Modelling and Solving Large-Scale Mathematical Programs with Complementarity Constraints

Anton E. Pozharskiy[1], Armin Nurkanović[1], François Pacaud[2], Moritz Diehl[1]

[1]Systems, Control, and Optimization Laboratory,
Department of Microsystems Engineering,
University of Freiburg, Germany

[2]Centre Automatique et Systèmes
Ecolé des Mines de Paris, France

JuMP-Dev 2026
University of Edinburgh
31st May, 2026, Edinburgh, UK

Outline

Outline

Mathematical Programs with Complementarity Constraints

MPCC

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_x}} \quad & f(x) \\ \text{s.t.} \quad & \\ 0 &= c(x), \\ 0 &\leq x, \\ 0 &\leq G(x) \perp H(x) \geq 0 \end{aligned}$$

- Recall that $a \perp b \coloneqq a_i b_i = 0$ for all i.
- Violates MFCQ at all feasible points.
- Complementarity feasible set is nonsmooth and non-differentiable.

Mathematical Programs with Complementarity Constraints

MPCC

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_x}} \quad & f(x) \\ \text{s.t.} \quad & \\ 0 &= c(x), \\ 0 &\leq x, \\ 0 &\leq G(x) \perp H(x) \geq 0 \end{aligned}$$

- Recall that $a \perp b := a_i b_i = 0$ for all i.
- Violates MFCQ at all feasible points.
- Complementarity feasible set is nonsmooth and non-differentiable.

MPCC in Vertical Form

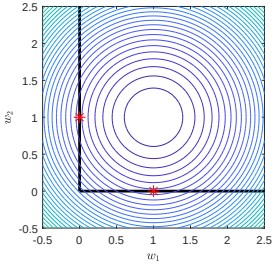
$$\begin{aligned} \min_{x \in \mathbb{R}^{n_x}} \quad & f(x) \\ \text{s.t.} \quad & \\ 0 &= c(x), \\ 0 &\leq x_0, \\ 0 &\leq x_1 \perp x_2 \geq 0 \end{aligned}$$

- Alternative form with variable partition $x = (x_0, x_1, x_2) \in \mathbb{R}^{n_x}$, $x_1, x_2 \in \mathbb{R}^{n_{cc}}$.
- All MPCCs can be written in this form, and we'll use it for the rest of the exposition.

Mathematical Programs with Complementarity Constraints (MPCC)

MPCC as an NLP

$$
\begin{aligned}
\min_{x \in \mathbb{R}^{n_x}} \quad & f(x) \\
\text{s.t.} \quad & \\
0 &= c(x), \\
0 &\leq x_0, \\
0 &\leq x_1, \\
0 &\leq x_2, \\
0 &\geq x_{1,i} x_{2,i}, \quad i = 1, \ldots, n_{cc}
\end{aligned}
$$



Toy MPCC example:

$$
\begin{aligned}
\min_{x \in \mathbb{R}^{n_x}} \quad & (x_1 - 1)^2 + (x_2 - 1)^2 \\
\text{s.t.} \quad & \\
0 &\leq x_1 \perp x_2 \geq 0
\end{aligned}
$$

Two local minimizers.
One local maximizer
(without constraint
qualification)

Outline

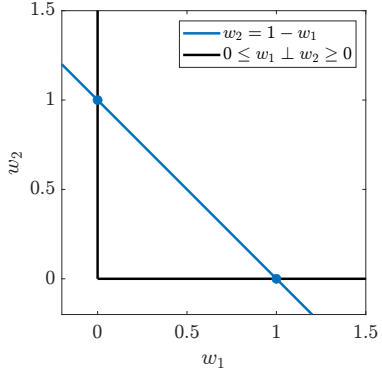
How complementarity constraints should **NOT** be used

Integrality conditions

$$w_1 \in \{0, 1\}$$

are equivalent to

$$w_2 = 1 - w_1$$
$$0 \leq w_1 \perp w_2 \geq 0$$



- Feasible set consists of two isolate points.
- May converge only if initialized very close to solution.
- Problems with disjoint feasible regions should be avoided.
- Bad idea as $\sin(w\pi) = 0$ to obtain $w \in \mathbb{Z}$.

Biegler's guidelines for good MPCC modeling

Can be found in Chapter 11 of [Biegler, 2010].

1. To model discrete decisions, piecewise smooth functions, ..., start with a convex *lower level* problem, e.g.,

$$\min_{y} F(w_0)y \quad \text{s.t.} \quad y_\ell \leq y \leq y_u.$$

   Problem is parametric in switching function $F(w_0)$.

Biegler's guidelines for good MPCC modeling

Can be found in Chapter 11 of [Biegler, 2010].

1. To model discrete decisions, piecewise smooth functions, ..., start with a convex *lower level* problem, e.g.,

$$\min_{y} F(w_0)y \quad \text{s.t.} \quad y_\ell \leq y \leq y_u.$$

   Problem is parametric in switching function $F(w_0)$.

2. Whenever possible, formulate the problem such that constraints on upper level variables w_0 do not interfere with lower level constraints.
   If there are no constraints connecting w_0 and y, for $F(w_0) = 0$, $y \in [y_\ell, y_u]$ - avoiding disconnected feasible sets.

Biegler's guidelines for good MPCC modeling

Can be found in Chapter 11 of [Biegler, 2010].

1. To model discrete decisions, piecewise smooth functions, ..., start with a convex *lower level* problem, e.g.,

$$\min_{y} F(w_0)y \quad \text{s.t.} \quad y_\ell \leq y \leq y_u.$$

   Problem is parametric in switching function $F(w_0)$.

2. Whenever possible, formulate the problem such that constraints on upper level variables w_0 do not interfere with lower level constraints.
   If there are no constraints connecting w_0 and y, for $F(w_0) = 0$, $y \in [y_\ell, y_u]$ - avoiding disconnected feasible sets.

3. Apply KKT conditions, to reformulate to MPCC:

$$\begin{aligned} &F(w_0) - \lambda_\ell + \lambda_u = 0, \\ &0 \leq y - y_\ell \perp \lambda_\ell \geq 0, \\ &0 \leq y_u - y \perp \lambda_u \geq 0. \end{aligned}$$

Biegler's guidelines for good MPCC modeling

Can be found in Chapter 11 of [Biegler, 2010].

1. To model discrete decisions, piecewise smooth functions, ..., start with a convex *lower level* problem, e.g.,

$$\min_{y} F(w_0)y \quad \text{s.t.} \quad y_\ell \leq y \leq y_u.$$

   Problem is parametric in switching function $F(w_0)$.

2. Whenever possible, formulate the problem such that constraints on upper level variables w_0 do not interfere with lower level constraints.
   If there are no constraints connecting w_0 and y, for $F(w_0) = 0$, $y \in [y_\ell, y_u]$ - avoiding disconnected feasible sets.

3. Apply KKT conditions, to reformulate to MPCC:

$$\begin{aligned} &F(w_0) - \lambda_\ell + \lambda_u = 0, \\ &0 \leq y - y_\ell \perp \lambda_\ell \geq 0, \\ &0 \leq y_u - y \perp \lambda_u \geq 0. \end{aligned}$$

4. If possible, do some variable eliminations in the KKT conditions.

Modeling the $\mathrm{sign}(\cdot)$ and Heaviside step function

Set-valued Heaviside step function

$$\mathrm{step}(F(w)) = \begin{cases} \{1\}, & F(w) > 0, \\ [0,1], & F(w) = 0, \\ \{0\}, & F(w) < 0. \end{cases}$$

$$\mathrm{step}(F(w)) = \underset{y \in \mathbb{R}}{\operatorname{argmin}} \quad -F(w)y \tag{1a}$$

$$\text{s.t.} \quad 0 \leq y \leq 1. \tag{1b}$$

Modeling the $\operatorname{sign}(\cdot)$ and Heaviside step function

Set-valued Heaviside step function

$$\operatorname{step}(F(w)) = \begin{cases} \{1\}, & F(w) > 0, \\ [0,1], & F(w) = 0, \\ \{0\}, & F(w) < 0. \end{cases}$$

$$\operatorname{step}(F(w)) = \underset{y \in \mathbb{R}}{\operatorname{argmin}} \quad -F(w)y \tag{1a}$$

$$\text{s.t.} \quad 0 \leq y \leq 1. \tag{1b}$$

KKT conditions of the LP (1):

$$\begin{aligned} &F(w) = \lambda_{\mathrm{p}} - \lambda_{\mathrm{n}}, \\ &0 \leq y \perp \lambda_{\mathrm{n}} \geq 0, \\ &0 \leq 1 - y \perp \lambda_{\mathrm{p}} \geq 0. \end{aligned}$$

Modeling the $\operatorname{sign}(\cdot)$ and Heaviside step function

Set-valued Heaviside step function

$$\operatorname{step}(F(w)) = \begin{cases} \{1\}, & F(w) > 0, \\ [0,1], & F(w) = 0, \\ \{0\}, & F(w) < 0. \end{cases}$$

$$\operatorname{step}(F(w)) = \underset{y \in \mathbb{R}}{\operatorname{argmin}} \quad -F(w)y \tag{1a}$$

$$\text{s.t.} \quad 0 \le y \le 1. \tag{1b}$$

KKT conditions of the LP (1):

$$\begin{aligned} &F(w) = \lambda_{\mathrm{p}} - \lambda_{\mathrm{n}}, \\ &0 \le y \perp \lambda_{\mathrm{n}} \ge 0, \\ &0 \le 1 - y \perp \lambda_{\mathrm{p}} \ge 0. \end{aligned}$$

- For $F(w) > 0 \implies y = 1$, $\lambda_{\mathrm{n}} = 0$ positive part: $\lambda_{\mathrm{p}} = \max(0, F(w))$.
- For $F(w) < 0 \implies y = 0$, $\lambda_{\mathrm{p}} = 0$, negative part: $\lambda_{\mathrm{n}} = \max(0, -F(w))$.
- $\operatorname{sign}(F(w))$ by setting lower bound to -1 in LP (1), in this case $|F(w)| = F(w)y$.
- Cannot model discontinuous functions, only their set-valued extensions.

Modeling other common nonsmooth functions

The $\mathrm{abs}(\cdot)$ function:

- To model $z = |F(w)|$, use single complementarity constraints:

$$F(w) = \lambda^{\mathrm{p}} - \lambda^{\mathrm{n}}$$
$$0 \leq \lambda^{\mathrm{p}} \perp \lambda^{\mathrm{n}} \geq 0$$

- Combining the positive and negative part, $z = \lambda^{\mathrm{p}} + \lambda^{\mathrm{n}}$

Modeling other common nonsmooth functions

The $\mathrm{abs}(\cdot)$ function:

- To model $z = |F(w)|$, use single complementarity constraints:

$$F(w) = \lambda^{\mathrm{p}} - \lambda^{\mathrm{n}}$$
$$0 \leq \lambda^{\mathrm{p}} \perp \lambda^{\mathrm{n}} \geq 0$$

- Combining the positive and negative part, $z = \lambda^{\mathrm{p}} + \lambda^{\mathrm{n}}$

The $\max(\cdot, \cdot)$ function: exploit that

$$z = \max(F_1(w), F_2(w)) = F_1(w) + \min(0, -(F_1(w) - F_2(w)))$$

and use that

Modeling other common nonsmooth functions

The $\mathrm{abs}(\cdot)$ function:

- To model $z = |F(w)|$, use single complementarity constraints:

$$F(w) = \lambda^{\mathrm{p}} - \lambda^{\mathrm{n}}$$
$$0 \leq \lambda^{\mathrm{p}} \perp \lambda^{\mathrm{n}} \geq 0$$

- Combining the positive and negative part, $z = \lambda^{\mathrm{p}} + \lambda^{\mathrm{n}}$

The $\max(\cdot, \cdot)$ function: exploit that

$$z = \max(F_1(w), F_2(w)) = F_1(w) + \min(0, -(F_1(w) - F_2(w)))$$

and use that

- $F_1(w) > F_2(w) \implies \min(0, -(F_1(w) - F_2(w))) = 0.$
- $F_1(w) < F_2(w) \implies \min(0, -(F_1(w) - F_2(w))) = -F_1(w) + F_2(w).$
- Using the positive and negative part of $F_1(w) - F_2(w)$ we have

$$z = F_1(w) + \lambda_{\mathrm{n}},$$
$$F_1(w) - F_2(w) = \lambda_{\mathrm{p}} - \lambda_{\mathrm{n}},$$
$$0 \leq \lambda_{\mathrm{p}} \perp \lambda_{\mathrm{n}} \geq 0.$$

Modeling other common nonsmooth functions

The $\mathrm{abs}(\cdot)$ function:

- To model $z = |F(w)|$, use single complementarity constraints:

$$F(w) = \lambda^{\mathrm{p}} - \lambda^{\mathrm{n}}$$
$$0 \leq \lambda^{\mathrm{p}} \perp \lambda^{\mathrm{n}} \geq 0$$

- Combining the positive and negative part, $z = \lambda^{\mathrm{p}} + \lambda^{\mathrm{n}}$

The $\max(\cdot, \cdot)$ function: exploit that

$$z = \max(F_1(w), F_2(w)) = F_1(w) + \min(0, -(F_1(w) - F_2(w)))$$

and use that

- $F_1(w) > F_2(w) \implies \min(0, -(F_1(w) - F_2(w))) = 0.$
- $F_1(w) < F_2(w) \implies \min(0, -(F_1(w) - F_2(w))) = -F_1(w) + F_2(w).$
- Using the positive and negative part of $F_1(w) - F_2(w)$ we have

$$z = F_1(w) + \lambda_{\mathrm{n}},$$
$$F_1(w) - F_2(w) = \lambda_{\mathrm{p}} - \lambda_{\mathrm{n}},$$
$$0 \leq \lambda_{\mathrm{p}} \perp \lambda_{\mathrm{n}} \geq 0.$$

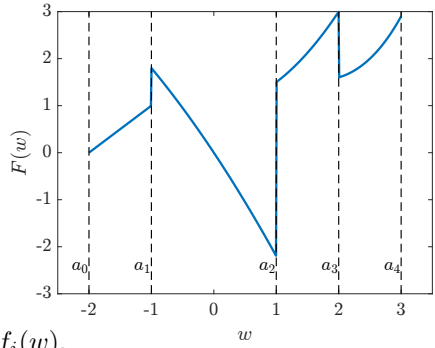
The $\min(\cdot, \cdot)$ function: can be modelled analogously.

Piecewise smooth functions and look-up tables

Introduced in [Raghunathan and Biegler, 2003]

Model a piecewise smooth functions $F : \mathbb{R} \to \mathbb{R}$ over intervals defined by grid points $a_0 < a_1 \ldots < a_n$

$$\begin{aligned} \min_{y \in \mathbb{R}^n} \quad & \sum_{i=1}^{n} (w - a_i)(w - a_{i+1}) y_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} y_i = 1, \\ & y \geq 0. \end{aligned}$$



- Compute selectors variables y_i, and $z = \sum_{i=1}^{n} y_i f_i(w)$.
- If $w \in [a_{i-1}, a_i]$, then $(w - a_i)(w - a_{i+1}) \leq 0$, otherwise positive.
- Generalization to multi dimensional input and output spaces via Stewart's LP (Introduced in [Stewart, 1990]) , very efficient if Voronoi regions can be used.

Some more modelling ideas

Far from exhaustive!

Some more modelling ideas

Far from exhaustive!

State triggered constraints [Szmuk et al., 2020]

If $H_i(w) > 0$ then $G_i(w) = 0$, otherwise if $H_i(w) \leq 0$ then $G_i(w) \in \mathbb{R}$

Some more modelling ideas

Far from exhaustive!

State triggered constraints [Szmuk et al., 2020]

If $H_i(w) > 0$ then $G_i(w) = 0$, otherwise if $H_i(w) \leq 0$ then $G_i(w) \in \mathbb{R}$

Vanishing constraint [Achtziger and Kanzow, 2008]

If $H_i(w) > 0$ then $G_i(w) \leq 0$, else if $H_i(w) = 0$ then $G_i(w) \in \mathbb{R}$ ($G_i(w) \leq 0$ vanishes)

Some more modelling ideas

Far from exhaustive!

State triggered constraints [Szmuk et al., 2020]

If $H_i(w) > 0$ then $G_i(w) = 0$, otherwise if $H_i(w) \leq 0$ then $G_i(w) \in \mathbb{R}$

Vanishing constraint [Achtziger and Kanzow, 2008]

If $H_i(w) > 0$ then $G_i(w) \leq 0$, else if $H_i(w) = 0$ then $G_i(w) \in \mathbb{R}$ ($G_i(w) \leq 0$ vanishes)

Sparsity optimization [Feng et al., 2018]

$$\begin{aligned} \min_{w \in \mathbb{R}^n} \quad & f(w) + \rho \|w\|_0 \\ \text{s.t.} \quad & g(w) = 0, \\ & h(w) \geq 0. \end{aligned}$$

Cardinality Constraints [Kanzow et al., 2024]

$$\|w\|_0 \leq \kappa, \ \kappa \in \mathbb{N}$$

Some more modelling ideas

Far from exhaustive!

State triggered constraints [Szmuk et al., 2020]

If $H_i(w) > 0$ then $G_i(w) = 0$, otherwise if $H_i(w) \leq 0$ then $G_i(w) \in \mathbb{R}$

Vanishing constraint [Achtziger and Kanzow, 2008]

If $H_i(w) > 0$ then $G_i(w) \leq 0$, else if $H_i(w) = 0$ then $G_i(w) \in \mathbb{R}$ ($G_i(w) \leq 0$ vanishes)

Sparsity optimization [Feng et al., 2018]

$$\begin{aligned} \min_{w \in \mathbb{R}^n} \quad & f(w) + \rho \|w\|_0 \\ \text{s.t.} \quad & g(w) = 0, \\ & h(w) \geq 0. \end{aligned}$$

Cardinality Constraints [Kanzow et al., 2024]

$$\|w\|_0 \leq \kappa, \ \kappa \in \mathbb{N}$$

Logical operators for $x, y \geq 0$ [Pozharskiy et al., 2024]

$$\begin{aligned} & z > 0 \ \ (\text{true}), \quad z = 0 \ \ (\text{false}) \\ & z = x \vee y \iff z \geq x, \ z \geq y, \ z \leq x + y, \\ & z = x \wedge y \iff z \leq x, \ z \leq y, \ z \geq x + y - \max(x, y), \\ & x \implies z \iff x \leq z \end{aligned}$$

An incomplete list of MPCC applications

- Optimal control of hybrid/nonsmooth systems [Baumrucker and Biegler, 2009, Guo and Ye, 2016, Vieira et al., 2019, Nurkanović, 2023]
- Optimization with piecewise smooth functions, abs-normal froms [Hegerhorst-Schultchen et al., 2020]
- Bi-level optimization (if the lower level problem is convex) [Kim et al., 2020]
- Modeling of logical constraints [Pozharskiy et al., 2024]
- Inverse optimization [Albrecht and Ulbrich, 2017, Hu et al., 2012]
- Process and chemical engineering [Baumrucker et al., 2008, Biegler, 2010]
- Robotics [Wensing et al., 2023]
- District heating networks [Krug et al., 2021]
- Optimal power flow [Pacaud et al., 2025]

Outline

On solving MPCCs

Solution methods for MPCCs:

1. **Mixed-integer reformulation:** reformulate into equivalent mixed-integer nonlinear program (MINLP).
   - Often impractical for anything but small easy (linear) problems.

On solving MPCCs

Solution methods for MPCCs:

1. **Mixed-integer reformulation:** reformulate into equivalent mixed-integer nonlinear program (MINLP).
   - Often impractical for anything but small easy (linear) problems.
2. **Active set methods**, e.g. `mpecopt`[Nurkanović and Leyffer, 2025].
   - Guaranteed to verify B-stationarity
   - May require many LPCC and NLP solves to converge or needs a "phase one" algorithm.

On solving MPCCs

Solution methods for MPCCs:

1. **Mixed-integer reformulation:** reformulate into equivalent mixed-integer nonlinear program (MINLP).
   - Often impractical for anything but small easy (linear) problems.
2. **Active set methods**, e.g. `mpecopt`[Nurkanović and Leyffer, 2025].
   - Guaranteed to verify B-stationarity
   - May require many LPCC and NLP solves to converge or needs a "phase one" algorithm.
3. **Regularization and penalty methods**.
   - Replace the complementarity constraints with relaxations or penalties.
   - Requires solving a sequence of NLPs.

On solving MPCCs

Solution methods for MPCCs:

1. **Mixed-integer reformulation:** reformulate into equivalent mixed-integer nonlinear program (MINLP).
   - Often impractical for anything but small easy (linear) problems.
2. **Active set methods**, e.g. `mpecopt`[Nurkanović and Leyffer, 2025].
   - Guaranteed to verify B-stationarity
   - May require many LPCC and NLP solves to converge or needs a "phase one" algorithm.
3. **Regularization and penalty methods**.
   - Replace the complementarity constraints with relaxations or penalties.
   - Requires solving a sequence of NLPs.
   - **We do this in a smart way via `CCOpt`.**

The main CCOpt idea

Classic approach: Solve:

REG(τ)

$$
\begin{aligned}
\min_{x \in \mathbb{R}^{n_x}} \quad & f(x) \\
\text{s.t.} \quad & c(x) = 0, \\
& x_0 \geq 0, \\
& x_1 \geq 0, \\
& x_2 \geq 0, \\
& X_1 x_2 - \tau \leq 0
\end{aligned}
$$

in a homotopy, with $\tau \to 0$.

A note on notation: $X = \mathrm{diag}(x)$.

The main CCOpt idea

Classic approach: Solve:

REG(τ)

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_x}} \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0, \\ & x_0 \geq 0, \\ & x_1 \geq 0, \\ & x_2 \geq 0, \\ & X_1 x_2 - \tau \leq 0 \end{aligned}$$

in a homotopy, with $\tau \to 0$.

A note on notation: $X = \mathrm{diag}(x)$.

The intuition behind CCOpt: Like in interior point methods, only solve each $\mathrm{REG}(\tau)$ approximately.

- This corresponds to (when using an interior point solver) driving the barrier parameter μ and Scholtes relaxation τ, simultaneously to zero.
- Inspired by [Raghunathan and Biegler, 2005], which sets $\mu = \tau$.

The main CCOpt idea

Classic approach: Solve:

REG(τ)

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_x}} \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0, \\ & x_0 \geq 0, \\ & x_1 \geq 0, \\ & x_2 \geq 0, \\ & X_1 x_2 - \tau \leq 0 \end{aligned}$$

in a homotopy, with $\tau \to 0$.

A note on notation: $X = \mathrm{diag}(x)$.

The intuition behind CCOpt: Like in interior point methods, only solve each REG(τ) approximately.

- This corresponds to (when using an interior point solver) driving the barrier parameter μ and Scholtes relaxation τ, simultaneously to zero.
- Inspired by [Raghunathan and Biegler, 2005], which sets $\mu = \tau$.
- However, this is not enough to develop a robust and performant solver for MPCCs.

CCOpt's Bag of Tricks

We additionally implement several tricks to improve performance:

The bag of tricks

- Relax lower bounds on x_1 or x_2 based on estimated multipliers to prevent numerical issues caused by the shrinking interior.
- Regularize the indefinite Hessian contributions of the scholtes bounds independently. This means in the case of LPCCs, (convex)-QPCCs, CCOpt avoids expensive refactorizations.
- Advanced monotone and adaptive, non-monotone, update rules for both μ and τ.

CCOpt's Bag of Tricks

We additionally implement several tricks to improve performance:

The bag of tricks

- Relax lower bounds on x_1 or x_2 based on estimated multipliers to prevent numerical issues caused by the shrinking interior.
- Regularize the indefinite Hessian contributions of the scholtes bounds independently. This means in the case of LPCCs, (convex)-QPCCs, CCOpt avoids expensive refactorizations.
- Advanced monotone and adaptive, non-monotone, update rules for both μ and τ.

The results

CCOpt's Bag of Tricks

We additionally implement several tricks to improve performance:

The bag of tricks

- Relax lower bounds on x_1 or x_2 based on estimated multipliers to prevent numerical issues caused by the shrinking interior.
- Regularize the indefinite Hessian contributions of the scholtes bounds independently. This means in the case of LPCCs, (convex)-QPCCs, CCOpt avoids expensive refactorizations.
- Advanced monotone and adaptive, non-monotone, update rules for both μ and τ.

The results

- On QPCCs from Real-Time non-smooth MPC [Nurkanović et al., 2026]: 10-50 times speedup over (Commercial) Mixed Integer (Gurobi), and QPCC tailored solvers (LCQPow, [Hall et al., 2024]), with equivalent robustness.

CCOpt's Bag of Tricks

We additionally implement several tricks to improve performance:

The bag of tricks

- Relax lower bounds on x_1 or x_2 based on estimated multipliers to prevent numerical issues caused by the shrinking interior.
- Regularize the indefinite Hessian contributions of the scholtes bounds independently. This means in the case of LPCCs, (convex)-QPCCs, CCOpt avoids expensive refactorizations.
- Advanced monotone and adaptive, non-monotone, update rules for both μ and τ.

The results

- On QPCCs from Real-Time non-smooth MPC [Nurkanović et al., 2026]: 10-50 times speedup over (Commercial) Mixed Integer (Gurobi), and QPCC tailored solvers (LCQPow, [Hall et al., 2024]), with equivalent robustness.
- On MPCCs from security-constrained optimal power flow [Pacaud et al., 2025]: 10 times speedup over Knitro (at the cost of some robustness). 1.1 to 10 times speedup on MPCCS from NOSBENCH [Nurkanović et al., 2024].

CCOpt's Bag of Tricks

We additionally implement several tricks to improve performance:

The bag of tricks

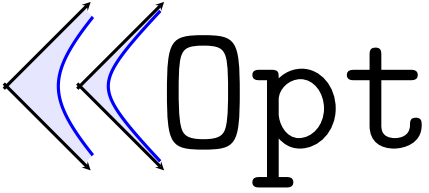
- Relax lower bounds on x_1 or x_2 based on estimated multipliers to prevent numerical issues caused by the shrinking interior.
- Regularize the indefinite Hessian contributions of the scholtes bounds independently. This means in the case of LPCCs, (convex)-QPCCs, CCOpt avoids expensive refactorizations.
- Advanced monotone and adaptive, non-monotone, update rules for both μ and τ.

The results

- On QPCCs from Real-Time non-smooth MPC [Nurkanović et al., 2026]: 10-50 times speedup over (Commercial) Mixed Integer (Gurobi), and QPCC tailored solvers (LCQPow, [Hall et al., 2024]), with equivalent robustness.
- On MPCCs from security-constrained optimal power flow [Pacaud et al., 2025]: 10 times speedup over Knitro (at the cost of some robustness). 1.1 to 10 times speedup on MPCCS from NOSBENCH [Nurkanović et al., 2024].
- More detailed results in [Pozharskiy et al., 2026].

Using CCOpt

We are still in the process of registering the CCOpt package, so it can be installed via:



To use via JuMP.jl: use NLPModelsJuMP.jl (and soon MPCCModels.jl, but an MOI interface is coming soon thanks to François Pacaud, and Benoît Legat!

Conclusions and future work

Conclusions

- 2x to 50x wall time speedups when compared to well tuned homotopy solvers, tailored QPCC solvers, and commercial MIQP solvers.

Conclusions and future work

Conclusions

- 2x to 50x wall time speedups when compared to well tuned homotopy solvers, tailored QPCC solvers, and commercial MIQP solvers.
- Work on a `CCOpt` `MOI` interface is ongoing, bringing an open source local solver for problems with `MOI.complements`!

Conclusions and future work

Conclusions

- 2x to 50x wall time speedups when compared to well tuned homotopy solvers, tailored QPCC solvers, and commercial MIQP solvers.
- Work on a `CCOpt` `MOI` interface is ongoing, bringing an open source local solver for problems with `MOI.complements`!
- Will interface with `ComplementOpt.jl`, via `MOI`.

Conclusions and future work

Conclusions

- 2x to 50x wall time speedups when compared to well tuned homotopy solvers, tailored QPCC solvers, and commercial MIQP solvers.
- Work on a `CCOpt` `MOI` interface is ongoing, bringing an open source local solver for problems with `MOI.complements`!
- Will interface with `ComplementOpt.jl`, via `MOI`.

Future work

Conclusions and future work

Conclusions

- 2x to 50x wall time speedups when compared to well tuned homotopy solvers, tailored QPCC solvers, and commercial MIQP solvers.
- Work on a `CCOpt` `MOI` interface is ongoing, bringing an open source local solver for problems with `MOI.complements`!
- Will interface with `ComplementOpt.jl`, via `MOI`.

Future work

- Support specialized GPU acceleration. (Currently can be done through equality relaxation and a "Condensed" KKT system, which limits accuracy, more work neede to avoid transfer overhead).

Conclusions and future work

Conclusions

- 2x to 50x wall time speedups when compared to well tuned homotopy solvers, tailored QPCC solvers, and commercial MIQP solvers.
- Work on a `CCOpt` `MOI` interface is ongoing, bringing an open source local solver for problems with `MOI.complements`!
- Will interface with `ComplementOpt.jl`, via `MOI`.

Future work

- Support specialized GPU acceleration. (Currently can be done through equality relaxation and a "Condensed" KKT system, which limits accuracy, more work neede to avoid transfer overhead).
- Specialized linear algebra for OCPs with complementarity constraints.

Conclusions and future work

Conclusions

- 2x to 50x wall time speedups when compared to well tuned homotopy solvers, tailored QPCC solvers, and commercial MIQP solvers.
- Work on a `CCOpt` `MOI` interface is ongoing, bringing an open source local solver for problems with `MOI.complements`!
- Will interface with `ComplementOpt.jl`, via `MOI`.

Future work

- Support specialized GPU acceleration. (Currently can be done through equality relaxation and a "Condensed" KKT system, which limits accuracy, more work neede to avoid transfer overhead).
- Specialized linear algebra for OCPs with complementarity constraints.
- Filling out the "bag of tricks".

Thank you very much for your attention!


Achtziger, W. and Kanzow, C. (2008).
Mathematical programs with vanishing constraints: optimality conditions and constraint qualifications.
Mathematical Programming, 114:69–99.


Albrecht, S. and Ulbrich, M. (2017).
Mathematical programs with complementarity constraints in the context of inverse optimal control for locomotion.
Optimization Methods and Software, 32(4):670–698.


Baumrucker, B., Renfro, J. G., and Biegler, L. T. (2008).
Mpec problem formulations and solution strategies with chemical engineering applications.
Computers & Chemical Engineering, 32(12):2903–2913.


Baumrucker, B. T. and Biegler, L. T. (2009).
Mpec strategies for optimization of a class of hybrid dynamic systems.
Journal of Process Control, 19(8):1248–1256.


Biegler, L. T. (2010).
Nonlinear Programming.

MOS-SIAM Series on Optimization. SIAM.


Feng, M., Mitchell, J. J., Pang, J. S., Shen, X., and Waechter, A. (2018).
Complementarity formulations of ℓ_0-norm optimization.
Pacific Journal of Optimization, 14(2):273–305.


Guo, L. and Ye, J. J. (2016).
Necessary optimality conditions for optimal control problems with equilibrium constraints.
SIAM Journal on Control and Optimization, 54(5):2710–2733.


Hall, J., Nurkanović, A., Messerer, F., and Diehl, M. (2024).
LCQPow: a solver for linear complementarity quadratic programs.
Mathematical Programming Computation.


Hegerhorst-Schultchen, L. C., Kirches, C., and Steinbach, M. C. (2020).
On the relation between mpecs and optimization problems in abs-normal form.
Optimization Methods and Software, 35(3):560–575.


Hu, J., Mitchell, J. E., Pang, J.-S., and Yu, B. (2012).
On linear programs with linear complementarity constraints.
Journal of Global Optimization, 53:29–51.


Kanzow, C., Schwartz, A., and Weiß, F. (2024).
The sparse (st) optimization problem: Reformulations, optimality, stationarity, and numerical results.
Computational Optimization and Applications, pages 1–36.


Kim, Y., Leyffer, S., and Munson, T. (2020).
Mpec methods for bilevel optimization problems.
In *Bilevel Optimization*, pages 335–360. Springer.


Krug, R., Mehrmann, V., and Schmidt, M. (2021).
Nonlinear optimization of district heating networks.
Optimization and Engineering, 22(2):783–819.


Nurkanović, A. (2023).
Numerical Methods for Optimal Control of Nonsmooth Dynamical Systems.
PhD thesis, University of Freiburg.


Nurkanović, A. and Leyffer, S. (2025).
A globally convergent method for computing b-stationary points of mathematical programs with equilibrium constraints.

arXiv preprint arXiv:2501.13835.


Nurkanović, A., Pozharskiy, A., and Diehl, M. (2024).
Solving mathematical programs with complementarity constraints arising in nonsmooth optimal control.
Vietnam Journal of Mathematics, pages 1–39.


Nurkanović, A., Pozharskiy, A., and Diehl, M. (2026).
Real-time algorithms for model predictive control of hybrid dynamical systems.
arXiv preprint.


Pacaud, F., Nurkanović, A., Pozharskiy, A., Montoison, A., and Shin, S. (2025).
An Augmented Lagrangian method on GPU for security-constrained AC optimal power flow.
In *PSCC2026 (submitted).*


Pozharskiy, A., Nurkanović, A., and Diehl, M. (2024).
Finite elements with switch detection for numerical optimal control of projected dynamical systems.
arXiv preprint arXiv:2404.05367.


Pozharskiy, A., Pacaud, F., Diehl, M., and Nurkanović, A. (2026).
CCOpt: an open-source solver for large-scale mathematical programs with complementarity constraints.
arXiv preprint.


Raghunathan, A. and Biegler, L. (2003).
Mathematical programs with equilibrium constraints (MPECs) in process engineering.
Computers and Chemical Engineering, 27:1381–1392.


Raghunathan, A. U. and Biegler, L. T. (2005).
An interior point method for mathematical programs with complementarity constraints (mpccs).
SIAM Journal on Optimization, 15(3):720–750.


Stewart, D. (1990).
A high accuracy method for solving odes with discontinuous right-hand side.
Numerische Mathematik, 58(1):299–328.


Szmuk, M., Reynolds, T. P., and Açıkmeşe, B. (2020).

Successive convexification for real-time six-degree-of-freedom powered descent guidance with state-triggered constraints.
Journal of Guidance, Control, and Dynamics, 43(8):1399–1413.


Vieira, A., Brogliato, B., and Prieur, C. (2019).
Quadratic optimal control of linear complementarity systems: First-order necessary conditions and numerical analysis.
IEEE Transactions on Automatic Control, 65(6):2743–2750.


Wensing, P. M., Posa, M., Hu, Y., Escande, A., Mansard, N., and Del Prete, A. (2023).
Optimization-based control for dynamic legged robots.
IEEE Transactions on Robotics.

Parameter Updates

Monotone Updates

As explored in [Raghunathan and Biegler, 2005] the monotone approach uses the Fiacco-McCormick rule:

$$\mu^{k+1} = \alpha_\mu (\mu^k)^\beta,$$

for the barrier parameter and a proportional rule for τ:

$$\tau(\mu) = \alpha_\tau (\mu)^{\beta_\tau}.$$

todo: 2 plots of this here.

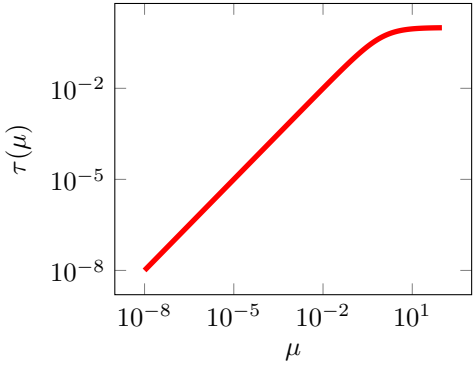
Parameter Updates

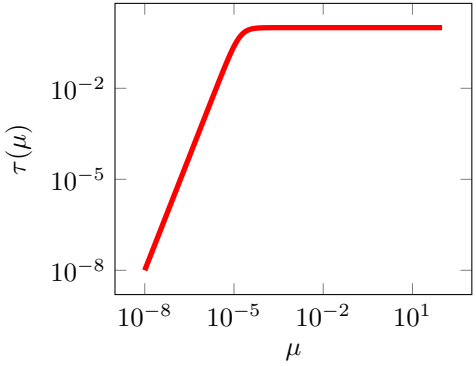
Monotone Updates

In practice, the proportional rule yields worse objectives. We implement an alternative:

$$\tau(\mu) = \frac{c\mu^a}{\sqrt{\mu^{2a}} + b}$$

with $a, b, c \in \mathbb{R}^+$. The tuned implementation uses $a = 2.5$, $b = 1e-12$, and $c = 1$





Parameter Updates

Adaptive updates

Alternatively, one can choose an adaptive rule for updating μ. Following classical approaches we use:

$$\mu^k = \sigma \frac{(x^k)^\top z^k + (x_1^k)^\top x_2^k}{n + n_{cc}}$$

with several approaches to selecting $\sigma \in \mathbb{R}^+$.

LOQO Rule

$$\sigma = \gamma \min\left((1-r)\frac{1-\xi}{\xi}, 2\right)^3, \ \xi = \frac{n_x \min(Xz)}{x^\top z},$$
$$r \in (0,1), \quad \gamma > 0$$

Quality Function Rule

Quality Function

$$\Psi(x, s, y, z) = \|\nabla \mathcal{L}(x, s, y, z)\|^2 + \|c(x)\|^2 + \|Zx\|^2 + \|X_1 x_2\|^2$$

Linearized Quality Function

$$\begin{aligned} q_L(\sigma) =& (1 - \alpha_{\text{du}}^{\max}(\sigma))^2 \|\nabla \mathcal{L}(x, s, y, z)\|^2 + (1 - \alpha_{\text{pr}}^{\max}(\sigma))^2 \|c(x)\|^2 + \\ & \left\|(Z + \alpha_{\text{du}}^{\max}(\sigma)\Delta Z(\sigma))(x + \alpha_{\text{pr}}^{\max}(\sigma)\Delta x(\sigma))\right\|^2 + \\ & \left\|(X_1 + \alpha_{\text{pr}}^{\max}(\sigma)\Delta X_1(\sigma))(x_2 + \alpha_{\text{pr}}^{\max}(\sigma)\Delta x_2(\sigma\right\|^2 \end{aligned}$$