GPU-accelerated nonlinear programming with MadNLP

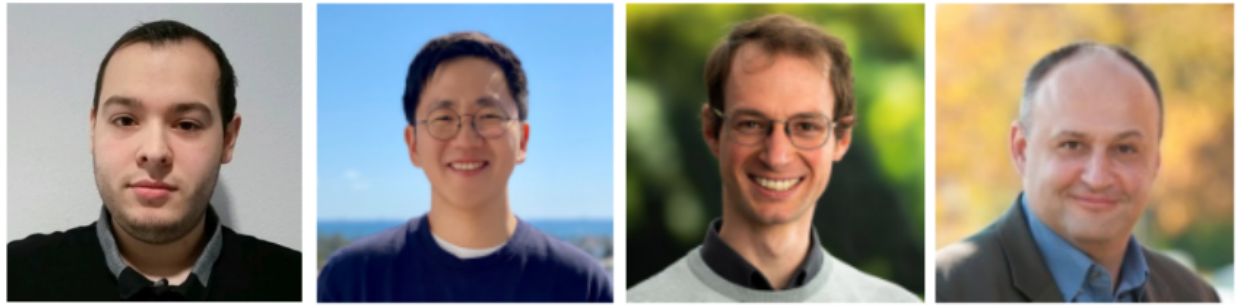
François Pacaud

CAS, Mines Paris - PSL

Systems Control and Optimization Laboratory
University of Freiburg
April, 10th 2025

Who are we?

- An international team looking at the future of nonlinear programming

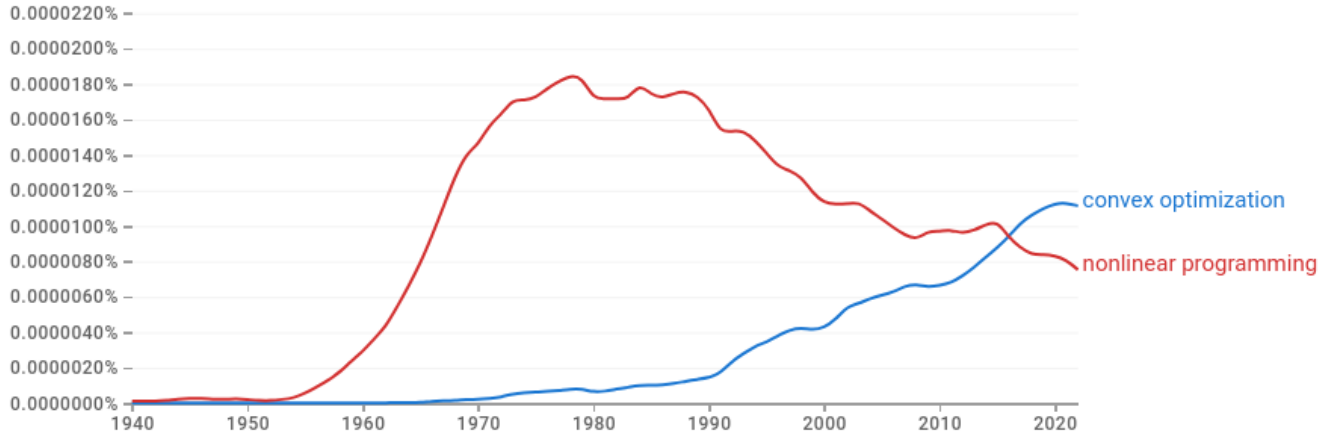


Outline: today we talk about nonlinear programming

1. We present a condensed-space interior-point method (IPM) that runs smoothly on NVIDIA GPUs using the new solver cuDSS
2. We adapt the method further and mix it with an Augmented Lagrangian method to solve degenerate problems

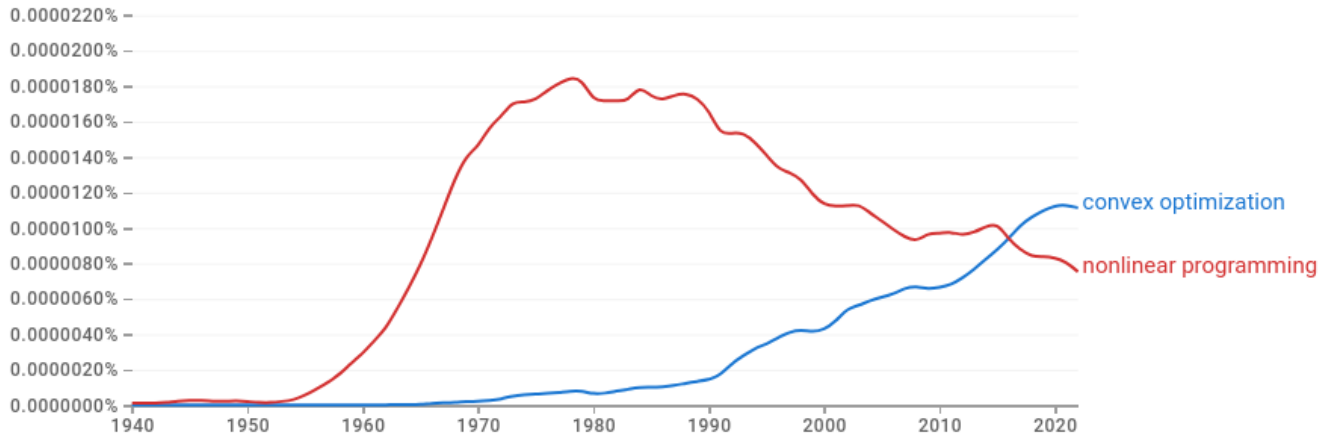
The sad truth...

Nonlinear programming has fallen out of fashion :-(



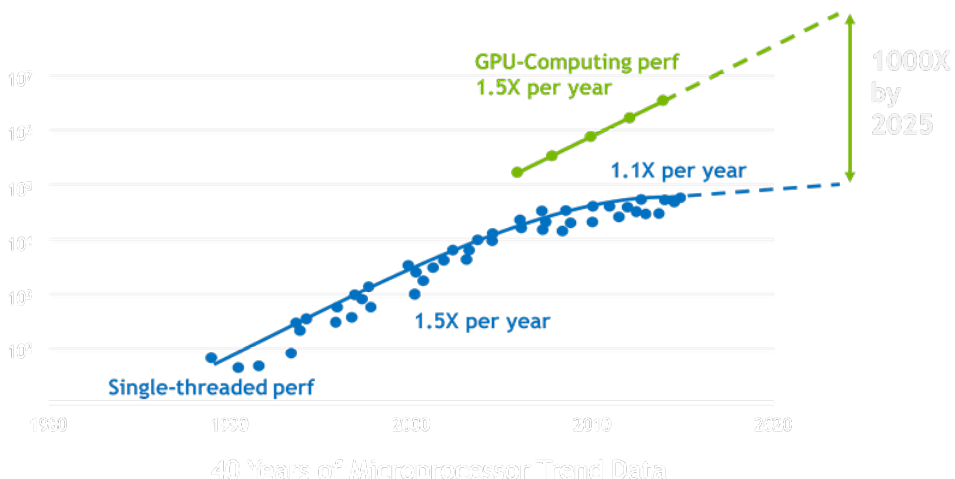
The sad truth...

Nonlinear programming has fallen out of fashion :-(



... but an open-door for new opportunities!

Can we make nonlinear programming great again using modern hardware?



Source: Google NGram, NVIDIA

Back to the basic: Nonlinear programming

Nonlinear programming in a nutshell

n variables, m inequality constraints, p equality constraints

Continuous nonlinear problems

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to} \quad \begin{cases} g(x) = 0 \\ h(x) \leq 0 \end{cases}$$

Objective

Equality cons.

Inequality cons.

The functions f, g, h are smooth, *possibly nonconvex*

- Useful framework to solve practical engineering problems
- Usually, we are interested only at finding a *local optimum*
- Mature solvers exist since the 2000s (Ipopt, Knitro, LOQO)

J. Nocedal, SJ. Wright. Numerical optimization.

Nonlinear programming in a nutshell

n variables, m inequality constraints, p equality constraints

Continuous nonlinear problems

Objective

$$\min_{x \in \mathbb{R}^n, s \in \mathbb{R}^m} f(x) \quad \text{subject to} \quad \begin{cases} g(x) = 0 \\ h(x) + s = 0\,, \quad s \geq 0 \end{cases}$$

Equality cons.

Slack

The functions f, g, h are smooth, *possibly nonconvex*

- Useful framework to solve practical engineering problems
- Usually, we are interested only at finding a *local optimum*
- Mature solvers exist since the 2000s (Ipopt, Knitro, LOQO)

J. Nocedal, SJ. Wright. Numerical optimization.

Interior-point method (IPM)

KKT stationary equations

$$\begin{cases} \nabla f(x) + \nabla g(x)^\top y + \nabla h(x)^\top z = 0 \\ z - \nu = 0 \\ g(x) = 0 \\ h(x) + s = 0 \\ 0 \le s \perp \nu \ge 0 \end{cases}$$

Complementarity cons

Rewrite the (nonsmooth) KKT system as a *smooth* nonlinear system

Dual variables

$$F_\mu(x, s; y, z, \nu) := \begin{bmatrix} \nabla f(x) + \nabla g(x)^\top y + \nabla h(x)^\top z \\ z - \nu \\ g(x) \\ h(x) + s \\ S\nu - \mu e \end{bmatrix} = 0$$

Homotopy, $S = \operatorname{diag}(s)$

Primal-dual interior-point method

Solve $F_\mu(x, s; y, z, \nu) = 0$ using Newton method while driving $\mu \to 0$.

Augmented KKT system

At iteration k,

1. Compute Newton step d^k as solution of the linear system

$$\nabla F_\mu(w^k)\, d^k = -F_\mu(w^k)$$

2. Update the primal-dual variable $w^k := (x^k, s^k, y^k, z^k, \nu^k)$ as

$$w^{k+1} = w^k + \alpha^k d^k$$

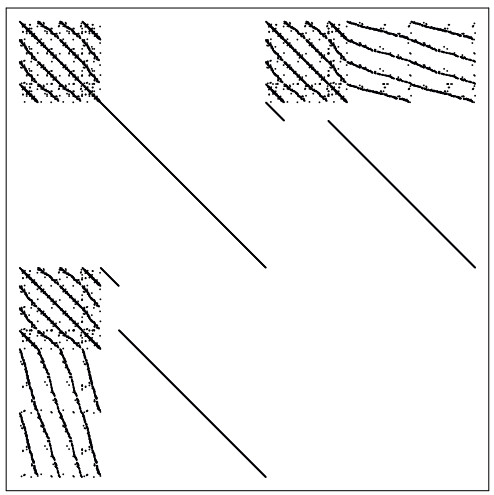


Figure: ∇F_μ

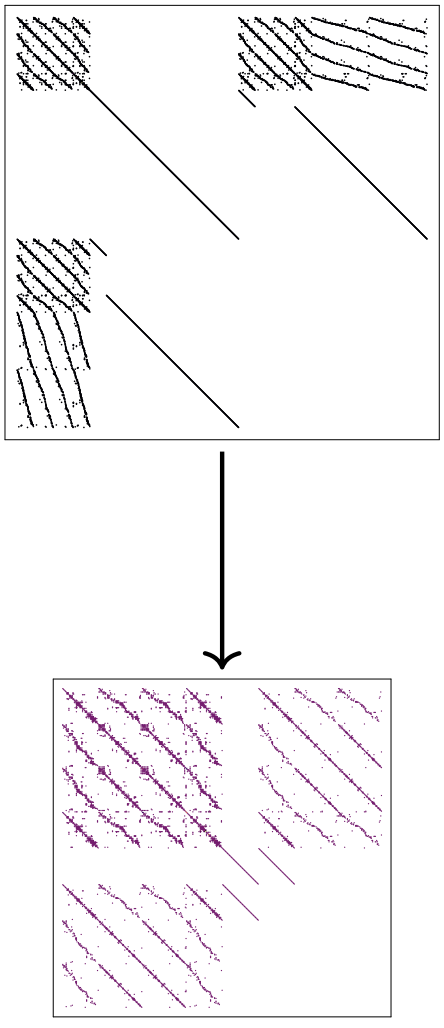
Augmented KKT system

After (slight) reformulation, the Newton step writes as

$$\begin{bmatrix} W & 0 & \nabla g^\top & \nabla h^\top \\ 0 & \Sigma_s & 0 & I \\ \nabla g & 0 & 0 & 0 \\ \nabla h & I & 0 & 0 \end{bmatrix} \begin{bmatrix} d_x \\ d_s \\ d_y \\ d_z \end{bmatrix} = - \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix}$$

with $W = \nabla^2_{xx} L(\cdot)$, $\Sigma_s = S^{-1}\mathrm{diag}(\nu)$

Condensed KKT system



Condensed KKT system

The augmented KKT system is equivalent to

$$\begin{bmatrix} K & \nabla g^\top \\ \nabla g & 0 \end{bmatrix} \begin{bmatrix} d_x \\ d_y \end{bmatrix} = - \begin{bmatrix} r_1 + (\nabla h)^\top (\Sigma_s r_4 + r_2) \\ r_3 \end{bmatrix}$$

with the *condensed matrix* $K = W + \nabla h^\top \Sigma_s \nabla h$.
We recover (d_s, d_z) as

$$d_s = -\Sigma_s^{-1}(r_3 + d_y)\,, \quad d_z = \Sigma_s\,(\nabla h\, d_x - r_4) - r_2\,.$$

- Symmetric indefinite
- Additional fill-in
- Useful when the number of inequality constraints m is large

Challenge: solving the sparse linear system on the GPU

- Ill-conditioning of the KKT system in IPM
  (= iterative solvers are often not practical)
- Direct solver requires **numerical pivoting** for stability
  (= difficult to parallelize)

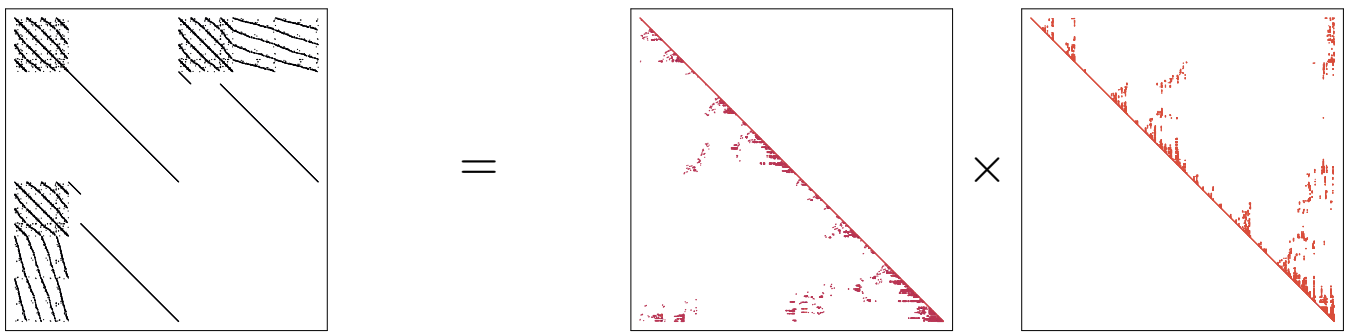


Figure: LU factorization using a direct solver

Solution: Condensation

Reduce the KKT system to a sparse positive definite matrix

- Sparse Cholesky is stable without numerical pivoting
- Runs in parallel on the GPU (`cuDSS`)

HyKKT (aka Golub & Greif method)

Idea: augmented Lagrangian reformulation

For $\gamma > 0$, the condensed KKT system is equivalent to

$$\begin{bmatrix} K_\gamma & \nabla g^\top \\ \nabla g & 0 \end{bmatrix} \begin{bmatrix} d_x \\ d_y \end{bmatrix} = - \begin{bmatrix} w_1 + \gamma \nabla g^\top w_2 \\ w_2 \end{bmatrix} \quad \text{with} \quad K_\gamma = K + \gamma \nabla g^\top \nabla g$$

- ☑ Suppose LICQ hold and the reduced Hessian is positive definite: Then for γ large-enough the matrix K_γ is positive definite
- ☑ The condensed KKT system reduces to the normal equations:

$$(\nabla g) K_\gamma^{-1} (\nabla g)^\top d_y = w_2 - K_\gamma^{-1} (w_1 + \gamma \nabla g^\top w_2)$$

- ☑ Keep K_γ^{-1} implicit by solving the normal equations *iteratively* with a conjugate gradient (CG) algorithm!
- ☑ For large γ, CG converges in few iterations

S. Regev et al., "HyKKT: a hybrid direct-iterative method for solving KKT linear systems." Optimization Methods and Software 38, no. 2 (2023)

Fast evaluation of derivatives with ExaModels.jl

- Large-scale optimization problems **almost always have repetitive patterns**

$$\min_{x^\flat \le x \le x^\sharp} \sum_{l \in [L]} \sum_{i \in [I_l]} f^{(l)}(x; p_i^{(l)}) \qquad \textbf{(SIMD abstraction)}$$

$$\text{subject to } \left[g^{(m)}(x; q_j) \right]_{j \in [J_m]} + \sum_{n \in [N_m]} \sum_{k \in [K_n]} h^{(n)}(x; s_k^{(n)}) = 0, \quad \forall m \in [M]$$

- Repeated patterns are made available by specifying the models as **iterable objects**

```
constraint(c, 3 * x[i+1]^3 + 2 * sin(x[i+2]) for i = 1:N-2)
```

- **For each repeatitive pattern**, the derivative evaluation kernel is constructed & compiled, and **executed in parallel over multiple data**

Application: AC-OPF problem

Observations

- Up to 10x speed-up compared to Ipopt

	HSL MA27				LiftedKKT+cuDSS				HyKKT+cuDSS			
Case	it	init	lin	**total**	it	init	lin	**total**	it	init	lin	**total**
13659_pegase	63	0.45	7.21	**10.14**	75	0.83	1.05	**2.96**	62	0.84	0.93	**2.47**
19402_goc	69	0.63	31.71	**36.92**	73	1.42	2.28	**5.38**	69	1.44	1.93	**4.31**
20758_epigrids	51	0.63	14.27	**18.21**	53	1.34	1.05	**3.57**	51	1.35	1.55	**3.51**
78484_epigrids	102	2.57	179.29	**207.79**	101	5.94	5.62	**18.03**	104	6.29	9.01	**18.90**

Table: OPF benchmark, solved by MadNLP with a tolerance `tol=1e-6`. (A100 GPU)

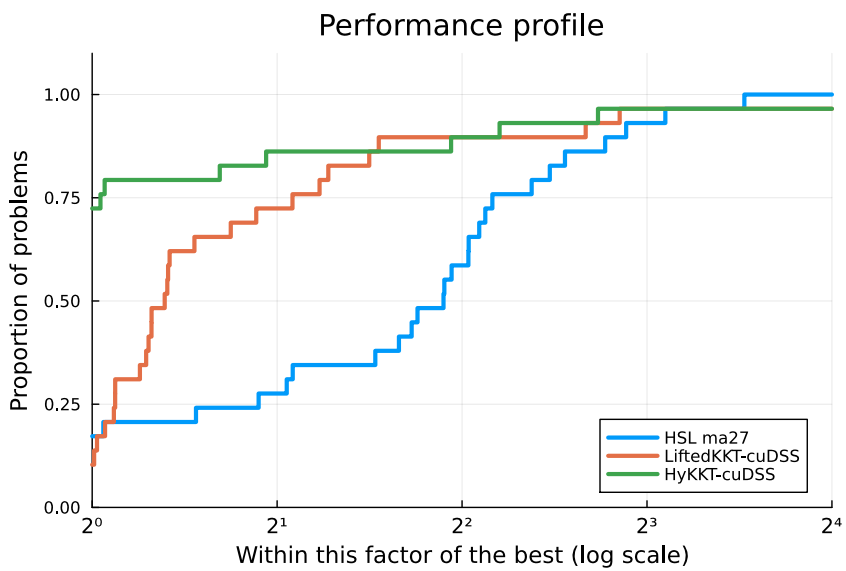


Figure: Performance profile

Application: optimal control

Observations

Textbook problem: Optimizing the operation of a distillation column over an horizon N

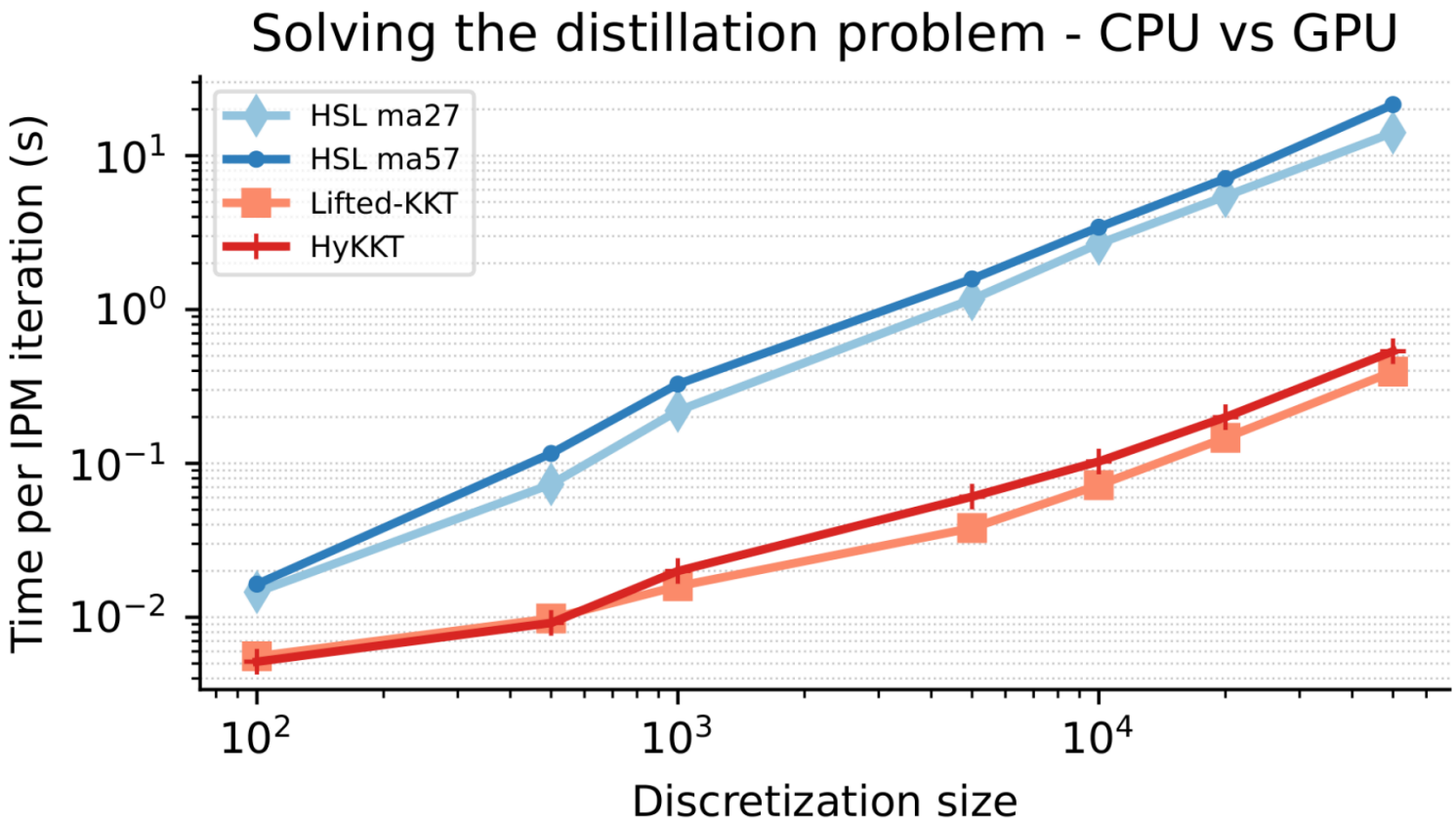


Figure: Time per IPM iteration (s), CPU versus GPU. Log-log scale.

How expensive should be your GPU?

Benchmarking different GPUs

- A100 (80GB) — HPC ($10,000)
- A30 (24GB) — workstation ($5,000)
- A1000 (4GB) — laptop

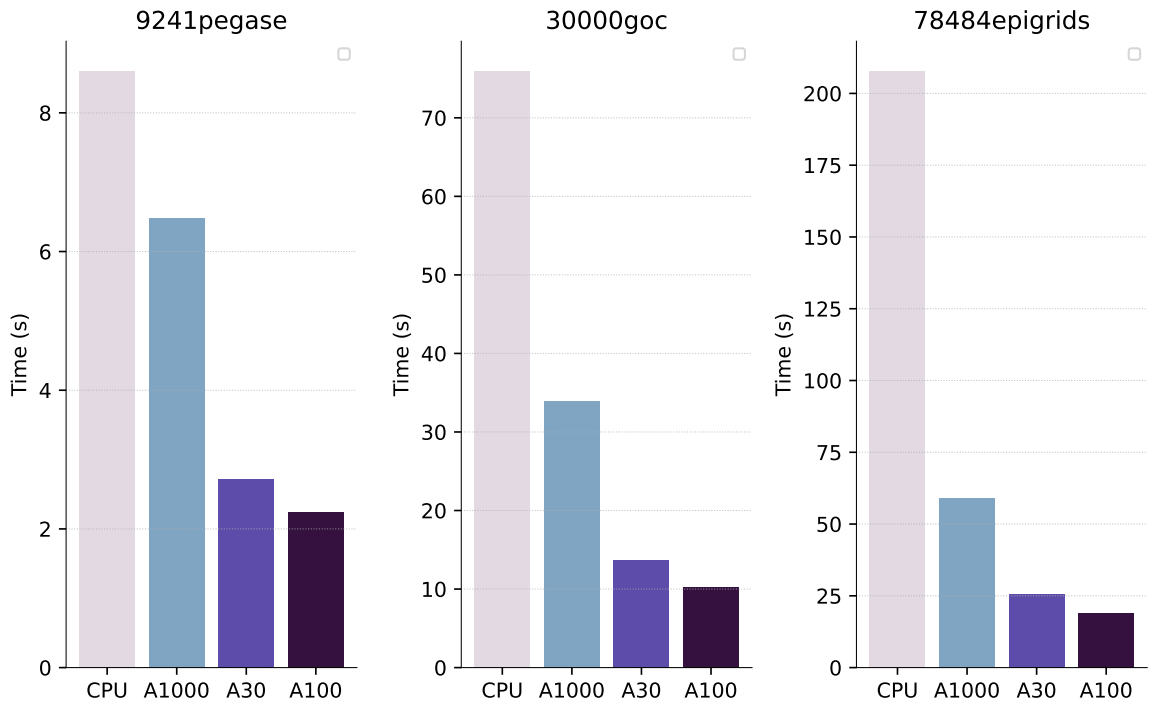


Figure: Time to optimality (in seconds).

Question

How to obtain more accurate solution? `tol=1e-8`

Back to Augmented Lagrangian (Auglag)

- Augmented Lagrangian methods are (super) robust
- Idea: solve Auglag subproblems with IPM

D. Fernández & M.V. Solodov (2012).
Local convergence of exact and inexact augmented Lagrangian methods under the second-order sufficient optimality condition.

Augmented Lagrangian

From now on, we simplify the nonlinear program as:

$$\min_{x \in \mathbb{R}^n} \; f(x) \quad \text{subject to} \quad c(x) = 0 \,, \; x \geq 0$$

Augmented Lagrangian subproblem

For $y_e \in \mathbb{R}^m, \rho > 0$,

$$\min_{x \in \mathbb{R}^n} \; f(x) + y_e^\top c(x) + \frac{\rho}{2} \|c(x)\|^2 \quad \text{subject to} \quad x \geq 0$$

Introducing the NCL algorithm

NCL reformulates the Auglag's subproblems as constrained optimization problems

At iteration k, the algorithm solves:

$$\begin{aligned} \min_{x,r} \; & f(x) - (y_k^e)^\top r + \frac{\rho_k}{2}\|r\|^2 \\ \text{subject to} \quad & c(x) + r = 0\,, \quad x \geq 0 \end{aligned} \tag{NCL$_k$}$$

- Subproblem (NCL$_k$) is always feasible, solvable by IPM!
- Only the objective changes
- Regularization r stabilizes internal IPM iterations

NCL algorithm $\equiv$ Auglag algorithm

- Solve (NCL$_k$) down to a tolerance ω_k
- Update parameters as
  - If $\|c(x_k)\| \leq \eta_k$, set $y_{k+1}^e = y_k^e - \rho_k r_k$
  - Else $\rho_{k+1} = 10 \times \rho_k$.

Highlighting the connection between HyKKT and Augmented Lagrangian

Observation #1

For $K_\gamma = W + \Sigma_x + \gamma J^\top J$, HyKKT solves the KKT system:

$$\begin{bmatrix} K_\gamma & J^\top \\ J & 0 \end{bmatrix} \begin{bmatrix} d_x \\ d_y \end{bmatrix} = - \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$$

Observation #2

For $K_\rho := W + \Sigma_x + \rho J^\top J$, Auglag solves the KKT system:

$$K_\rho \, d_x = -n_1$$

$\gamma \equiv \rho$: tune γ automatically using an Augmented Lagrangian strategy!

Writing the KKT system

Compared to raw IPM, each NCL's subproblem has an additional variable r

KKT system: IPM

IPM search direction is computed by solving

$$\begin{bmatrix} W + \Sigma_x & J^\top \\ J & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = - \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} \qquad (K_{aug})$$

KKT system: NCL

NCL search direction is computed by solving

$$\begin{bmatrix} W + \Sigma_x & 0 & J^\top \\ 0 & \rho_k I & I \\ J & I & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta r \\ \Delta y \end{bmatrix} = - \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} \qquad (K_2)$$

☞Note that the NCL Jacobian is always full row-rank

Condensation strategy

To port the method on the GPU, we use the same condensation strategy as before

Step 1: removing Δr

Eliminate $\Delta r = (n_2 - \Delta y)/\rho_k$:

$$\begin{bmatrix} W + \Sigma_x & J^\top \\ J & -\frac{1}{\rho_k} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = - \begin{bmatrix} n_1 \\ n_3 - \frac{1}{\rho_k} n_2 \end{bmatrix} \qquad (K_{2r})$$

Step 2: removing Δy

Eliminate $\Delta y = n_2 - \rho_k(n_3 - J\Delta x)$:

$$(W + \Sigma_x + \rho_k J^\top J)\Delta x = J^\top(n_3 + \rho_k r_1) - r_2 \qquad (K_{1s})$$

The original problem is nonconvex, hence:

- K_{2r} is (almost) SQD (LDL)
- K_{1s} is (almost) positive definite (Cholesky)

What do we have so far?

- NCL is an augmented Lagrangian algorithm whose subproblems are solvable by MadNLP
- By exploiting again the structure of the KKT system, the method runs on the GPU

What can go wrong?

- Augmented Lagrangian has only a linear rate of convergence
- Meaning it takes time to find highly accurate solution...

Solution: Dussault's extrapolation method

Denote

- $w_k = (x_k, r_k, y_k)$ the primal-dual iterate and
- $F(w_k)$ the Auglag residual at w_k.

NCL with extrapolation step

- Compute an extrapolation step w_k^+ solution of

$$\nabla_w F(w_k)(w_k^+ - w_k) + F(w_k) = 0$$

- If $\|F(w_k^+)\| \leq \theta_k \|F(w_k)\| + \varepsilon_k$ set $w_{k+1} = w_k^+$. Else
  - Solve (NCL_k) down to a tolerance ω_k
  - Update parameters as
    - If $\|c(x_k)\| \leq \eta_k$, set $y_{k+1}^e = y_k^e - \rho_k r_k$
    - Else $\rho_{k+1} = 10 \times \rho_k$.

Observations

- According to Dominique Orban, the method can be traced back to JP Dussault in the 1990s
- We can prove NCL achieves superlinear local convergence: asymptotically, the method becomes equivalent to a stabilized SQP method

J.P. Dussault, Numerical stability and efficiency of penalty algorithms, SIAM Journal on Numerical Analysis 32 (1995), pp. 296–317.
P. Armand, J. Benoist, D. Orban. From global to local convergence of interior methods for nonlinear optimization. OMS (2013).

Safeguarding convergence for degenerate problems

Auglag is more robust than interior-point

Property 1: Problem with redundant constraints

With the dual regularization r, the subproblems (NCL_k) automatically satisfy LICQ

Property 2: Infeasible problems

For infeasible problem, NCL converges to a stationary point of:

$$\min_{x,r} \ \frac{\rho}{2}\|r\|^2 \quad \text{subject to} \quad c(x) + r = 0\ ,\ x \geq 0$$

Property 3: Problems with complementarity constraints (MPCC)

For MPCC problem, Auglag converges to a strong-stationary point (under specific technical assumptions)

AF. Izmailov, MV. Solodov, and EI. Uskov. "Global convergence of augmented Lagrangian methods applied to optimization problems with degenerate constraints, including problems with complementarity constraints." SIAM Journal on Optimization 22 (2012)

Open research question

Is Auglag effective at solving MPCC problems?

Nonlinear program with complementarity constraints

Mathematical program with complementarity constraints (MPCC)

Let $f : \mathbb{R}^n \to \mathbb{R}$, $G : \mathbb{R}^n \to \mathbb{R}^m$ and $H : \mathbb{R}^n \to \mathbb{R}^m$. We define the MPCC as

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

Notation $0 \leq x \perp y \geq 0$ stands for:

$$0 \leq x_i \quad \text{and} \quad 0 \leq y_i \quad \text{and} \quad x_i y_i = 0 \quad \forall i = 1, \cdots, m$$

NLP reformulation

MPCC is equivalent to the NLP:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \ & f(x) \\ \text{s.t.} \quad & G(x) \geq 0\,, \quad H(x) \geq 0 \\ & G_i(x) H_i(x) \leq 0 \quad \forall i = 1, \cdots, m \end{aligned}$$

☢The previous NLP is degenerate, in the sense that MFCQ fails at all x

A set of optimality conditions for MPCC

- Active sets:

$$I_G = \{i \in \{1, \cdots, m\} \mid G_i(x) = 0\}\,, \quad I_H = \{i \in \{1, \cdots, m\} \mid H_i(x) = 0\}\,.$$

- MPCC Lagrangian:

$$L(x, \lambda_G, \lambda_H) := f(x) - \lambda_G^\top G(x) - \lambda_H^\top H(x)\,.$$

Stationarity conditions

Weak stationary $\nabla_x L(x, \lambda) = 0$ with

$$(\lambda_G)_i = 0 \quad \text{for} \quad i \notin I_G \quad \text{and} \quad (\lambda_H)_i = 0 \quad \text{for} \quad i \notin I_H$$

Clarke stationary: weak stationary and

$$(\lambda_G)_i (\lambda_H)_i \geq 0 \quad \forall i \in I_G \cap I_H$$

Mordukhovich stationary: Clarke stationary and

$$\text{Either}\Big((\lambda_G)_i > 0 \text{ and } (\lambda_H)_i > 0\Big) \text{ or } (\lambda_G)_i (\lambda_H)_i = 0 \qquad \forall i \in I_G \cap I_H$$

Strong stationary: Mordukhovich stationary and

$$(\lambda_G)_i \geq 0 \quad \text{and} \quad (\lambda_H)_i \geq 0 \quad \forall i \in I_G \cap I_H$$

Usual solution methods

- **Relaxation:** For $\tau > 0$, replace complementarity constraints by

$$G_i(x)H_i(x) \leq \tau \ , \quad \forall i = 1, \cdots, m$$

- **Smoothening:** Use a smooth approximation parameterized by $\varepsilon > 0$:

$$\Phi_\varepsilon\big(G_i(x), H_i(x)\big) = 0 \quad \forall i = 1, \cdots, m$$

- **Exact penalty:** Move the complementarity constraints in the objective:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \ & f(x) + \nu G(x)^\top H(x) \\ \text{s.t.} \quad & G(x) \geq 0 \ , \quad H(x) \geq 0 \end{aligned}$$

  and solve resulting problem with interior-point

Research question

Can we solve instead the original problem using NCL?

R. Fletcher and S. Leyffer. "Solving mathematical programs with complementarity constraints as nonlinear programs." Optimization Methods and Software

An application to Corrective Security-Constraint OPF

Suppose we have K potential contingencies, with for each $k = 1, \cdots, K$,

Automatic generation control system (droop control)

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

or, equivalently,

$$\rho_{g,+}^k - \rho_{g,-}^k = p_g^k - (p_g^0 + \alpha_g \Delta)$$
$$0 \leq \rho_{g,-}^k \perp \overline{p}_g - p_g^k \geq 0$$
$$0 \leq \rho_{g,+}^k \perp p_g^k - \underline{p}_g \geq 0$$

Voltage control (PV/PQ switches)

$$\nu_+^k - \nu_-^k = v_m^k - v_m^0$$
$$0 \leq \nu_-^k \perp \overline{q}_g - q_g^k \geq 0$$
$$0 \leq \nu_+^k \perp q_g^k - \underline{q}_g \geq 0$$

I. Aravena et al. "Recent developments in security-constrained AC optimal power flow: Overview of challenge 1 in the ARPA-E grid optimization competition." Operations research 71 (2023)

SCOPF problem

Objective: adjusting the base case x_0

We keep the K_s most important contingencies in the problem (generally, $K_s \approx 10$)

Corrective SCOPF

$$
\begin{aligned}
\min_{x0, x_1, \cdots, x_{K_s}} \quad & f(x_0) \\
\text{subject to} \quad & c_0(x_0) = 0\,, \quad x_0 \geq 0 \\
& c_k(x_k) = 0\,, \quad x_k \geq 0 && \forall k = 1, \cdots, K_s \\
& 0 \leq \tau_k^L(x_0, x_k) \perp \tau_k^U(x_k) \geq 0 && \forall k = 1, \cdots, K_s
\end{aligned}
$$

(Preliminary) Numerical results on corrective SCOPF

Observations

- Both MadNLP and Ipopt **fail** at converging on these instances
- NCL converges (empirically) to stationary points
- Code runs on the GPU, but is less robust than on the CPU – yet (ill-conditioning in K_{1s})
- If convergence achieved, we observe again a 10x speed-up w.r.t. the CPU

		NCL+K2r+MA27					**NCL+K2r+CUDSS**					**NCL+K1s+CUDSS**				
	K	st	obj	it	linsolve	**total**	st	obj	it	linsolve	**total**	st	obj	it	linsolve	**total**
1354pegase	16	1	7.4	282	235.3	**259.7**	-3	7.4	295	30.0	**35.0**	1	7.4	231	17.9	**21.1**
ACTIVSg2000	8	1	122.9	296	543.2	**564.1**	1	122.9	314	29.1	**33.9**	-3	122.9	429	31.7	**37.0**
2869pegase	8	-3	13.4	331	305.0	**340.0**	1	13.4	211	21.5	**26.7**	-3	13.4	244	19.1	**23.2**

Table: st: return status (1 if locally optimal, -3 if step is becoming too small)

Take away

1. Large-scale optimization is practical on modern GPU hardware
2. On OPF problems, we observe a **x10** speed-up compared to state-of-the-art
3. Exciting new developments are coming!