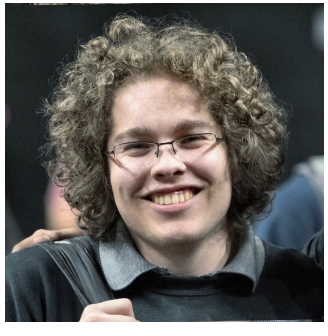
An augmented Lagrangian method on GPU for AC-SCOPF

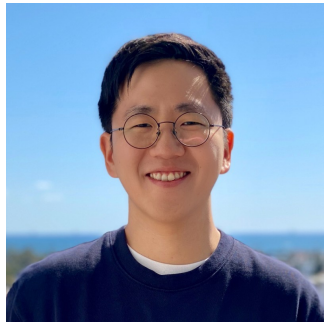
François Pacaud
Armin Nurkanović Anton Pozharskiy Alexis Montoison Sungho Shin

CAS, Mines Paris - PSL

PSCC 2026
June 10th







What do we mean by SCOPF?

Corrective SCOPF

- Compute a base case solution that remains **feasible** for K given contingencies
- **Nonlinear** AC power flow formulation
- Two types of corrective actions in post-contingency state:
  1. Automatic generation control (AGC)
  2. PV/PQ switches
- We suppose contingency screening has been applied before-hand
- No structure exploiting method!

Complementarity constraints

Complementarity constraint $0 \leq a \perp b \geq 0$

We say that a complements b if

$$a \geq 0\,, \quad b \geq 0\,, \quad a \times b = 0$$

- Useful paradigm for expliciting non-convex or non-smooth structures!
- Already used by the power system community in the 2000s:
  - Rosehart W, Roman C, Schellenberg A. *Optimal power flow with complementarity constraints*. IEEE Transactions on Power Systems. 2005
  - Capitanescu F, Rosehart W, Wehenkel L. *Optimal power flow computations with constraints limiting the number of control actions*. 2009

Where do complementarity constraints appear in SCOPF?

Automatic generation control (AGC)

In contingency k, the power generations are adapted with Δ^k

$$p_g^k = \min\left(\max\left(p_g^0 + \alpha_g \Delta^k,\ \underline{p}_g\right),\ \overline{p}_g\right)$$

or, equivalently,

$$\begin{aligned}
&\pi_{g+}^k - \pi_{g-}^k = p_g^k - (p_g^0 + \alpha_g \Delta) \\
&0 \le \pi_{g-}^k \perp \overline{p}_g - p_g^k \ge 0 \\
&0 \le \pi_{g+}^k \perp p_g^k - \underline{p}_g \ge 0
\end{aligned}$$

PV/PQ switches

If reactive power q_g reaches its bounds, the connected bus behave as a PQ node:

$$\begin{aligned}
&\nu_{b+}^k - \nu_{b-}^k = v_b^k - v_b^0\ , \\
&0 \le \nu_{b-}^k \perp \overline{q}_g - q_g^k \ge 0\ , \\
&0 \le \nu_{b+}^k \perp q_g^k - \underline{q}_g \ge 0\ .
\end{aligned}$$

Mathematical program with complementarity constraints (MPCC)

Here, we formulate the SCOPF as a large-scale MPCC:

$$
\begin{aligned}
&\min_{x,u} \; f(x_0, u_0) \\
\text{s.t.} \quad & g_0(x_0, u_0) = 0 \;,\; h_0(x_0, u_0) \leq 0 \;, \\
& \forall k \in \{1, \cdots, K\} : \\
& g_k(u_0, x_k, u_k) = 0 \;,\; h_k(x_k, u_k) \leq 0 \;, \\
& 0 \leq G(x_k, u_k) \perp H(x_k, u_k) \geq 0 \;,
\end{aligned}
$$

with

- (x_k, u_k): state and control in contingency k
- $f(\cdot)$: objective in base case scenario
- $g_k(\cdot)$: power-flow and linear constraints
- $h_k(\cdot)$: operational constraints and line flow limits
- $G(\cdot), H(\cdot)$: complementarity left and right-hand-sides

MPCC in standard form

Reformulated as:

$$
\min_{w \in \mathbb{R}^n} \; \phi(w) \quad \text{s.t.} \quad \begin{cases} c(w) = 0 \\ 0 \leq w_1 \perp w_2 \geq 0 \end{cases}
$$

Solution methods

The MPCC reformulates as the (degenerate) nonlinear program:

$$\min_{w \in \mathbb{R}^n} \phi(w) \quad \text{s.t.} \quad \begin{cases} c(w) = 0 \\ W_1 W_2 e \leq 0 \end{cases}$$

Violate all constraint qualifications (including MFCQ)!

Scholtes method

Relax the degenerate constraint with a positive parameter $\tau > 0$:

$$\min_{w \in \mathbb{R}^n} \phi(w) \quad \text{s.t.} \quad \begin{cases} c(w) = 0 \\ W_1 W_2 e \leq \tau e \end{cases}$$

Elastic reformulation

Move the complementarity constraints to the objective:

$$\min_{w \in \mathbb{R}^n} \phi(w) + \rho w_1^\top w_2 \quad \text{s.t.} \quad c(w) = 0\,, \quad (w_1, w_2) \geq 0$$

Augmented Lagrangian algorithm

Lagrangian

For multipliers (λ, ξ, ν), define

$$\mathcal{L}(w, \lambda, \xi, \nu) = \phi(w) + \lambda^\top c(w) - \xi^\top w_0 - \nu_1^\top w_1 - \nu_2^\top w_2 + \nu_0^\top W_1 W_2 e$$

Augmented Lagrangian

For multiplier estimates $(\lambda^{(n)}, \nu_0^{(n)})$ and penalty $\rho^{(n)}$, define

$$\mathcal{L}_\rho(w, \lambda^{(n)}, \nu_0^{(n)}) := \phi(w) + \frac{1}{2\rho^{(n)}} \left(\|c(w) + \lambda^{(n)}\|^2 + \|\max\{0, W_1 W_2 e + \nu_0^{(n)}\}\|^2 \right)$$

Have to handle inequality constraints!

Algorithm: At each iteration n, solve

$$\min_{(w_1, w_2) \geq 0} \mathcal{L}_\rho(w, \lambda^{(n)}, \nu_0^{(n)})$$

Update the multipliers and penalty according to threshold $\eta^{(n)}$:

- If current primal infeasibility less than $\eta^{(n)}$, update $(\lambda^{(n)}, \nu_0^{(n)})$;
- Otherwise, increase $\rho^{(n)}$.

Linear system solve

Remember that the decision variable decomposes as $w = (w_0, w_1, w_2)$

At each iteration, NCL solves a *KKT system* with the saddle-point structure

$$\begin{bmatrix} A & B^\top \\ B & -C \end{bmatrix} \begin{bmatrix} \Delta w \\ \Delta y \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$$

with

$$A = \begin{bmatrix} H_{00} & H_{01} & H_{02} \\ H_{10} & H_{11} & H_{12} \\ H_{20} & H_{21} & H_{22} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & W_1^{-1} V_1 & V_0 \\ 0 & V_0 & W_2^{-1} V_2 \end{bmatrix}$$

and

$$B = \begin{bmatrix} J_0 & J_1 & J_2 \\ & W_2 & W_1 \end{bmatrix}, \quad C = \begin{bmatrix} \rho^{-1} I & & \\ & & \rho^{-1} I + V_0^{-1} S \end{bmatrix}$$

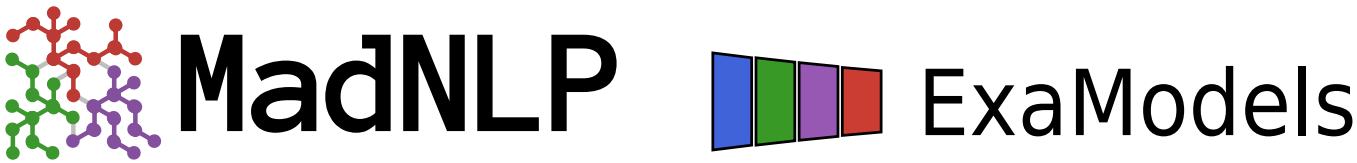
Observations

- The complementarity terms increase the indefinitess in A
- If $\mathcal{I}_{00}(w) \neq \emptyset$, the matrix B is not full row-rank
- Problem with unbounded multipliers!

GPU acceleration with MadNLP

Umbrella project: madsuite.org

- The algorithm has been built on top of MadNLP
- Derivatives (Jacobian and Hessian) are evaluated in parallel using ExaModels
- Benefit directly from GPU-acceleration



Bottleneck: Factorization algorithm

- Factorize the KKT system with a $LDL^\top$ factorization
- Require slight regularization for the pivots, but allow for a solution on the GPU

Experiment #1: time per iteration

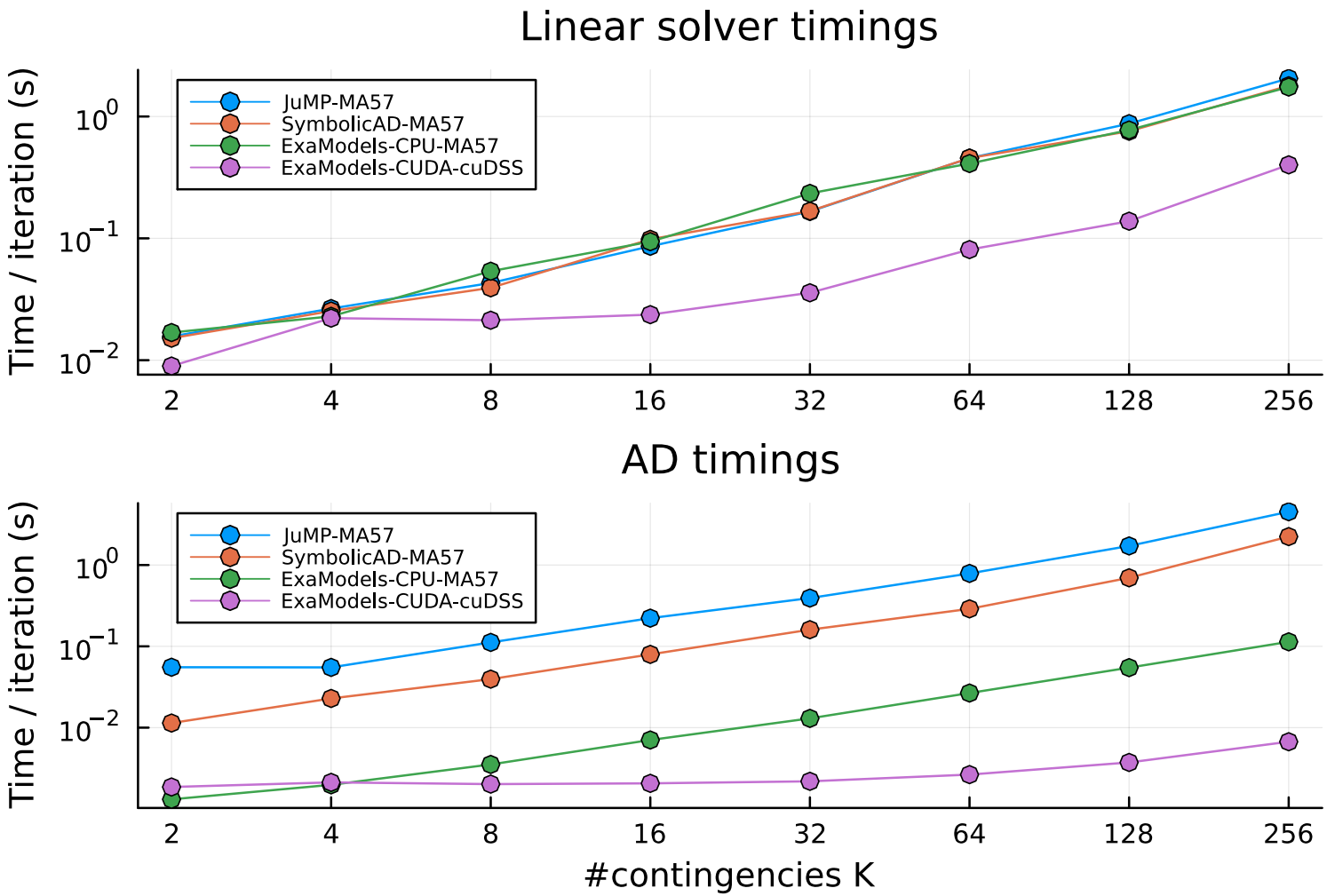


Figure: Scalability as we increase the number of contingencies K for a network with 500 buses.

Experiment #1: time per iteration

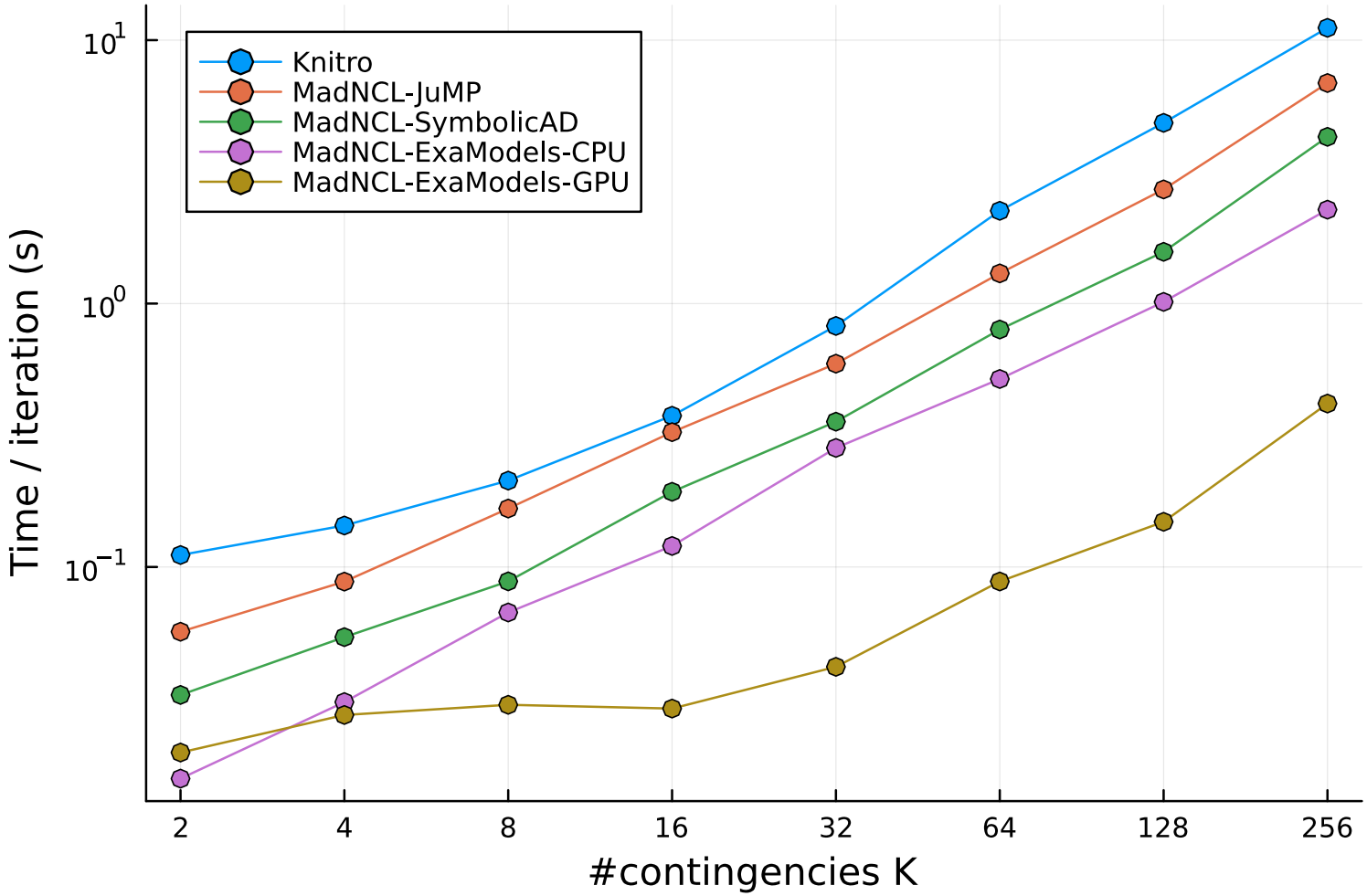


Figure: Scalability as we increase the number of contingencies K for a network with 500 buses.

Experiment #2: time to solution

Instances from the ARPAE GO competition (Challenge 1)

			Knitro (ref)				MadNCL-GPU			
Name	K	nvar	Iter	Obj.	**Time (s)**	Time/it (s)	Iter	Obj.	**Time (s)**	Time/it (s)
Network07O-122	5	111k	135	0.04	**219.90**	1.63	786	0.04	**42.80**	0.05
Network07O-122	10	203k	303	0.04	**952.59**	3.14	1050	0.04	**100.45**	0.10
Network09O-070	5	245k	46	0.05	**245.72**	5.34	81	0.05	**12.60**	0.16
Network07O-122	20	388k	257	0.04	**981.03**	3.82	1093	0.04	**116.36**	0.11
Network09O-070	10	451k	-	0.06	**3600.08**	8.51	101	0.05	**18.66**	0.18
Network12O-030	5	518k	-	10.39	**3601.25**	9.89	-	10.41	**265.41**	0.44
Network09O-070	20	861k	135	0.05	**2493.12**	18.47	501	0.05	**152.38**	0.30
Network12O-030	10	950k	-	10.39	**3611.73**	19.42	875	10.39	**413.43**	0.47
Network12O-030	20	1814k	-	12.96	**3646.33**	33.76	679	10.39	**474.33**	0.70

The symbol "-" indicates that the solver failed to find a solution.

Take-home messages

Final answer

Yes, it is now practical to solve large-scale SCOPFs on the GPU!

We are looking for challenging problems to solve!

The end of complementarity winter?

- `MathOptComplement.jl`: improve the support of complementarity in JuMP (including automatic problem's reformulation)
- `MadNCL.jl`: A robust nonlinear solver that support the solution of MPCCs
- `COpt.jl`: A custom solver for MPCCs