CuClarabel: A conic interior point solver with GPU acceleration

Yuwen Chen[1,4], Danny Tse[2], Parth Nobel[3], Paul Goulart[4], Stephen Boyd[3]

JuMP-dev 2025

1 Automatic Control Lab, EPFL, Lausanne, CH
2 Department of Computer Science, Stanford University, Stanford, CA
3 Department of Electrical Engineering, Stanford University, Stanford, CA
4 Department of Engineering Science, University of Oxford, Oxford, UK

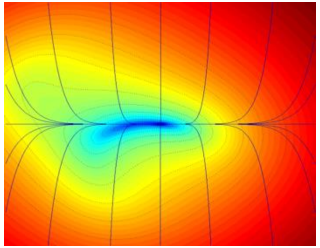
EPFL

Applications of convex optimization

Industries

- control
- quantitative finance
- machine learning
- signal processing
- robotics
- civil
- energy
- ……

Accelerator control

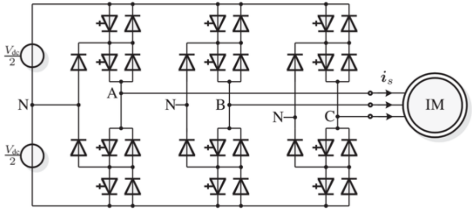


Finite-element model

Robotics

Quantitative finance

Optimal power flow



Power electronics

Outline

- Supported features
- Interior point method with homogeneous embedding
- GPU implementation for Clarabel

Outline

- Supported features
- Interior point method with homogeneous embedding
- GPU implementation for Clarabel

Problem formulation in Clarabel solver

$$\begin{array}{rl} \text{minimize} & \frac{1}{2} x^\top P x + q^\top x \\ \text{subject to} & Ax + s = b \\ & s \in \mathcal{K} \end{array}$$

P is positive semidefinite

$\mathcal{K}$: convex cone

- Zero cone
- Nonnegative cone
- Second-order cone
- Positive semidefinite cone
- Exponential cone
- Power cone

$\mathcal{K} := \{\mathbf{0}\}^m$

$\mathcal{K} := \mathbb{R}_+^m$

$\mathcal{K}_{\text{soc}} = \{(t, x) : t \geq \|x\|, t \geq 0, x \in \mathbb{R}^{m-1}\}$

$\mathcal{K}_{\succeq}^n := \{\text{mat}(x) \in \mathbb{S}^n : \text{mat}(x) \succeq 0\}$

$\mathcal{K}_{\exp} = \left\{(x, y, z) : y > 0, ye^{x/y} \leq z\right\}$

$\mathcal{K}_{\text{pow}(\alpha)} = \{(x, y, z) : x^\alpha y^{1-\alpha} \geq |z|, x, y \geq 0, \alpha \in (0, 1)\}$

Quadratic programming

Zero cone

$\mathcal{K} := \{\mathbf{0}\}^m$

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^\top P x + q^\top x \\ \text{subject to} & Ax = b \end{array}$$

Quadratic programming

Nonnegative cone

$$\mathcal{K} := \mathbb{R}^m_+$$

$$\begin{array}{ll} \text{minimize} & \frac{1}{2} x^\top P x + q^\top x \\ \text{subject to} & Ax \leq b \end{array}$$

Conic programming

Conic form

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^\top Px + q^\top x \\ \text{subject to} & Ax + s = b \\ & s \in \mathcal{K} \end{array}$$

- Second-order cone

$$\mathcal{K}_{\text{soc}} = \left\{(t, x) : t \geq \|x\|, t \geq 0, x \in \mathbb{R}^{m-1}\right\}$$

- Positive semidefinite cone

$$\mathcal{K}^n_{\succeq} := \{\mathsf{mat}(x) \in \mathbb{S}^n : \mathsf{mat}(x) \succeq 0\}$$

- Exponential cone

$$\mathcal{K}_{\text{exp}} = \left\{(x, y, z) : y > 0, ye^{x/y} \leq z\right\}$$

- Power cone

$$\mathcal{K}_{\text{pow}(\alpha)} = \left\{(x, y, z) : x^\alpha y^{1-\alpha} \geq |z|, x, y \geq 0, \alpha \in (0, 1)\right\}$$

Second order cone programming (SOCP)

Second-order cone

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} x^\top P x + q^\top x \\ \text{subject to} \quad & Ax + s = b \\ & s \in \mathcal{K} \end{aligned}$$

$$\mathcal{K}_{\text{soc}} = \left\{ (t, x) : t \geq \|x\|, t \geq 0, x \in \mathbb{R}^{m-1} \right\}$$

2-norm

Semidefinite programming (SDP)

Positive semidefinite cone

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^\top P x + q^\top x \\ \text{subject to} & Ax + s = b \\ & s \in \mathcal{K} \end{array}$$

$$\mathcal{K}^n_{\succeq} := \{\mathsf{mat}(x) \in \mathbb{S}^n : \mathsf{mat}(x) \succeq 0\}$$

$$0 \preceq X \preceq I$$

Eigenvalue problems

Exponential cone programming

Exponential cone

$$\begin{array}{ll} \text{minimize} & q^\top x \\ \text{subject to} & Ax + s = b \\ & s \in \mathcal{K} \end{array}$$

$$\mathcal{K}_{\exp} = \left\{(x, y, z) : y > 0, ye^{x/y} \leq z\right\}$$

$$t \geq e^x \iff (t, 1, x) \in \mathcal{K}_{\exp}$$

Exponentials

$$t \leq -x \log x \iff t \leq x \log(1/x) \iff (1, x, t) \in \mathcal{K}_{\exp}$$

Entropy

Power cone programming

Power cone

$$
\begin{array}{ll}
\text{minimize} & q^\top x \\
\text{subject to} & Ax + s = b \\
& s \in \mathcal{K}
\end{array}
$$

$$\mathcal{K}_{\mathrm{pow}(\alpha)} = \{(x, y, z) : x^\alpha y^{1-\alpha} \geq |z|, x, y \geq 0, \alpha \in (0, 1)\}$$

$p > 1$

$$t \geq |x|^p \iff (t, 1, x) \in \mathcal{K}_{\mathrm{pow}}\left(\frac{1}{p}\right)$$ **Polynomials**

$$t \geq \|x\|_p \iff (r_i, t, x_i) \in \mathcal{K}_{\mathrm{pow}}\left(\frac{1}{p}\right), \sum r_i = t$$ **p-norm**

Outline

- Supported features
- Interior point method with homogeneous embedding
- GPU implementation for Clarabel

Problem formulation

Primal problem

$$
\begin{aligned}
\text{minimize} \quad & \frac{1}{2}x^\top P x + q^\top x \\
\text{subject to} \quad & Ax + s = b \\
& s \in \mathcal{K}
\end{aligned}
$$

Dual problem

$$
\begin{aligned}
\text{maximize} \quad & -\frac{1}{2}x^\top P x - b^\top z \\
\text{subject to} \quad & Px + A^\top z + q = 0 \\
& z \in \mathcal{K}^*
\end{aligned}
$$

Optimality: KKT condition

$$
\begin{aligned}
\text{find} \quad & (x, s, z) \\
\text{subject to} \quad & -Ax + b = s \\
& Px + A^\top z + q = 0 \\
& \langle s, z \rangle = 0 \\
& (s, z) \in (\mathcal{K}, \mathcal{K}^*)
\end{aligned}
$$

Homogeneous self-dual embedding (HSDE) [1]

Primal problem

$$
\begin{aligned}
\text{minimize} \quad & q^\top x \\
\text{subject to} \quad & Ax + s = b \\
& s \in \mathcal{K}
\end{aligned}
$$

Dual problem

$$
\begin{aligned}
\text{maximize} \quad & -b^\top z \\
\text{subject to} \quad & A^\top z + q = 0 \\
& z \in \mathcal{K}^*
\end{aligned}
$$

Add scaling terms τ, κ

$$
\begin{aligned}
\text{find} \quad & (x, s, z, \tau, \kappa) \\
\text{subject to} \quad & -Ax + b\tau = s \\
& Px + A^\top z + q\tau = 0 \\
& q^\top x + b^\top z = -\kappa \\
& (s, z, \tau, \kappa) \in (\mathcal{K}, \mathcal{K}^*, \mathbb{R}_+, \mathbb{R}_+)
\end{aligned}
$$

[1] Yinyu Ye, Michael J. Todd, and Shinji Mizuno. *An O(√nL)-iteration homogeneous and self-dual linear programming algorithm.* Mathematics of Operations Research, 19(1):53–67, 1994.

HSDE for infeasibility detection

$$
\begin{aligned}
\text{find} \quad & (x, s, z, \tau, \kappa) \\
\text{subject to} \quad & -Ax + b\tau = s \\
& A^\top z + q\tau = 0 \\
& q^\top x + b^\top z = -\kappa \\
& (s, z, \tau, \kappa) \in (\mathcal{K}, \mathcal{K}^*, \mathbb{R}_+, \mathbb{R}_+)
\end{aligned}
$$

- Problem is always feasible
- The problem is homogeneous and self-dual

HSDE for infeasibility detection

$$
\begin{aligned}
\text{find}\ & (x, s, z, \tau, \kappa) \\
\text{subject to}\ & -Ax + b\tau = s \\
& A^\top z + q\tau = 0 \\
& q^\top x + b^\top z = -\kappa \\
& (s, z, \tau, \kappa) \in (\mathcal{K}, \mathcal{K}^*, \mathbb{R}_+, \mathbb{R}_+)
\end{aligned}
$$

- Problem is always feasible
- The problem is homogeneous and self-dual
- $\tau^* > 0,\ \kappa^* = 0 \Rightarrow$ optimal solution $(x^*/\tau^*, s^*/\tau^*, z^*/\tau^*)$
- $\tau^* = 0,\ \kappa^* > 0 \Rightarrow$ strongly infeasible certificate $(x^*/\kappa^*, s^*/\kappa^*, z^*/\kappa^*)$

Extension for quadratic cost [2]

Primal problem

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}x^\top P x + q^\top x \\ \text{subject to} \quad & Ax + s = b \\ & s \in \mathcal{K} \end{aligned}$$

Dual problem

$$\begin{aligned} \text{maximize} \quad & -\frac{1}{2}x^\top P x - b^\top z \\ \text{subject to} \quad & Px + A^\top z + q = 0 \\ & z \in \mathcal{K}^* \end{aligned}$$

Add scaling terms τ, κ:

$$\begin{aligned} \text{find} \quad & (x, s, z, \tau, \kappa) \\ \text{subject to} \quad & -Ax + b\tau = s \\ & Px + A^\top z + q\tau = 0 \\ & q^\top x + b^\top z = -\kappa - \frac{1}{\tau}x^\top P x \\ & (s, z, \tau, \kappa) \in (\mathcal{K}, \mathcal{K}^*, \mathbb{R}_+, \mathbb{R}_+) \end{aligned}$$

[2] Erling D. Andersen and Yinyu Ye. *On a homogeneous algorithm for the monotone complementarity problem.* Mathematical Programming, 84(2):375–399, 1999.

Homogeneous embedding for infeasibility detection

$$
\begin{aligned}
\text{find } & (x, s, z, \tau, \kappa) \\
\text{subject to } & -Ax + b\tau = s \\
& Px + A^\top z + q\tau = 0 \\
& q^\top x + b^\top z = -\kappa - \frac{1}{\tau} x^\top P x \\
& (s, z, \tau, \kappa) \in (\mathcal{K}, \mathcal{K}^*, \mathbb{R}_+, \mathbb{R}_+)
\end{aligned}
$$

- Problem is always (asymptotically) feasible
- The problem is homogeneous, not self-dual
- $\tau^* > 0,\ \kappa^* = 0 \Rightarrow$ optimal solution $(x^*/\tau^*, s^*/\tau^*, z^*/\tau^*)$
- $\tau^* = 0,\ \kappa^* > 0 \Rightarrow$ strongly infeasible certificate $(x^*/\kappa^*, s^*/\kappa^*, z^*/\kappa^*)$

EPFL

Practical benefit for quadratic costs

ECOS (quadratic cost to SOC)

$$x^\top P x = \|P^{1/2}x\|^2 \leq 2t + 1$$

$$\Leftrightarrow (t+1, t, P^{1/2}x) \in \mathcal{K}_{\text{soc}}$$

$$\begin{bmatrix} 0 & A^T & [P^{\frac{1}{2}}]^T \\ A & -H & \\ P^{\frac{1}{2}} & & -H_{\mathcal{K}} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta z \\ \Delta z_{soc} \end{bmatrix} = \begin{bmatrix} r_x \\ r_z \\ r_{z_{soc}} \end{bmatrix}$$

Clarabel (quadratic cost)

$$\begin{bmatrix} P & A^T \\ A & -H \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta z \end{bmatrix} = \begin{bmatrix} r_x \\ r_z \end{bmatrix}$$

- Direct LDL solve for step direction calculation (with homogeneous embedding) ↑

Practical benefit for quadratic costs

ECOS (quadratic cost to SOC)

$$x^\top P x = \|P^{1/2}x\|^2 \leq 2t + 1$$

$$\Leftrightarrow (t+1, t, P^{1/2}x) \in \mathcal{K}_{\mathrm{soc}}$$

$$\begin{bmatrix} 0 & A^T & [P^{\frac{1}{2}}]^T \\ A & -H & \\ P^{\frac{1}{2}} & & -H_{\mathcal{K}} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta z \\ \Delta z_{soc} \end{bmatrix} = \begin{bmatrix} r_x \\ r_z \\ r_{z_{soc}} \end{bmatrix}$$

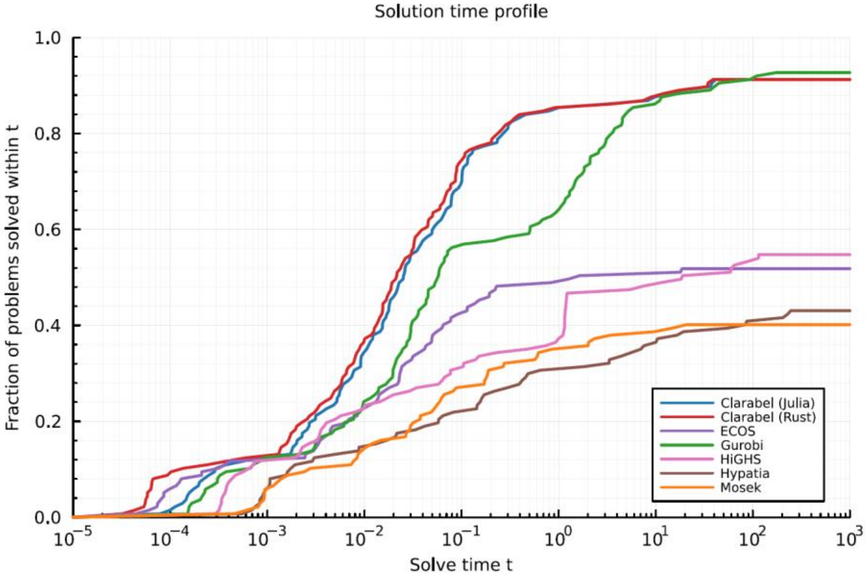
Clarabel (quadratic cost)

$$\begin{bmatrix} P & A^T \\ A & -H \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta z \end{bmatrix} = \begin{bmatrix} r_x \\ r_z \end{bmatrix}$$

- Direct LDL solve for step direction calculation (with homogeneous embedding) ⬆
- Limited to QP, SOCP, SDP [3] ⬇

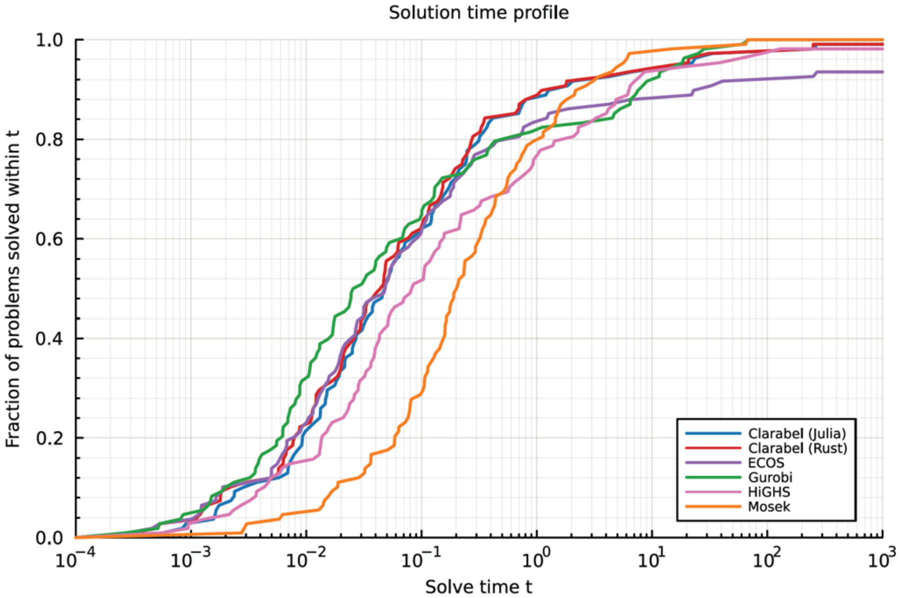
[3] Akiko Yoshise. *Interior point trajectories and a homogeneous model for nonlinear complementarity problems over symmetric cones.* SIAM Journal on Optimization, 17(4):1129–1153, 2007.

Maros-Meszaros: (QP)



		ClarabelRs	Clarabel	ECOS	Gurobi	HiGHS	Hypatia	Mosek
Shifted GM	Full Acc.	1.0	1.02	15.49	1.64	17.92	36.61	32.67
	Low Acc.	1.0	1.1	19.8	2.9	39.99	42.99	4.07
Failure Rate (%)	Full Acc.	8.8	8.8	48.2	7.3	45.3	56.9	59.9
	Low Acc.	2.2	2.9	38.0	3.6	45.3	42.3	10.2

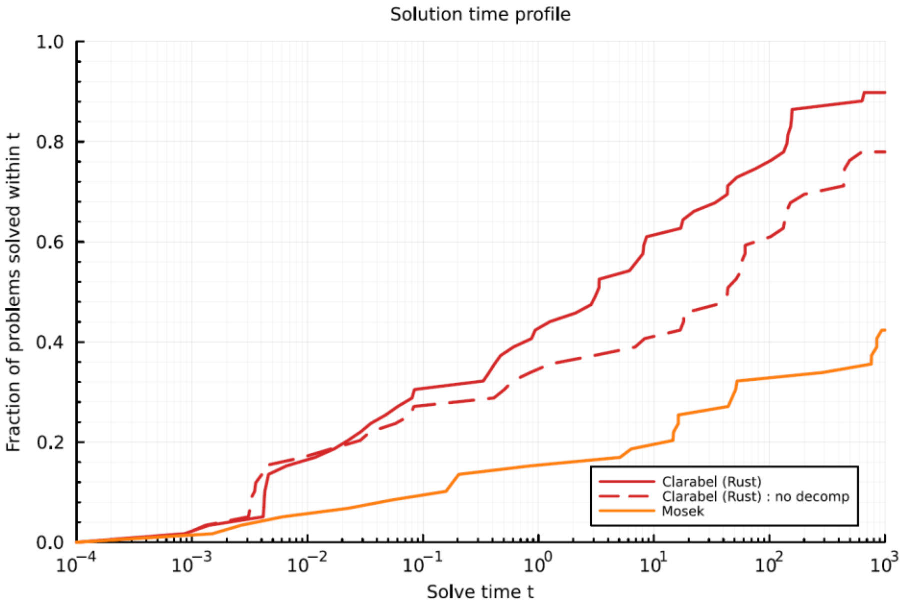
NETLIB (LP)



		ClarabelRs	Clarabel	ECOS	Gurobi	HiGHS	Mosek
Shifted GM	Full Acc.	1.0	1.04	2.13	1.3	1.79	1.2
	Low Acc.	1.0	1.04	1.08	1.3	1.79	1.2
Failure Rate (%)	Full Acc.	0.9	0.9	6.5	0.0	1.9	0.0
	Low Acc.	0.9	0.9	0.9	0.0	1.9	0.0

SDPLIB

- Chordal decomposition with clique merging [4]

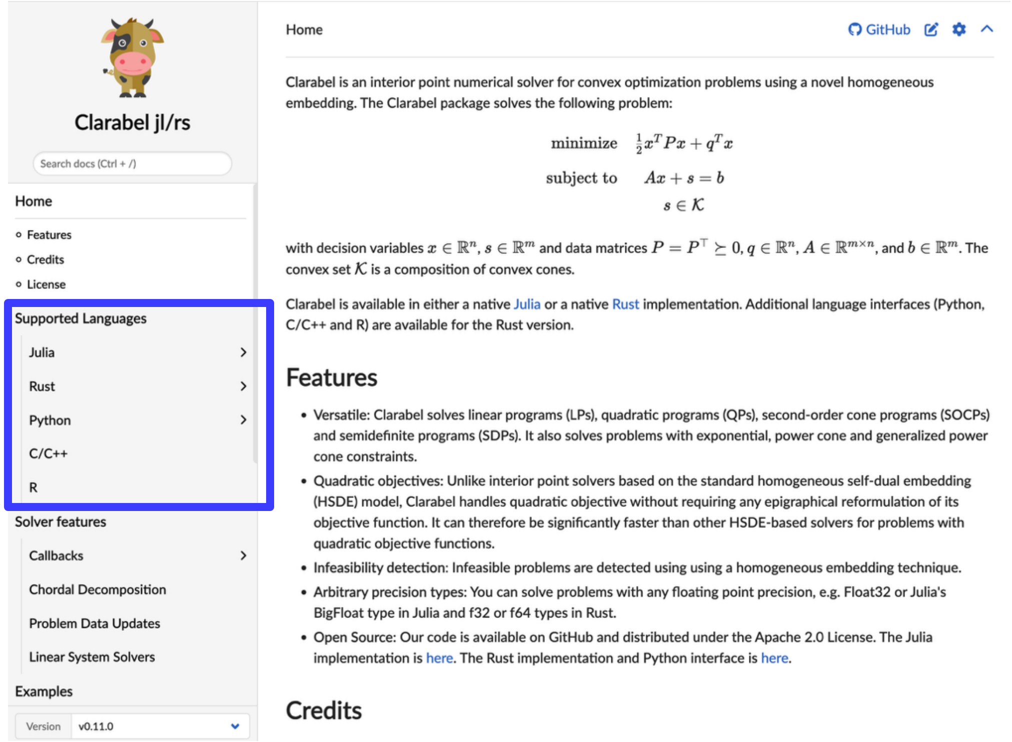


		ClarabelRs	ClarabelRs (no decomp)	Mosek
Shifted GM	Full Acc.	1.0	2.53	9.95
	Low Acc.	1.0	2.17	1.26
Failure Rate (%)	Full Acc.	10.2	22.0	57.6
	Low Acc.	1.7	10.2	1.7

[4] Michael Garstka, Mark Cannon, and Paul Goulart. *COSMO: A conic operator splitting method for convex conic problems.* Journal of Optimization Theory and Applications, 190(3):779–810, 2021.

Start with Clarabel

- Julia&Rust 0.11.0
- Python, C/C++, R wrappers
- Arbitrary precision (Julia)
- Default from cvxpy 1.5



Clarabel jl/rs

Search docs (Ctrl + /)

Home

- Features
- Credits
- License

Supported Languages

- Julia
- Rust
- Python
- C/C++
- R

Solver features

- Callbacks
- Chordal Decomposition
- Problem Data Updates
- Linear System Solvers

Examples

Version v0.11.0

Home

GitHub

Clarabel is an interior point numerical solver for convex optimization problems using a novel homogeneous embedding. The Clarabel package solves the following problem:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & Ax + s = b \\ & s \in \mathcal{K} \end{array}$$

with decision variables $x \in \mathbb{R}^n$, $s \in \mathbb{R}^m$ and data matrices $P = P^\top \succeq 0$, $q \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $b \in \mathbb{R}^m$. The convex set $\mathcal{K}$ is a composition of convex cones.

Clarabel is available in either a native Julia or a native Rust implementation. Additional language interfaces (Python, C/C++ and R) are available for the Rust version.

Features

- Versatile: Clarabel solves linear programs (LPs), quadratic programs (QPs), second-order cone programs (SOCPs) and semidefinite programs (SDPs). It also solves problems with exponential, power cone and generalized power cone constraints.
- Quadratic objectives: Unlike interior point solvers based on the standard homogeneous self-dual embedding (HSDE) model, Clarabel handles quadratic objective without requiring any epigraphical reformulation of its objective function. It can therefore be significantly faster than other HSDE-based solvers for problems with quadratic objective functions.
- Infeasibility detection: Infeasible problems are detected using using a homogeneous embedding technique.
- Arbitrary precision types: You can solve problems with any floating point precision, e.g. Float32 or Julia's BigFloat type in Julia and f32 or f64 types in Rust.
- Open Source: Our code is available on GitHub and distributed under the Apache 2.0 License. The Julia implementation is here. The Rust implementation and Python interface is here.

Credits

- Industrial use:

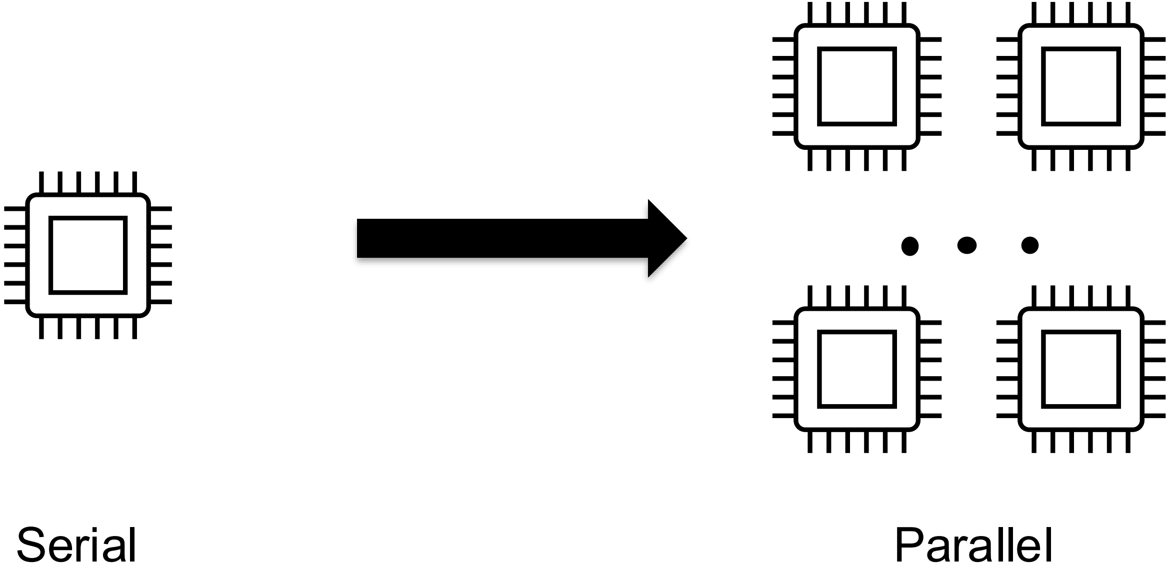
  control, finance, energy, civil……

https://oxfordcontrol.github.io/ClarabelDocs/stable/

Outline

- Supported features
- Interior point method with homogeneous embedding
- GPU implementation for Clarabel

When the dimensionality scales up…



Classes of operations

Basic parallel operations

- Add, multiplication…
- Sum, min/max…

Cone-wise kernel operations

- Update scaling matrices
- Step size computation
- Compute r.h.s. of the KKT system
- …

$$\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_p$$

Operations related to linear systems (cuDSS [5,6])

- KKT matrix factorization
- Linear system solves

[5] https://docs.nvidia.com/cuda/cudss/index.html
[6] https://github.com/exanauts/CUDSS.jl

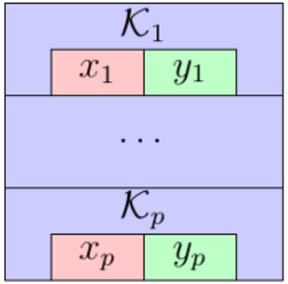
Data structure

Basic parallel operations

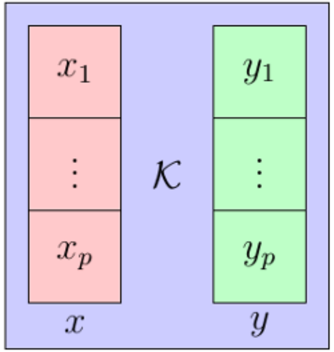
- Add, multiplication…
- Sum, min/max…

Cone-wise kernel operations

- Update Hessian
- Step size computation
- Compute r.h.s. of the KKT system
- …



AoS



SoA

Operations related to linear systems (cuDSS)

- KKT matrix factorization
- Linear system solves

Stream-wise parallelism

Algorithm 1 Algorithmic sketch for the scaling matrix H^k update

Require: Current iterate v^k, streams $st_{soc}, st_{exp}, st_{pow}, st_{sdp}$.

```
function _update_H(v^k)
    Update H^k_zero        // Zero cone (t = 1)
    Update H^k_nn          // Nonnegative cone (t = 2)

    // Second-order cones
    H^k_soc = _kernel_soc_update_H<st_soc>() // t = 3 to i

    // Exponential cones
    H^k_exp = _kernel_exp_update_H<st_exp>() // t = i + 1 to j

    // Power cones
    H^k_pow = _kernel_pow_update_H<st_pow>() // t = j + 1 to l

    // Positive semidefinite cones
    H^k_sdp = _kernel_sdp_update_H<st_sdp>() // t = l + 1 to p

    // Synchronize scaling update
    CuDeviceSynchronize()

    return H^k        // Output the scaling matrix H^k
end
```

$$\begin{bmatrix} P & A^T \\ A & -H \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta z \end{bmatrix} = \begin{bmatrix} b_x \\ b_z \end{bmatrix}.$$

$$H = \begin{bmatrix} H_1 & & \\ & \ddots & \\ & & H_p \end{bmatrix}$$

Dynamic parallelism for SOCP

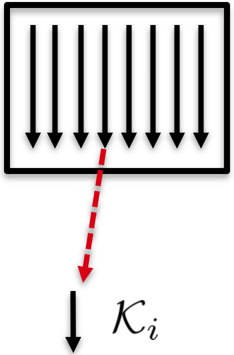
Barrier: $F(t_i, x_i) = -\ln\left(t_i^2 - \|x_i\|_2^2\right), \qquad (t_i, x_i) \in \mathbb{R}^{n+1}$

Uncertainty in dimension n

Dynamic parallelism for SOCP

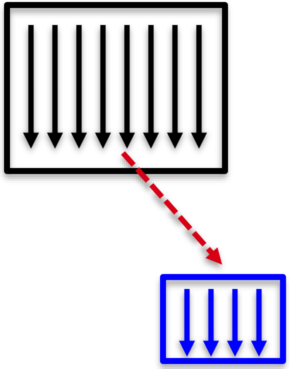
Barrier: $F(t_i, x_i) = -\ln\left(t_i^2 - \|x_i\|_2^2\right), \qquad (t_i, x_i) \in \mathbb{R}^{n+1}$

Simple parallelism (1-layer)



Thread i: $\mathcal{K}_i$

Dynamic parallelism (2-layers)



$\mathcal{K}_i$

Reduction for $\|x_i\|^2$

Batched SDP

Cone related Matrix factorizations

- Cholesky
- SVD

Supported batched operations

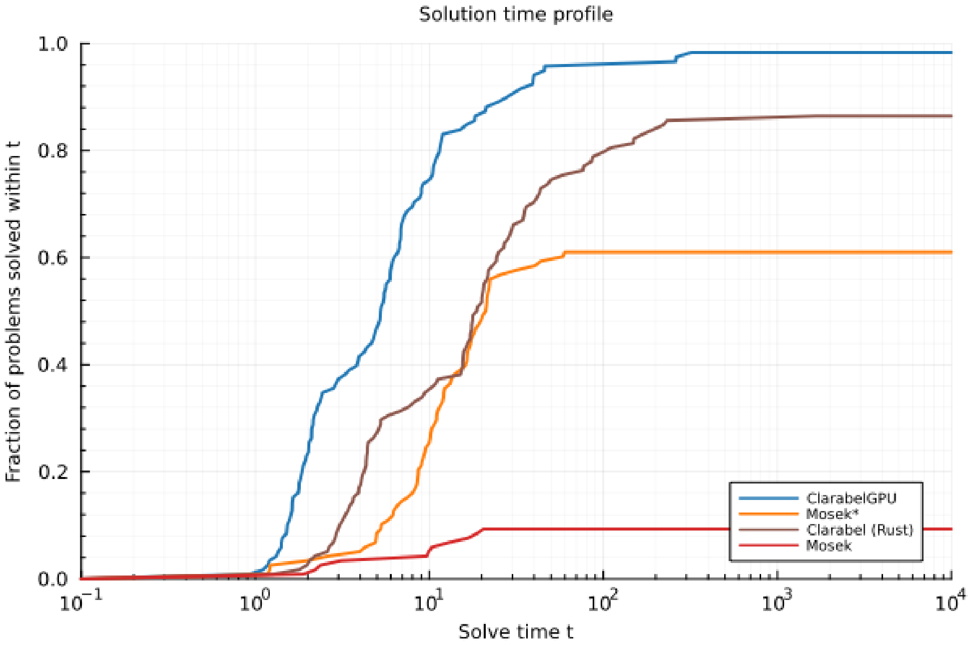
- cuSolver (limited to the same dimensionality <=32)

Hardwares

- CPU
  - Intel (R) Xeon (R) w9-3475X
  - 72 cores, 4.8GHz, 256GB DDR5
- GPU
  - NVIDIA GeForce RTX 4090
  - 1.29 TFLOPS (FP64), 24GB GDDR6X

SOCP relaxation for OPF

- Pglib-opf



Multi-stages portfolio optimization

Horizon	iterations			total time (s)		
	ClarabelGPU	Mosek*	Gurobi	ClarabelGPU	Mosek*	Gurobi
5	15	**12**	21	**0.803**	1.33	1.73
10	16	**9**	21	**1.78**	3.16	6.43
15	16	**11**	23	**2.86**	6.75	15.8
20	17	**12**	22	**4.49**	11.3	26.2
25	17	**9**	22	**5.73**	14.9	37.4
30	17	**9**	22	**7.18**	19.6	73.8

Multi-stages portfolio optimization

Horizon	iterations			total time (s)		
	ClarabelGPU	Mosek*	Gurobi	ClarabelGPU	Mosek*	Gurobi
5	15	**12**	21	**0.803**	1.33	1.73
10	16	**9**	21	**1.78**	3.16	6.43
15	16	**11**	23	**2.86**	6.75	15.8
20	17	**12**	22	**4.49**	11.3	26.2
25	17	**9**	22	**5.73**	14.9	37.4
30	17	**9**	22	**7.18**	19.6	73.8

T = 30: nnz(P)=225000, nnz(A) = 5398830

Setup time: 4.70s, ≈65%

Gain more for parametric programming!

Multi-stages portfolio optimization (SOCP)

	iterations				total time (s)			
Horizon	ClarabelGPU	Mosek*	Clarabel	Mosek	ClarabelGPU	Mosek*	Clarabel	Mosek
5	21	15	21	**14**	**0.882**	1.24	12.1	1.65
10	22	**15**	22	**15**	**1.79**	3.87	53.7	4.63
15	22	14	22	**12**	**2.8**	7.49	99.5	7.42
20	22	15	22	**11**	**3.91**	12.5	167	13.2
25	24	18	24	**14**	**9.82**	20.2	245	19.8
30	23	**12**	23	13	**11.3**	22.1	312	28

Exponential programming (entropy optimization)

	iterations				total time (s)			
Cone	ClarabelGPU	Mosek*	Clarabel	Mosek	ClarabelGPU	Mosek*	Clarabel	Mosek
2000	20	15	20	**10**	**0.476**	0.794	2.84	0.609
4000	20	16	20	**9**	**1.4**	3.02	14	2.28
6000	21	16	21	**9**	**3.16**	8.85	90.5	5.89
8000	21	16	21	**10**	**5.98**	17.7	104	12.1
10000	21	16	23	**10**	**19.2**	31.6	239	36.5

SDP-constrained FEM

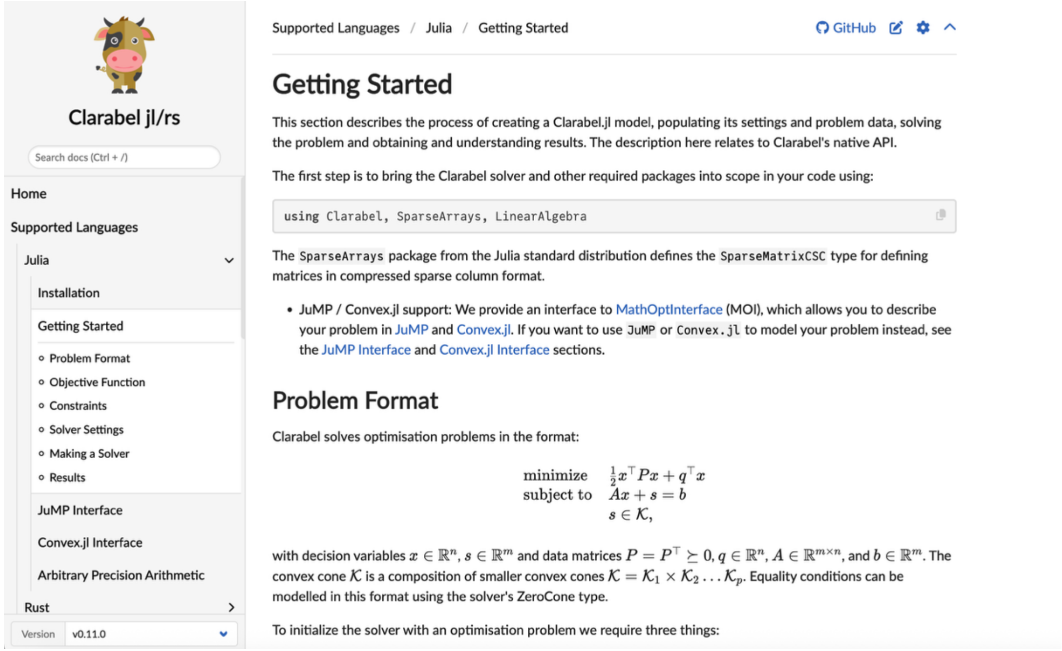
SDP cones $\in \mathbb{S}_+^3$

Problem	iterations			total time (s)		
	ClarabelGPU	Mosek*	Mosek	ClarabelGPU	Mosek*	Mosek
GEO3D_FELA_SDP_1048	23	24	**22**	**2.18**	6.38	2.66
GEO3D_OBEFM_SDP_1048	19	**13**	15	**2.47**	5.73	3.59
GEO3D_FELA_SDP_4444	**30**	31	38	**10.6**	38.8	18.6
GEO3D_OBEFM_SDP_4444	20	**13**	-	**10.8**	33.2	-
GEO3D_FELA_SDP_9263	-	**33**	34	-	55.6	**40.2**
GEO3D_OBEFM_SDP_9263	21	**14**	-	**24.4**	34.5	-

SDP_9263 classes: 74008 sdp cones

How to use it in Julia

- Julia
  - Default input
  - JuMP


Clarabel jl/rs

Search docs (Ctrl + /)

Home

Supported Languages

Julia

Installation

Getting Started

- Problem Format
- Objective Function
- Constraints
- Solver Settings
- Making a Solver
- Results

JuMP Interface

Convex.jl Interface

Arbitrary Precision Arithmetic

Rust

Version v0.11.0

Supported Languages / Julia / Getting Started GitHub

Getting Started

This section describes the process of creating a Clarabel.jl model, populating its settings and problem data, solving the problem and obtaining and understanding results. The description here relates to Clarabel's native API.

The first step is to bring the Clarabel solver and other required packages into scope in your code using:

```
using Clarabel, SparseArrays, LinearAlgebra
```

The `SparseArrays` package from the Julia standard distribution defines the `SparseMatrixCSC` type for defining matrices in compressed sparse column format.

- JuMP / Convex.jl support: We provide an interface to MathOptInterface (MOI), which allows you to describe your problem in JuMP and Convex.jl. If you want to use `JuMP` or `Convex.jl` to model your problem instead, see the JuMP Interface and Convex.jl Interface sections.

Problem Format

Clarabel solves optimisation problems in the format:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^\top Px + q^\top x \\ \text{subject to} & Ax + s = b \\ & s \in \mathcal{K}, \end{array}$$

with decision variables $x \in \mathbb{R}^n$, $s \in \mathbb{R}^m$ and data matrices $P = P^\top \succeq 0$, $q \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, and $b \in \mathbb{R}^m$. The convex cone $\mathcal{K}$ is a composition of smaller convex cones $\mathcal{K} = \mathcal{K}_1 \times \mathcal{K}_2 \ldots \mathcal{K}_p$. Equality conditions can be modelled in this format using the solver's ZeroCone type.

To initialize the solver with an optimisation problem we require three things:

set_optimizer_attribute(model, "direct_solve_method", :cudss)

Python use

- Python
  - juliacall

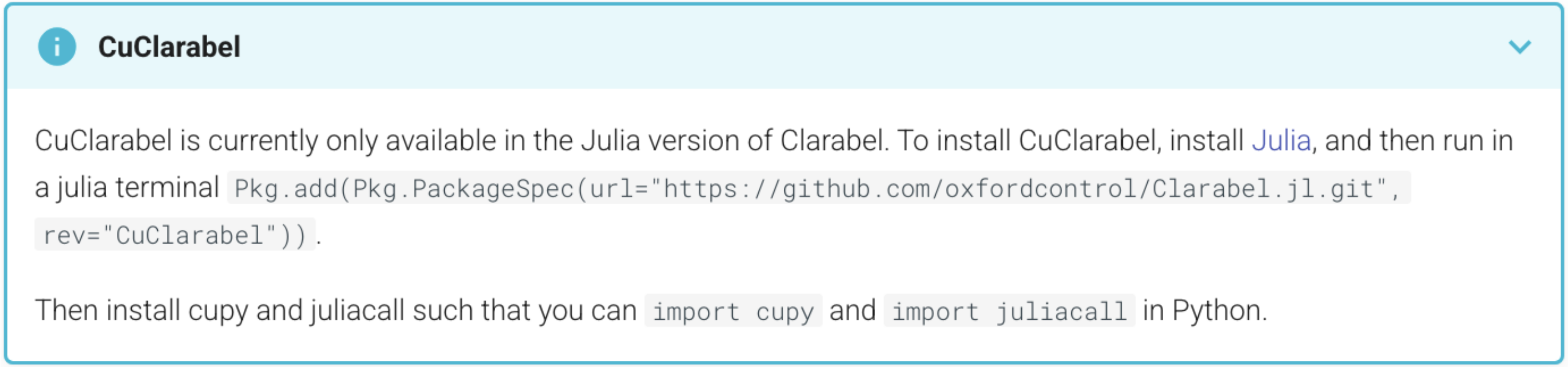
```
from juliacall import Main as jl
import numpy as np
import cupy as cp
from cupyx.scipy.sparse import csr_matrix
# Load Clarabel in Julia
jl.seval('using Clarabel, LinearAlgebra, SparseArrays')
jl.seval('using CUDA, CUDA.CUSPARSE')
```

```
# Update b vector
bpy = cp.array([2.0, 1.0, 1.0, 1.0], dtype=cp.float64)
bjl = jl.Clarabel.cupy_to_cuvector(jl.Float64, int(bpy.data.ptr), bpy.size)

# "_b" is the replacement of "!" in julia function
jl.Clarabel.update_b_b(jl.solver,bjl)          #Clarabel.update_b!()
```

Python use

- Python
  - juliacall
  - cvxpy (CuClarabel)



What’s next

- Warm-starting
- Efficient data loading
- More powerful SDP support on GPU

Clarabel [6]

CuClarabel [7]

Thank you

[6] Paul Goulart and Yuwen Chen. *Clarabel: An interior-point solver for conic programs with quadratic objectives.* arXiv, 2024

[7] Yuwen Chen and Danny Tse and Parth Nobel and Paul Goulart and Stephen Boyd. *CuClarabel: GPU Acceleration for a Conic Optimization Solver.* arXiv, 2024