Generation of Custom Solvers in Rust for Convex Optimization

Hao Zhu

Department of Computer Science
University of Freiburg

July 24, 2026

Parameterized convex optimization

$$\begin{array}{lll} \text{minimize} & f_0(x;\omega) & \\ \text{subject to} & f_i(x;\omega) \leq 0, & i = 1,\ldots,m \\ & h_i(x;\omega) = 0, & i = 1,\ldots,p \end{array} \tag{1}$$

- $x \in \mathbf{R}^n$ is the *variable*
- $\omega \in \mathbf{R}^k$ is the *parameter* or *data*
- for each fixed ω,
  - $f_0,\ldots,f_m$: $\mathbf{R}^n \times \mathbf{R}^k \to \mathbf{R}$ are convex
  - $h_1,\ldots,h_p$: $\mathbf{R}^n \times \mathbf{R}^k \to \mathbf{R}$ are affine

  so each *instance* of (1) is a convex optimization problem
- for all $\omega \in \mathbf{R}^k$, (1) forms a convex *problem family*

Embedded optimization

- solving convex problems is a part of larger systems, $e.g.$, MPC
- instances in a problem family are solved repeatedly with different parameters
- additional requirements:
  - small memory footprint, $e.g.$, no heavy interpreter like Python or MATLAB
  - fast execution time, $e.g.$, in miliseconds
  - deterministic behavior, $e.g.$, no random seeds involved
- applications in control, finance, signal processing, . . .

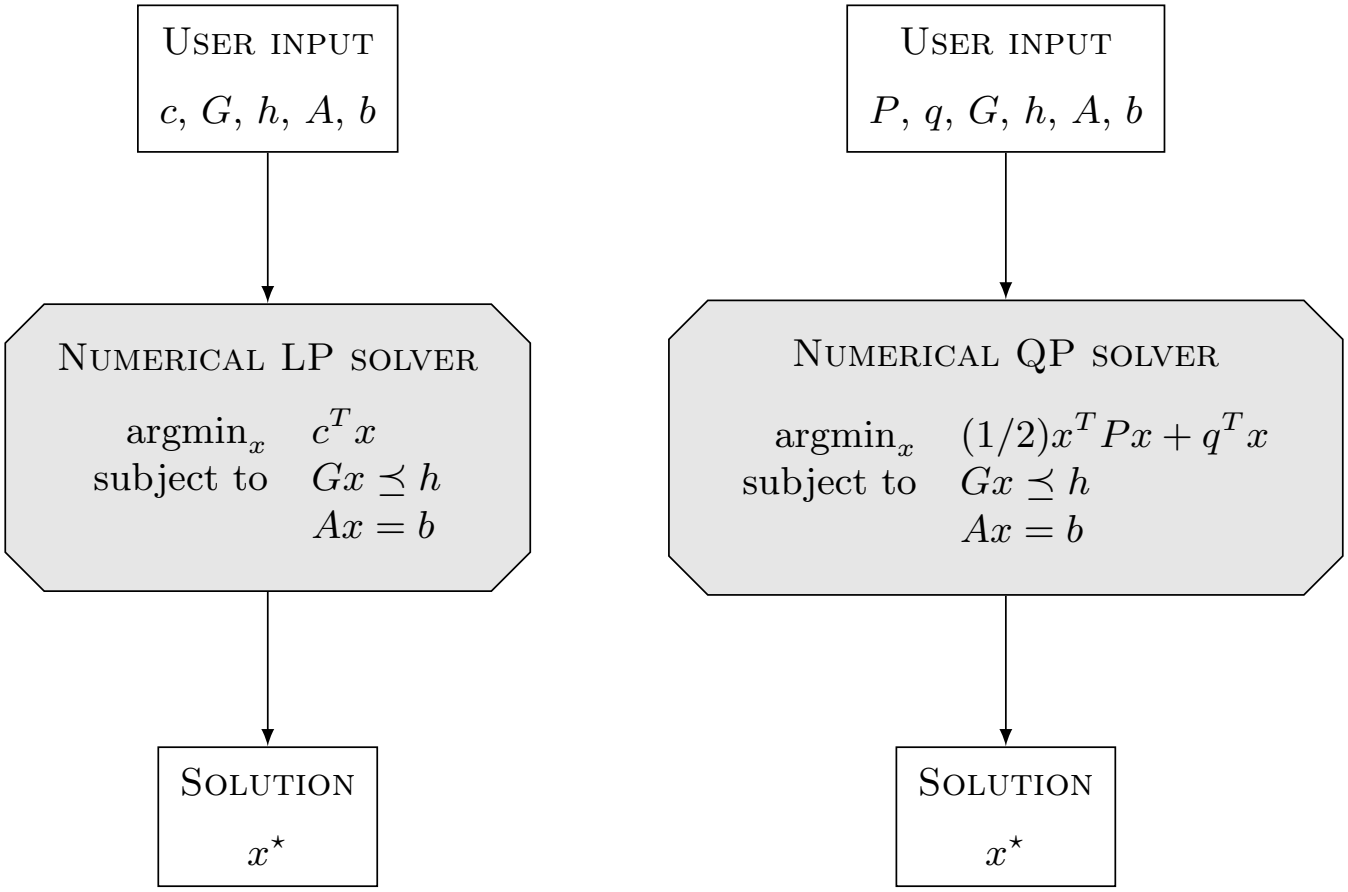
Numerical solver

- targets a specific canonical problem class, $e.g.$, LP, QP, SOCP, SDP
- implements a specific numerical algorithm, $e.g.$, interior-point method, subgradient method
- hard-coded and highly optimized, so expected to be very fast

- only accepts canonical problem data (derived from ω) as input and returns the solution
- a user has to manually convert the original problem into the accepted canonical form and call the solver

requires heavy convex analysis and optimization knowledge and experience

example: two numerical solvers for LP and QP problems

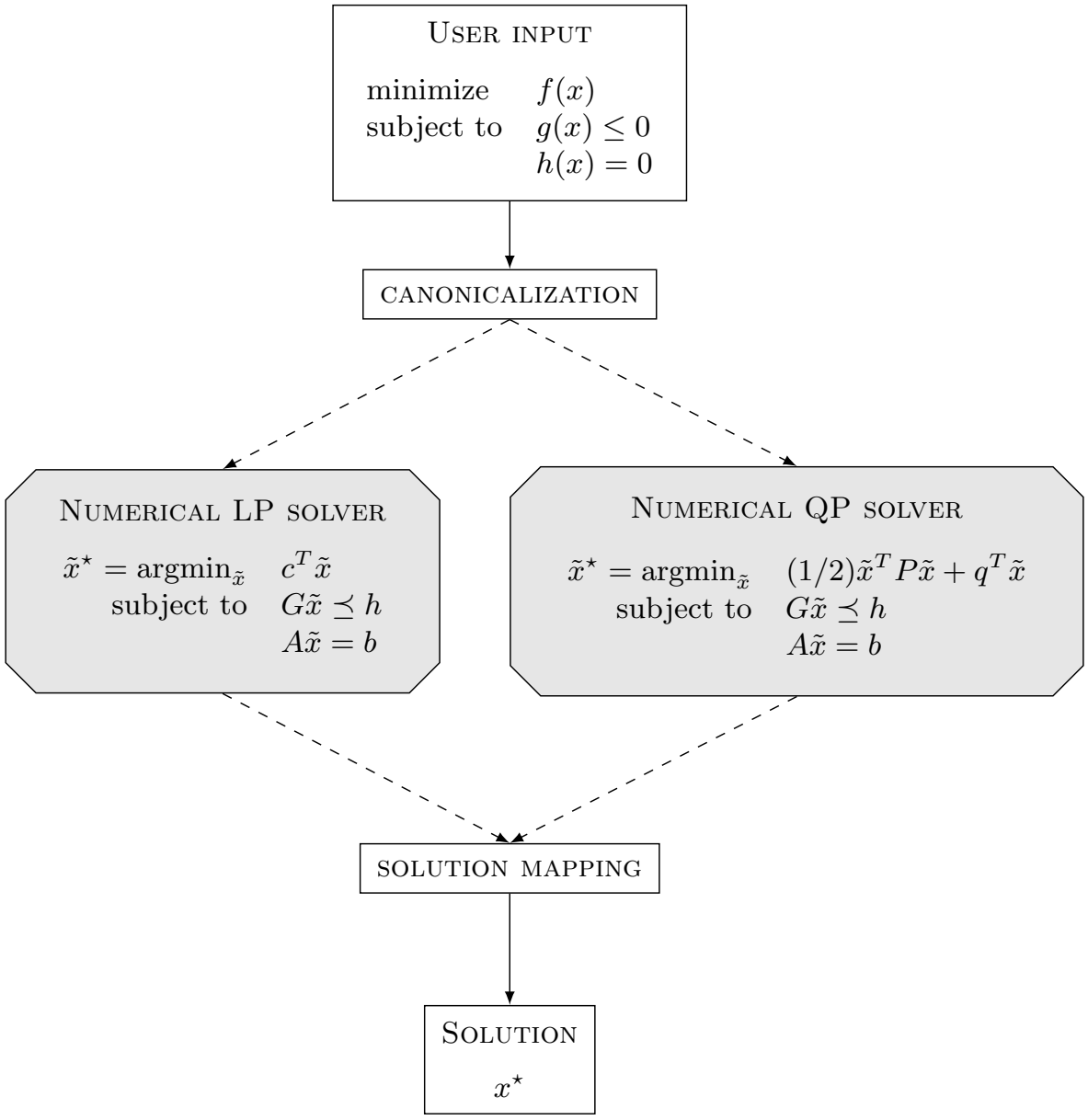


Parser solver

- accepts a symbolic problem description in a high-level modeling language, $e.g.$, CVXPY
- automatically converts the original problem into a canonical form and calls a compatible numerical solver
- avoids manual problem canonicalization

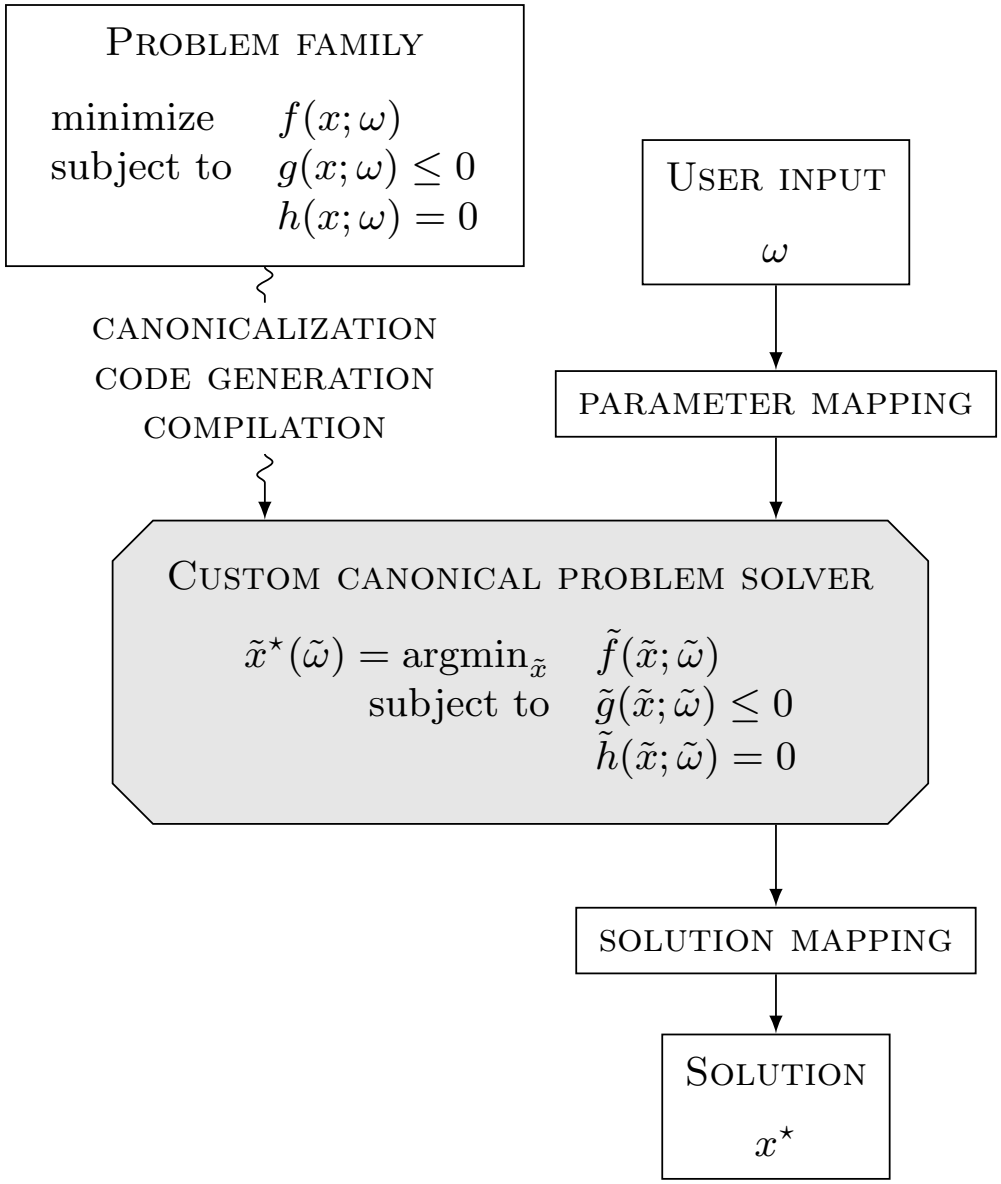
- requires a heavy interpreter environment, $e.g.$, Python or MATLAB
- canonicalization is done for each problem instance at run time

can be slow for embedded applications


User input
minimize $f(x)$
subject to $g(x) \leq 0$
$h(x) = 0$
canonicalization
Numerical LP solver
$\tilde{x}^\star = \operatorname{argmin}_{\tilde{x}} \; c^T\tilde{x}$
subject to $G\tilde{x} \preceq h$
$A\tilde{x} = b$
Numerical QP solver
$\tilde{x}^\star = \operatorname{argmin}_{\tilde{x}} \; (1/2)\tilde{x}^T P\tilde{x} + q^T\tilde{x}$
subject to $G\tilde{x} \preceq h$
$A\tilde{x} = b$
solution mapping
Solution
$x^\star$

Code generator

- combines the advantages of numerical and parser solvers
- takes a symbolic modeling of a problem *family* as input
- canonicalize the problem family *once* at code generation time
- generates a custom solver that accepts the parameter ω as input and returns the solution
- generated solver implemented in a compiled language, $e.g.$, C or Rust, so can be compiled into executable binaries


Problem family
minimize $f(x;\omega)$
subject to $g(x;\omega) \leq 0$
$h(x;\omega) = 0$
User input
ω
canonicalization
code generation
compilation
parameter mapping
Custom canonical problem solver
$\tilde{x}^\star(\tilde{\omega}) = \text{argmin}_{\tilde{x}}$ $\tilde{f}(\tilde{x};\tilde{\omega})$
subject to $\tilde{g}(\tilde{x};\tilde{\omega}) \leq 0$
$\tilde{h}(\tilde{x};\tilde{\omega}) = 0$
solution mapping
Solution
$x^\star$

examples:

- CVXGEN [MB12], CVXPYgen [SBD+22], etc.
  - targets C language
  - supports LP, QP, up to SOCP
  - used in onboard flight code generation for precision landing of space rockets by SpaceX [Bla16].

- acados [VFK+22], etc.
  - supports specific problem classes, $e.g.$, linear and nonlinear MPC

CVXGenRust

- a code generator that targets Rust language
  - fast (comparable to C)
  - memory safe
  - modern dependency management and build system
- supports general conic problems, including
  - LP and QP: control and finance
  - SOCP and SDP: signal processing and machine learning
  - exponential cone: statistics and information theory, $e.g.$, problems involving relative entropy, log-sum-exp, KL-divergence, etc.
  - power cone: problems involving, $e.g.$, geometric mean and ℓ_p-norms

Canonical quadratic cone programs

CVXGenRust canonicalizes (1) into

$$\begin{array}{ll} \text{minimize} & (1/2)\tilde{x}^T P\tilde{x} + q^T\tilde{x} \\ \text{subject to} & A\tilde{x} + s = b \\ & s \in \mathcal{K}, \end{array}$$

- $\tilde{x}$ and s: canonical variable (mapped from x)
- P, q and A, b: canonical problem data (mapped from ω)
- $\mathcal{K}$: product cone

- backend numerical solver: Clarabel [GC24]
  - interior-point solver: very fast for embedded problems
  - native Rust implementation
  - fully open-source

  customized for a problem *family* and generate reusable implementation

Supported cone types

zero cone

$$\{0\}^n = \{x \in \mathbf{R}^n \mid x = 0\}$$

for representing equality constraints

nonnegative orthant

$$\begin{aligned} \mathbf{R}^n_+ &= \{x \in \mathbf{R}^n \mid x \succeq 0\} \\ &= \{x \in \mathbf{R}^n \mid x_i \geq 0,\ i = 1, \ldots, n\} \end{aligned}$$

for representing inequality constraints

second-order cone

$$\mathcal{Q}^{n+1} = \{(x, t) \in \mathbf{R}^{n+1} \mid \|x\|_2 \leq t\}$$

appears when (1) involves quadratic or ℓ_2-norm functions

SOCP example: norm bounded least squares (in epigraph form)

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \|Ax-b\|_2 \leq t \\ & \|x\|_2 \leq \rho \end{array}$$

with variables $x \in \mathbf{R}^n$ and $t \in \mathbf{R}$

semidefinite cone

$$\begin{aligned} \mathbf{S}_+^n &= \{X \in \mathbf{S}^n \mid X \succeq 0\} \\ &= \{X \in \mathbf{S}^n \mid v^T X v \geq 0 \text{ for all } v \in \mathbf{R}^n\} \end{aligned}$$

is the standard cone associated with SDPs
example: covariance estimation with variable $X \in \mathbf{S}^n$

$$\begin{array}{ll} \text{minimize} & \|X-S\|_F^2 \\ \text{subject to} & X \succeq 0, \quad \mathbf{diag}(X) = \mathbf{diag}(S) \end{array}$$

exponential cone

$$\begin{aligned}\mathcal{K}^{\exp} &= \mathbf{cl}\{(x,y,z) \in \mathbf{R}^3 \mid y > 0,\ ye^{x/y} \leq z\} \\ &= \{(x,y,z) \in \mathbf{R}^3 \mid y > 0,\ ye^{x/y} \leq z\} \\ &\qquad \cup \{(x,0,z) \in \mathbf{R}^3 \mid x \leq 0,\ z \geq 0\}\end{aligned}$$

example: entropy maximization with variable $x \in \mathbf{R}^n$

$$\begin{array}{ll}\text{minimize} & \sum_{i=1}^n x_i \log x_i \\ \text{subject to} & Cx = d, \quad x \succeq 0\end{array}$$

in canonical form

$$\begin{array}{ll}\text{minimize} & \mathbf{1}^T t \\ \text{subject to} & Cx = d \\ & (-t_i, x_i, 1) \in \mathcal{K}^{\exp}, \quad i = 1, \ldots, n\end{array}$$

with variables $x, t \in \mathbf{R}^n$

power cone

$$\mathcal{K}^{\mathrm{pow}} = \{(x, y, z) \in \mathbf{R}^3 \mid x^\alpha y^{1-\alpha} \geq |z|,\ x \geq 0,\ y \geq 0\}$$

where $\alpha \in (0, 1)$ is a fixed parameter; appears when (1) involves power functions, $e.g.$, geometric mean, ℓ_p-norms, etc.

Affine-solve-affine workflow

data-free canonicalization: reusable canonical solver requires

- *structure* of canonical problem data P, q, A, b and cone $\mathcal{K}$ to be independent of specific *values* of ω

for DPP [AAB$^+$19] compatible convex problems (1), we have

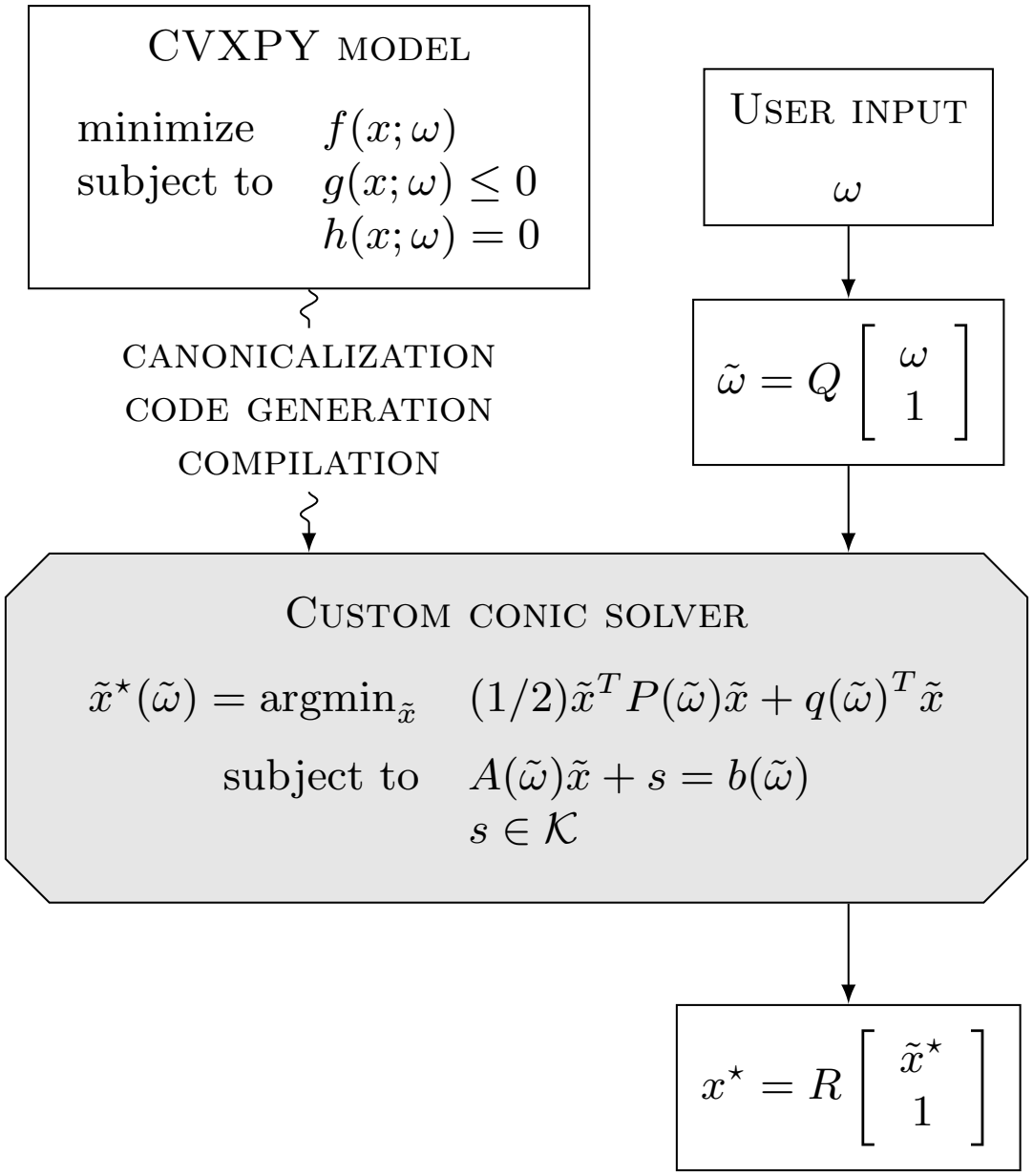
$$\tilde{\omega} = Q \begin{bmatrix} \omega \\ 1 \end{bmatrix}, \qquad x^\star = R \begin{bmatrix} \tilde{x}^\star \\ 1 \end{bmatrix}$$

where

- $\tilde{\omega}$: stacked canonical problem parameters
- $\tilde{x}$: stacked canonical variable

$i.e.$, maps between original and canonical variables and parameters are both affine, independent of specific values of ω

$\implies$ once ω changes, only $\tilde{\omega}$ in the custom solver should be updated


CVXPY model
minimize $f(x;\omega)$
subject to $g(x;\omega) \leq 0$
$h(x;\omega) = 0$
User input
ω
canonicalization
code generation
compilation
$\tilde{\omega} = Q \begin{bmatrix} \omega \\ 1 \end{bmatrix}$
Custom conic solver
$\tilde{x}^\star(\tilde{\omega}) = \text{argmin}_{\tilde{x}} \quad (1/2)\tilde{x}^T P(\tilde{\omega})\tilde{x} + q(\tilde{\omega})^T\tilde{x}$
subject to $A(\tilde{\omega})\tilde{x} + s = b(\tilde{\omega})$
$s \in \mathcal{K}$
$x^\star = R \begin{bmatrix} \tilde{x}^\star \\ 1 \end{bmatrix}$

example: norm approximation

$$\begin{array}{ll} \text{minimize} & \|Cx - d\|_2 + \lambda\|x\|_2 \\ \text{subject to} & x \succeq 0 \end{array}$$

with variable $x \in \mathbf{R}^n$ and parameters $C \in \mathbf{R}^{m\times n}$, $d \in \mathbf{R}^n$, $\lambda > 0$

equivalent problem in epigraph form

$$\begin{array}{ll} \text{minimize} & t_1 + \lambda t_2 \\ \text{subject to} & \|Cx - d\|_2 \leq t_1, \\ & \|x\|_2 \leq t_2, \\ & x \succeq 0 \end{array}$$

with variables $x \in \mathbf{R}^n$, $t_1 \in \mathbf{R}$, $t_2 \in \mathbf{R}$

canonical quadratic cone problem formulation

$$\begin{array}{ll} \text{minimize} & \tilde{x}^T P \tilde{x} + q^T \tilde{x} \\ \text{subject to} & A\tilde{x} + s = b \\ & s \in \mathcal{Q}^{m+1} \times \mathcal{Q}^{n+1} \times \mathbf{R}^n_+ \end{array}$$

where

$$\tilde{x} = \begin{bmatrix} x \\ t_1 \\ t_2 \end{bmatrix}, \qquad s = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

and

$$P = 0, \quad q = \begin{bmatrix} 0 \\ 1 \\ \lambda \end{bmatrix}, \qquad A = \left[\begin{array}{ccc} -C & 0 & 0 \\ 0 & -1 & 0 \\ \hline -I & 0 & 0 \\ 0 & 0 & -1 \\ \hline -I & 0 & 0 \end{array}\right], \qquad b = \left[\begin{array}{c} -d \\ 0 \\ \hline 0 \\ 0 \\ \hline 0 \end{array}\right]$$

Code generation

consider the nonnegative least squares problem

$$\begin{array}{ll} \text{minimize} & \|Ax - b\|_2^2 \\ \text{subject to} & x \succeq 0 \end{array}$$

with variables $x \in \mathbf{R}^n$ and parameters $A \in \mathbf{R}^{m \times n}$, $b \in \mathbf{R}^m$

specifying the problem family

```
import cvxpy as cp

x = cp.Variable(n, name="x")
A = cp.Parameter((m, n), name="A")
b = cp.Parameter(m, name="b")

problem = cp.Problem(
    cp.Minimize(cp.sum_squares(A @ x - b)),
    [x >= 0]
)
```

code generation

```python
import cvxgenrust as cgr  # import CVXGenRust

# generate custom solver code
cgr.generate_code(
    problem,
    module_name="nonneg_ls"
)
```

Rust integration

```rust
// create a new problem instance
use nonneg_ls::CGRProblem;
let mut problem = CGRProblem::new();

// update the problem parameters
problem.set_a(&a_values)?;
problem.set_b(&b_values)?;

// solve and extract the solution
let solution = problem.solve()?;
let x = problem.extract_x(&solution.x)?;
```

Python integration

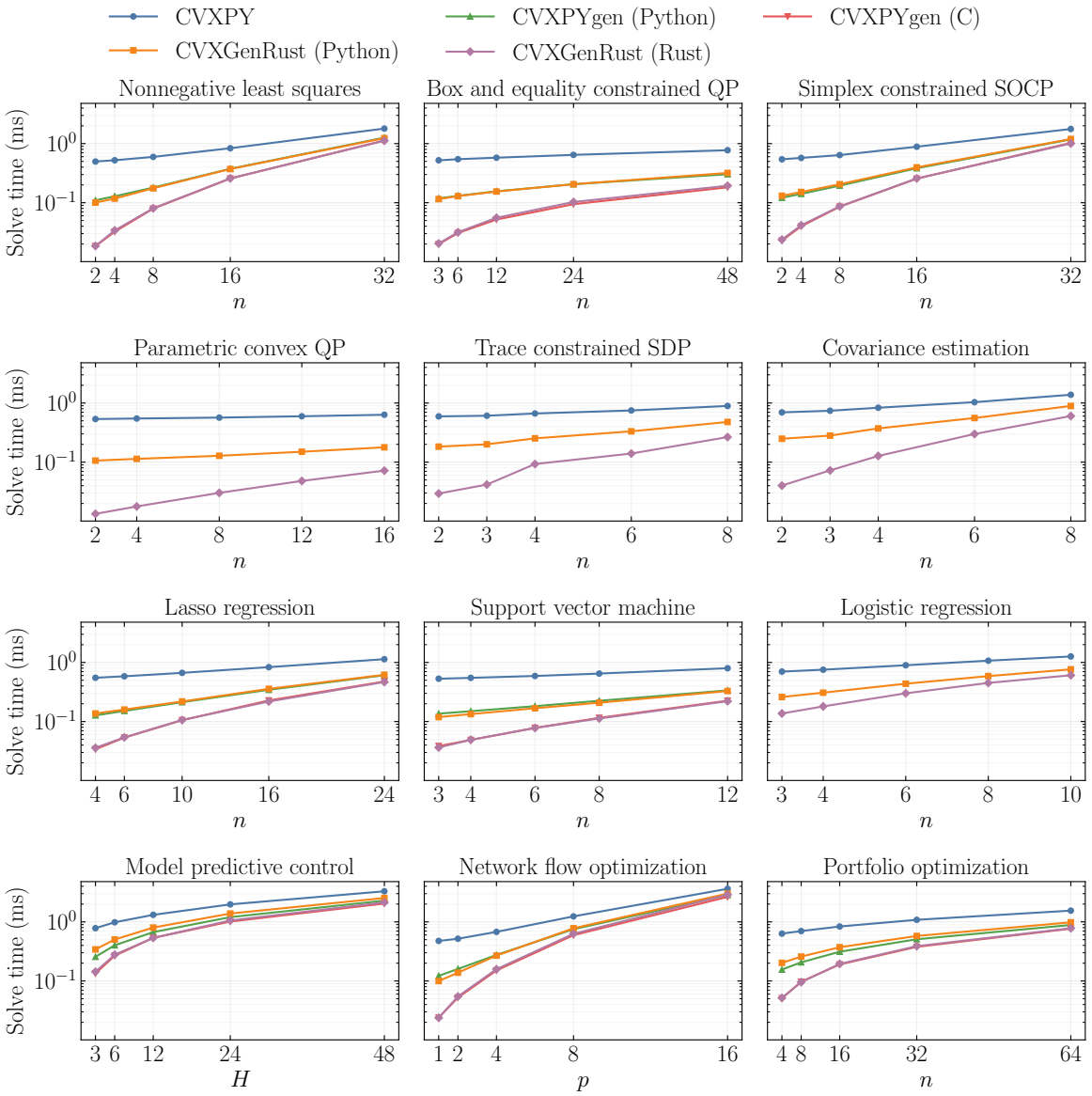
```python
from nonneg_ls_wrapper.cgr_solver import cgr_solve

# register the generated solver in CVXPY
problem.register_solve("CGR", cgr_solve)

# update the problem parameters
A.value = A_value
b.value = b_value

# solve with the generated solver
problem.solve(
    method="CGR",
    updated_params=["A", "b"]
)
```

Benchmarks


CVXPY
CVXPYgen (Python)
CVXPYgen (C)
CVXGenRust (Python)
CVXGenRust (Rust)
Nonnegative least squares
Box and equality constrained QP
Simplex constrained SOCP
Parametric convex QP
Trace constrained SDP
Covariance estimation
Lasso regression
Support vector machine
Logistic regression
Model predictive control
Network flow optimization
Portfolio optimization
Solve time (ms)
10^0
10^{-1}
n
H
p

Resources

- CVXGenRust paper:

  `https://haozhu10015.github.io/papers/cvxgenrust`

- CVXGenRust code:

  `https://github.com/dxogrp/cvxgenrust`

  – installation:

  ```
  pip install cvxgenrust
  ```

- examples:

  `https://github.com/dxogrp/cvxgenrust/tree/main/examples`

References

[AAB+19] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter. Differentiable convex optimization layers. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[Bla16] L. Blackmore. Autonomous precision landing of space rockets. In *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2016 Symposium*, pages 33–41. The National Academies Press, 2016.

[GC24] P. J. Goulart and Y. Chen. Clarabel: An interior-point solver for conic programs with quadratic objectives. *arXiv*, 2405.12762, 2024.

[MB12] J. Mattingley and S. Boyd. CVXGEN: A code generator for embedded convex optimization. *Optimization and Engineering*, 13:1–27, 2012.

[SBD+22] M. Schaller, G. Banjac, S. Diamond, A. Agrawal, B. Stellato, and S. Boyd. Embedded code generation with CVXPY. *IEEE Control Systems Letters*, 6:2653–2658, 2022.

[VFK+22] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl. acados — a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 14:147–183, 2022.