GPU-accelerated interior-point solvers

Are second-order methods relevant on GPUs?

François Pacaud

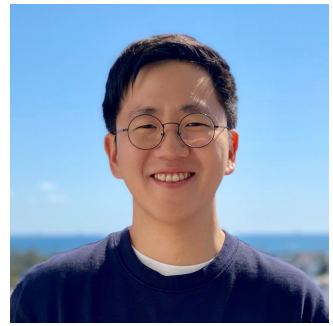
Centre Automatique et Systèmes (CAS), Mines Paris - PSL

SPOT Seminar
February 2nd 2026

Who are we?

https://madsuite.org/





- Alexis Montoison @ Argonne National Laboratory
- François Pacaud @ Mines Paris-PSL
- Sungho Shin @ MIT

GPU-accelerated optimization?

- GPUs are now broadly available in scientific computing
- In optimization, first-order methods are getting the most traction (e.g., cuPDLP)
- Achieving high precision remains challenging (tol$=10^{-8}$)

Early recollection of the barrier method in the 1960s:

> First they improved the line search (trying bisection, then a one-dimensional version of Newton's method, and finally golden-section search). Various unconstrained optimization algorithms were used to solve the barrier subproblems, at first with little success. The only thing they did not try was Newton's method, because the "conventional wisdom" among experts was that Newton's method would be impractical because of the computational expense associated with it. Computers in the early 1960s were slow and had small memories.
>
> Nevertheless, in desperation they finally programmed Newton's method, but without telling their colleagues. The program was written and then run on the diet problem. It stopped after a few seconds. Fiacco and McCormick assumed that there must be a programming error. After much hunting, though, they realized that there was no bug in the program and that Newton's method had solved the optimization problem with astonishing speed.

Research question (yes, we start all over again)

Are Newton and the barrier method still relevant with modern GPUs?

Stephen G. Nash. SUMT (Revisited). 1998

It all started with Newton method

Let $F : \mathbb{R}^n \to \mathbb{R}^n$ a smooth function. We look for a root for the system of nonlinear equations

$$F(x) = 0$$

Newton method

Starting from x_0, proceeds as

$$x_{k+1} = x_k - \partial F(x_k)^{-1} F(x_k)$$

If $F = \nabla f$, we recover the second-order method in optimization

Often require a globalization mechanism outside Newton's basins of attraction:

- Line-search
- Trust-region

Izmailov, Solodov. Newton-type methods for optimization and variational problems. 2014.

Why are we using sparse direct linear solvers?

The Newton iterations rewrites equivalently as

$$x_{k+1} = x_k + \alpha_k d_k \quad \text{with } d_k \text{ solution of } \partial F(x_k) d_k = -F(x_k)$$

Two computational bottlenecks:

- Computation of the (sparse) Jacobian ∂F using automatic differentiation
- (Sparse) Factorization of the Jacobian ∂F

Observation

If $\partial F(x_k) = LU$, solving $\partial F(x_k) d_k = -F(x_k)$ translates to two backsolves:

$$\text{Solve} \quad Lv = -F(x_k) \quad \text{then} \quad Ud_k = v$$

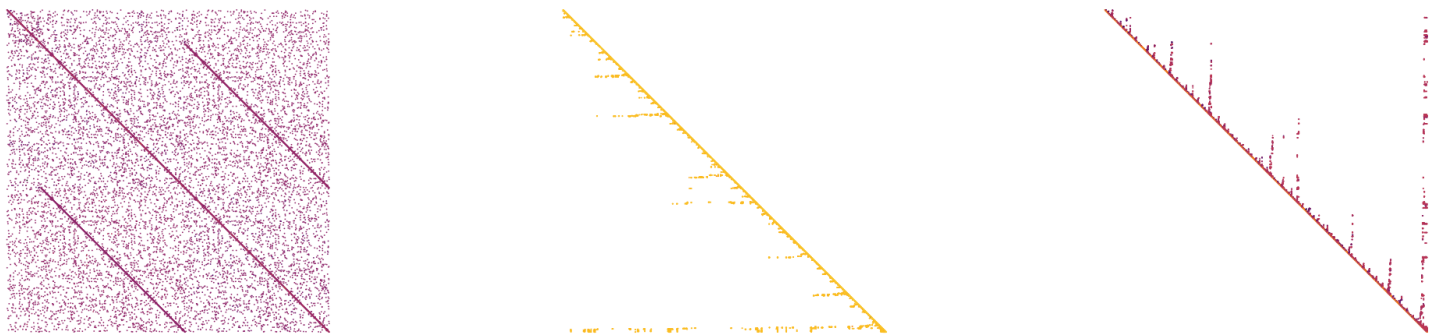


Figure: LU decomposition of a sparse Jacobian (computed using KLU)

Outline

How to solve a linear program (LP) with the interior-point method?

LP in standard form

$$\min_{x \in \mathbb{R}^n} \; c^\top x \quad \text{subject to} \quad Ax = b \,, \; x \geq 0 \,, \tag{LP}$$

with $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ problem's data

LPs are ubiquitous in operational research and mathematical optimization...

Mehrotra. On the implementation of a primal-dual interior point method. 1992.

Variational problem

We write the KKT equations for the LP in standard form

The primal-dual point $w := (x, y, z)$ is a solution of (LP) if and only if

$$
\begin{aligned}
c + A^\top y - z &= 0 \\
Ax - b &= 0 \\
0 \leq x \perp z &\geq 0
\end{aligned}
$$

Central path

For a barrier parameter $\mu > 0$, the *central path* is the set of primal-dual point $w(\mu) = (x(\mu), y(\mu), z(\mu))$ satisfying

$$
\begin{aligned}
c + A^\top y - z &= 0 \\
Ax - b &= 0 \\
XZe = \mu e \ , \ (x, z) &> 0
\end{aligned}
$$

As $\mu \to 0$, the solution $w(\mu)$ converges to a solution of (LP)

- An interior-point method (IPM) solves (LP) by tracking the *central path*
- The **Mehrotra's predictor-corrector** remains the most efficient IPM for (LP)

Wright. Primal-dual interior-point methods. 1997.

A small example is always better than a long discussion

$$\begin{aligned} \min \quad & 2x + 5y \\ \text{subject to} \quad & x + 2y \geq 0.5 \\ & 0 \leq x \leq 1 \\ & 0 \leq y \leq 1 \end{aligned}$$

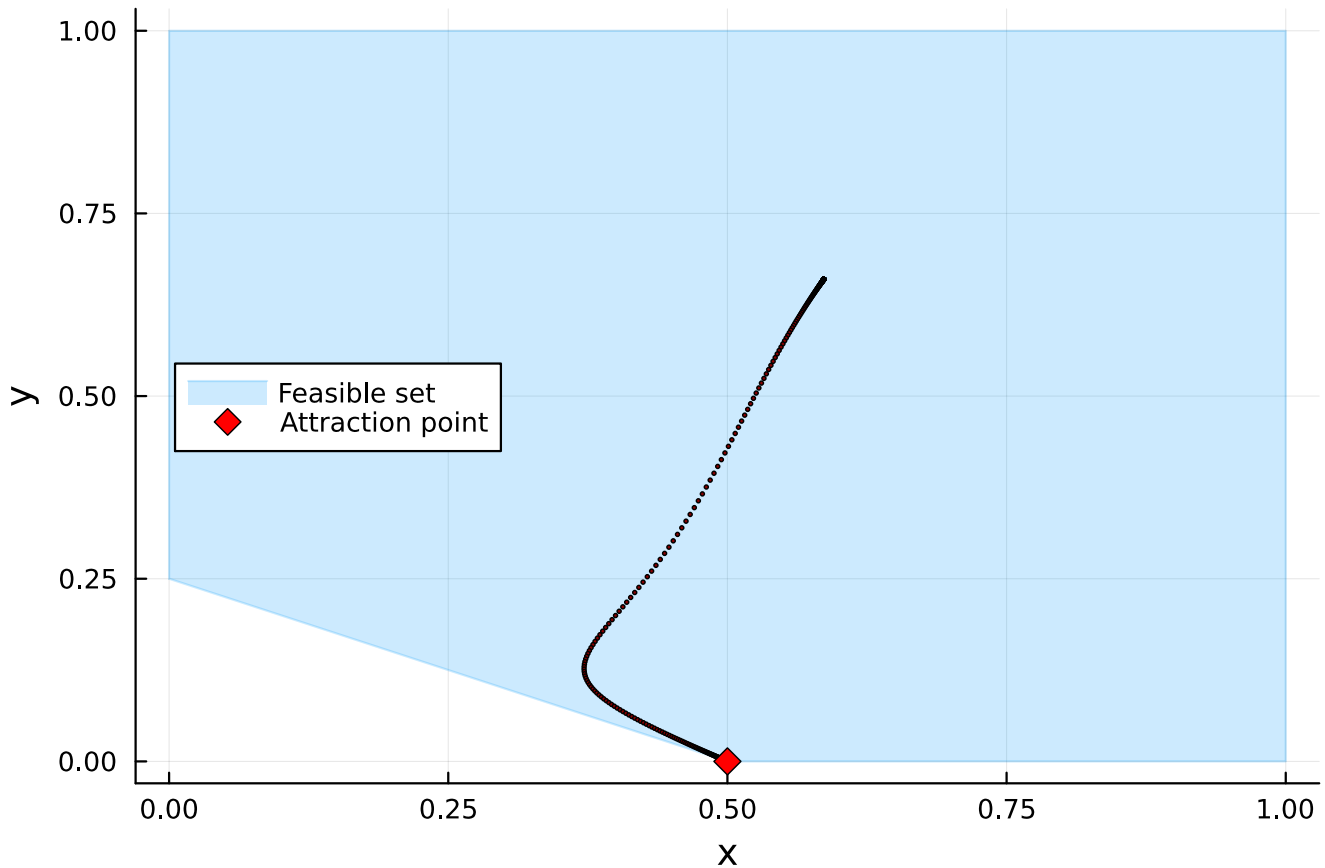


Figure: A unique central path

Tracking the central path with Mehrotra's predictor-corrector

Decomposing one IPM iteration

Barrier update: Set the barrier at the average complementarity:

$$\mu = \frac{z^\top x}{n} \, .$$

Affine step: Compute Δ^{aff} by solving

$$\begin{bmatrix} 0 & A^\top & -I \\ A & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x^{\text{aff}} \\ \Delta y^{\text{aff}} \\ \Delta z^{\text{aff}} \end{bmatrix} = - \begin{bmatrix} c + A^\top y - z \\ Ax - b \\ XZe \end{bmatrix} \, .$$

Corrector step: Compute Δ^{corr} by solving

$$\begin{bmatrix} 0 & A^\top & -I \\ A & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x^{\text{corr}} \\ \Delta y^{\text{corr}} \\ \Delta z^{\text{corr}} \end{bmatrix} = - \begin{bmatrix} 0 \\ 0 \\ \sigma\mu e - \Delta Z^{\text{aff}} \Delta X^{\text{aff}} e \end{bmatrix} \, ,$$

with σ given by Mehrotra's heuristic

Next iterate: For $\Delta_k = \Delta^{\text{aff}} + \Delta^{\text{corr}}$, set

$$w_{k+1} = w_k + \alpha_k \Delta_k$$

with α_k a step computed using a *fraction-to-boundary rule* to keep $(x_{k+1}, z_{k+1}) > 0$

Mehrotra. On the implementation of a primal-dual interior point method. 1992.

The reliance on sparse linear solvers

Both the affine step and the corrector step are solution of:

$$\begin{bmatrix} 0 & A^\top & -I \\ A & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} .$$

$\rightarrow$ sparse, not symmetric

Augmented KKT system

The previous system is equivalent to the symmetric indefinite system:

$$\begin{bmatrix} \Sigma & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r_1 + X^{-1} r_3 \\ r_2 \end{bmatrix} ,$$

with the diagonal matrix $\Sigma := X^{-1}Z$.

Normal KKT system

We can also eliminate Δy to recover a positive-definite system:

$$\left(A\Sigma^{-1}A^\top\right) \Delta y = A\Sigma^{-1}\left(r_1 + X^{-1} r_3\right) - r_2 .$$

Most IPM solvers use the normal KKT system.

Primal-dual regularization

It is the key for GPU performance!

Both the augmented and normal KKT systems have issues if problem has:

- Free variables
- Rank-deficient Jacobian A
- Dense rows in A (leads to a dense matrix $A\Sigma^{-1}A^\top$)

Regularized IPM

For parameters $(\rho, \delta) > 0$, solve

$$\begin{bmatrix} \Sigma + \rho I & A^\top \\ A & -\delta I \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r_1 + X^{-1} r_3 \\ r_2 \end{bmatrix} .$$

- The matrix is **symmetric quasi-definite** (SQD), meaning that it is strongly factorizable using a signed Cholesky factorization.
- Regularized IPM is the standard IPM applied to the **regularized LP**:

$$\min_{x,r} \; c^\top x + \frac{\rho}{2}\|x\|^2 + \frac{1}{2\delta}\|r\|^2$$
$$\text{subject to } Ax + r = b \;,\; x \geq 0$$

Altman, Gondzio. Regularized symmetric indefinite systems in interior point methods for linear and quadratic optimization. 1999.
Friedlander, Orban. A primal–dual regularized interior-point method for convex quadratic programs. 2012.
Gondzio. Matrix-free interior point method. 2012.

The signed Cholesky factorization

Cholesky factorization

If K is symmetric positive definite, there exists a lower triangular matrix L such that

$$K = LL^\top$$

Generalize to symmetric quasidefinite matrix

$$K = \begin{bmatrix} A & B^\top \\ B & -C \end{bmatrix}$$

with A and C symmetric positive definite matrices

Signed Cholesky factorization

If K is symmetric quasidefinite, there exists a unit lower triangular matrix L and a diagonal matrix D such that

$$K = LDL^\top$$

Same properties as the Cholesky factorization, but with more flexibility:

- The LDL factorization is numerically stable and does not require expensive numerical pivoting
- Easy to implement in parallel

Vanderbei. Symmetric Quasidefinite matrices. 1995

MadIPM

https://github.com/MadNLP/MadIPM.jl

In a nutshell,

- MadIPM is a GPU-accelerated interior-point solver for LPs
- Implement Mehrotra's predictor-corrector in pure Julia
- Solve the regularized KKT system with NVIDIA cuDSS

Experiments

We compare MadIPM with Gurobi, running in parallel using 16 threads.

MadIPM: performance profile on MIPLIB

The upper, the better

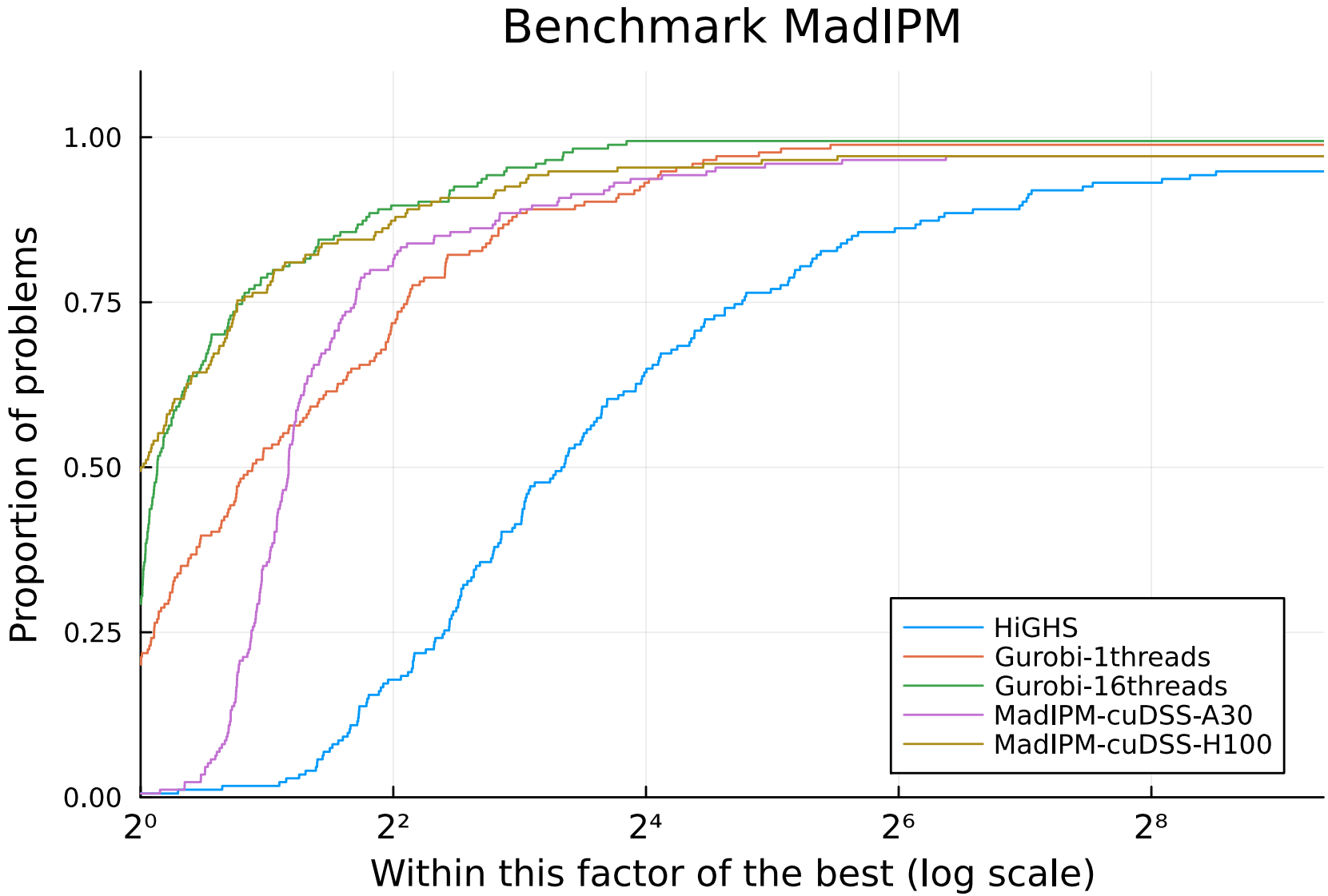


Figure: Benchmarking MadIPM, Gurobi and HiGHS on 174 large-scale LP instances from MIPLIB

MadIPM: raw performance

The lower, the better

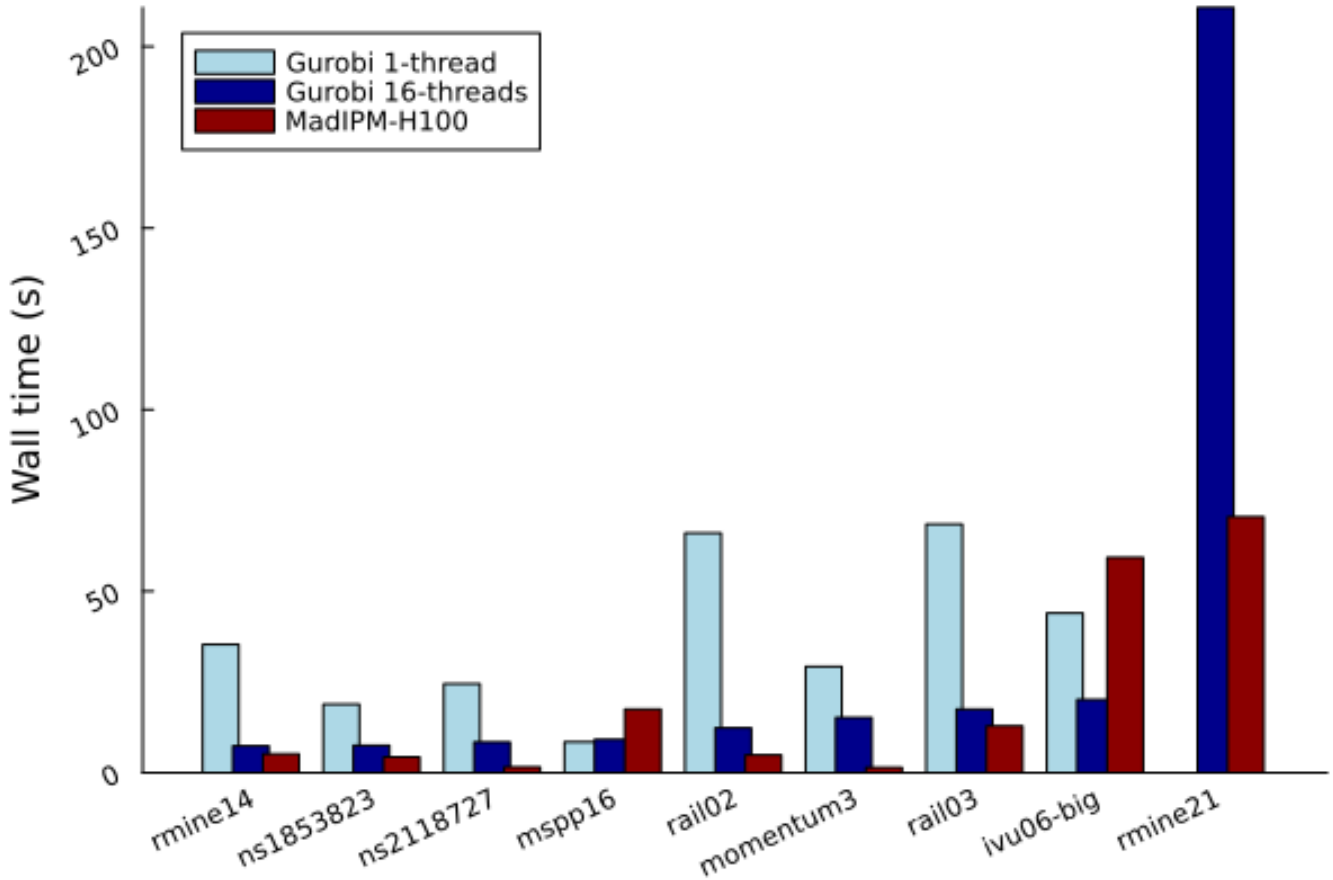


Figure: Zoom on the largest instances in MIPLIB

Outline

How to solve a nonlinear program (NLP) with interior-point?

NLP in standard form

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

with $f(\cdot)$ and $c(\cdot)$ to twice continuously differentiable functions

Most use-cases in power engineering, chemical engineering and optimal control

(NLP) is much harder than (LP):

- Non-convex, existence of spurious solutions
- The central path is not a well defined notion for (NLP)

Two reference interior-point solvers:

- Ipopt: Filter line-search
- Knitro: Byrd-Omojokun algorithm

Variational problem

We write the KKT equations for (NLP) in standard form

The primal-dual point $w := (x, y, z)$ is a stationary solution of (NLP) if

$$\begin{aligned}
&\nabla f(x) + \nabla c(x)^\top y - z = 0 \\
&c(x) = 0 \\
&0 \leq x \perp z \geq 0
\end{aligned}$$

For a barrier parameter $\mu > 0$, IPM aims at solving:

$$\begin{aligned}
&\nabla f(x) + \nabla c(x)^\top y - z = 0 \\
&c(x) = 0 \\
&XZe = \mu e
\end{aligned}$$

Primal-dual interior-point method

Under some regularity assumptions, globalized Newton algorithm
(with proper update for μ) converges to a local stationary solution

A second example showing why NLPs are difficult

$$
\begin{aligned}
\min \quad & 100\left(-x1^2 + x2\right)^2 + (x1 - 1)^2 \\
\text{subject to} \quad & x1 \times x2 \geq 1 \\
& x2^2 + x1 \geq 0 \\
& x1 \leq 0.5
\end{aligned}
$$

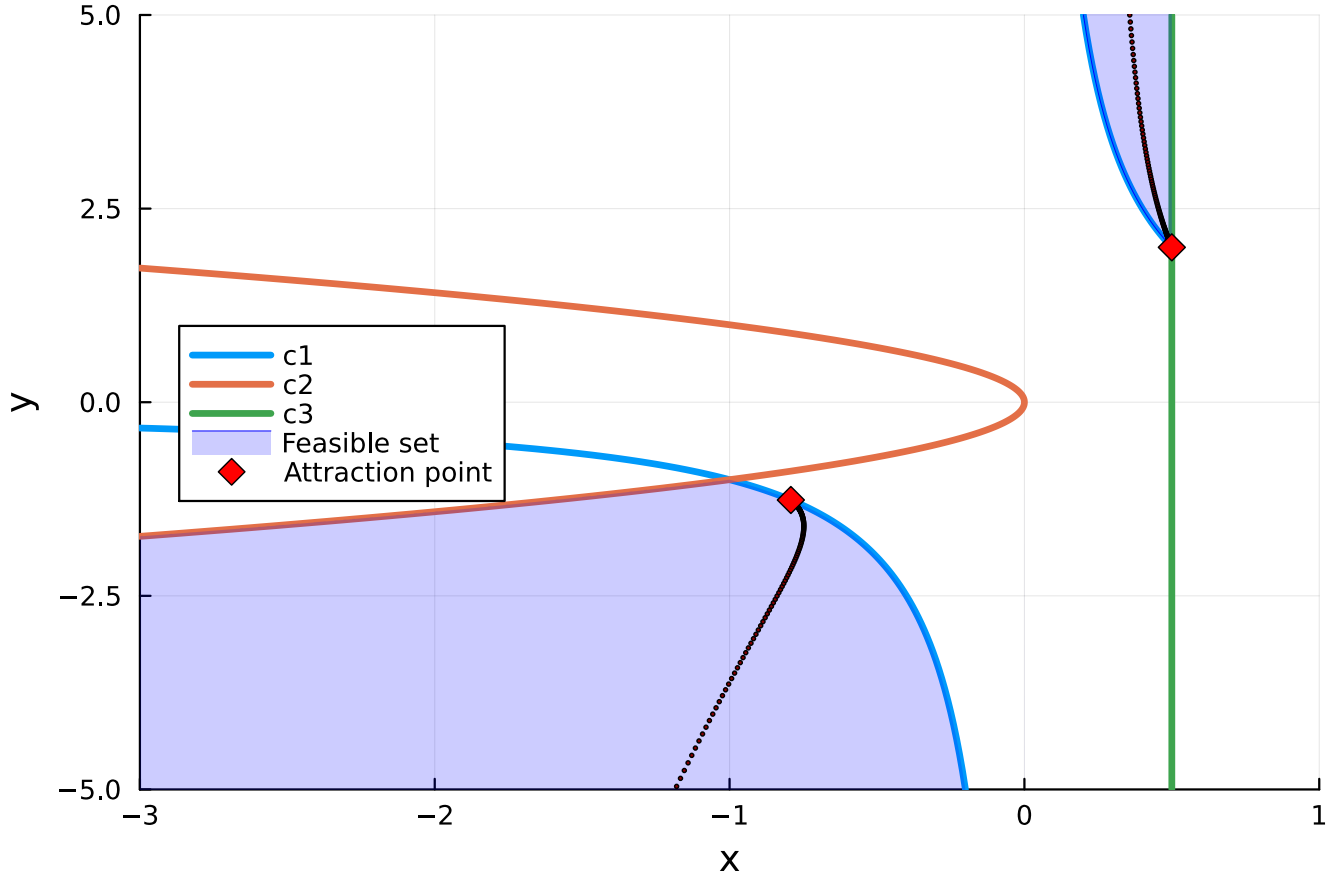


Figure: Multiple solutions, multiple central paths

Filter line-search IPM (ala Ipopt)

Barrier update: For a given primal-dual iterate $w = (x, y, z)$, update the barrier parameter using a *monotone* rule.

Sensitivities: Compute the Jacobian $J := \nabla c(x)^\top$ and the Hessian $W := \nabla^2_{xx} L(x, y)$ using *sparse* automatic differentiation

Newton system: Compute the descent direction Δw by solving

$$\begin{bmatrix} W + \rho I & J^\top & -I \\ J & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = - \begin{bmatrix} \nabla f(x) + J^\top y - z \\ c(x) \\ XZe \end{bmatrix} .$$

with ρ a regularization parameter ensuring that

$$Z^\top (W + \rho I) Z \succ 0 \quad \text{for} \quad Z \text{ a basis for } \mathrm{Ker}(J)$$

(ρ usually computed using an *inertia-correction* procedure)

Next iterate: Set

$$w_{k+1} = w_k + \alpha_k \Delta w$$

with α_k a step computed using a filter line-search

Again, we rely on sparse linear solvers

This time, the **unsymmetric linear system** writes as:

$$\begin{bmatrix} W + \rho I & J^\top & -I \\ J & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix}$$

Augmented KKT system

The unsymmetric system reduces to:

$$\begin{bmatrix} W + \Sigma + \rho I & J^\top \\ J & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r_1 + X^{-1} r_3 \\ r_2 \end{bmatrix},$$

with the diagonal matrix $\Sigma := X^{-1}Z$.

By default, most solvers look at the *symmetric indefinite* augmented KKT system (no equivalent of the normal KKT system there)

The Duff-Reid LBL factorization

Static and numerical pivoting

Duff-Reid factorization

$$A = PQ\,LBL^\top\,Q^\top P^\top$$

with

- P: fill-in minimization matrix (= static pivoting)
- Q: additional pivoting for numerical stability (= numerical pivoting)
- L: unit lower-triangular matrix
- B: block diagonal matrix with blocks of dimension 1×1 or 2×2

- The LBL factorization has become competitive only in the 1990s, using a technique known as *matching-based preprocessing*
- Getting an efficient algorithm for the numerical pivoting Q remains highly non trivial
- Does not parallelize well

Duff, Erisman, Reid. Direct methods for sparse matrices. 2017.

Idea: augmented Lagrangian reformulation

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

$$\begin{bmatrix} W_\gamma & J^\top \\ J & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = - \begin{bmatrix} r_1 + X^{-1} r_3 + \gamma J^\top r_2 \\ r_2 \end{bmatrix}$$

with $W_\gamma := W + \Sigma + \rho I + \gamma J^\top J$

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

$$J W_\gamma^{-1} J^\top \Delta y = r_2 - W_\gamma^{-1} \gamma$$

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

Regev et al., HyKKT: a hybrid direct-iterative method for solving KKT linear systems. 2023.

Faster (sparse) derivative evaluation with ExaModels.jl

Coming back to factorable programming

Remark

We also need to compute the sparse derivatives J and W on the GPU using automatic differentiation

- 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**

MadNLP

https://github.com/MadNLP/MadNLP.jl

In a nutshell,

- MadNLP is a GPU-accelerated interior-point solver for NLPs
- Implement filter line-search in pure Julia
- Solve the Golub & Grief KKT system with NVIDIA cuDSS

Experiment

Compare the time to solution (in seconds) with Ipopt running with the HSL solvers

The method gives excellent results on AC-OPF problems...

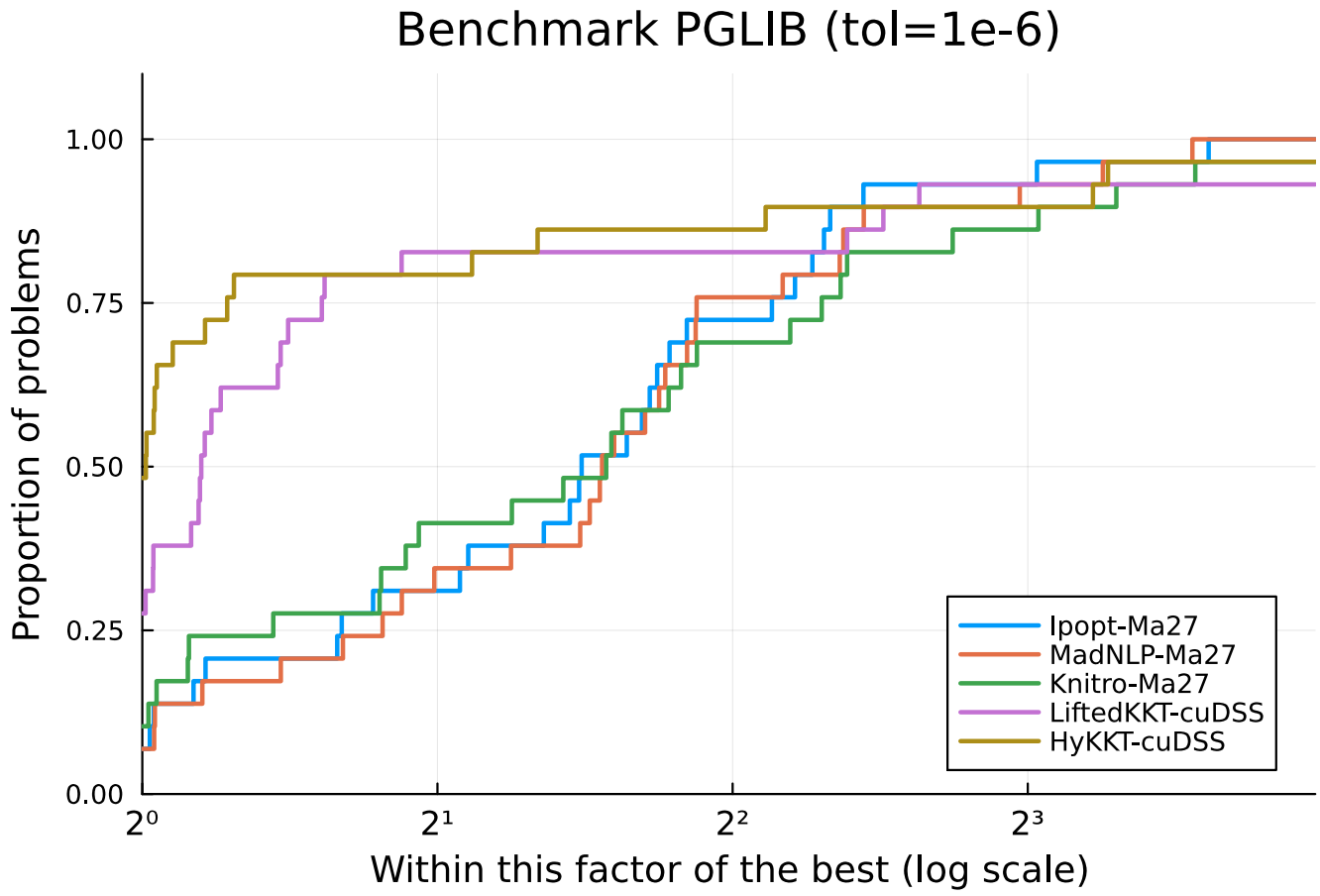


Figure: Performance profile, MadNLP against Ipopt on large-scale AC-OPF instances

...but results are more mitigated on CUTEst

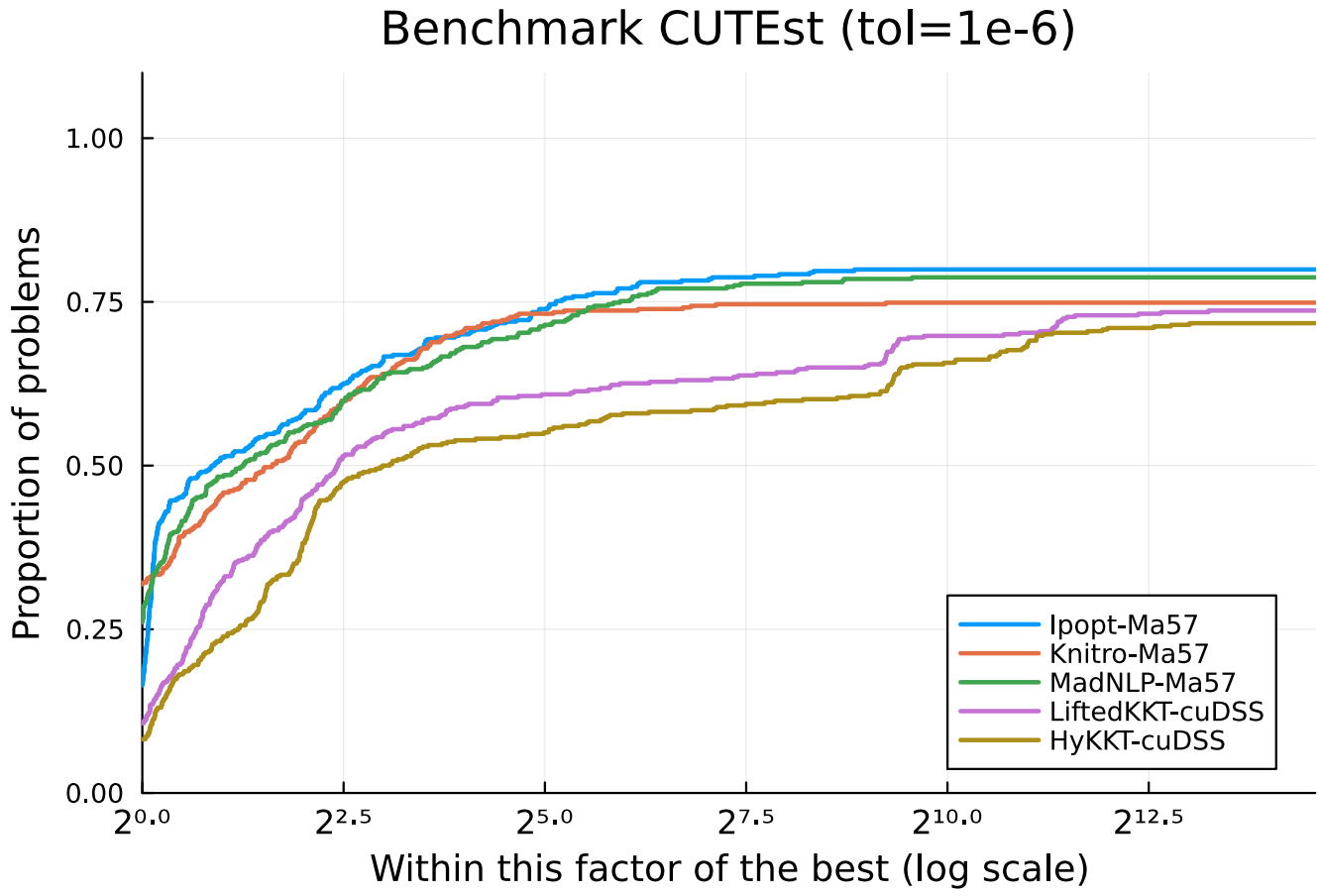


Figure: Performance profile, MadNLP against Ipopt on large-scale CUTEst instances

But...

Instances in CUTEst are too small ...

How expensive should be your GPU?

Image courtesy of Sungho Shin

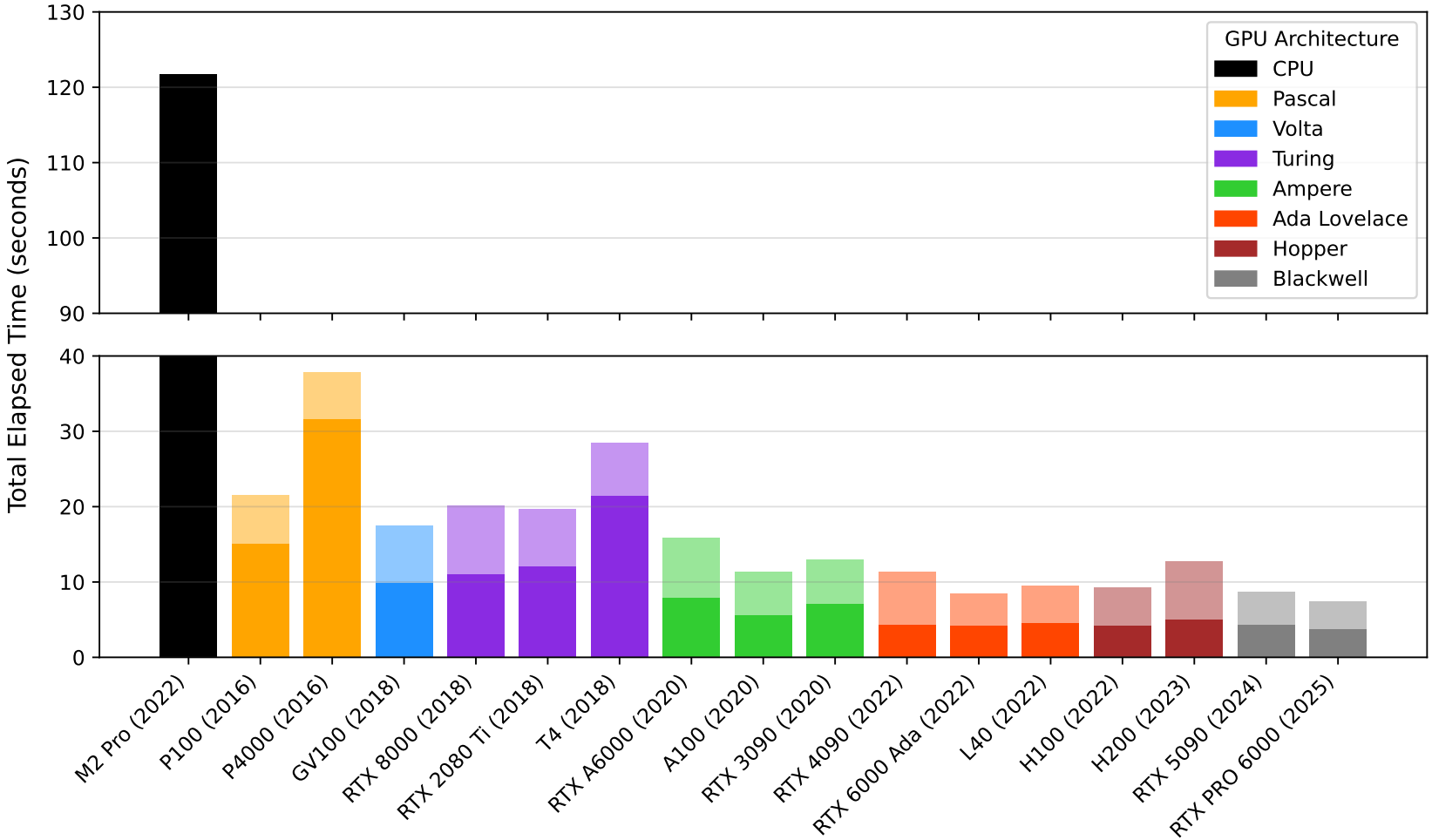


Figure: Time to optimality, here for a large-scale optimal power flow instance.

No need to buy a professional GPU for fast performance!

Outline

Extensions (WIP)

Semi-definite programs (SDP)

$$\min_{X \in \mathbb{S}^n} \ \mathrm{tr}(CX) \quad \text{subject to} \quad \mathcal{A}X = b \ , \ X \succeq 0$$

Mathematical programs with complementarity constraints (MPCC)

$$\min_{x \in \mathbb{R}^n} \ f(x) \quad \text{subject to} \quad c(x) = 0 \ , \ 0 \leq x_1 \perp x_2 \geq 0$$

Challenges:

- Both problems exhibit *significant* ill-conditioning in their respective KKT systems
- They break state-of-the-art sparse direct solver running on the CPU...

Conclusion

Take-away

- Yes, Newton method is practical on the GPU!
- Sparse direct linear solvers are key
- With GPUs, you can expect up to x10 speed-up on difficult instances, but not always!

Next step

Solve LPs in **batch** on the GPU!

- Solve multiple LPs **simultaneously**, assuming the same KKT sparsity pattern
- Reuse **symbolic analysis** across all sparse linear systems
- Different central paths $\rightarrow$ real-time rebalancing when some systems converge earlier.
- Exploit low-rank structure in solution matrix?