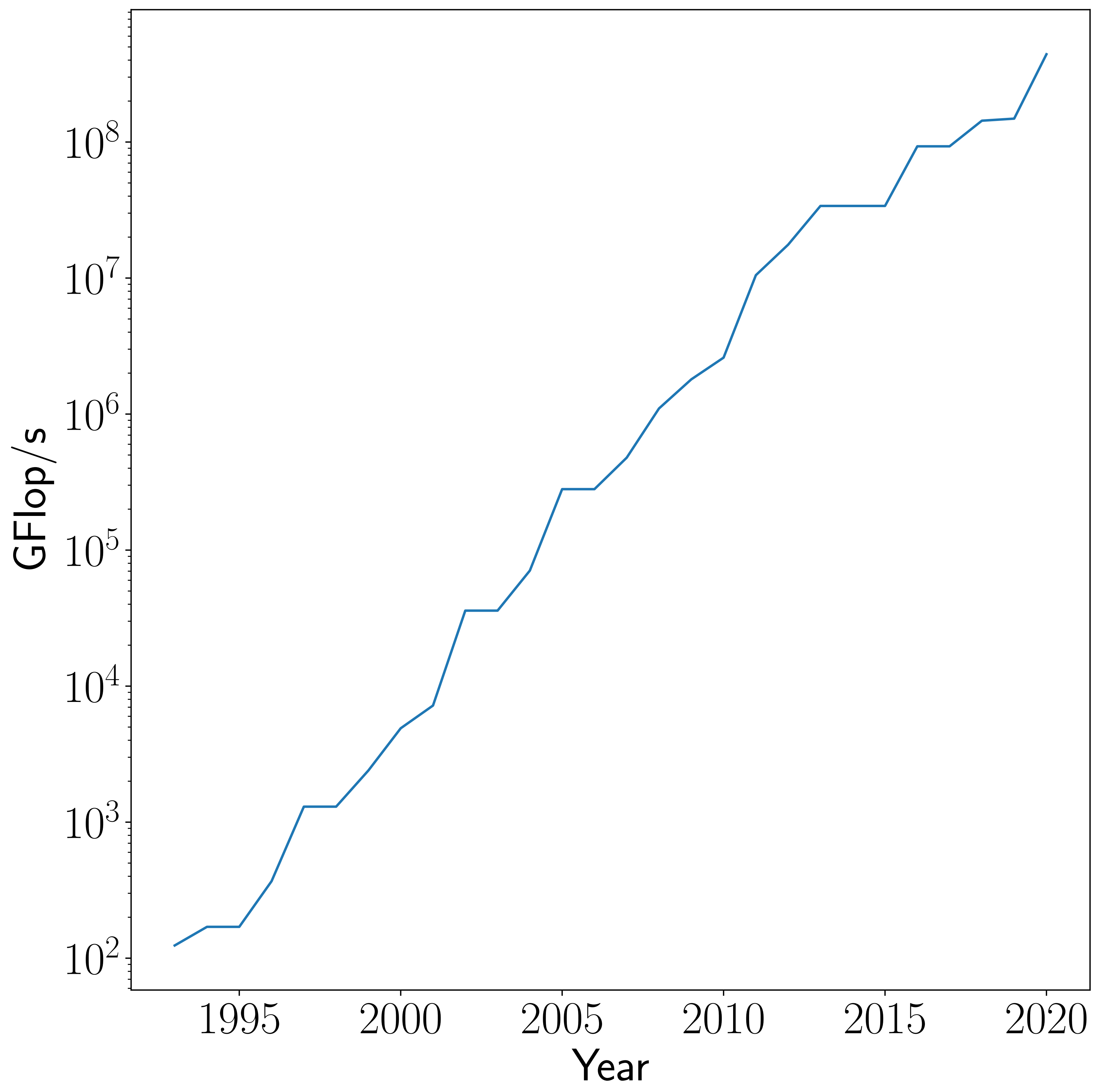
Data-Driven Embedded Optimization for Control

Bartolomeo Stellato — RTRC Seminar Series on Intelligent Cyber-Physical Systems, Jan 2021

Tremendous progress in optimization

Top500 peak CPU power



Hardware + Software

400 billion times speedups!

400,000 years → 30 seconds

[Our World in Data, https://github.com/owid/owid-datasets]

Is it enough?

400,000 years → 30 seconds

Robotics

< 10 milliseconds

High-Frequency Trading

< 1 millisecond

Same problem with varying data

data

$$\begin{array}{ll} \text{minimize} & f(x, \theta) \\ \text{subject to} & g(x, \theta) \leq 0 \end{array}$$

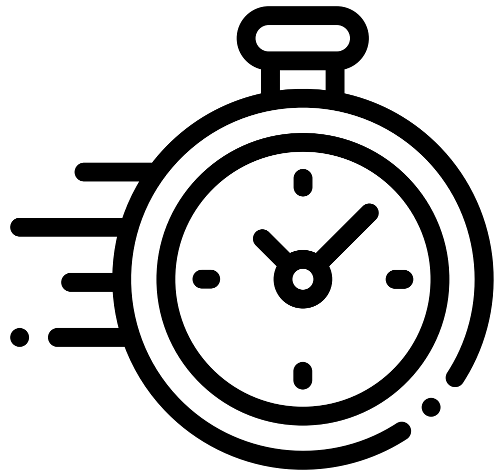
decisions

Can we solve it in **milliseconds** *or* **microseconds**?

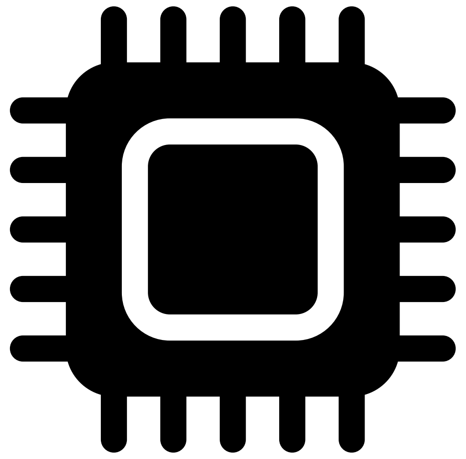
Challenges in real-time optimization

Hardware

Real-Time

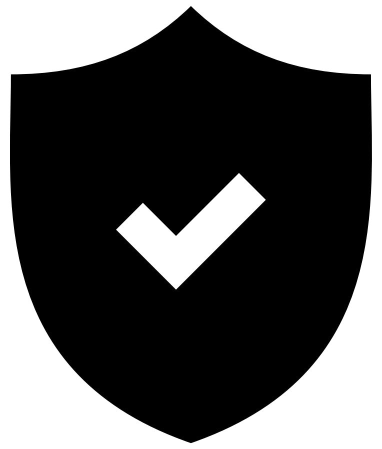


Limited resources

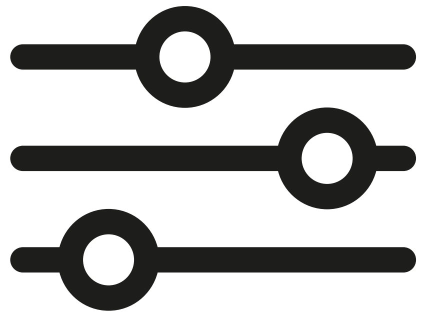


Software

Reliability



Easy
tuning

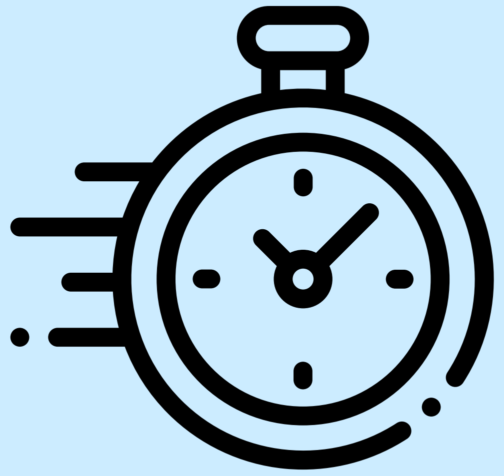


Today's talk

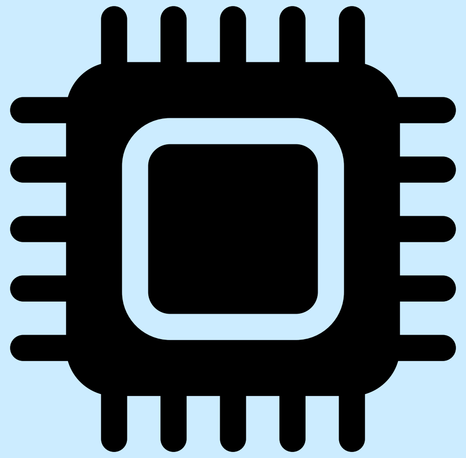
Data-Driven Embedded Optimization for Control

OSQP Solver

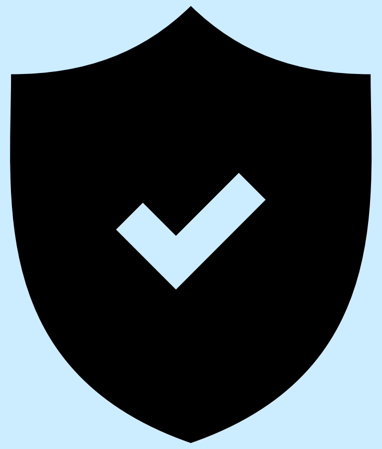
Real-Time



Limited resources

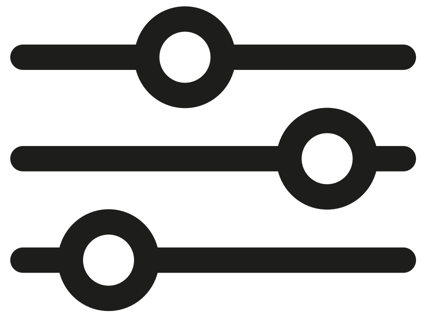


Reliability



Learning Convex Optimization Control Policies

Easy tuning



OSQP Solver

Still quadratic programming?

AN ALGORITHM FOR QUADRATIC PROGRAMMING

Marguerite Frank and Philip Wolfe[1]
Princeton University

A finite iteration method for calculating the solution of quadratic programming problems is described. Extensions to more general non-linear problems are suggested.

1. INTRODUCTION

The problem of maximizing a concave quadratic function whose variables are subject to linear inequality constraints has been the subject of several recent studies, from both the computational side and the theoretical (see Bibliography). Our aim here has been to develop a method for solving this non-linear programming problem which should be particularly well adapted to high-speed machine computation.

March 1956!


145
140
135
130
125
Tues.
142.78
Chg.
1.64
1.98
Day

First-order methods

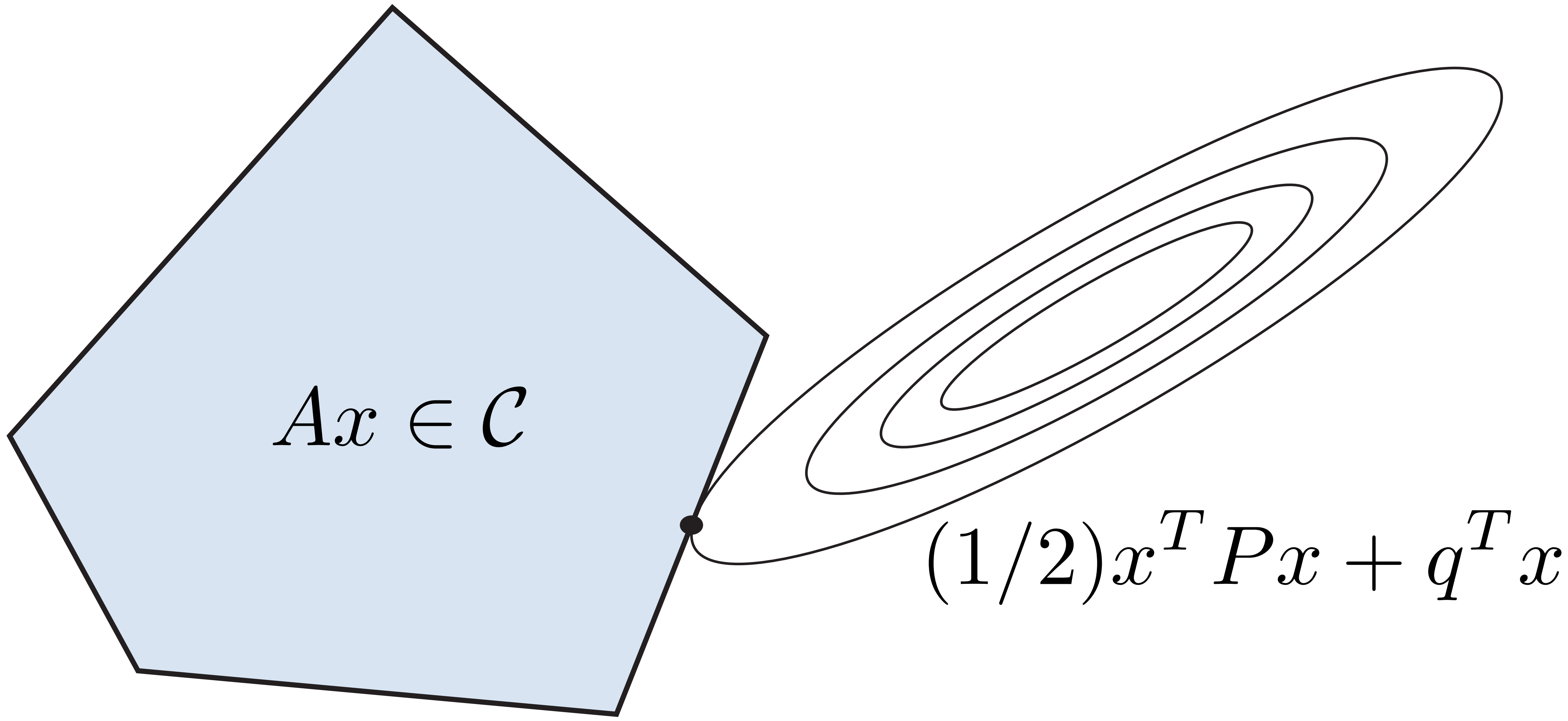
Wide popularity

Pros	Cons		OSQP
Warm-starting	Low quality solutions		High-quality solutions
Large-scale problems	Can’t detect infeasibility	→	Detects infeasibility
Embeddable	Problem data dependent		Robust

The problem

$$\begin{array}{ll} \text{minimize} & (1/2)x^T P x + q^T x \\ \text{subject to} & Ax \in \mathcal{C} \end{array}$$

Quadratic program: $\mathcal{C} = [l, u]$



ADMM

Alternating Direction Method of Multipliers

$$\text{minimize} \quad f(x) + g(x) \longrightarrow$$

Splitting

$$\begin{array}{ll} \text{minimize} & \boxed{f(\tilde{x})} + \boxed{g(x)} \\ \text{subject to} & \tilde{x} = x \end{array}$$

Iterations

$$\tilde{x}^{k+1} \leftarrow \operatorname*{argmin}_{\tilde{x}} \left(\boxed{f(\tilde{x})} + \rho/2 \left\| \tilde{x} - (x^k - y^k/\rho) \right\|^2 \right)$$

$$x^{k+1} \leftarrow \operatorname*{argmin}_{x} \left(\boxed{g(x)} + \rho/2 \left\| x - (\tilde{x}^{k+1} + y^k/\rho) \right\|^2 \right)$$

$$y^{k+1} \leftarrow y^k + \rho \left(\tilde{x}^{k+1} - x^{k+1} \right)$$

How do we split the QP?

$$
\begin{array}{lll}
\text{minimize} & (1/2)x^T P x + q^T x & \multirow{2}{*}{f} \\
\text{subject to} & Ax = z & \\
 & z \in \mathcal{C} & g
\end{array}
$$

Splitting formulation

$$
\begin{array}{ll}
\text{minimize} & \overbrace{(1/2)\tilde{x}^T P \tilde{x} + q^T \tilde{x} + \mathcal{I}_{Ax=z}(\tilde{x}, \tilde{z})}^{f} + \overbrace{\mathcal{I}_{\mathcal{C}}(z)}^{g} \\
\text{subject to} & (\tilde{x}, \tilde{z}) = (x, z)
\end{array}
$$

ADMM iterations

Inner QP

$$\left(x^{k+1}, \tilde{z}^{k+1}\right) \leftarrow \underset{(x,z):Ax=z}{\operatorname{argmin}} \; (1/2)x^T P x + q^T x + \sigma/2 \left\|x - x^k\right\|^2 + \rho/2 \left\|z - z^k + y^k/\rho\right\|^2$$

$$z^{k+1} \leftarrow \Pi\left(\tilde{z}^{k+1} + y^k/\rho\right)$$

Projection onto $\mathcal{C}$

$$y^{k+1} \leftarrow y^k + \rho\left(\tilde{z}^{k+1} - z^{k+1}\right)$$

Solving the inner QP

Equality-constrained

$$\begin{array}{ll} \text{minimize} & (1/2)x^T P x + q^T x + \sigma/2 \left\| x - x^k \right\|^2 + \rho/2 \left\| z - z^k + y^k/\rho \right\|^2 \\ \text{subject to} & Ax = z \end{array}$$

Reduced KKT system

Always solvable!

$$\begin{bmatrix} P + \sigma I & A^T \\ A & -\frac{1}{\rho} I \end{bmatrix} \begin{bmatrix} x \\ \nu \end{bmatrix} = \begin{bmatrix} \sigma x^k - q \\ z^k - \frac{1}{\rho} y^k \end{bmatrix}$$

Solving the linear system

Direct method (small to medium scale)

Quasi-definite matrix

$$\begin{bmatrix} P + \sigma I & A^T \\ A & -\frac{1}{\rho} I \end{bmatrix} \begin{bmatrix} x \\ \nu \end{bmatrix} = \begin{bmatrix} \sigma x^k - q \\ z^k - \frac{1}{\rho} y^k \end{bmatrix}$$

Well-defined LDL^T factorization

Factorization caching

$\longrightarrow$

QDLDL
Free quasi-definite linear system solver

[https://github.com/oxfordcontrol/qdldl]

Solving the linear system

Indirect method (large scale)

Positive-definite matrix

$$\boxed{\left(P + \sigma I + \rho A^T A\right)} x = \sigma x^k - q + A^T(\rho z^k - y^k)$$

Conjugate gradient

Solve very large systems

→

GPU implementation

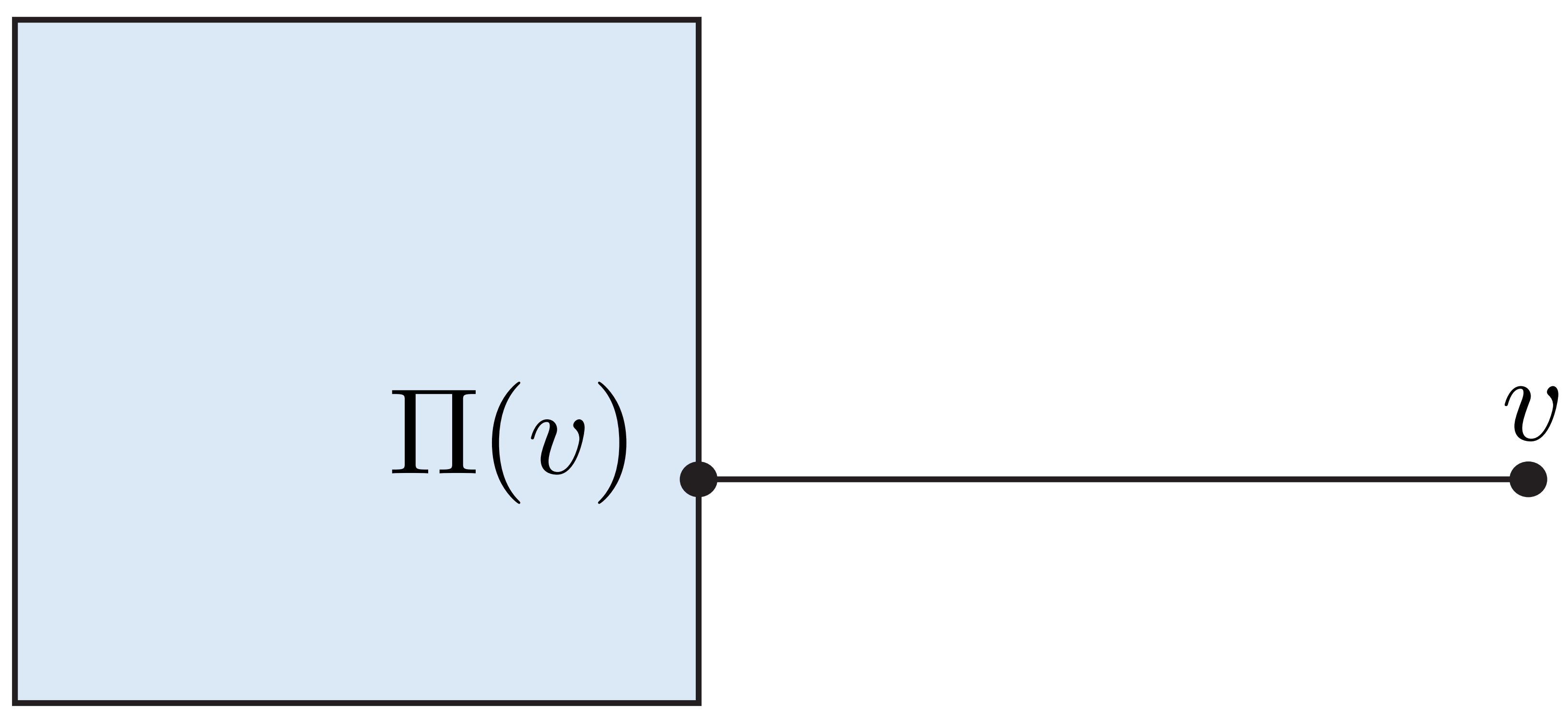
[https://github.com/oxfordcontrol/cuosqp]

Computing the projection

Quadratic program: $\mathcal{C} = [l, u]$

Box projection

$$\Pi(v) = \max(\min(v, u), l)$$



Complete algorithm

Problem

minimize $\quad (1/2)x^T P x + q^T x$

subject to $\quad l \leq Ax \leq u$

Algorithm

Linear system solve

$$(x^{k+1}, \nu^{k+1}) \leftarrow \text{solve} \begin{bmatrix} P + \sigma I & A^T \\ A & -\frac{1}{\rho} I \end{bmatrix} \begin{bmatrix} x^{k+1} \\ \nu^{k+1} \end{bmatrix} = \begin{bmatrix} \sigma x^k - q \\ z^k - \frac{1}{\rho} y^k \end{bmatrix}$$

Easy operations

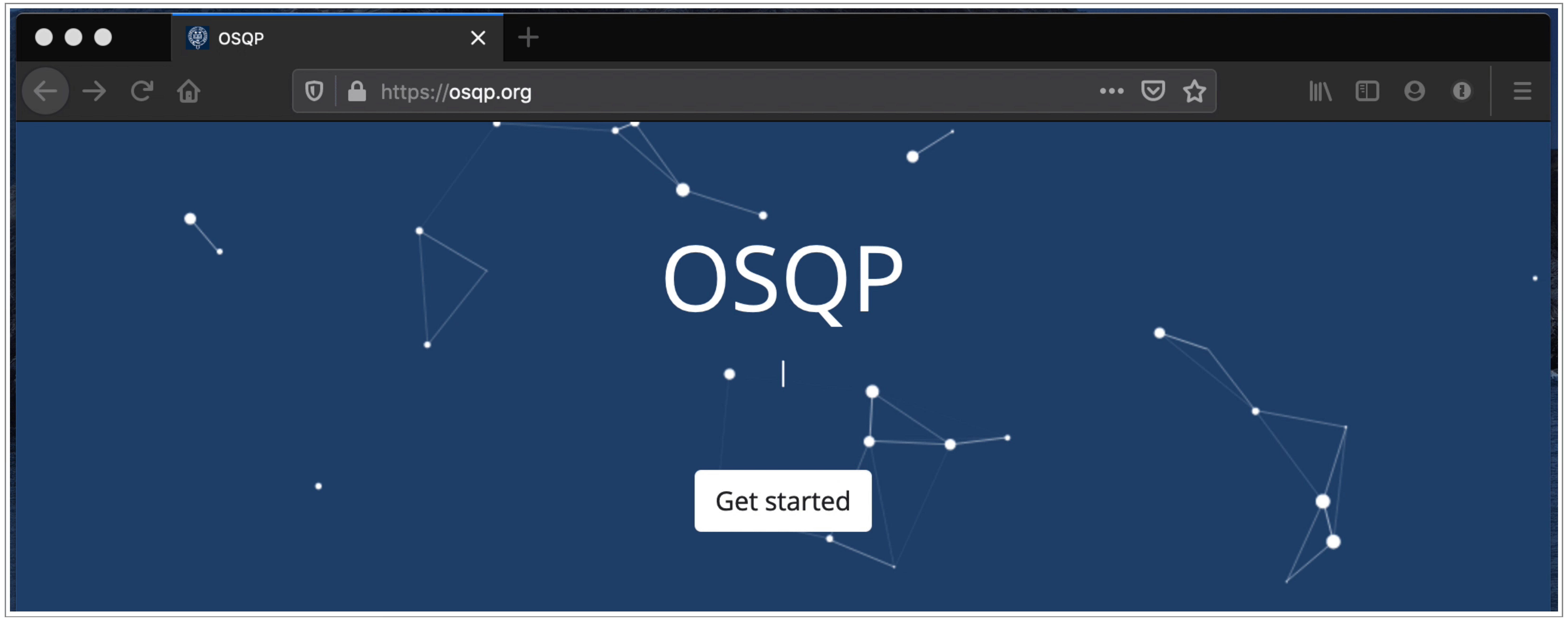
$$\tilde{z}^{k+1} \leftarrow z^k + (\nu^{k+1} - y^k)/\rho$$

$$z^{k+1} \leftarrow \Pi\left(\tilde{z}^{k+1} + y^k/\rho\right)$$

$$y^{k+1} \leftarrow y^k + \rho\left(\tilde{z}^{k+1} - z^{k+1}\right)$$

OSQP

Operator Splitting solver for Quadratic Programs



Embeddable (can be division free!)

Supports warm-starting

Detects infeasibility

Solves large-scale problems

Users

More than 2 million downloads!

MIT

Los Alamos National Laboratory EST. 1943

Berkeley University of California

NYU

Toyota Let's Go Places

Google

UC Davis University of California

IIT Istituto Italiano di Tecnologia

ETH zürich

Lund University

BlackRock®

Aptiv

Stanford University

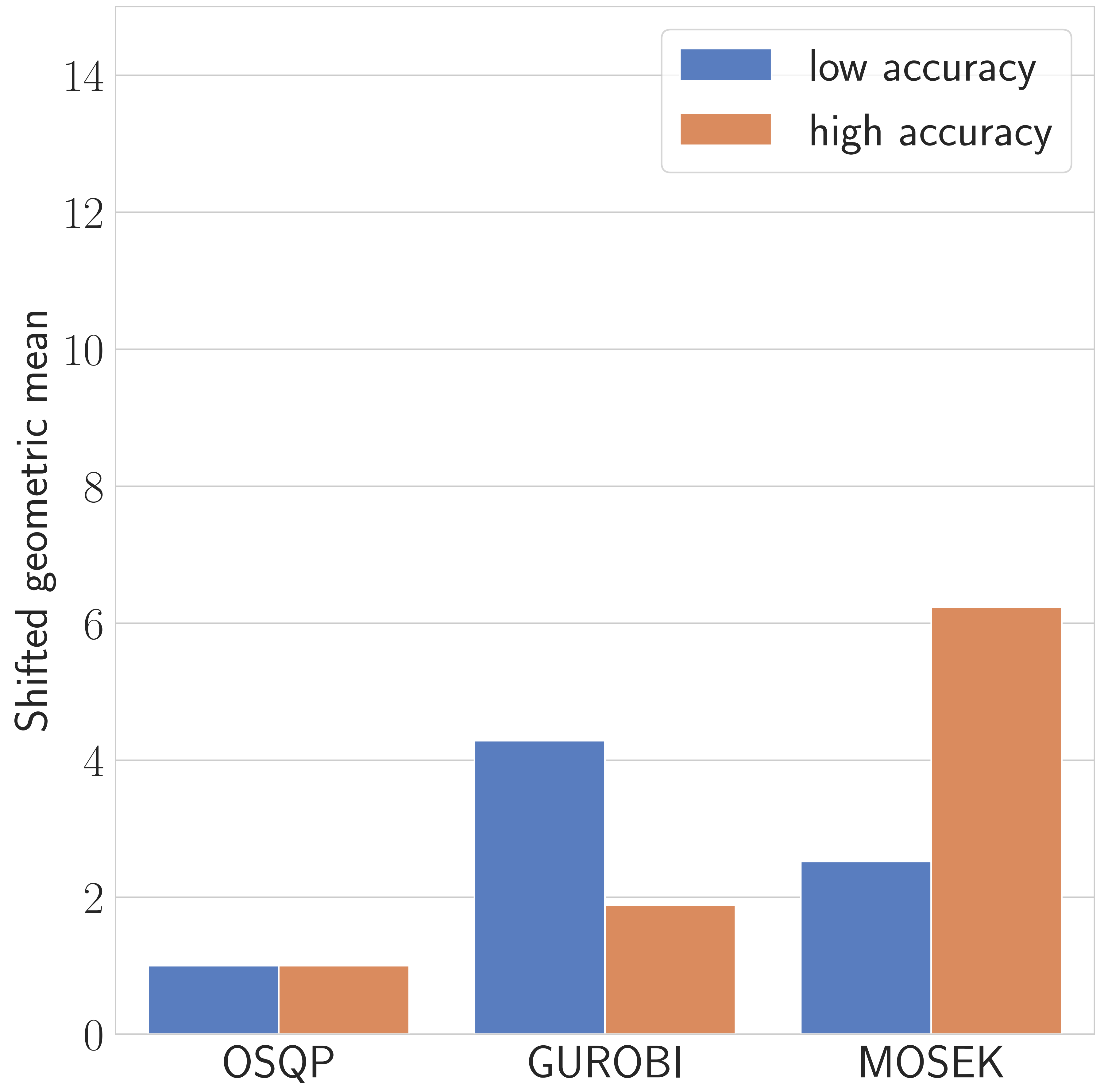
KU Leuven

Universität Stuttgart

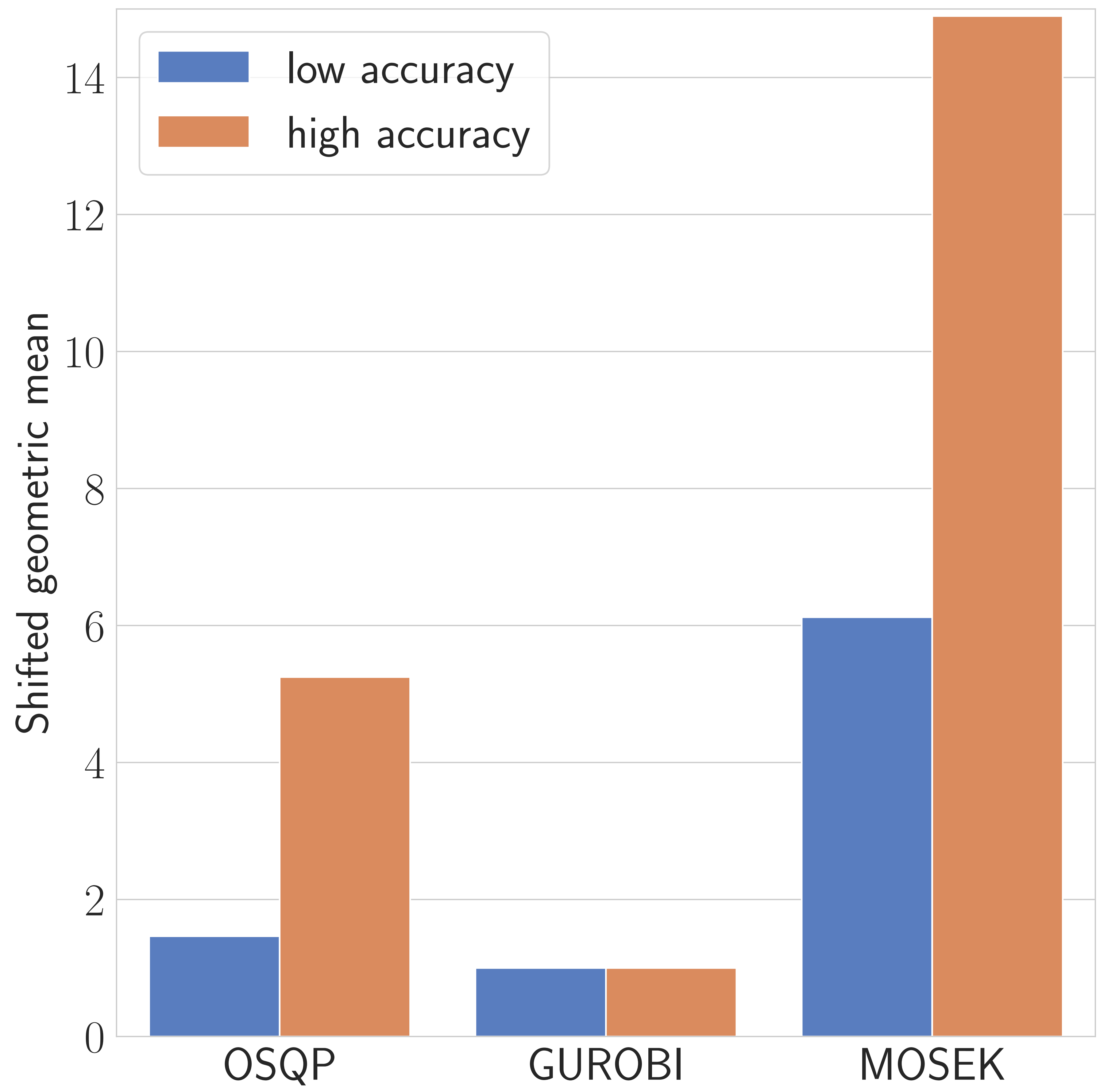
[pepy.tech/project/osqp]

Performance benchmarks

OSQP Benchmarks
(control, portfolio, lasso, SVM, etc.)



Maroz-Meszaros

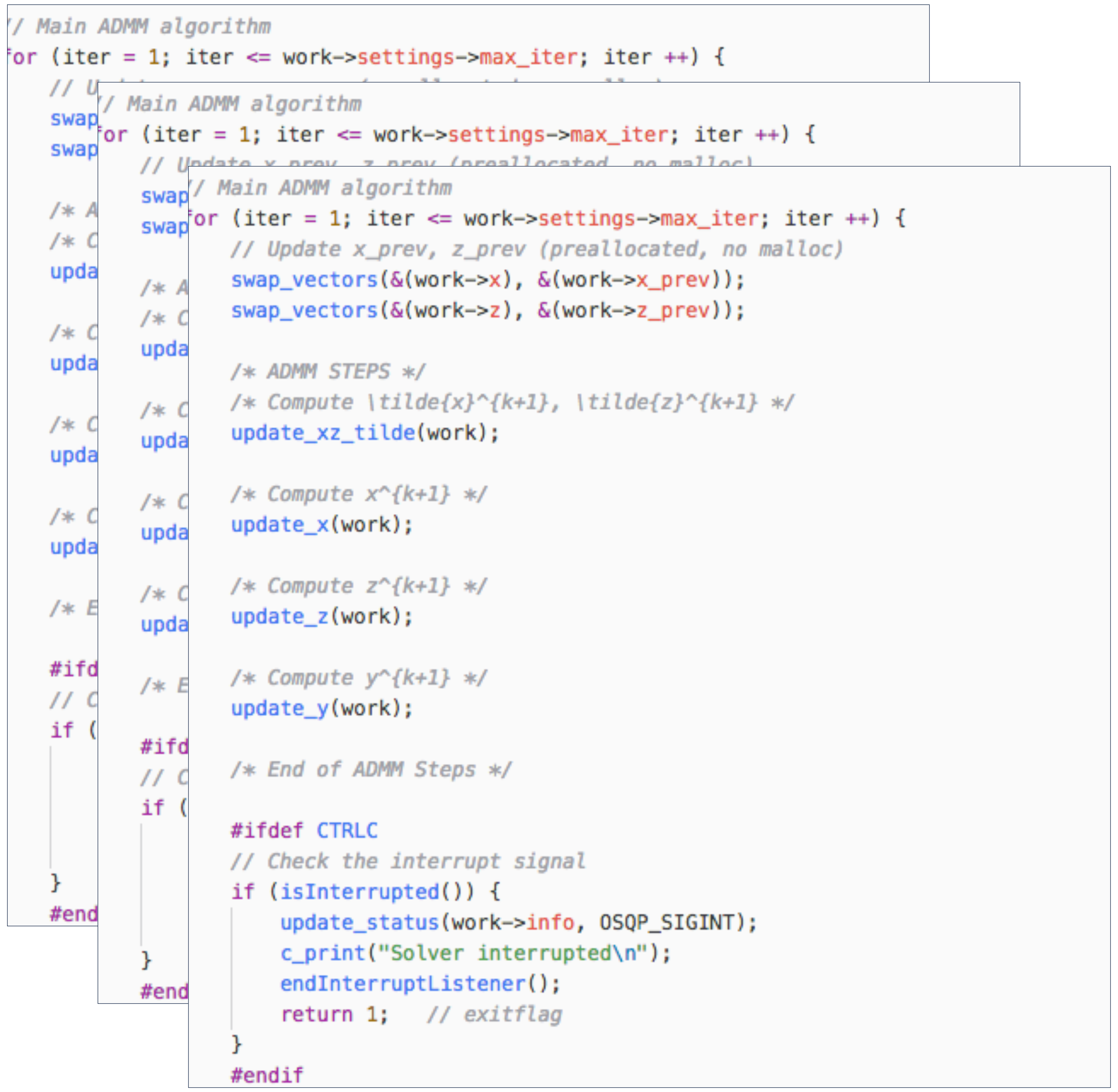


[github.com/oxfordcontrol/osqp_benchmarks]

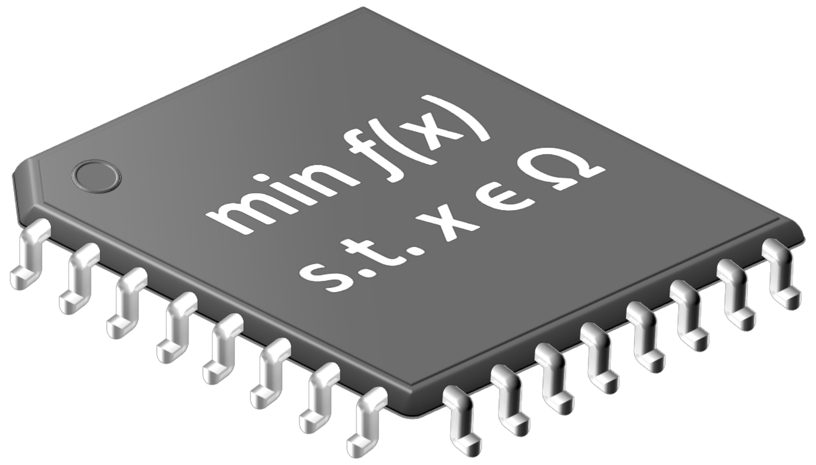
Code generation

```
# Create OSQP object
m = osqp.OSQP()

# Initialize solver
m.setup(P, q, A, l, u,
        settings)

# Generate C code
m.codegen('folder_name')
```

Optimized C code

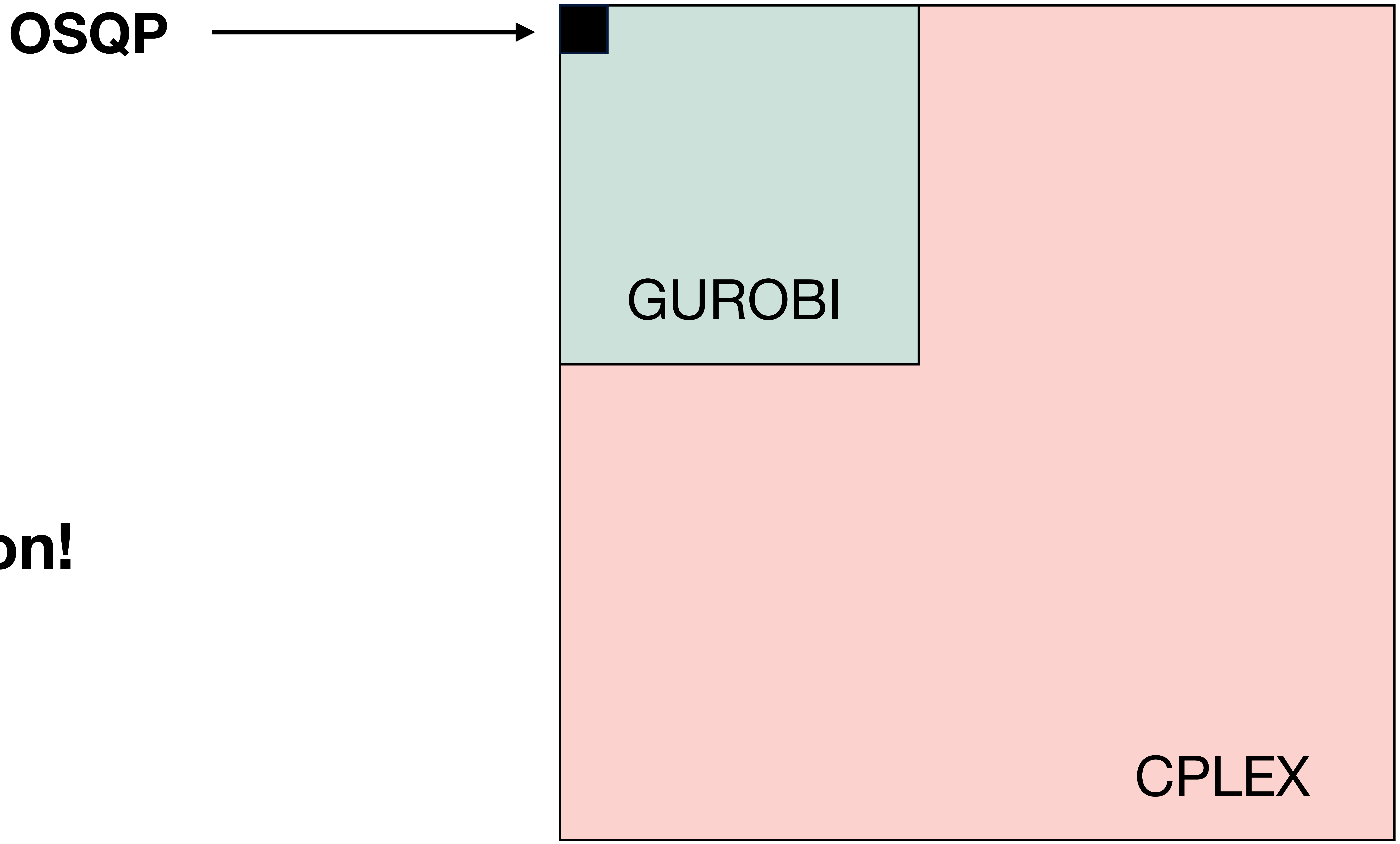


Embedded Hardware



It can be compiled into division-free

Compiled code size ~80kb (low footprint)



300x Reduction!

OSQP summary

Robust

Embeddable
(can be division free!)

Supports
warm-starting

Detects
infeasibility

Solves large-scale
problems

Future work

Algorithms

- Improvements: acceleration, restarts
- Semidefinite optimization (SDP)
- Sequential quadratic programming (SQP)
- Mixed-integer optimization

Architecture

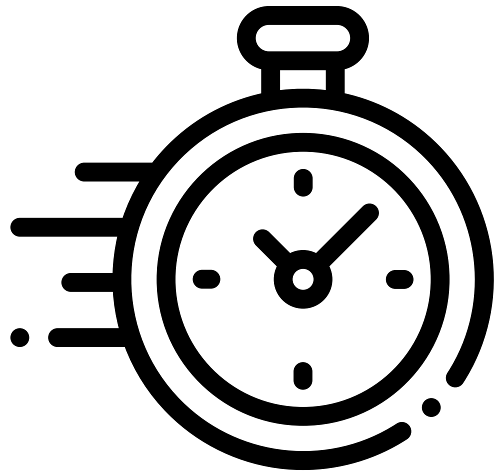
- New linear algebra
- New linear system solvers
- New languages supported

Today's talk

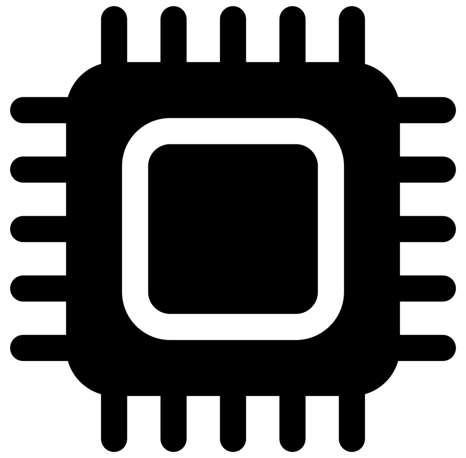
Data-Driven Embedded Optimization for Control

OSQP Solver

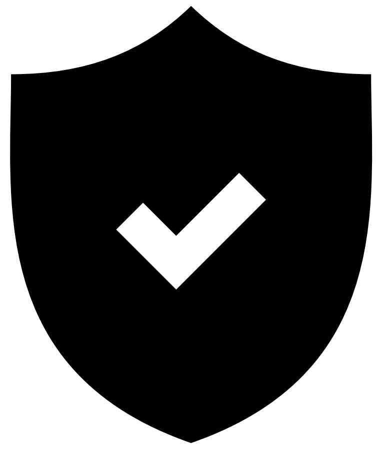
Real-Time



Limited resources

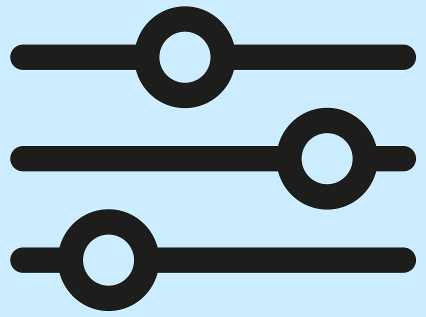


Reliability



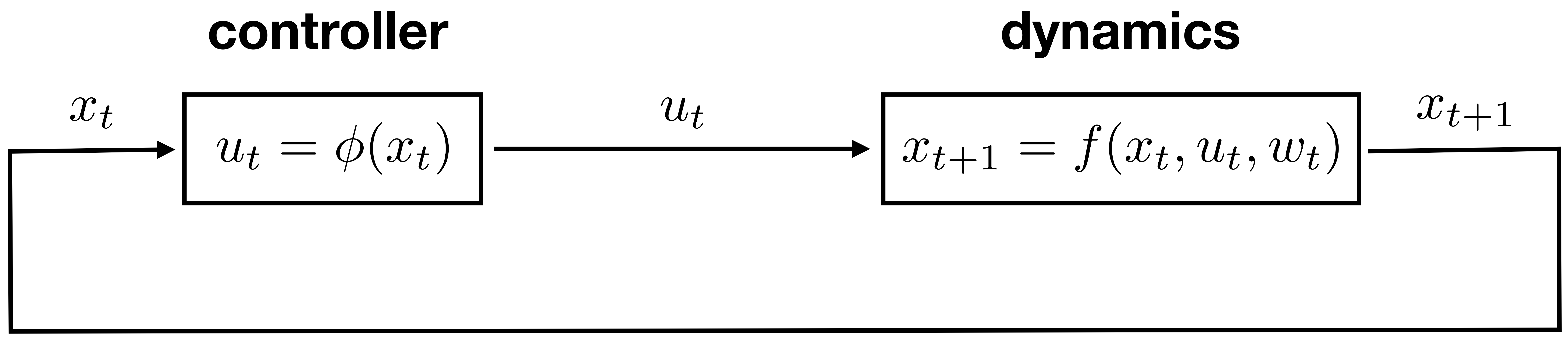
Learning Convex Optimization Control Policies

Interpretable tuning



Learning Convex Optimization Control Policies

Control loop



x_t state
u_t input
w_t (random) disturbance

$\phi(x_t)$ **control policy**

Explicit vs implicit control policies

Explicit

Complete control specification

Example: PI Controller

$$u_t = -K_P e_t - K_I \sum_{\tau=0}^{t} e_\tau$$

Implicit (optimization-based)

Designer specifies goal and requirements → **Optimizer** computes the action

Example: LQR Controller

dynamics: $x_{k+1} = Ax_t + Bu_t + w_t$
stage cost: $x^T Q x + u^T R u$

$$\begin{aligned} u_t &= \underset{u}{\text{argmin}}\, u^T R u + (Ax_t + Bu)^T P (Ax_t + Bu) \\ &= K x_t \end{aligned}$$

Convex optimization control policies (COCPs)

$$\begin{array}{rll} u_t = & \underset{u}{\text{argmin}} & f(x_t, u, \theta) \\ & \text{subject to} & g(x_t, u, \theta) \leq 0 \\ & & A(x_t, \theta)u = b(x_t, \theta) \end{array}$$

x_t state
θ parameters to tune
f, g convex functions

Many control policies are COCPs

Examples

- Linear Quadratic Regulator (LQR)
- Model predictive control (MPC)
- Actuator allocation
- Resource allocation
- Portfolio trading

Advantages

Interpretable

Satisfy constraints

Handle varying dynamics

Efficient and reliable (even division-free: OSQP)

Judging COCPs

Given a policy, state and input trajectories form a ***stochastic process***

Trajectories

$$X = (x_0, \ldots, x_{T-1}, x_T)$$
$$U = (u_0, \ldots, u_{T-1})$$
$$W = (w_0, \ldots, w_{T-1})$$

$\longrightarrow$

Policy cost

$$J(\theta) = \mathbf{E}\, \psi(X, U, W)$$

Approximate $J(\theta)$ from data (monte carlo simulation)

$$\hat{J}(\theta) = \frac{1}{K} \sum_{i=1}^{K} \psi(X^i, U^i, W^i)$$

COCP Example: dynamic programming

Time-separable cost

$$\psi(X, U, W) = \sum_{t=0}^{T-1} g(x_t, u_t, w_t)$$

Optimal policy as $T \to \infty$

$$\phi(x_t) = \underset{u}{\text{argmin}}\, \mathbf{E}\left(g(x_t, u, w_t) + \underbrace{V}_{\text{Value function}}(f(x_t, u, w_t))\right)$$

COCP if

- f affine in x and u
- g convex in x and u
- V is convex

COCP Example: approximate dynamic programming

$$\phi(x_t) = \underset{u}{\text{argmin}}\, \mathbf{E}\left(g(x_t, u, w_t) + \hat{V}(f(x_t, u, w_t))\right)$$

Approximate value function

COCP if

- f affine in x and u
- g convex in x and u
- $\hat{V}$ is convex $\longrightarrow$ **(even when V is not)**

Controller tuning problem

Goal

minimize $\quad J(\theta)$

Nonconvex and difficult to solve

Traditional approaches

- **Hand-tuning** (few parameters, simple dependencies)
- **Derivative-free method** (very slow)

Learning scheme

Auto-tuning

Stochastic gradient descent

$$\theta^{k+1} = \theta^k - \underbrace{t^k}_{\text{step size}} \underbrace{\nabla_\theta \hat{J}(\theta^k)}_{\text{stochastic gradient from simulation}}$$

Generalization

Split simulation data in training, validation and testing

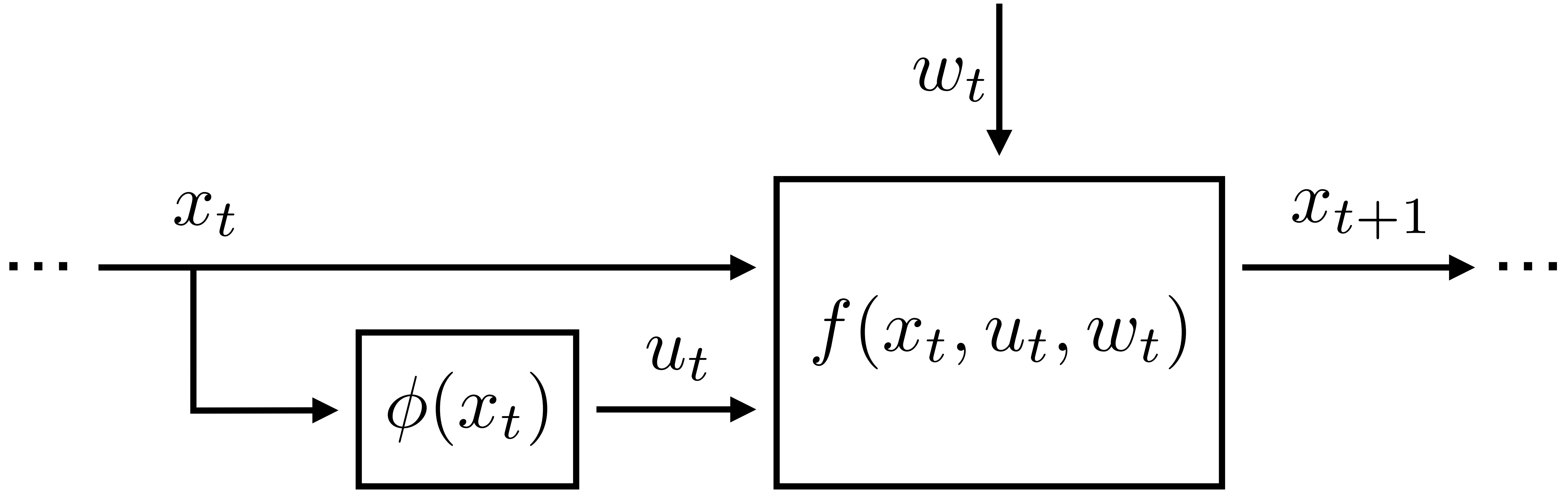
Non differentiable $\hat{J}(\theta)$?

Still get a descent direction (common in NN community)

Implementation

Automatic differentiation

- **Build** computation graph (simulate)
- **Backpropagate** using PyTorch



CVXPYLayers

Backpropagate through COCPs
(differentiate KKT optimality conditions)

[https://github.com/cvxgrp/cvxpylayers/]

[Differentiable convex optimization layers. Agrawal, Amos, Barratt, Boyd, Diamond, and Kolter. NeurIPS 2019]

[Differentiable optimization-based modeling for machine learning. Amos. PhD thesis 2019]

Box-constrained LQR

Problem setup

- dynamics: $x_{t+1} = Ax_t + Bu_t + w_t$
- actuator limit: $\|u_t\|_\infty \leq 1$
- stage cost: $x_t^T Q x_t + u_t^T R u_t$

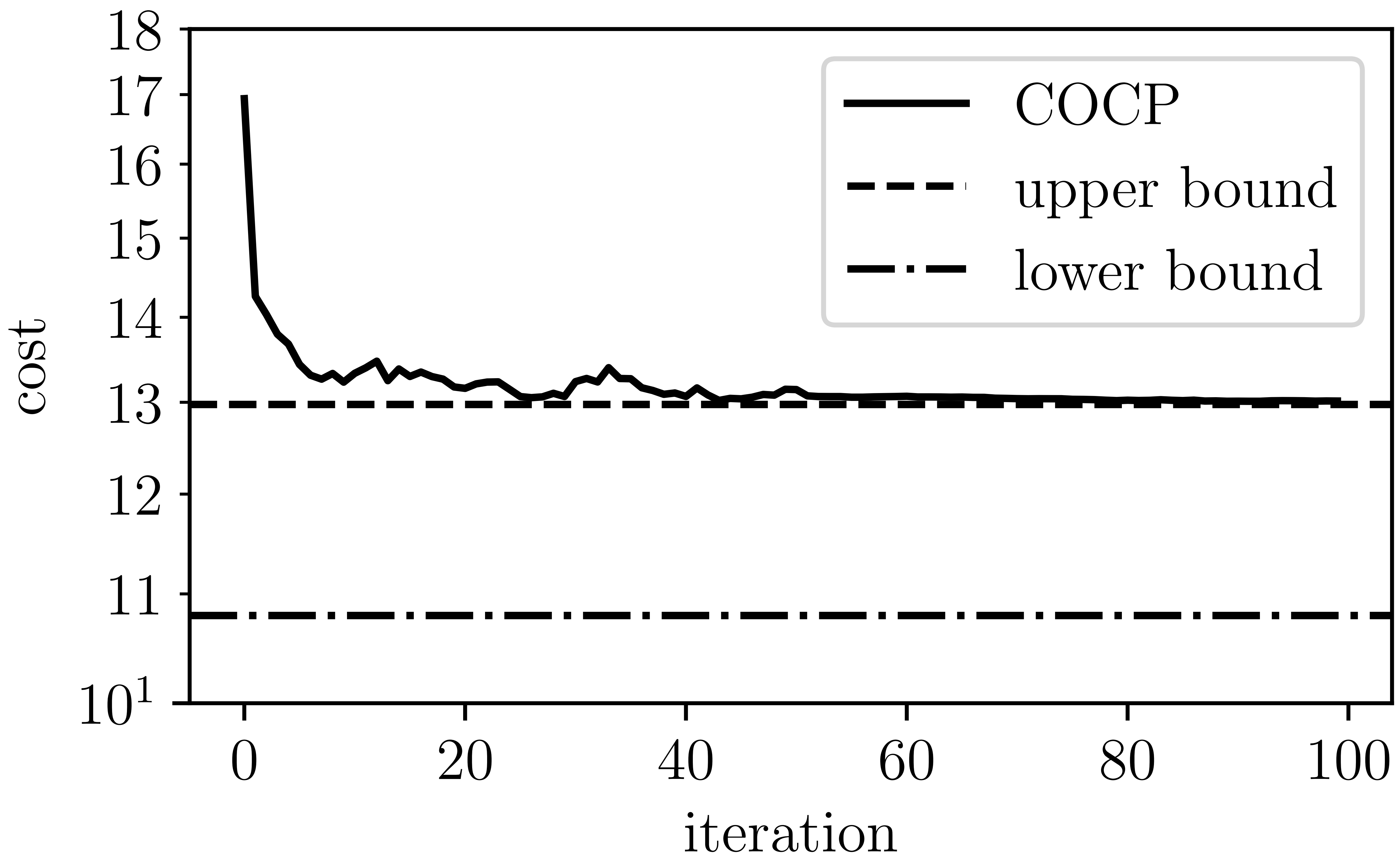
COCP Policy (QP)

$$u_t = \begin{array}{ll} \underset{u}{\text{argmin}} & u^T R u + \|\theta(Ax_t + Bu)\|_2^2 \\ \text{subject to} & \|u\|_\infty \leq 1 \end{array}$$

parameters

Box-constrained LQR

Performance



Standard upper/lower bounds from SDPs

Hard to generalize
(other dynamics, disturbances, etc)

Supply chain distribution

supplier price

quantity held

consumer demand

State: $x_t = (h_t, p_t, d_t)$

sell

buy

ship

Input: $u_t = (b_t, s_t, z_t)$

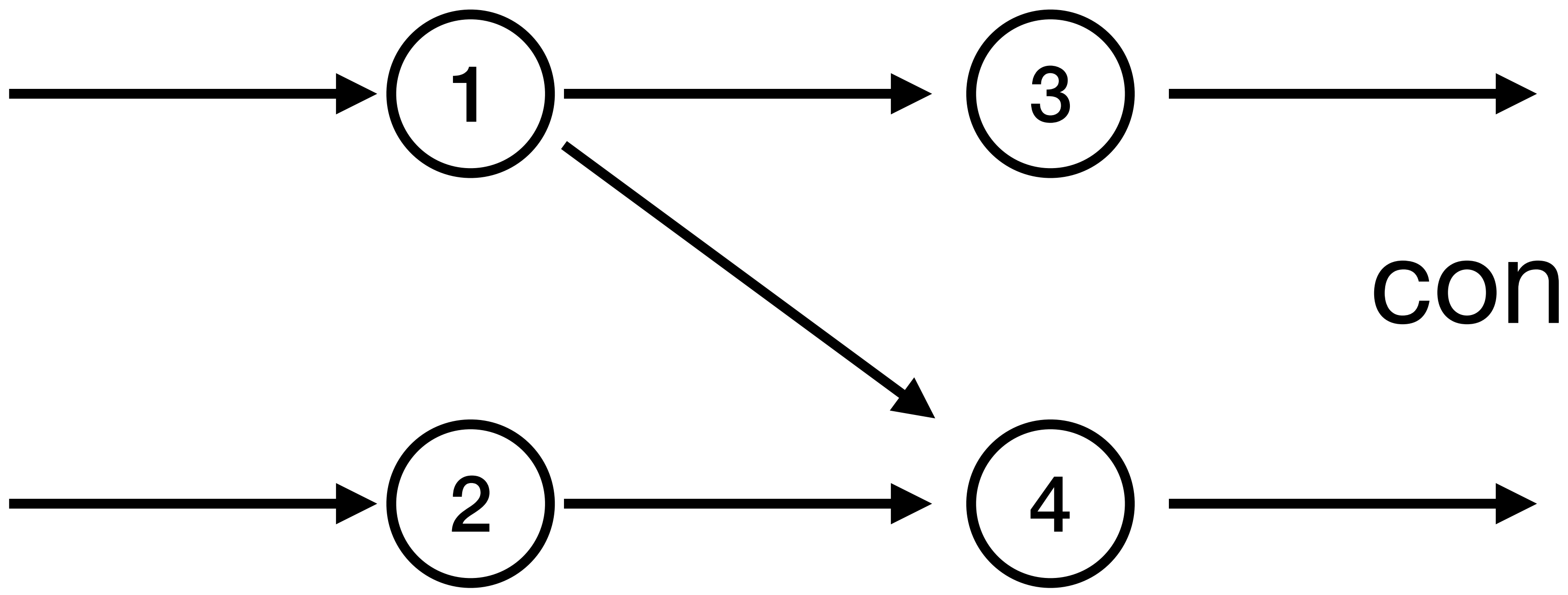
Dynamics

$$h_{t+1} = h_t + (A^{\text{in}} - A^{\text{out}})u_t$$

p_{t+1} and d_{t+1} are log-normal

Network example

suppliers



consumers

Supply chain distribution

Cost and constraints

Stage cost

sale revenues

constraints

$$\underbrace{p_t^T b_t - r^T s_t + \tau^T z_t} + \underbrace{\alpha^T h_t + \beta^T h_t^2} + \underbrace{\mathcal{I}(x_t, u_t)}$$

suppliers payment

shipment cost

storage cost

Constraints

$$0 \leq h_t \leq h_{\max}, \qquad 0 \leq u_t \leq u_{\max}$$

$$A^{\text{out}} u_t \leq h_t, \qquad s \leq d_t$$

Supply chain distribution COCP

COCP

QP problem **parameters**

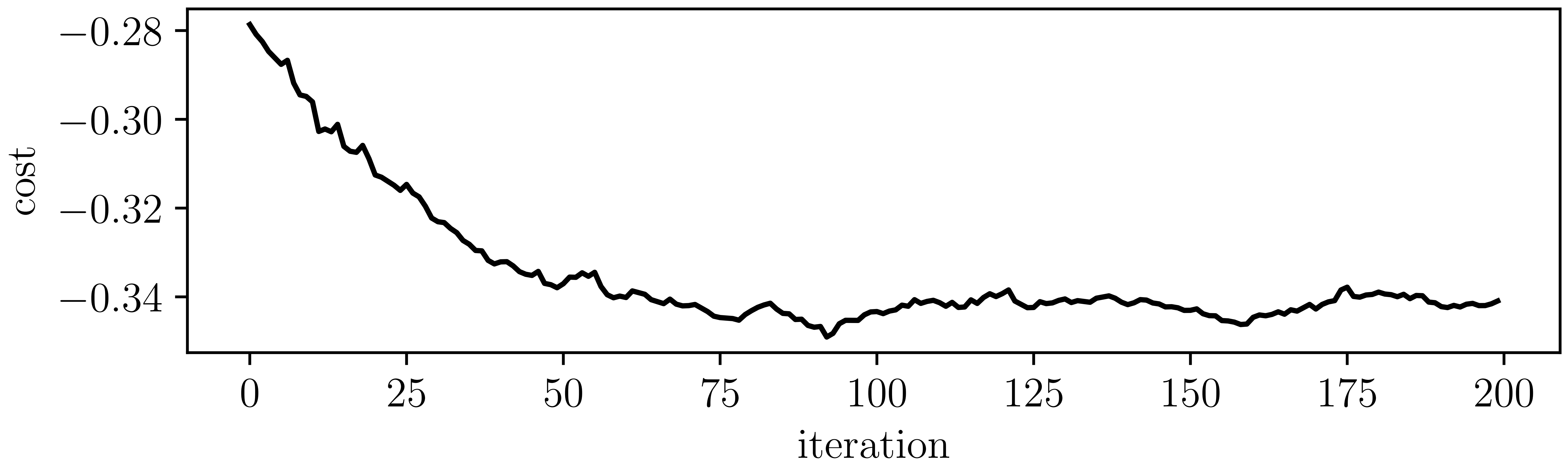
$$
u_t = (b_t, s_t, z_t) = \quad \text{argmin} \quad p_t^T b - r^T s + \tau^T z + \|\boxed{S} h^+\|_2^2 + \boxed{q}^T h^+
$$

$$
\begin{array}{ll}
\text{subject to} & h^+ = h_t + (A^{\text{in}} - A^{\text{out}})(b, s, z) \\
& 0 \le h^+ \le h_{\max}, \quad 0 \le (b, s, z) \le u_{\max} \\
& A^{\text{out}}(b, s, z) \le h_t, \quad s \le d_t
\end{array}
$$

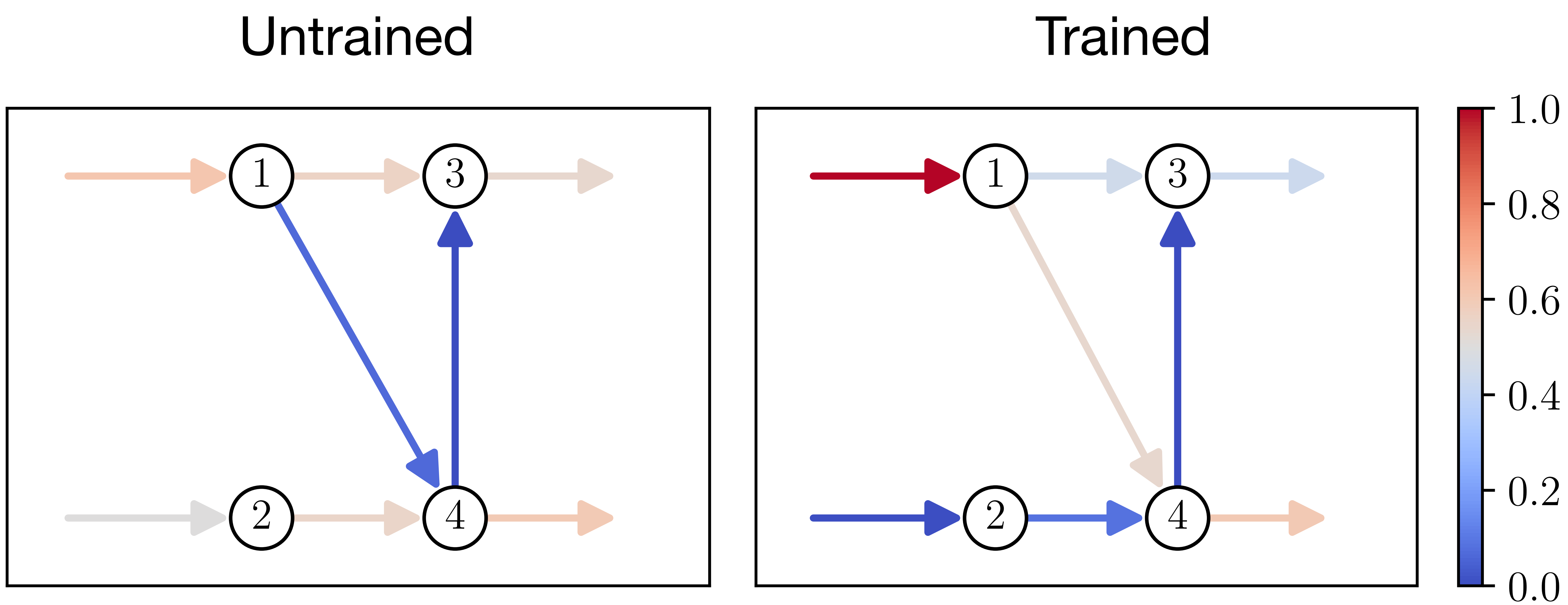
Supply chain distribution

Results

Validation loss



Normalized shipments



Learning COCPs summary

Interpretable

Satisfy constraints

Handle varying dynamics

Efficient and reliable (even division-free: OSQP)

Easy to tune from data

Future work

- Support hybrid (mixed-integer) control policies
- Integrate tuning and deployment with code generation (e.g., OSQP)
- Stochastic policies

Conclusions

Acknowledgements

Goran Banjac

Paul Goulart

Alberto Bemporad

Stephen Boyd

Akshay Agrawal

Shane Barratt

References

OSQP (osqp.org)

[OSQP: An Operator Splitting Solver for Quadratic Programs. Stellato, Banjac, Goulart, Bemporad, and Boyd. Mathematical Programming Computation 2020]

[Infeasibility detection in the alternating direction method of multipliers for convex optimization. Banjac, Goulart, Stellato, and Boyd. Journal of Optimization Theory and Applications 2019]

[Embedded code generation using the OSQP solver. Stellato, Banjac, Stellato, Moehle, Goulart, Bemporad, and Boyd. IEEE Conf. on Decision and Control 2017]

[Embedded mixed-integer quadratic optimization using the OSQP solver. Stellato, Naik, Bemporad, Goulart, and Boyd. European Control Conference, 2018]

Learning COCPs (https://github.com/cvxgrp/cocp)

[Learning Convex Optimization Control Policies. Agrawal, Amos, Barratt, Boyd, and Stellato. L4DC 2020]

[Differentiable convex optimization layers. Agrawal, Amos, Barratt, Boyd, Diamond, and Kolter. NeurIPS 2019]

[Differentiable optimization-based modeling for machine learning. Amos. PhD thesis 2019]

Conclusions

Real-time and embedded optimization

will soon **become a technology**

Thanks to

Efficient and reliable optimizers

Easy-to-tune control policies

stellato.io

bstellato@princeton.edu

@b_stellato

github.com/bstellato

Backup

OSQP Parameter selection

$$\sigma = 10^{-6}$$

$$\rho \leftarrow \rho \sqrt{\frac{\|r_{\text{prim}}\|}{\|r_{\text{dual}}\|}}$$