; t \in [0, 2]$$
$$x(0) = 1$$

Runge-Kutta method examples

Explicit Runge-Kutta of order 4

$$
\begin{aligned}
k_{n,1} &= f\left(t_n, x_n\right) \\
k_{n,2} &= f\left(t_n + \frac{h}{2}, x_n + h\frac{k_{n,1}}{2}\right) \\
k_{n,3} &= f\left(t_n + \frac{h}{2}, x_n + h\frac{k_{n,2}}{2}\right) \\
k_{n,5} &= f\left(t_n + h, x_n + hk_{n,3}\right) \\
x_{n+1} &= x_n + h\left(\frac{1}{6}k_{n,1} + \frac{2}{6}k_{n,2} + \frac{2}{6}k_{n,3} + \frac{1}{6}k_{n,4}\right)
\end{aligned}
$$

- All $k_{n,i}$ can be found by explicit function evaluations.

Runge-Kutta method examples

Explicit Runge-Kutta of order 4

$$
\begin{aligned}
k_{n,1} &= f\left(t_n, x_n\right) \\
k_{n,2} &= f\left(t_n + \frac{h}{2}, x_n + h\frac{k_{n,1}}{2}\right) \\
k_{n,3} &= f\left(t_n + \frac{h}{2}, x_n + h\frac{k_{n,2}}{2}\right) \\
k_{n,5} &= f\left(t_n + h, x_n + hk_{n,3}\right) \\
x_{n+1} &= x_n + h\left(\frac{1}{6}k_{n,1} + \frac{2}{6}k_{n,2} + \frac{2}{6}k_{n,3} + \frac{1}{6}k_{n,4}\right)
\end{aligned}
$$

- All $k_{n,i}$ can be found by explicit function evaluations.

Implicit Euler Method

$$
\begin{aligned}
k_{n,1} &= f(t_n, x_n + hk_{n,1}) \\
x_{n+1} &= x_n + hk_{n,1}
\end{aligned}
$$

- $k_{n,1}$ is found implicitly by solving $k_{n,1} - f(t_n, x_n + hk_{n,1}) = 0$.

Continuous time OCP into Nonlinear Programs (NLP)

Continuous time OCP

$$\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_c(x(t),u(t))\,\mathrm{d}t + M(x(T)) \\
\text{s.t.} \quad & x(0) = \bar{x}_0 \\
& \dot{x}(t) = f(x(t),u(t)) \\
& 0 \geq h(x(t),u(t)),\ t \in [0,T] \\
& 0 \geq r(x(T))
\end{aligned}$$

- Direct methods: first discretize, then optimize

Continuous time OCP into Nonlinear Programs (NLP)

Continuous time OCP

$$\min_{x(\cdot),u(\cdot)} \int_0^T L_\mathrm{c}(x(t),u(t))\,\mathrm{d}t + M(x(T))$$

$$\begin{aligned} \text{s.t.} \quad & x(0) = \bar{x}_0 \\ & \dot{x}(t) = f(x(t),u(t)) \\ & 0 \geq h(x(t),u(t)),\ t \in [0,T] \\ & 0 \geq r(x(T)) \end{aligned}$$

- Direct methods: first discretize, then optimize

1. Parametrize controls, e.g. $u(t) = u_n, t \in [t_n, t_{n+1}]$.

Continuous time OCP into Nonlinear Programs (NLP)

Continuous time OCP

$$\begin{aligned}\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_c(x(t),u(t))\,\mathrm{d}t + M(x(T)) \\ \text{s.t.} \quad & x(0) = \bar{x}_0 \\ & \dot{x}(t) = f(x(t),u(t)) \\ & 0 \geq h(x(t),u(t)),\ t \in [0,T] \\ & 0 \geq r(x(T))\end{aligned}$$

- Direct methods: first discretize, then optimize

1. Parametrize controls, e.g. $u(t) = u_n, t \in [t_n, t_{n+1}]$.
2. Discretize cost and dynamics

$$l(x_n,u_n) \approx \int_{t_n}^{t_{n+1}} L_c(x(t),u(t))\,\mathrm{d}t.$$

Replace $\dot{x} = f(x,u)$ by

$$x_{n+1} = \phi(x_n,u_n).$$

Continuous time OCP into Nonlinear Programs (NLP)

Continuous time OCP

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_c(x(t),u(t))\,\mathrm{d}t + M(x(T)) \\
\text{s.t.} \quad & x(0) = \bar{x}_0 \\
& \dot{x}(t) = f(x(t),u(t)) \\
& 0 \geq h(x(t),u(t)),\ t \in [0,T] \\
& 0 \geq r(x(T))
\end{aligned}
$$

- Direct methods: first discretize, then optimize

1. Parametrize controls, e.g. $u(t) = u_n, t \in [t_n, t_{n+1}]$.
2. Discretize cost and dynamics

$$
l(x_n,u_n) \approx \int_{t_n}^{t_{n+1}} L_c(x(t),u(t))\,\mathrm{d}t.
$$

Replace $\dot{x} = f(x,u)$ by

$$
x_{n+1} = \phi(x_n,u_n).
$$

3. Relax path constraints, e.g., evaluate only at $t = t_n$

$$
0 \geq h(x_n,u_n),\ n = 0,\ldots N-1.
$$

Continuous time OCP into Nonlinear Programs (NLP)

Continuous time OCP

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_c(x(t),u(t))\,\mathrm{d}t + M(x(T)) \\
\text{s.t.} \quad & x(0) = \bar{x}_0 \\
& \dot{x}(t) = f(x(t),u(t)) \\
& 0 \geq h(x(t),u(t)), \; t \in [0,T] \\
& 0 \geq r(x(T))
\end{aligned}
$$

- Direct methods: first discretize, then optimize

Discrete time OCP (an NLP)

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{N-1} l(x_k,u_k) + M(x_N) \\
\text{s.t.} \quad & x_0 = \bar{x}_0 \\
& x_{n+1} = \phi(x_n,u_n) \\
& 0 \geq h(x_n,u_n), \; n = 0,\ldots,N-1 \\
& 0 \geq r(x_N)
\end{aligned}
$$

Variables $\mathbf{x} = (x_0,\ldots,x_N)$ and $\mathbf{u} = (u_0,\ldots,u_{N-1})$.

Direct optimal control methods solve Nonlinear Programs (NLP)

Discrete time OCP – Multiple Shooting Formulation

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{u}} \quad & \textstyle\sum_{k=0}^{N-1} l(x_k, u_k) + E(x_N) \\
\text{s.t.} \quad x_0 &= \bar{x}_0 \\
x_{n+1} &= \phi(x_n, u_n) \\
0 &\geq h(x_n, u_n), \; n = 0, \dots, N-1 \\
0 &\geq r(x_N)
\end{aligned}
$$

Variables $w = (\mathbf{x}, \mathbf{u})$

Direct optimal control methods solve Nonlinear Programs (NLP)

Discrete time OCP – Multiple Shooting Formulation

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{u}} \quad & \textstyle\sum_{k=0}^{N-1} l(x_k, u_k) + E(x_N) \\
\text{s.t.} \quad & x_0 = \bar{x}_0 \\
& x_{n+1} = \phi(x_n, u_n) \\
& 0 \geq h(x_n, u_n), \; n = 0, \ldots, N-1 \\
& 0 \geq r(x_N)
\end{aligned}
$$

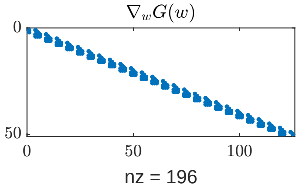
Variables $w = (\mathbf{x}, \mathbf{u})$

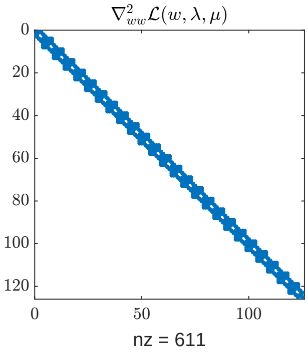
Nonlinear Program (NLP)

$$
\begin{aligned}
\min_{w \in \mathbb{R}^{n_x}} \quad & F(w) \\
\text{s.t.} \quad & G(w) = 0 \\
& H(w) \geq 0
\end{aligned}
$$

Obtain large and sparse NLP

Direct optimal control methods solve Nonlinear Programs (NLP)





Nonlinear Program (NLP)

$$\min_{w \in \mathbb{R}^{n_x}} \; F(w)$$
$$\text{s.t. } G(w) = 0$$
$$H(w) \geq 0$$

Obtain large and sparse NLP

Variables $w = (\mathbf{x}, \mathbf{u})$

Direct optimal control methods solve Nonlinear Programs (NLP)

Discrete time OCP – Collocation Formulation

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{k},\mathbf{u}} \quad & \textstyle\sum_{k=0}^{N-1} l(x_k, u_k) + E(x_N) \\
\text{s.t.} \quad x_0 &= \bar{x}_0 \\
x_{n+1} &= \phi(x_n, u_n, k_n) \\
0 &= \phi_{\text{coll}}(x_n, u_n, k_n) \\
0 &\geq h(x_n, u_n), \; n = 0, \dots, N-1 \\
0 &\geq r(x_N)
\end{aligned}
$$

Variables $w = (\mathbf{x}, \mathbf{k}, \mathbf{u})$

Direct optimal control methods solve Nonlinear Programs (NLP)

Discrete time OCP – Collocation Formulation

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{k},\mathbf{u}} \ & \textstyle\sum_{k=0}^{N-1} l(x_k, u_k) + E(x_N) \\
\text{s.t.} \quad & x_0 = \bar{x}_0 \\
& x_{n+1} = \phi(x_n, u_n, k_n) \\
& 0 = \phi_{\text{coll}}(x_n, u_n, k_n) \\
& 0 \geq h(x_n, u_n), \ n = 0, \dots, N-1 \\
& 0 \geq r(x_N)
\end{aligned}
$$

Variables $w = (\mathbf{x}, \mathbf{k}, \mathbf{u})$

Nonlinear Program (NLP)

$$
\begin{aligned}
\min_{w \in \mathbb{R}^{n_x}} \ & F(w) \\
\text{s.t.} \ & G(w) = 0 \\
& H(w) \geq 0
\end{aligned}
$$

Obtain large and sparse NLP

Summary

- Numerical simulation methods (integrators) used to solve ODEs and DAEs approximately.
- Integration accuracy order and stability play key roles.
- Within the multiple shooting framework, integrators are a key building block for discretization of the continuous OCP.
- The resulting discrete-time OCP is large, but very sparse

acados is an open-source software package for nonlinear optimal control developed and maintain by the group of Prof. Diehl.

acados provides several building blocks for nonlinear optimal control

- Integrators for ODEs and DAEs
  - explicit and (structure-exploiting) implicit Runge-Kutta schemes
  - efficient sensitivity propagation
- SQP-type solver for nonlinear optimal control problems
  - Hessian approximation exploiting convex-over-nonlinear structures in costs and constraints
  - real-time iteration
  - (partial) condensing routines
- Interfaces to state-of-the-art QP solvers
  - `HPIPM`, `qpOASES`, `qpDUNES`, `OSQP`, `DAQP`
- Generation of self-contained `C` code for embedded deployment as well as convenient user interfaces to `MATLAB` and `python`.

Open-Source Dependencies/Foundations

acados builds on

- CasADi[2] for describing the problem functions and their derivatives via algorithmic differentiation (AD)
- HPIPM[2] for efficient condensing routines
- BLASFEO[3] for high-performance linear algebra tailored to the embedded hardware
- various open-source QP solvers, HPIPM[2], qpOASES[4], qpDUNES[5], OSQP[6], DAQP, for solving the SQP-subproblems

[2]Andersson et al., 2019; [2]Frison & Diehl, 2020; [3]Frison et al., 2018; [4]Ferreau et al., 2014; [5]Frasch et al., 2015; [6]Stellato et al., 2020; [7]Arnstrom et al., 2022;

Recent applications of `acados`

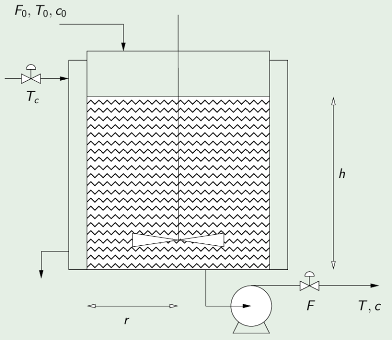
Recent applications of `acados` in real-world experiments.

- Obstacle Avoidance for Mobile Robotics (Gao et al., 2023)
- Quadrotor Control (Salzmann et al., 2023; Romero et al., 2022; Carlos et al., 2020)
- Combustion Engine and Air Path Control (Hänggi et al., 2022; Gordon et al., 2022)
- Electric Motor Control (Zanelli et al., 2021)

Advanced NMPC problem formulations and implementations.

- Robust MPC (Gao et al., 2023)
- Deep Neural Networks (DNN) and Gaussian Processes (GP) as dynamics model (Salzmann et al., 2023; Lahr et al., 2023)
- Convenient and efficient access to the SQP subproblem for custom modifications (Frey, Gao, et al., 2023)
- Custom sensitivity propagation for accurate cost integration for convex-over-nonlinear costs (Frey, Baumgärtner, & Diehl, 2023)

- We consider a continuously stirred tank reactor as in Pannocchia & Rawlings (2003).
- An irreversible, first-order reaction $A \to B$ occurs in the liquid phase and the reactor temperature is regulated with external cooling.



Mass and energy balances lead to the following nonlinear state space model:

$$\dot{c} = \frac{F_0(c_0 - c)}{\pi r^2 h} - k_0 \exp\left(-\frac{E}{RT}\right) c$$

$$\dot{T} = \frac{F_0(T_0 - T)}{\pi r^2 h} - \frac{\Delta H}{\rho C_p} k_0 \exp\left(-\frac{E}{RT}\right) c + \frac{2U}{r \rho C_p}(T_c - T)$$

$$\dot{h} = \frac{F_0 - F}{\pi r^2}$$

- The controls are T_c, the coolant liquid temperature, and F, the outlet flowrate.

References I

Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., & Diehl, M. (2019). CasADi – a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, *11*(1), 1–36. doi: 10.1007/s12532-018-0139-4

Arnstrom, D., Bemporad, A., & Axehill, D. (2022). A dual active-set solver for embedded quadratic programming using recursive LDL^T updates. *IEEE Transactions on Automatic Control*. doi: 10.1109/TAC.2022.3176430

Carlos, B. B., Sartor, T., Zanelli, A., Frison, G., Burgard, W., Diehl, M., & Oriolo, G. (2020). An efficient real-time nmpc for quadrotor position control under communication time-delay. In *2020 16th international conference on control, automation, robotics and vision (icarcv)* (p. 982-989). doi: 10.1109/ICARCV50220.2020.9305513

Ferreau, H. J., Kirches, C., Potschka, A., Bock, H. G., & Diehl, M. (2014). qpOASES: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, *6*(4), 327–363.

Frasch, J. V., Sager, S., & Diehl, M. (2015). A parallel quadratic programming method for dynamic optimization problems. *Mathematical Programming Computations*, *7*(3), 289–329.

References II

Frey, J., Baumgärtner, K., & Diehl, M. (2023). Gauss-newton runge-kutta integration for efficient discretization of optimal control problems with long horizons and least-squares costs..

Frey, J., Gao, Y., Messerer, F., Lahr, A., Zeilinger, M., & Diehl, M. (2023). Efficient zero-order robust optimization for real-time model predictive control with acados..

Frison, G., & Diehl, M. (2020, July). HPIPM: a high-performance quadratic programming framework for model predictive control. In *Proceedings of the ifac world congress.* Berlin, Germany.

Frison, G., Kouzoupis, D., Sartor, T., Zanelli, A., & Diehl, M. (2018). BLASFEO: Basic linear algebra subroutines for embedded optimization. *ACM Transactions on Mathematical Software (TOMS)*, *44*(4), 42:1–42:30.

Gao, Y., Messerer, F., Frey, J., van Duijkeren, N., & Diehl, M. (2023). Collision-free motion planning for mobile robots by zero-order robust optimization-based mpc. In *Proceedings of the european control conference (ecc).*

Gordon, D. C., Norouzi, A., Winkler, A., McNally, J., Nuss, E., Abel, D., ... Koch, C. R. (2022). End-to-end deep neural network based nonlinear model predictive control: experimental implementation on diesel engine emission control. *Energies*, *15*(24), 9335.

Hänggi, S., Frey, J., van Dooren, S., Diehl, M., & Onder, C. H. (2022). A modular approach for diesel engine air path control based on nonlinear mpc. *IEEE Transactions on Control Systems Technology*, 1-16. doi: 10.1109/TCST.2022.3228203

Lahr, A., Zanelli, A., Carron, A., & Zeilinger, M. N. (2023). Zero-order optimization for Gaussian process-based model predictive control. *European Journal of Control*, 100862.

Romero, A., Penicka, R., & Scaramuzza, D. (2022). Time-optimal online replanning for agile quadrotor flight. *IEEE Robotics and Automation Letters*, *7*(3), 7730–7737.

Salzmann, T., Kaufmann, E., Arrizabalaga, J., Pavone, M., Scaramuzza, D., & Ryll, M. (2023). Real-time neural MPC: Deep learning model predictive control for quadrotors and agile robotic platforms. *IEEE Robotics and Automation Letters*, *8*(4), 2397–2404.

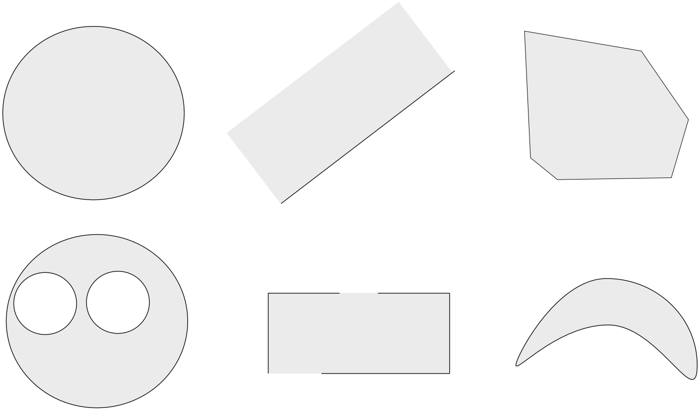
Stellato, B., Banjac, G., Goulart, P., Bemporad, A., & Boyd, S. (2020). OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, *12*(4), 637–672. Retrieved from `https://doi.org/10.1007/s12532-020-00179-2` doi: 10.1007/s12532-020-00179-2

Zanelli, A., Kullick, J., Eldeeb, H., Frison, G., Hackl, C., & Diehl, M. (2021). Continuous control set nonlinear model predictive control of reluctance synchronous machines. *IEEE Transactions on Control Systems Technology*, 1-12. doi: 10.1109/TCST.2020.3043956

Appendix

Convex sets

A key concept in optimization is convexity



A set Ω is said to be convex if for any w_1, w_2 and any $\theta \in [0,1]$ it holds $\theta w_1 + (1-\theta) w_2 \in \Omega$

Convex functions

- A function F is convex if for every $w_1, w_2 \in \mathbb{R}^n$ and $\theta \in [0, 1]$ it holds that

$$F(\theta w_1 + (1-\theta) w_2) \leq \theta F(w_1) + (1-\theta) F(w_2)$$

- F is concave if and only if $-F$ is convex
- F is convex if and only if the epigraph

$$\text{epi}F = \{(w, t) \in \mathbb{R}^{n_w+1} \mid F(w) \leq t\}$$

is a convex set