GPU-accelerated optimal control

A nonlinear programming point-of-view

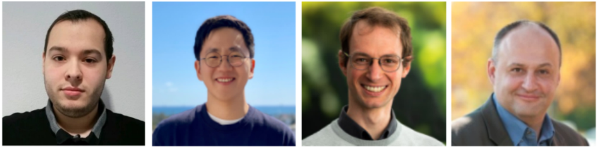
François Pacaud
frapac.github.io

Centre Automatique et Systèmes, Mines Paris - PSL

Guest lecture
Georgia Institute of Technology
October 17th, 2025

Who are we?

https://madsuite.org/



- Alexis Montoison @ ANL
- Sungho Shin @ MIT
- Mihai Anitescu @ ANL

Outline: today we talk about GPU-accelerated optimal control

- We leverage the GPU-accelerated optimization solver MadNLP to solve large-scale optimal control on the GPU

Motivation for today's lecture

☞**New generation of optimization solvers is under way**

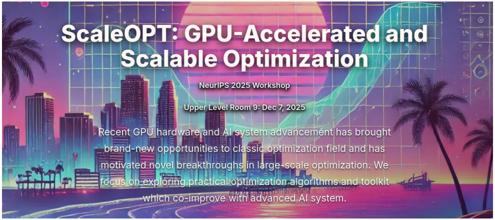


Figure: Go check Sungho's talk at ScaleOpt!

☞**Massive investments in robotics**

Nvidia Bets Big on Robots

The chipmaker, which has led a rally in artificial intelligence stocks, laid out a vision for dominating so-called physical A.I. Investors appeared impressed.

Share full article

By Andrew Ross Sorkin, Ravi Mattu, Bernhard Warner, Sarah Kessler, Michael J. de la Merced and Lauren Hirsch

Jan. 7, 2025

Figure: NVIDIA Jetson

Outline

What am I talking about when talking about optimal control?

Primal-dual interior-point method

How to solve the Newton systems on the GPU?

Using MadNLP and ExaModels on the GPU

Applications of optimal control

Finding optimal trajectory in problems with a dynamic structure

- Trajectory planning
- Real-time optimization with model predictive control (MPC)

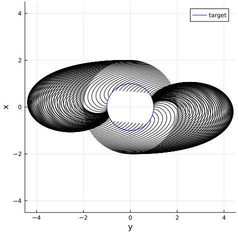


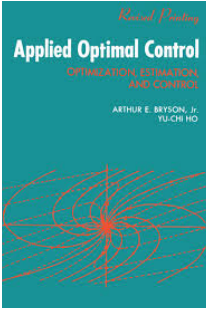
Figure: Trajectory flow for moving a satellite locally close to its orbit

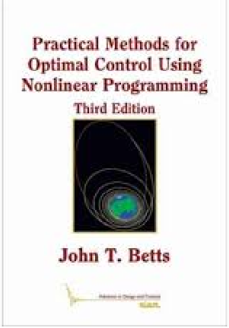
A mature ecosystem already exists!

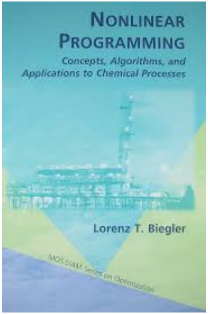
Classical workflow:

- Modeling with casadi
- Solving with Ipopt (or any structure exploiting solver)

Useful references







What do I mean by optimal control?

Disclaimer: We write everything in discrete time

At discrete time k, the *state* of the system is encoded in a vector $x_k \in \mathbb{R}^{n_x}$.
After applying a *control* $u_k \in \mathbb{R}^{n_u}$, the system evolves between k and $k+1$ as

$$x_{k+1} = g_k(x_k, u_k)$$

Most of the time, the dynamics $g_k(\cdot)$ discretizes an ordinary differential equations (e.g. using collocation)

The system is subject to the operational constraints:

$$h_k(x_k, u_k) \leq 0$$

Optimal control problem

Starting from $\underline{x} \in \mathbb{R}^{n_x}$, we minimize the functional $\ell_k(\cdot)$ over an horizon N:

$$\begin{aligned} \min_{x,u} \; & \sum_{k=0}^{N-1} \ell_k(x_k, u_k) + \ell_N(x_N) \\ \text{s.t. } & x_{k+1} = g_k(x_k, u_k) \; , \; x_0 = \underline{x} \\ & h_k(x_k, u_k) \leq 0 \end{aligned}$$

Let $x := (x_0, \cdots, x_N)$ and $u := (u_0, \cdots, u_{N-1})$.

Introducing the nonlinear program

Many degrees-of-freedom approach

For $w := (x, u)$, we abstract the previous dynamic program as:

$$\begin{aligned} &\min_{w} f(w) \\ &\text{s.t. } g(w) = 0\,,\ h(w) \leq 0\,. \end{aligned} \tag{NLP}$$

Nonlinear programming

- Problem is nonlinear nonconvex :
  we are interested in finding only a **local** optimum solution
- Solvable using classical nonlinear solvers:
  - IPM (Ipopt, Knitro, MadNLP)
  - SQP (FilterSQP, SNOPT)
  - Augmented-Lagrangian (LANCELOT, Algencan)

J. Nocedal, SJ. Wright. Numerical optimization.

Karush-Kuhn-Tucker (KKT) conditions

Using a slack variable $s \geq 0$, NLP is equivalent to

$$\begin{aligned} \min_{w,s} \; & f(w) \\ \text{s.t. } & g(w) = 0 \,, \; h(w) + \underbrace{\boxed{s}}_{\text{Slack}} = 0 \\ & s \geq 0 \end{aligned} \tag{NLP}$$

We introduce the *Lagrangian*:

$$\mathcal{L}(w, y, z) = f(w) + y^\top g(w) + z^\top h(w) \,.$$

KKT stationary equations

If w is a *regular* local solution, then there exist dual multipliers (y, z) satisfying

$$\begin{cases} \nabla f(w) + \nabla g(w)^\top y + \nabla h(w)^\top z = 0 \\ g(w) = 0 \\ h(w) + s = 0 \\ \underbrace{\boxed{0 \leq s \perp z \geq 0}}_{\text{Complementarity conditions}} \end{cases}$$

☞ Solving the NLP is equivalent to the solution of a system of *nonsmooth* nonlinear equations

Regularity conditions: first-order

Denote the active set $\mathcal{A}(w) = \{i \in [m] \mid h_i(w) = 0\}$ and the active Jacobian

$$J(w) = \begin{bmatrix} \nabla g(w) \\ \nabla h_{\mathcal{A}}(w) \end{bmatrix}$$

First-order constraint qualification

We say that the point w is *qualified* is the local geometry of the feasible can be captured by a *linearized* model near w

(in math language, the tangent cone is equal to the set of linearized feasible directions)

- The *Linear Independence Constraint Qualification* (LICQ) holds if the active Jacobian $J(w)$ is full row-rank
- The *Mangasarian-Fromovitz Constraint Qualification* (MFCQ) holds if $(y, z) = (0, 0)$ is the unique solution to the linear system

$$\begin{cases} \nabla g(w)^\top y + \nabla h(w)^\top z = 0 & \\ z_i \geq 0 & \forall i \in \mathcal{A}(w) \end{cases}$$

Gauvin's theorem (1979)

MFCQ holds if and only if the set of multipliers (y, z) satisfying KKT is bounded

Regularity conditions: second-order

Let (w, y, z) a primal-dual point satisfying KKT. We define the *critical cone*:

$$\begin{aligned} \mathcal{C}(w, y, z) = \{d \in \mathbb{R}^n \mid \nabla g_i(w)^\top d = 0\ ,\ i \in [m_e]\ , \\ \nabla h_i(w)^\top d = 0\ ,\ i \in \mathcal{A}(w) \text{ with } z_i > 0\ , \\ \nabla h_i(w)^\top d \leq 0\ ,\ i \in \mathcal{A}(w) \text{ with } z_i = 0\}\ . \end{aligned}$$

Strict complementarity (SC) holds if $z_i > 0$ for all $i \in \mathcal{A}(w)$.

Second-order sufficient condition (SOSC)

We say that (w, y, z) satisfies SOSC if

$$d^\top \nabla^2_{ww} \mathcal{L}(w, y, z) d > 0 \quad \forall d \in \mathcal{C}(w, y, z) \setminus \{0\}$$

Under SOSC, the problem is locally convex near w and the solution is isolated

Proposition

Suppose LICQ and SC hold. Then SOSC holds if and only if

$$Z^\top \nabla^2_{ww} \mathcal{L}(w, y, z) Z \succ 0$$

with Z a basis of the null-space of the active Jacobian $J(w)$.

Are optimal control problems regular?

Well... it depends

State constraints

- Pure state constraints: $h_k(x_k) \leq 0$
- Mixed state constraints: $h_k(x_k, u_k) \leq 0$

☞ Problems with active *state constraints* do not satisfy LICQ
(not enough degrees of freedom)

☞ Problems with *singular arcs* do not satisfy SOSC
Example: Goddard rocket's problem

Observation

Oftentimes, nonlinear solvers are struggling to solve optimal control instances

Outline

Interior-point method (IPM)

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

Dual variables

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

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

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

The Newton algorithm translates to the solution of a sequence of linear systems

Denote the primal-dual variable by $v^k := (w^k, s^k, y^k, z^k)$. At iteration k,

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

$$\boxed{\nabla F_\mu(v^k)}\; d^k = -F_\mu(v^k)$$

2. Update the as

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

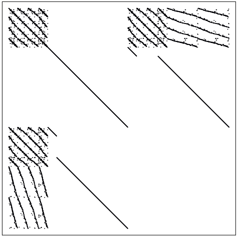


Figure: ∇F_μ

Augmented KKT system

After (slight) reformulation, the Newton step writes as

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

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

☞ This is the system solved by default in Ipopt

Generic case: sparse linear solver for nonlinear programming

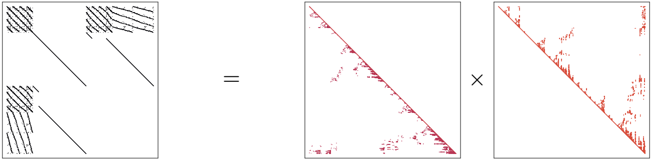


Figure: Matrix factorization using a direct solver

Direct method for sparse indefinite linear systems

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

B. Tasseff, C. Coffrin, A. Wächter, C. Laird. "Exploring benefits of linear solver parallelism on modern nonlinear optimization applications.", 2019

Computing sparse factorizations

Duff-Reid factorization

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

with

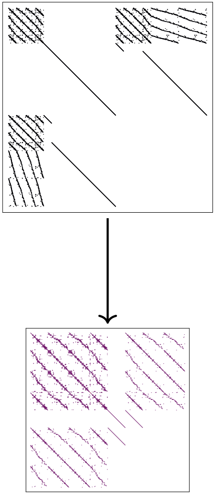
- P: fill-in minimization matrix
- Q: additional pivoting for numerical stability
- L: unit lower-triangular matrix
- B: block diagonal matrix with blocks of dimension 1×1 or 2×2

- ☞ The LBL factorization has become competitive only in the 1990s, using a technique known as matching-based preprocessing
- ☞ The progress in sparse linear solvers has directly benefited to nonlinear optimization solvers such as Ipopt or Knitro
- ☞ Numerical pivoting Q impairs the parallelism in the algorithm

I. Duff, J. Koster. (2001). "On algorithms for permuting large entries to the diagonal of a sparse matrix.", 2001

Condensed KKT system

We condense the linear system by removing the inequality blocks



Condensed KKT system

The augmented KKT system condenses to

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

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

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

- ☞ Additional fill-in
- ☞ Useful when large number of inequality constraints m

F. Pacaud, S. Shin, M. Schanen, DA. Maldonado, M. Anitescu. "Accelerating condensed interior-point methods on SIMD/GPU architectures." 2024

Condensed KKT systems in optimal control

Writing the control as a feedback on the state

Proposition

For K positive definite, the solution of the saddle-point linear system

$$\begin{bmatrix} K & G^\top \\ G & 0 \end{bmatrix} \begin{bmatrix} w \\ y \end{bmatrix} = \begin{bmatrix} -c \\ b \end{bmatrix}$$

is the primal-dual solution of the convex QP

$$\min_w \frac{1}{2} w^\top K w + c^\top w \quad \text{s.t.} \quad Gw = b$$

For problem with a dynamic structure, the condensed KKT system yields the LQR

$$\min_{d_x, d_u} \sum_{k=0}^{N-1} \begin{bmatrix} d_{x,k} \\ d_{u,k} \end{bmatrix}^\top \begin{bmatrix} K_{xx}^k & K_{xu}^k \\ K_{ux}^k & K_{uu}^k \end{bmatrix} \begin{bmatrix} d_{x,k} \\ d_{u,k} \end{bmatrix} + \begin{bmatrix} r_{x,k} \\ r_{u,k} \end{bmatrix}^\top \begin{bmatrix} d_{x,k} \\ d_{u,k} \end{bmatrix}$$

$$\text{s.t.} \quad d_{x,k+1} = G_{x,k} d_{x,k} + G_{u,k} d_{u,k}$$

- ☞ No need to use a sparse linear solver
- ☞ Efficient solution with (backward) Riccati recursions
- ☞ Require convexification in nonlinear programming

JC. Dunn, D. Bertsekas. "Efficient dynamic programming implementations of Newton's method for unconstrained optimal control problems." 1989
R. Verschueren, M. Zanon, R. Quirynen, M. Diehl. "A sparsity preserving convexification procedure for indefinite quadratic programs arising in direct optimal contro

Remember that the IPM solution is always inaccurate!

IPM is NOT an active-set method

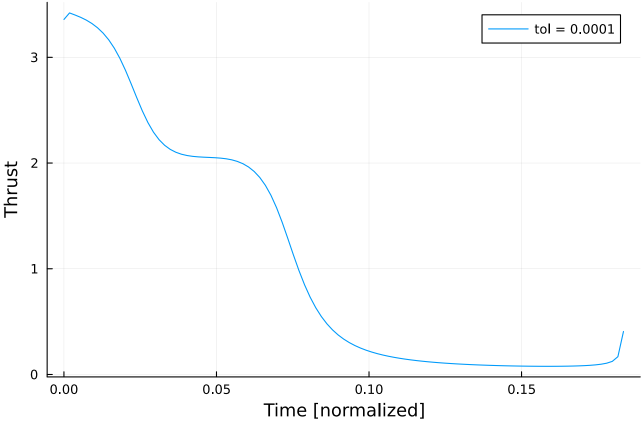


Figure: Optimal solution of the Goddard rocket problem with $nh = 100$

Remember that the IPM solution is always inaccurate!

IPM is NOT an active-set method

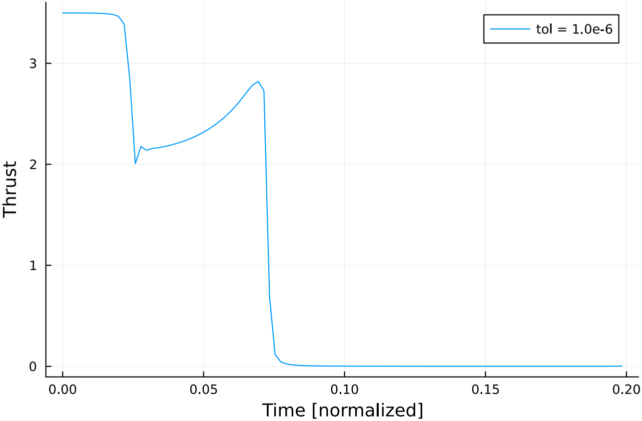


Figure: Optimal solution of the Goddard rocket problem with $nh = 100$

Remember that the IPM solution is always inaccurate!

IPM is NOT an active-set method

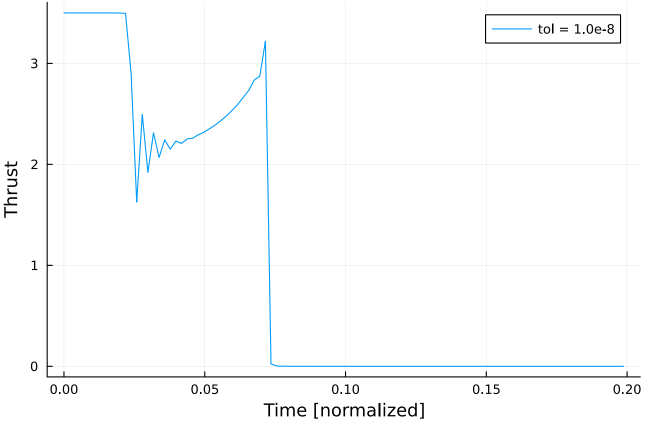


Figure: Optimal solution of the Goddard rocket problem with $nh = 100$

Remember that the IPM solution is always inaccurate!

IPM is NOT an active-set method

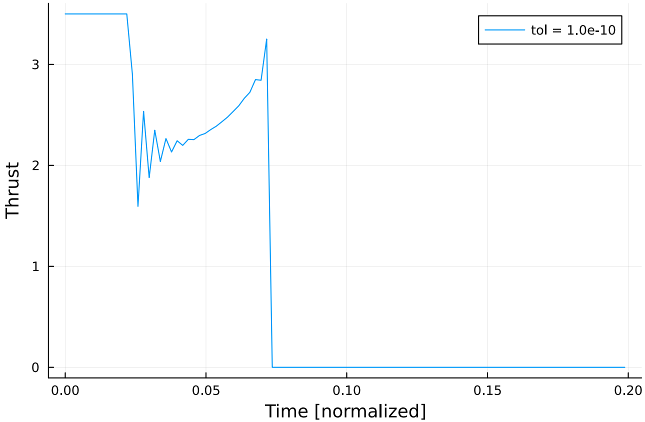


Figure: Optimal solution of the Goddard rocket problem with $nh = 100$

Remember that the IPM solution is always inaccurate!

IPM is NOT an active-set method

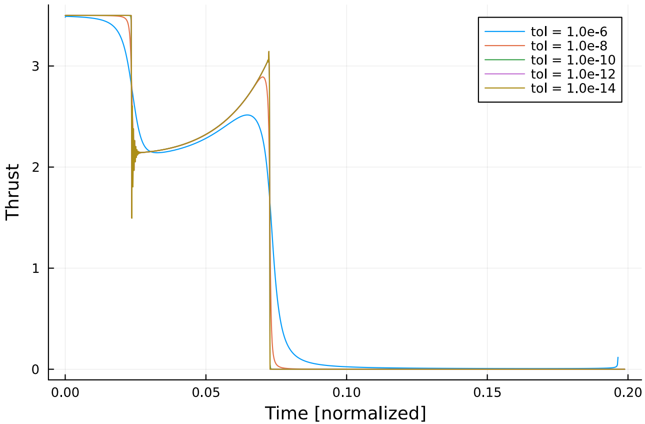


Figure: Optimal solution of the Goddard rocket problem with $nh = 1000$

Outline

What am I talking about when talking about optimal control?

Primal-dual interior-point method

How to solve the Newton systems on the GPU?

Using MadNLP and ExaModels on the GPU

Why is it challenging?

Current status

- Implementing a sparse linear solver on the GPU is highly non-trivial
- Before the release of NVIDIA cuDSS, there was no efficient sparse linear solver available on the GPU
- Solving the KKT systems in IPM with an iterative solver remains an open research question

Two solution methods for GPU-accelerated optimization method:

1. Rewrite the KKT system as a dense matrix ;
2. Use a pivoting-free factorization (at the price of sacrificing slightly the accuracy).

Reduced-space approach: densify the solution!

Few degrees of freedom

Proposition

Suppose the Jacobian ∇g is full rank. Let Z be a matrix whose columns encode a basis of the Jacobian null-space ($(\nabla g)Z = 0$). Then the condensed KKT system is equivalent to

$$Z^\top K Z d_u = \bar{r}_{red}$$

Oftentimes, the matrix $Z^\top K Z$ is dense.

Example of null space for optimal control:

$$\nabla g = \begin{bmatrix} G_x & G_u \end{bmatrix} \quad \Longrightarrow \quad Z = \begin{bmatrix} -G_x^{-1} G_u \\ I \end{bmatrix}$$

Works best if number of control is small (e.g. PDE constrained optimization)!

D. Cole, S. Shin, F. Pacaud, VM. Zavala. "Exploiting GPU/SIMD architectures for solving linear-quadratic MPC problems." 2022
F. Pacaud, S. Shin, M. Schanen, DA. Maldonado, M. Anitescu. "Accelerating condensed interior-point methods on SIMD/GPU architectures." 2024

How to avoid numerical pivoting?

Reformulate the KKT system either as

- a positive definite (PD) system

$$A \succ 0$$

- a symmetric quasidefinite (SQD) system

$$\begin{bmatrix} A & B^\top \\ B & -C \end{bmatrix} \quad \text{with} \quad A \succ 0\,,\; C \succ 0$$

In both cases, the sparse system can be factorized using only *static pivoting*, e.g. using a signed Cholesky factorization

$$A = PLDL^\top P^\top$$

with P static pivoting, L lower triangular, D diagonal matrix.

RJ. Vanderbei. "Symmetric quasidefinite matrices." 1995

A. Montoison, F. Pacaud, S. Shin, M. Anitescu. "GPU Implementation of Second-Order Linear and Nonlinear Programming Solvers." 2025

How to avoid numerical pivoting in nonlinear programming?

Objective

How to avoid numerical pivoting in the algorithm?

We look again at the condensed KKT system (sparse symmetric indefinite):

$$\begin{bmatrix} \boxed{K} & \nabla g^\top \\ \nabla g & 0 \end{bmatrix} \begin{bmatrix} d_w \\ d_y \end{bmatrix} = - \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$$

with the *condensed matrix* $K = W + \nabla h^\top \Sigma_s \nabla h$.

☞Three strategies to avoid numerical pivoting:

1. LiftedKKT
2. HyKKT
3. NCL (Augmented Lagrangian)

S. Shin, F. Pacaud, M. Anitescu. *Accelerating optimal power flow with GPUs: SIMD abstraction of nonlinear programs and condensed-space interior-point method.*
S. Regev et al., "HyKKT: a hybrid direct-iterative method for solving KKT linear systems." Optimization Methods and Software 38, no. 2 (2023)

Strategy 1: LiftedKKT

Idea: equality relaxation

For a $\tau > 0$ small enough, solve the relaxed problem

$$\min_{w \in \mathbb{R}^n} \; f(w) \quad \text{s.t.} \quad \begin{cases} -\tau \leq g(w) \leq \tau \\ \qquad h(w) \leq 0 \end{cases}$$

Reformulating the problem with slack variables:

$$\min_{w \in \mathbb{R}^n, s \in \mathbb{R}^m} \; f(w) \quad \text{subject to} \quad h^\tau(w) + s = 0 \, , \; s \geq 0$$

with $h^\tau(w) = (g(w) - \tau, \; -g(w) - \tau, \; h(w))$

Condensed KKT system

The augmented KKT system is equivalent to

$$K_\tau \, d_w = -r_1 + (\nabla h^\tau)^\top (\Sigma_s r_4 + r_2)$$

with the *condensed matrix* $K_\tau = W + (\nabla h^\tau)^\top \, \Sigma_s \, (\nabla h^\tau)$.

$\rightarrow$ the condensed KKT system can be solved without numerical pivoting!

Strategy 2: HyKKT (aka Golub & Greif method)

Idea: augmented Lagrangian reformulation

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

$$\begin{bmatrix} K_\gamma & \nabla g^\top \\ \nabla g & 0 \end{bmatrix} \begin{bmatrix} d_w \\ d_y \end{bmatrix} = - \begin{bmatrix} r_1 + \gamma \nabla g^\top r_2 \\ r_2 \end{bmatrix}$$

with $K_\gamma = K + \gamma \nabla g^\top \nabla g$

- For γ large-enough the matrix K_γ is positive definite
  ☞solve the condensed KKT system using 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)

Strategy 3: NCL

Augmented Lagrangian methods have always been popular in optimal control

At iteration k, the algorithm solves:

$$\begin{aligned} \min_{w,r} \; & f(w) - (y_k^e)^\top r + \frac{\rho_k}{2}\|r\|^2 \\ \text{subject to} \quad & g(w) + r_g = 0 \,, \\ & h(w) + r_h \leq 0 \,, \end{aligned} \tag{NCL$_k$}$$

with $r = (r_g, r_h)$ *regularization* variables

- Subproblem (NCL$_k$) is always feasible, solvable by IPM!
- Only the objective changes between k and $k+1$
- Regularization r stabilizes internal IPM iterations

NCL algorithm $\equiv$ Auglag algorithm

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

Algorithm NCL is more robust, but converges in more iterations than IPM.

D. Ma, KL. Judd, D. Orban and MA Saunders. "Stabilized optimization via an NCL algorithm".

Strategy 3: Stabilized KKT system

Rockafellar: Augmented Lagrangian adds a natural dual regularization to the KKT system!

For the two diagonal matrices $C_g := \frac{1}{\rho_k} I$ and $C_h := \frac{1}{\rho_k} I + Z^{-1}S$, let

$$\begin{bmatrix} W & \nabla g^\top & \nabla h^\top \\ \nabla g & -C_g & 0 \\ \nabla h & 0 & -C_h \end{bmatrix} \begin{bmatrix} d_w \\ d_y \\ d_z \end{bmatrix} = - \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} \tag{K_{2r}}$$

Observation

NCL can compute its descent direction by solving K_{2r}.

- For a penalty ρ_k high-enough, there exists a LDL factorization for K_{2r}
- Otherwise, use a *pivot regularization* strategy inside LDL to return the factorization of a *perturbed* matrix
- Recover the original descent direction using iterative refinement

A. Montoison, F. Pacaud, M. Saunders, S. Shin, D. Orban.
"MadNCL: a GPU implementation of algorithm NCL for large-scale, degenerate nonlinear programs." 2025

Outline

GPU-accelerated optimization

New solvers are blossoming everywhere

Most solvers are specialized on convex problems:

- ☞ First-order
  - cuPDLP and all affiliated variants (LP, QP)
  - OSQP (QP)
  - cuHALLaR / cuLORADS (low-rank SDP)
- ☞ Second-order
  - QPTH (dense QP)
  - CuClarabel (conic)

MadSuite: an optimization software suite for GPUs madsuite.org

- **ExaModels** (NLP modeler)
- **MadNLP** (NLP)
- **MadNCL** (degenerate NLP)
- **MadIPM** (LP)

State-of-the-art solvers for optimal control

Support for GPU-accelerated optimization is coming

Observations

State-of-the-art solvers are *all* exploiting the problem's dynamic structure

- ☞ Differential dynamic programming (Altro, ProxDDP)
- ☞ Condensation strategy (HPIPM)
- ☞ Riccati recursion (FATROP)

Further, the ecosystem leverages mature tools:

- BLAS library for embedded system: BLASFEO
- Efficient modeler: Casadi

Question

So, what do we gain by solving optimal control instances on the GPU?

GPU-accelerated sparse automatic differentiation with ExaModels.jl

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

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

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

- 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**

☞**Perfect settings for optimal control**
(repeated pattern: cost and dynamics)

S. Shin, M. Anitescu, F. Pacaud. *Accelerating optimal power flow with GPUs: SIMD abstraction of nonlinear programs and condensed-space interior-point methods*

How fast can we get with ExaModels?

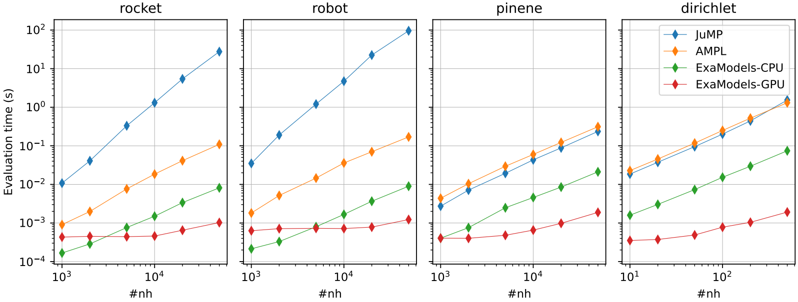


Figure: Time to evaluate the derivatives (Jacobian + Hessian) on various COPS instances with a dynamic structure

F. Pacaud, S. Shin, A. Montoison, M. Schanen, M. Anitescu. "Condensed-space methods for nonlinear programming on GPUs." 2024

Now, combining ExaModels with NVIDIA cuDSS

N.B.: here, we are not exploiting the problem's dynamic structure

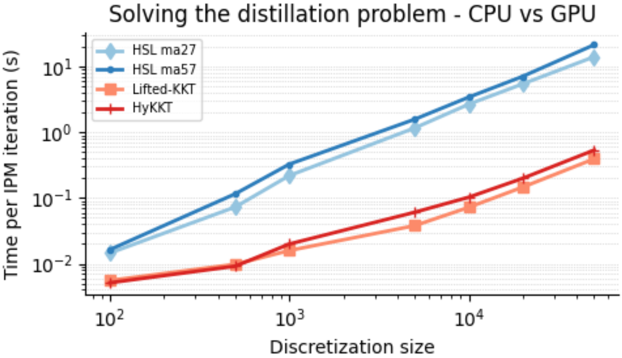


Figure: Solving the infamous distillation column instance

F. Pacaud, S. Shin. "GPU-accelerated dynamic nonlinear optimization with ExaModels and MadNLP." 2024

How expensive should be your GPU?

Observation

☑ No need to buy a professional GPU to get fast performance with ExaModels+cuDSS

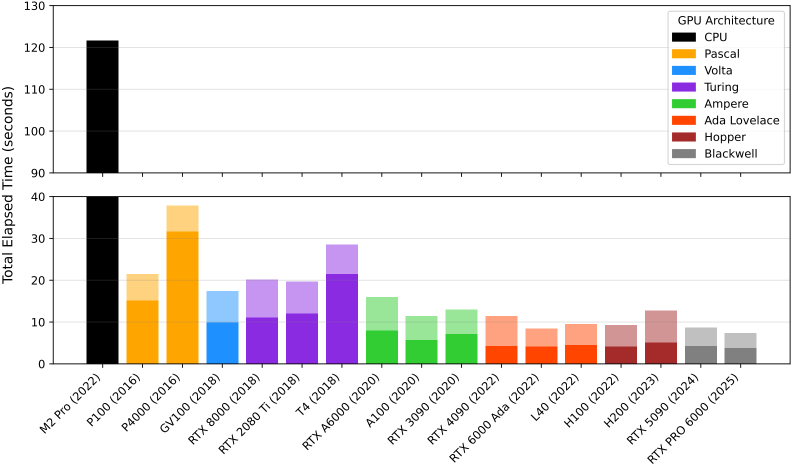


Figure: Time to optimality, here for a large-scale optimal power flow instance.

Image courtesy of Sungho Shin

Next step: Solving the system in parallel

Current research effort

The KKT system is a block tridiagonal system:

$$\begin{bmatrix} A_{1,1} & A_{2,1}^\top & & & \\ A_{2,1}^\top & \ddots & \ddots & & \\ & \ddots & \ddots & A_{N,N-1}^\top \\ & & A_{N,N-1} & A_{N,N} \end{bmatrix} \begin{bmatrix} d_{w1} \\ \vdots \\ \vdots \\ d_{wN} \end{bmatrix} = \begin{bmatrix} r_1 \\ \vdots \\ \vdots \\ r_N \end{bmatrix}$$

Two different path way:

- ☞ Parallel Cycling Reduction (PCR)
- ☞ Partitionned Dynamic Programming

Integration is under way in MadNLP!

D. Jin, A. Montoison, S. Shin. "Harnessing Batched BLAS/LAPACK Kernels on GPUs for Parallel Solutions of Block Tridiagonal Systems." 2025

Conclusion

GPU-accelerated optimal control is currently happening!

ExaModels

Fast evaluations of derivatives in nonlinear models

`frapac.github.io/tutorials/powertech/`

MadNLP

Fast solution of nonlinear programs on the GPU