# Accelerating Quadratic Optimization with Reinforcement Learning

PRINCETON UNIVERSITY

ORFE

Bartolomeo Stellato — INFORMS Annual Meeting, October 24 2021

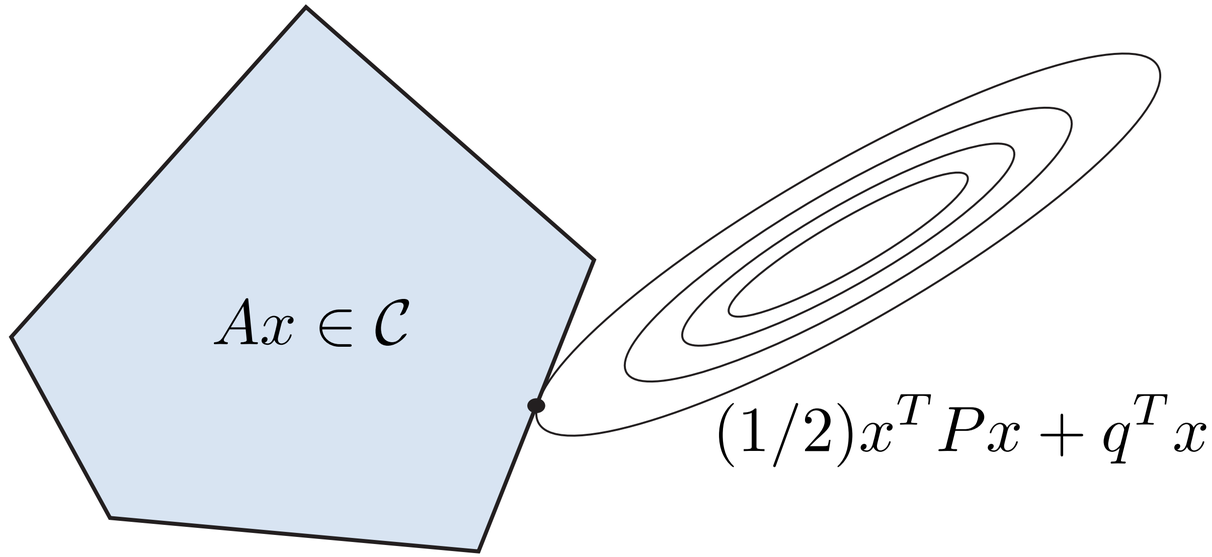

145
140
135
130
125
123.74
Tues.
142.78
Chg.
1.64
Pct. chg
Day

# The problem

minimize $(1/2)x^T Px + q^T x$

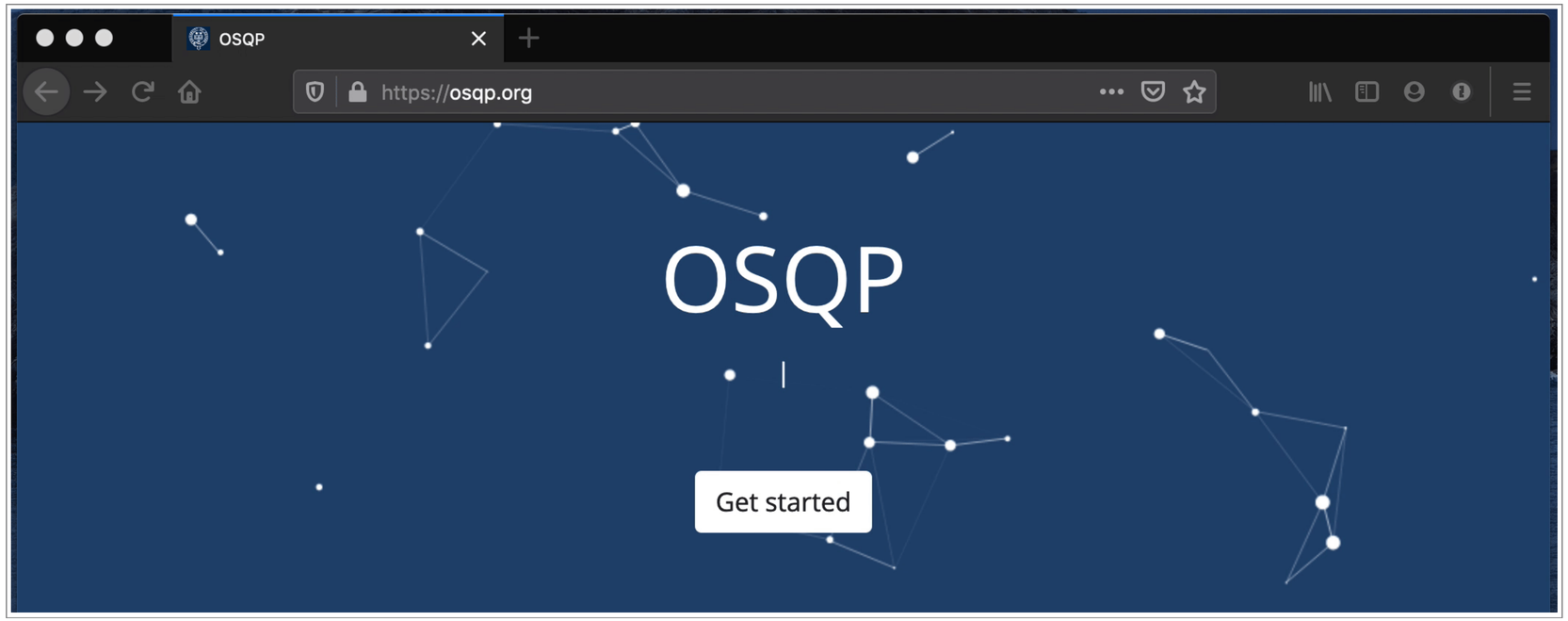
subject to $Ax \in \mathcal{C}$

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



# OSQP

## Operator Splitting solver for Quadratic Programs




Embeddable
(can be division free!)

Supports
warm-starting

Detects
infeasibility

Solves large-scale
problems

# OSQP algorithm

$$\begin{aligned} \text{minimize} \quad & (1/2)x^T Px + q^T x \\ \text{subject to} \quad & l \leq Ax \leq u \end{aligned} \longrightarrow \begin{aligned} \text{minimize} \quad & (1/2)x^T Px + q^T x \\ \text{subject to} \quad & Ax = z \\ & l \leq z \leq u \end{aligned}$$

KKT System

**Algorithm**

Solve $$\begin{bmatrix} P & A^T \\ A & -\mathbf{diag}(\rho)^{-1} \end{bmatrix} \begin{bmatrix} x^{k+1} \\ \nu^{k+1} \end{bmatrix} = \begin{bmatrix} -q \\ z^k - \mathbf{diag}(\rho)^{-1} y^k \end{bmatrix}$$

$$\begin{aligned} \tilde{z}^{k+1} &\leftarrow z^k + \mathbf{diag}(\rho)^{-1}(\nu^{k+1} - y^k) \\ z^{k+1} &\leftarrow \Pi\left(\tilde{z}^{k+1} + \mathbf{diag}(\rho)^{-1} y^k\right) \\ y^{k+1} &\leftarrow y^k + \mathbf{diag}(\rho)\left(\tilde{z}^{k+1} - z^{k+1}\right) \end{aligned}$$

**Critical step size**
$\rho = (\rho_1, \ldots, \rho_m)$

# Hand-tuned step size

minimize $(1/2)x^T Px + q^T x$

subject to $l \leq Ax \leq u$

**Tight constraints**

$l_i = (Ax^\star)_i$ or $(Ax^\star)_i = u_i$

**Vector step size**

$\rho = (\rho_1, \ldots, \rho_m)$

**Never tight**

$l_i = -\infty$ and $u_i = \infty$

↓

$\rho_i = 0$

**Otherwise**

↓

**Balance residuals**

$\rho_i^{k+1} \leftarrow \rho_i^k \sqrt{\|r_{\text{prim}}\| / \|r_{\text{dual}}\|}$

**Always tight**

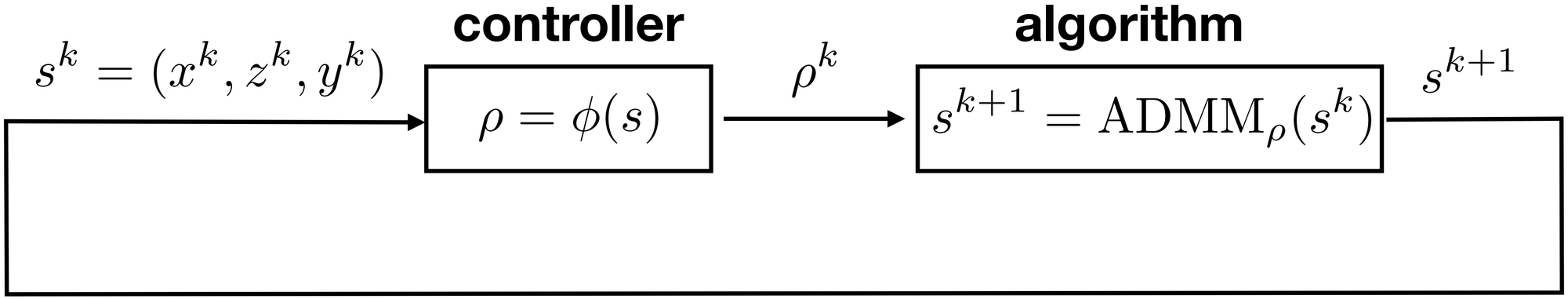
$l_i = u_i \neq \infty$

↓

$\rho_i = \infty$

**Can we learn a better update rule from data?**

# Step size choice as a control problem




**Stage cost**

$$\ell(s) = \begin{cases} 1 & \text{if not converged} \\ 0 & \text{if converged} \end{cases}$$

**Cumulative cost**

$$J = \mathbf{E} \sum_{k=1}^{\infty} \gamma^k \ell(s^k)$$

**Train with**
**Deep Policy Gradient methods (TD3)**

# Constraint-wise control policy

$$\phi(s) = \begin{bmatrix} \phi_c(s_1) \\ \phi_c(s_2) \\ \vdots \\ \phi_c(s_m) \end{bmatrix}$$

**Per-constraint update rule**

$$\rho_i = \phi_c(s_i)$$

**Per-constraint state**

$$s_i = \begin{bmatrix} \min(z_i - l_i, u_i - z_i) \\ (Ax)_i - z_i \\ y_i \end{bmatrix}$$

**slacks**

**infeasibility**

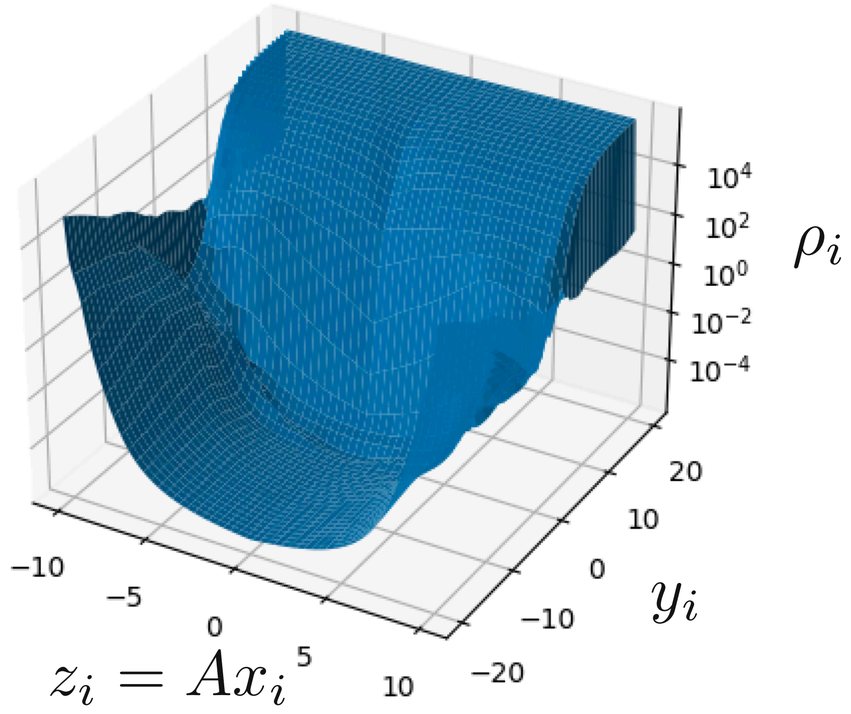
**dual variable**

Generalize to different dimensions

Low-dimensional state per constraint

Small NN policy $\phi_c(s_i)$

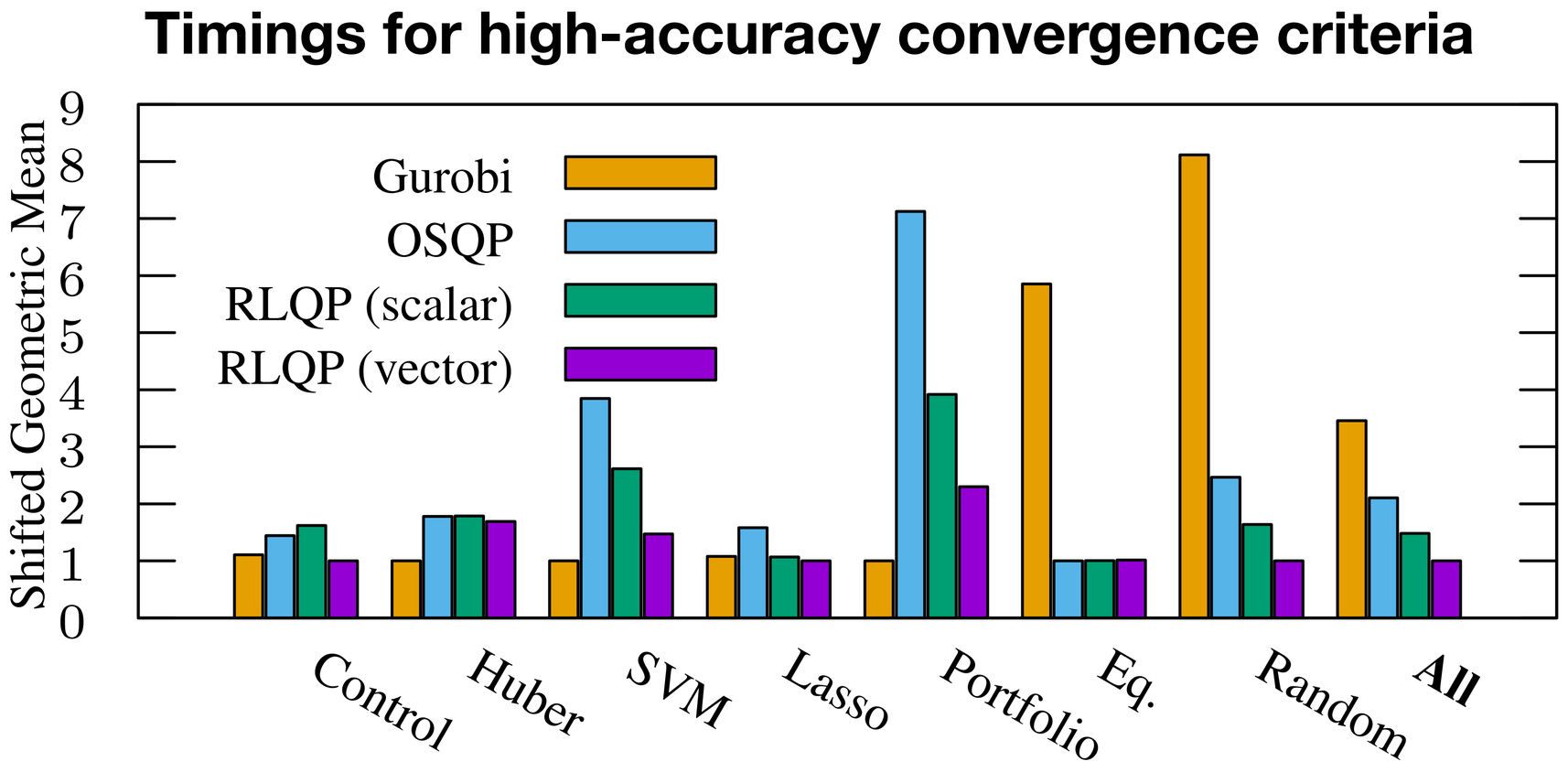
# Visualize learned policy




**Interpretable policy**

**High step size** $\rho_i$
when we reach bounds
$z_i \approx l_i$ and $z_i \approx u_i$.

# Performance with step size learning




Up to 3x faster than Gurobi

# Conclusions

**Integrate RL and ADMM to dynamically tune parameters**

- Faster convergence
- Very low overhead
- Interpretable policy


stellato.io

bstellato@princeton.edu

github.com/bstellato

@b_stellato


## References


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

[**Accelerating Quadratic Optimization with Reinforcement Learning.**
Ichnowski, Jain, Stellato, Banjac, Luo, Gonzalez, Stoica, Borrelli, and Goldberg. NeurIPS 2021]