

OSQP

An Operator Splitting Solver for Quadratic Programs

Bartolomeo Stellato

joint work with Goran Banjac,

Nicholas Moehle, Paul Goulart, Alberto Bemporad, Stephen Boyd

Cambridge Control Group Seminar, 29 Jun 2017

Why quadratic programming?

AN ALGORITHM FOR QUADRATIC PROGRAMMING

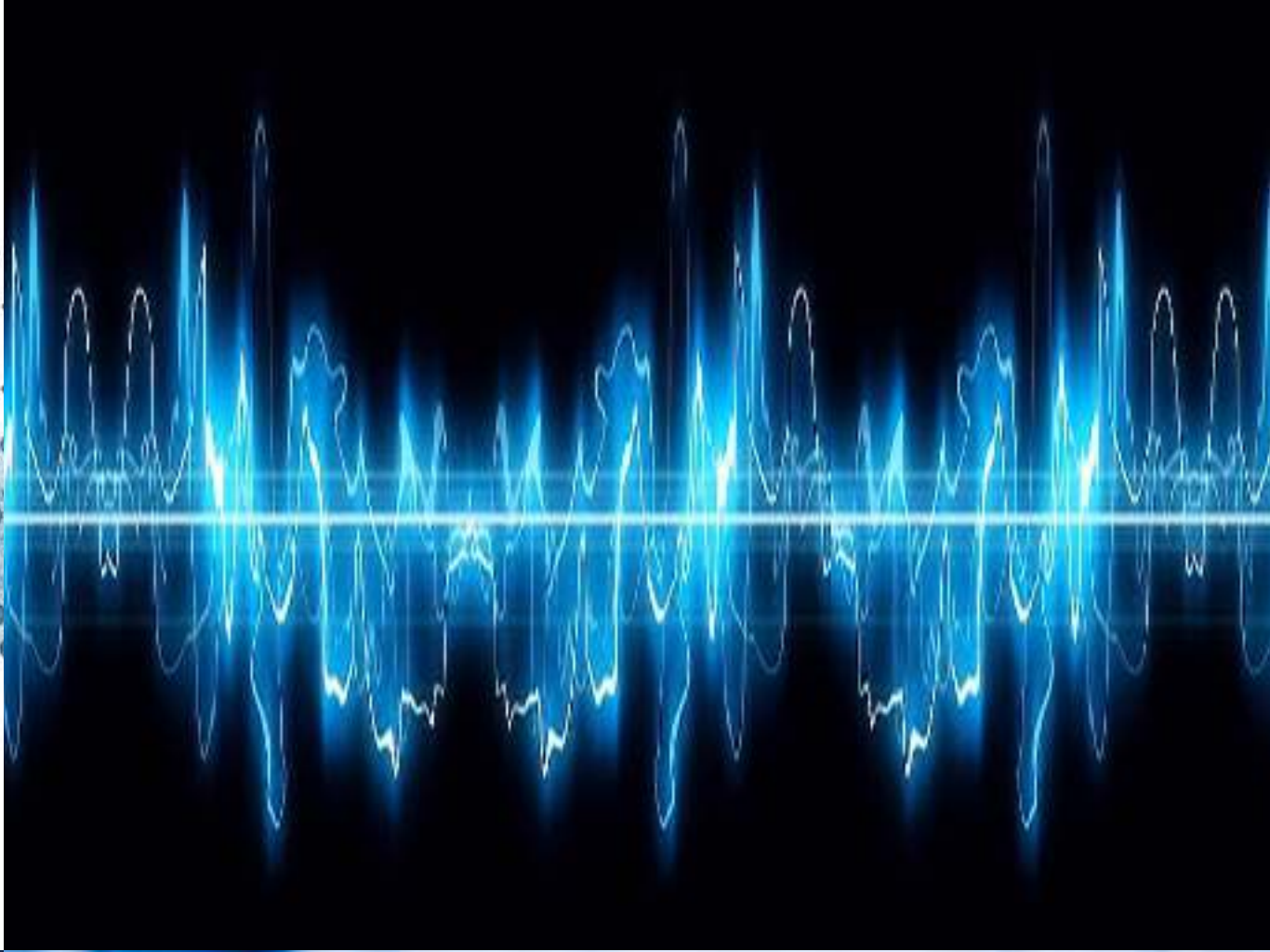
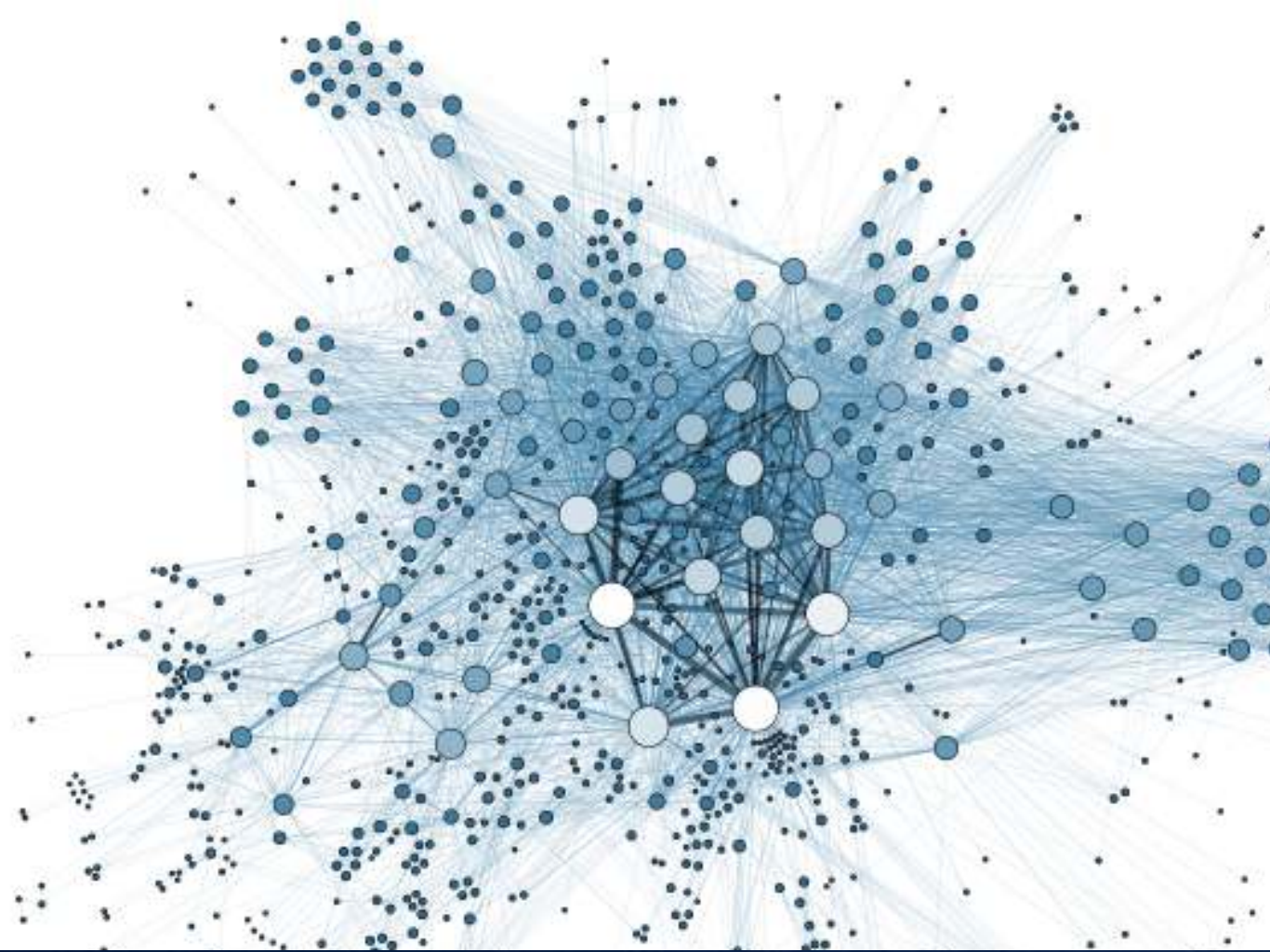
Marguerite Frank and Philip Wolfe¹
Princeton University

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!



First-order methods

Pros

Warm starting

Handle large-scale problems

Embeddable

Cons

Low accuracy solutions

Can't detect infeasibility

Problem data dependent

General Purpose Solver

Based on first-order
methods

Robust

Accurate

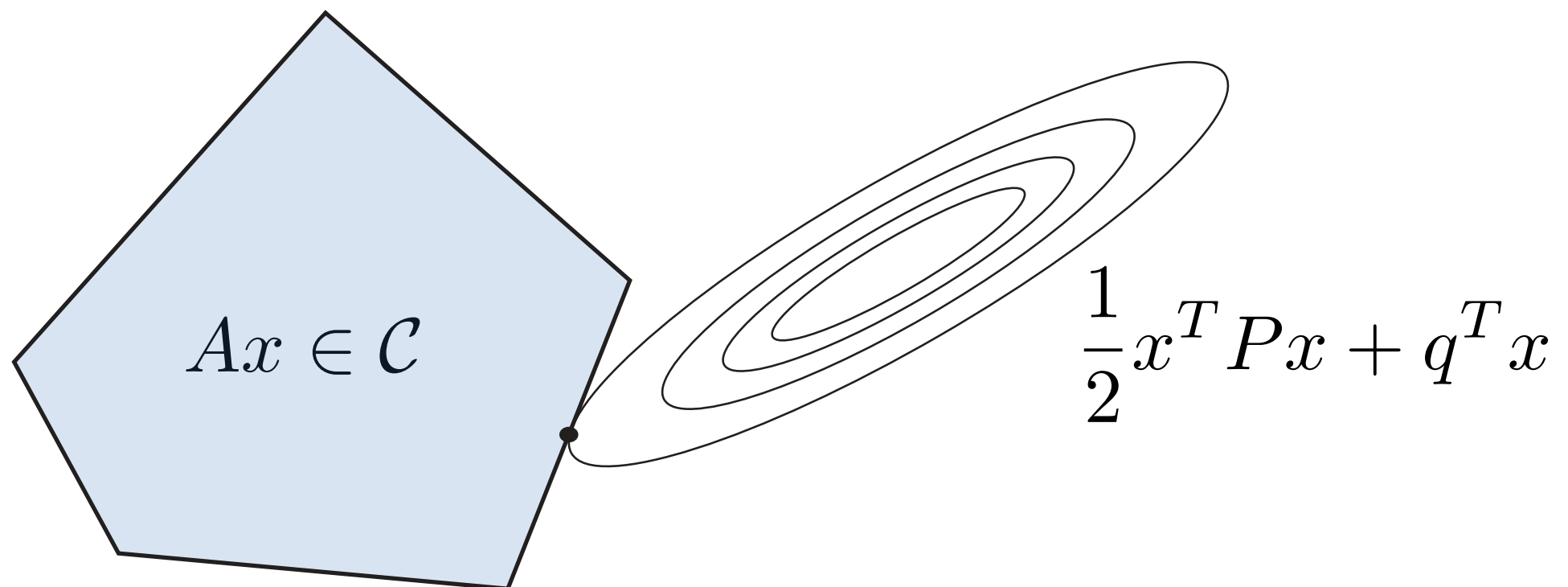
Detects
Infeasibility

The OSQP Solver

The problem

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

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



ADMM

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

ADMM

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

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

$$2 \quad x^{k+1} \leftarrow \operatorname{argmin}_x \left(g(x) + \frac{\rho}{2} \left\| x - \tilde{x}^{k+1} - \frac{y^k}{\rho} \right\|^2 \right)$$

$$3 \quad y^{k+1} \leftarrow y^k + \rho (\tilde{x}^{k+1} - x^{k+1})$$

How to split the QP?

$$\begin{array}{ll}\text{minimize} & \frac{1}{2}x^T Px + q^T x \\ \text{subject to} & Ax = z \\ & z \in \mathcal{C}\end{array}$$

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

How to split the QP?

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T Px + q^T x \\ \text{subject to} & Ax = z \\ & z \in \mathcal{C} \end{array} \quad \Bigg\} f$$

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

How to split the QP?

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T Px + q^T x \\ \text{subject to} & Ax = z \\ & z \in \mathcal{C} \end{array} \quad \left. \begin{array}{l} \} f \\ \} g \end{array} \right.$$

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

ADMM iterations

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

$$2 \quad z^{k+1} \leftarrow \Pi \left(\tilde{z}^{k+1} + \frac{y^k}{\rho} \right)$$

$$3 \quad y^{k+1} \leftarrow y^k + \rho (\tilde{z}^{k+1} - z^{k+1})$$

ADMM iterations

Inner QP

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

2 $z^{k+1} \leftarrow \Pi \left(\tilde{z}^{k+1} + \frac{y^k}{\rho} \right)$

3 $y^{k+1} \leftarrow y^k + \rho (\tilde{z}^{k+1} - z^{k+1})$

ADMM iterations

Inner QP

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

2 $z^{k+1} \leftarrow \Pi \left(\tilde{z}^{k+1} + \frac{y^k}{\rho} \right)$ Projection onto $\mathcal{C}$

3 $y^{k+1} \leftarrow y^k + \rho (\tilde{z}^{k+1} - z^{k+1})$

Solving the inner QP

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

Reduced KKT system

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

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

Solving the inner QP

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

Reduced KKT system

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

Always
solvable!

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

Solving the linear system

Direct Method

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

Solving the linear system

Direct Method

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

Solving the linear system

Direct Method

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

Solving the linear system

Indirect Method

$$(P + \sigma I + \rho A^T A) x = \sigma x^k - q + A^T (\rho z^k - y^k)$$

Solving the linear system

Indirect Method

Positive definite
matrix

$$(P + \sigma I + \rho A^T A) x = \sigma x^k - q + A^T (\rho z^k - y^k)$$

Conjugate
gradient

Solving the linear system

Indirect Method

Positive definite
matrix

$$(P + \sigma I + \rho A^T A) x = \sigma x^k - q + A^T (\rho z^k - y^k)$$

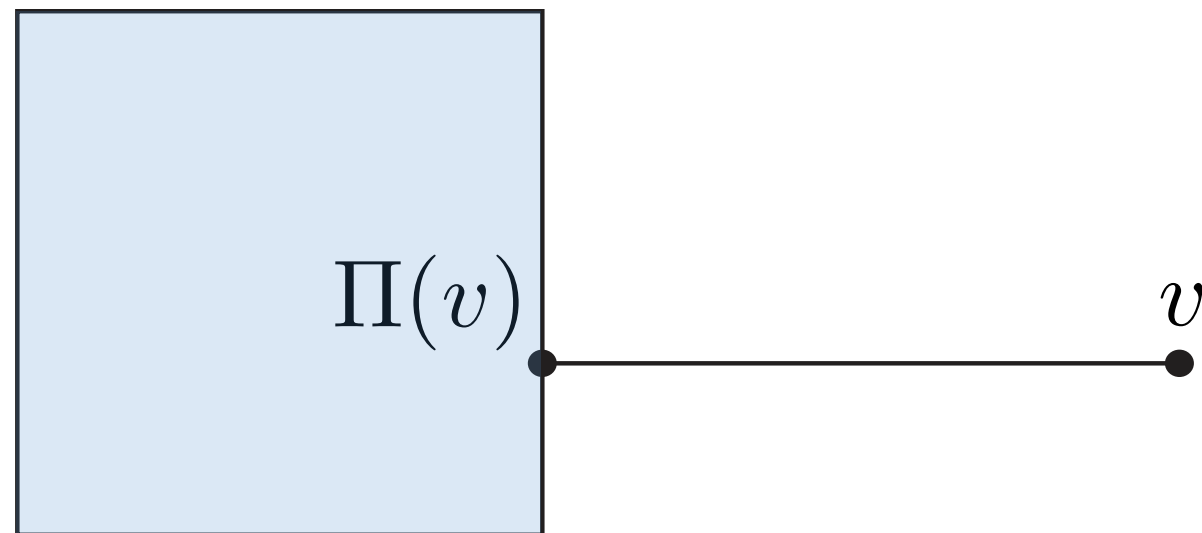
Conjugate
gradient

Solve very
large
systems

Computing the projection

Box projection

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



Final algorithm

Problem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & l \leq A x \leq u \end{array}$$

Algorithm

$$\begin{array}{l} \mathbf{1} \left\{ \begin{array}{l} (x^{k+1}, \nu^{k+1}) \leftarrow \text{solve} \begin{bmatrix} P + \sigma I & A^T \\ A & -\frac{1}{\rho} I \end{bmatrix} \begin{bmatrix} x^{k+1} \\ \nu^{k+1} \end{bmatrix} = \begin{bmatrix} \sigma x^k - q \\ z^k - \frac{1}{\rho} y^k \end{bmatrix} \\ \tilde{z}^{k+1} \leftarrow z^k + \frac{1}{\rho} (\nu^{k+1} - y^k) \end{array} \right. \\ \mathbf{2} \left\{ z^{k+1} \leftarrow \Pi \left(\tilde{z}^{k+1} + \frac{1}{\rho} y^k \right) \right. \\ \mathbf{3} \left\{ y^{k+1} \leftarrow y^k + \rho (\tilde{z}^{k+1} - z^{k+1}) \right. \end{array}$$

Final algorithm

Problem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & l \leq A x \leq u \end{array}$$

Algorithm

Linear system
solve

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

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

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

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

Final algorithm

Problem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & l \leq A x \leq u \end{array}$$

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

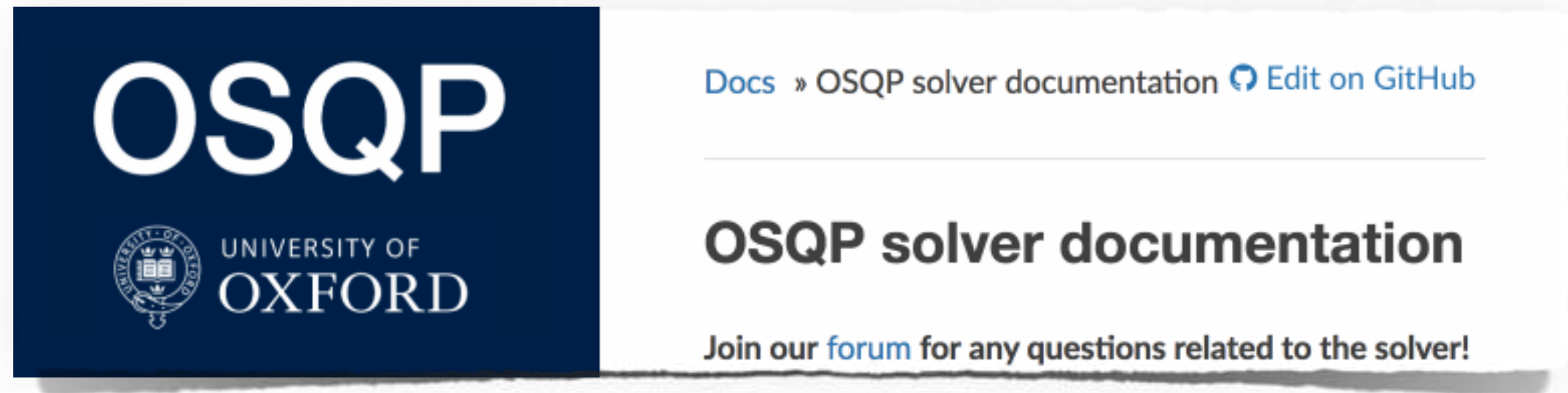
$$\tilde{z}^{k+1} \leftarrow z^k + \frac{1}{\rho} (\nu^{k+1} - y^k)$$

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

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

OSQP

`osqp.readthedocs.io`



Library
free

Multiple
interfaces

Embeddable

Interfaces

Languages



Parsers

JuMP

CVXPY

YALMIP

OSQP interface



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

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

# Solve
results = m.solve()

# Update cost with q_new
m.update(q=q_new)

# Solve again
results_new = m.solve()
```



```
% Create OSQP object
m = osqp();

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

% Solve
results = m.solve();

% Update cost with q_new
m.update('q', q_new);

% Solve again
results_new = m.solve();
```

Numerical Example

Lasso

$$\text{minimize} \quad \|Ax - b\|_2^2 + \lambda \|x\|_1$$

Features
 n

Data points
 $m = 100n$

Lasso

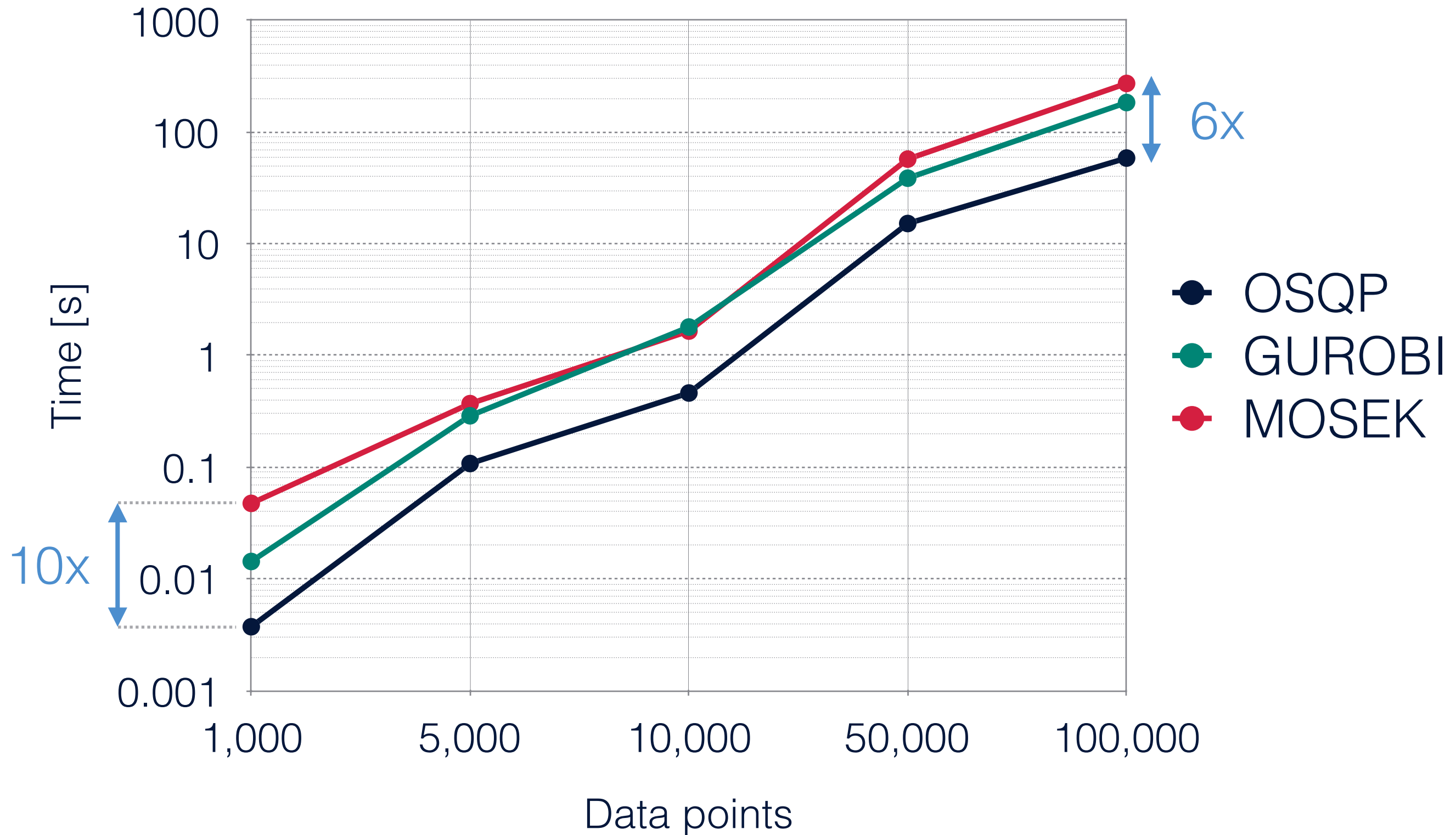
Weighting
parameter

minimize $\|Ax - b\|_2^2 + \lambda \|x\|_1$

Features
 n

Data points
 $m = 100n$

Lasso timings



Solution Polishing

Obtaining the active set

Lower active

$$\mathcal{L} = \{i \mid (Ax)_i = l_i \wedge y_i < 0\}$$

Upper active

$$\mathcal{U} = \{i \mid (Ax)_i = u_i \wedge y_i > 0\}$$

Solving single linear system

$$\begin{bmatrix} P & A_{\mathcal{L}}^T & A_{\mathcal{U}}^T \\ A_{\mathcal{L}} & & \\ A_{\mathcal{U}} & & \end{bmatrix} \begin{bmatrix} x \\ y_{\mathcal{L}} \\ y_{\mathcal{U}} \end{bmatrix} = \begin{bmatrix} -q \\ l_{\mathcal{L}} \\ u_{\mathcal{U}} \end{bmatrix}$$

Inactive constraints

$$y_i = 0 \quad i \notin (\mathcal{L} \cup \mathcal{U})$$

Solving single linear system

$$\underbrace{\begin{bmatrix} P & A_{\mathcal{L}}^T & A_{\mathcal{U}}^T \\ A_{\mathcal{L}} & & \\ A_{\mathcal{U}} & & \end{bmatrix}}_M \begin{bmatrix} x \\ y_{\mathcal{L}} \\ y_{\mathcal{U}} \end{bmatrix} = \underbrace{\begin{bmatrix} -q \\ l_{\mathcal{L}} \\ u_{\mathcal{U}} \end{bmatrix}}_g$$

Inactive constraints

$$y_i = 0 \quad i \notin (\mathcal{L} \cup \mathcal{U})$$

Iterative refinement

Perturbed system

$$\begin{bmatrix} P + \delta I & A_{\mathcal{L}}^T & A_{\mathcal{U}}^T \\ A_{\mathcal{L}} & -\delta I & \\ A_{\mathcal{U}} & & -\delta I \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{y}_{\mathcal{L}} \\ \hat{y}_{\mathcal{U}} \end{bmatrix} = \begin{bmatrix} -q \\ l_{\mathcal{L}} \\ u_{\mathcal{U}} \end{bmatrix}$$

Iterative refinement

Perturbed system

Quasi-definite



$$\begin{bmatrix} P + \delta I & A_{\mathcal{L}}^T & A_{\mathcal{U}}^T \\ A_{\mathcal{L}} & -\delta I & \\ A_{\mathcal{U}} & & -\delta I \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{y}_{\mathcal{L}} \\ \hat{y}_{\mathcal{U}} \end{bmatrix} = \begin{bmatrix} -q \\ l_{\mathcal{L}} \\ u_{\mathcal{U}} \end{bmatrix}$$

Iterative refinement

Quasi-definite $\longrightarrow$

Perturbed system

$$\underbrace{\begin{bmatrix} P + \delta I & A_{\mathcal{L}}^T & A_{\mathcal{U}}^T \\ A_{\mathcal{L}} & -\delta I & \\ A_{\mathcal{U}} & & -\delta I \end{bmatrix}}_{M + \Delta} \begin{bmatrix} \hat{x} \\ \hat{y}_{\mathcal{L}} \\ \hat{y}_{\mathcal{U}} \end{bmatrix} = \underbrace{\begin{bmatrix} -q \\ l_{\mathcal{L}} \\ u_{\mathcal{U}} \end{bmatrix}}_g$$

Iterative refinement

Quasi-definite $\longrightarrow$ Perturbed system

$$\underbrace{\begin{bmatrix} P + \delta I & A_{\mathcal{L}}^T & A_{\mathcal{U}}^T \\ A_{\mathcal{L}} & -\delta I & \\ A_{\mathcal{U}} & & -\delta I \end{bmatrix}}_{M + \Delta} \begin{bmatrix} \hat{x} \\ \hat{y}_{\mathcal{L}} \\ \hat{y}_{\mathcal{U}} \end{bmatrix} = \underbrace{\begin{bmatrix} -q \\ l_{\mathcal{L}} \\ u_{\mathcal{U}} \end{bmatrix}}_g$$

Iterations

$$\begin{aligned} \delta t^k &\leftarrow \text{solve } (M + \Delta)\delta t^k = g - Kt^k \\ t^{k+1} &\leftarrow t^k + \delta t^k \end{aligned}$$

Infeasibility detection

What happens if the problem is infeasible?

$$y^{k+1} \leftarrow y^k + \rho \underbrace{(\tilde{z}^{k+1} - z^{k+1})}_{\neq 0}$$

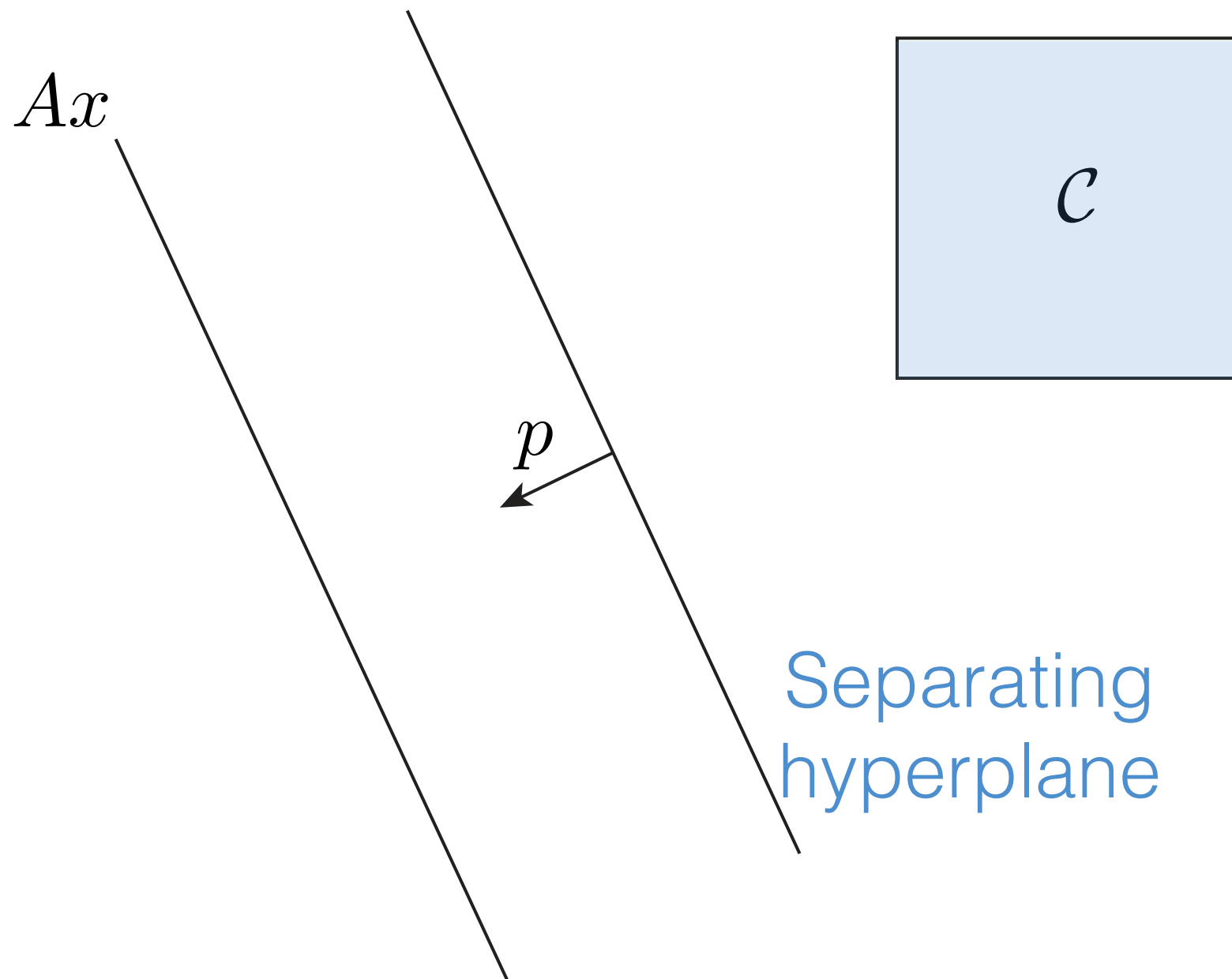
$\downarrow$

y^k diverge!

Farkas' Lemma

Primal infeasibility

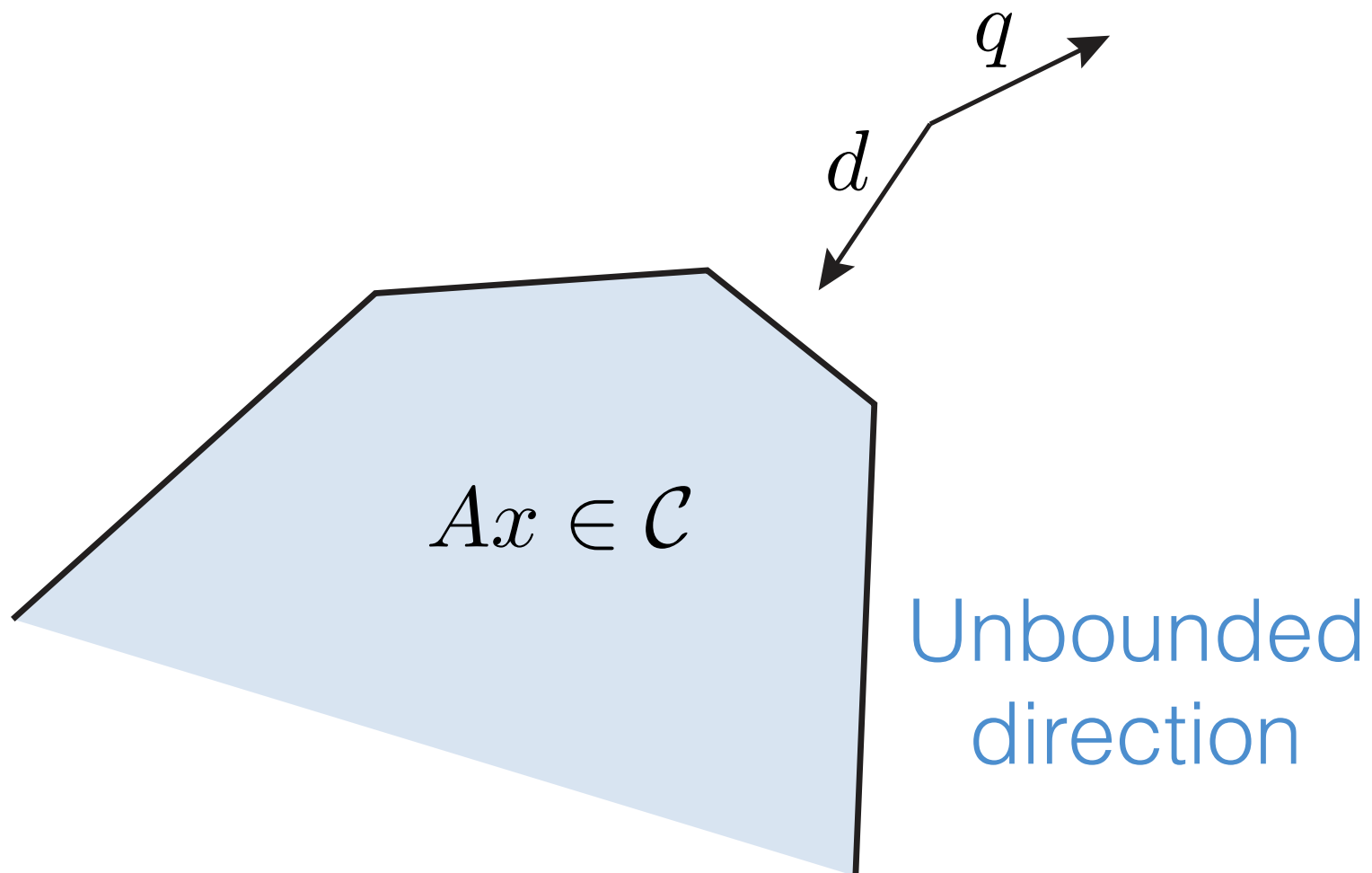
$$A^T p = 0 \quad u^T p_+ + l^T p_- < 0$$



Farkas' Lemma

Dual infeasibility

$$Pd = 0 \quad q^T d < 0 \quad (Ad)_i \begin{cases} = 0 & l_i, u_i \neq \infty \\ \geq 0 & u_i = +\infty \\ \leq 0 & l_i = -\infty \end{cases}$$

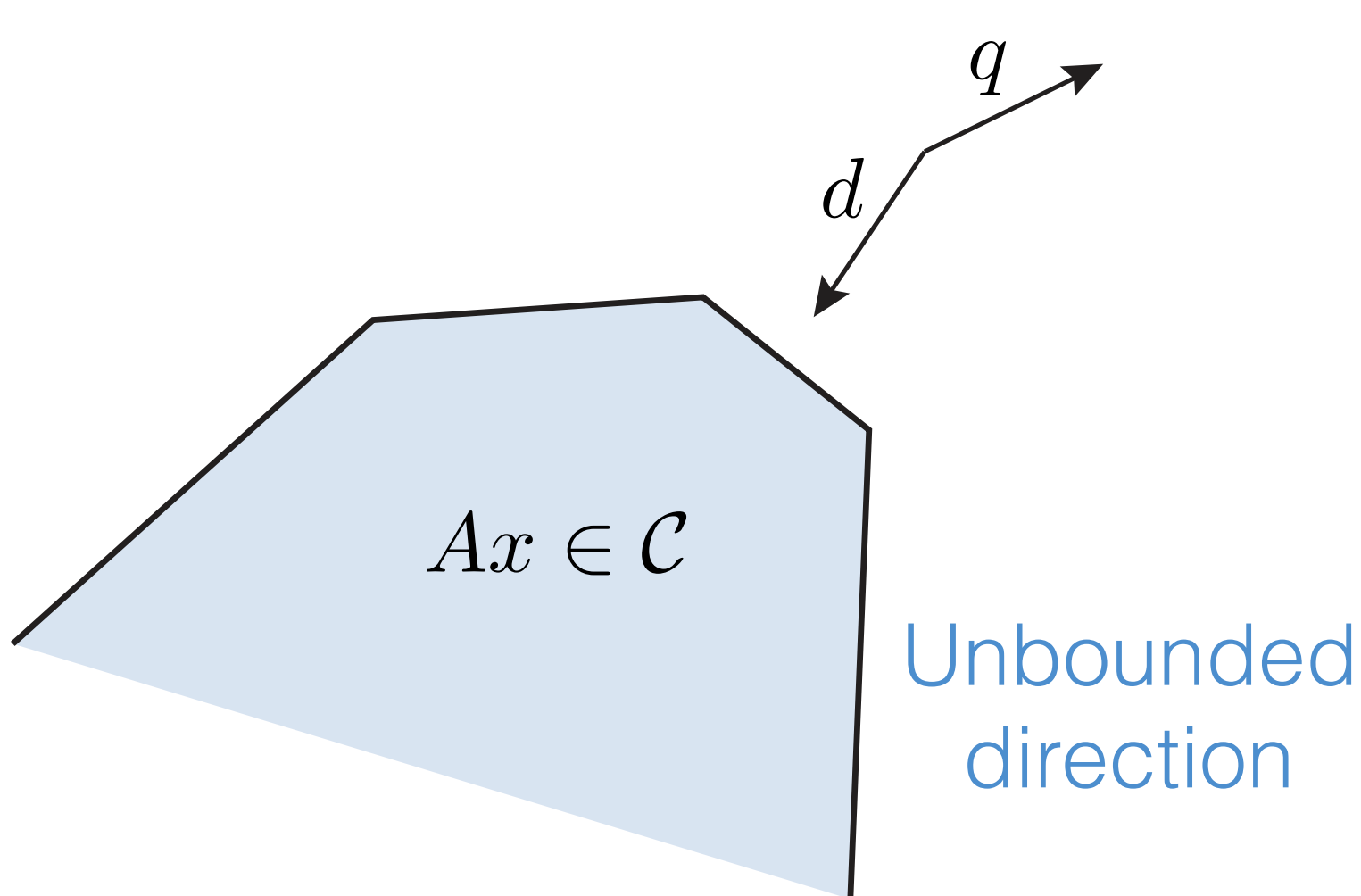


Farkas' Lemma

Dual infeasibility

$$Pd = 0 \quad q^T d < 0$$

$$(Ad)_i \begin{cases} = 0 & l_i, u_i \neq \infty \\ \geq 0 & u_i = +\infty \\ \leq 0 & l_i = -\infty \end{cases}$$



$Ad \in \mathcal{C}^\infty$ Recession cone

OSQP Algorithm

Reformulation

$$(x^{k+1}, \tilde{z}^k) \leftarrow \operatorname{argmin}_{(x,z): Ax=z} \frac{1}{2} x^T P x + q^T x + \frac{\sigma}{2} \|x - x^k\|^2 + \frac{\rho}{2} \|z - (2\Pi - I)(v^k)\|^2$$

$$v^{k+1} = \tilde{z}^k + (I - \Pi)(v^k)$$

Averaged non-expansive operator

$$(x^{k+1}, v^{k+1}) = T(x^k, v^k)$$

Original variables

$$z^k = \Pi(v^k) \qquad y^k = \rho(I - \Pi)(v^k)$$

Asymptotic behavior

Averaged non-expansive operator

$$(x^{k+1}, v^{k+1}) = T(x^k, v^k)$$

Differences

$$\delta x^k = x^k - x^{k-1} \quad \delta v^k = v^k - v^{k-1}$$

Convergence to smallest vector

$$\lim_{k \rightarrow \infty} (\delta x^k, \delta v^k) = (\delta x, \delta v) \in \operatorname{argmin}_{\overline{\operatorname{ran}}(T-I)} \|(\delta x, \delta v)\|$$

[A. Pazy, 1971]

Auxiliary results

Difference of dual iterates: δy

$$A^T \delta y = 0 \qquad u^T \delta y_+ + l^T \delta y_- = -\frac{1}{\rho} \|\delta y\|^2$$

Difference of primal iterates: δx

$$P \delta x = 0 \qquad q^T \delta x = -\sigma \|\delta x\|^2 - \rho \|A \delta x\|^2 \qquad A \delta x \in C^\infty$$

Infeasibility detection

Main result

Primal infeasibility: $\delta y \neq 0$

$$A^T \delta y = 0 \quad u^T \delta y_+ + l^T \delta y_- < 0$$

Dual infeasibility: $\delta x \neq 0$

$$P \delta x = 0 \quad q^T \delta x < 0 \quad (A \delta x)_i \begin{cases} = 0 & l_i, u_i \neq \infty \\ \geq 0 & u_i = +\infty \\ \leq 0 & l_i = -\infty \end{cases}$$

Example

Simple QP

$$\begin{array}{ll}\text{minimize} & \frac{1}{2}x^T \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} x \\ \text{subject to} & 1 \leq \begin{bmatrix} 1 & 0 \end{bmatrix} x \leq 2\end{array}$$

Optimal solution

$$x = (1, 0)$$

Example

Primal infeasible

$$\text{minimize} \quad \frac{1}{2} x^T \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} x$$

$$\text{subject to} \quad \begin{bmatrix} 1 \\ 1 \end{bmatrix} \leq \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix} \leq \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

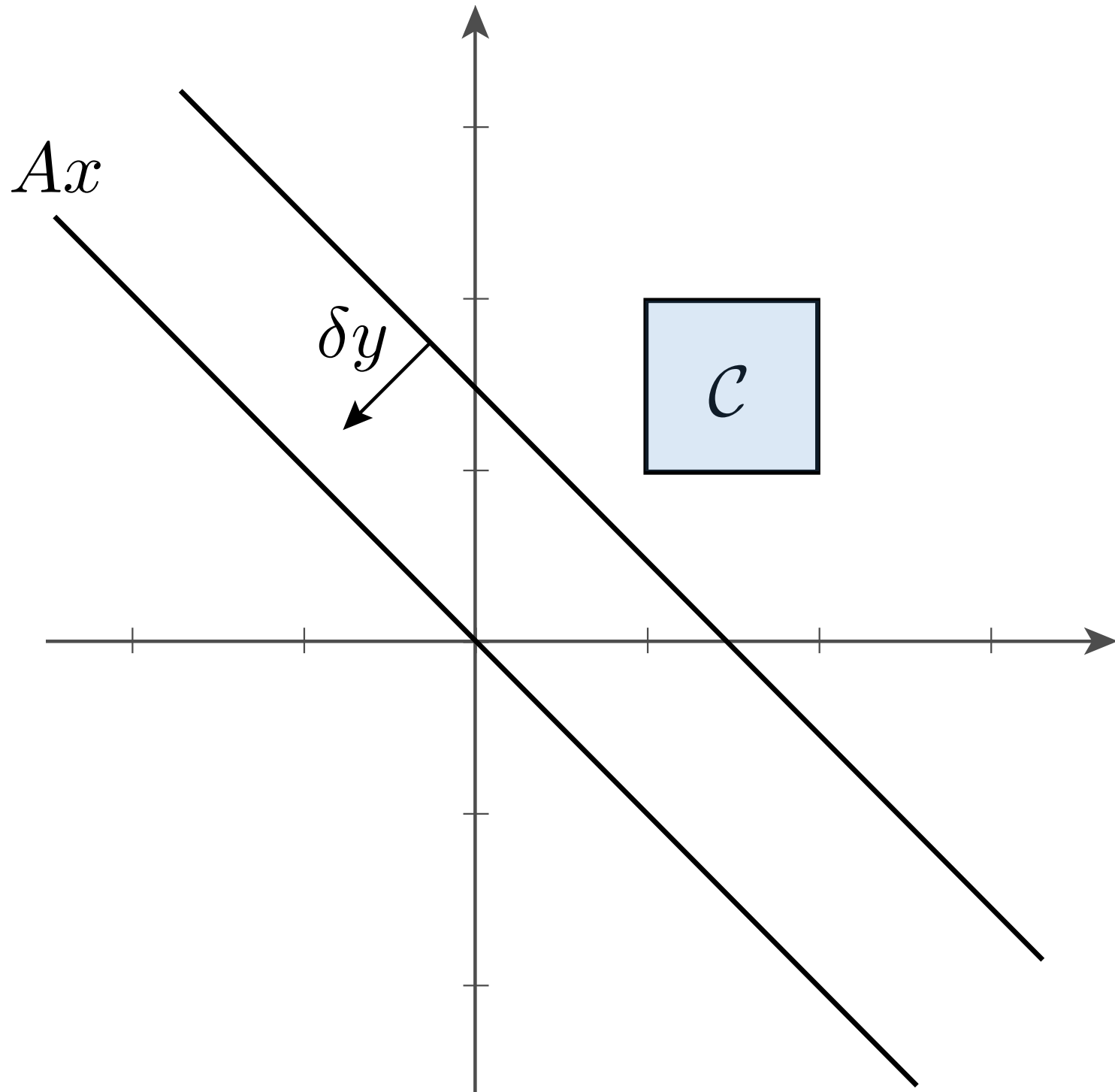
Conflicting
constraint

Certificate

$$\delta y = (-1, -1)$$

Example

Primal infeasible



Certificate

$$\delta y = (-1, -1)$$

Example

Dual infeasible

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 & 1 \end{bmatrix} x \\ \text{subject to} & 1 \leq \begin{bmatrix} 1 & 0 \end{bmatrix} x \leq 2 \end{array}$$

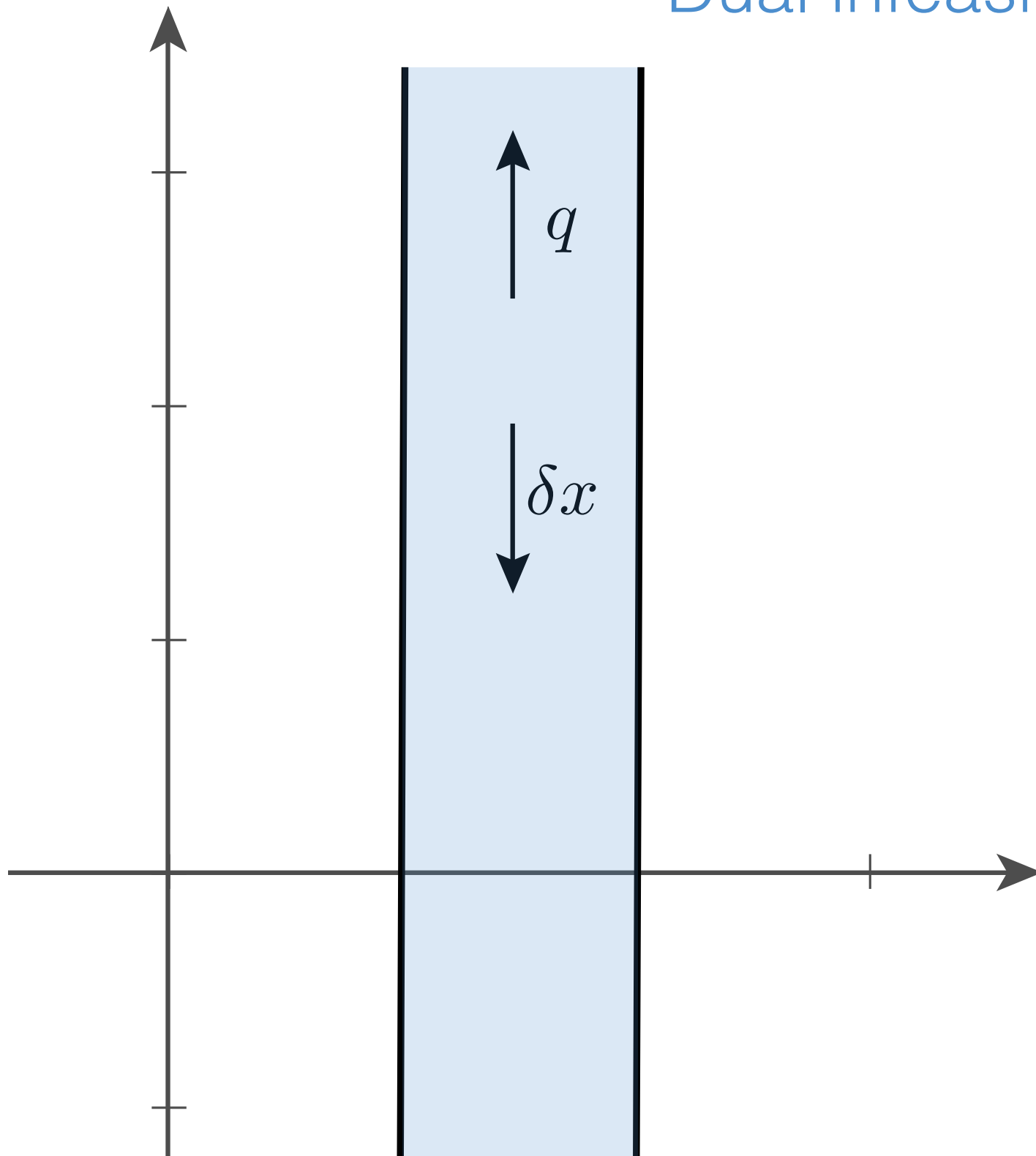
Unbounded
direction

Certificate

$$\delta x = (0, -1)$$

Example

Dual infeasible



Certificate

$$\delta x = (0, -1)$$

Example

Primal and dual infeasible

minimize $\frac{1}{2}x^T \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 & 1 \end{bmatrix} x$ Unbounded direction

subject to $\begin{bmatrix} 1 \\ 1 \end{bmatrix} \leq \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix} \leq \begin{bmatrix} 2 \\ 2 \end{bmatrix}$ Conflicting constraint

Certificates

$$\delta x = (0, -1)$$

$$\delta y = (-1, -1)$$

Extensions

Code generation

Optimized
C code

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

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

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



```
// Main ADMM algorithm
for (iter = 1; iter <= work->settings->max_iter; iter++) {
    // Update x and z
    swap_vectors(&work->x, &work->x_prev);
    swap_vectors(&work->z, &work->z_prev);

    // Main ADMM algorithm
    for (iter = 1; iter <= work->settings->max_iter; iter++) {
        // Update x and z
        swap_vectors(&work->x, &work->x_prev);
        swap_vectors(&work->z, &work->z_prev);

        // ADMM STEPS
        // Compute t_tilde(x)^{k+1}, t_tilde(z)^{k+1}
        update_xz_tilde(work);

        // Compute x^{k+1}
        update_x(work);

        // Compute z^{k+1}
        update_z(work);

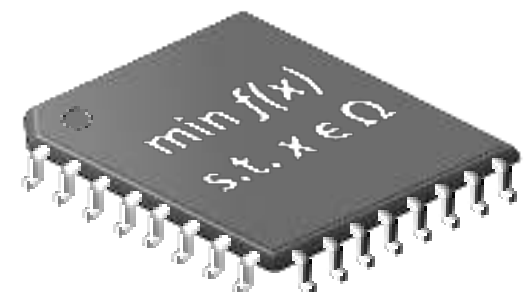
        // Compute y^{k+1}
        update_y(work);

        // End of ADMM Steps
    }

    // Check the interrupt signal
    if (isInterrupted()) {
        update_status(work->info, OSQP_SIGINT);
        c_print("Solver interrupted\n");
        endInterruptListener();
        return 1; // exitflag
    }
}
```

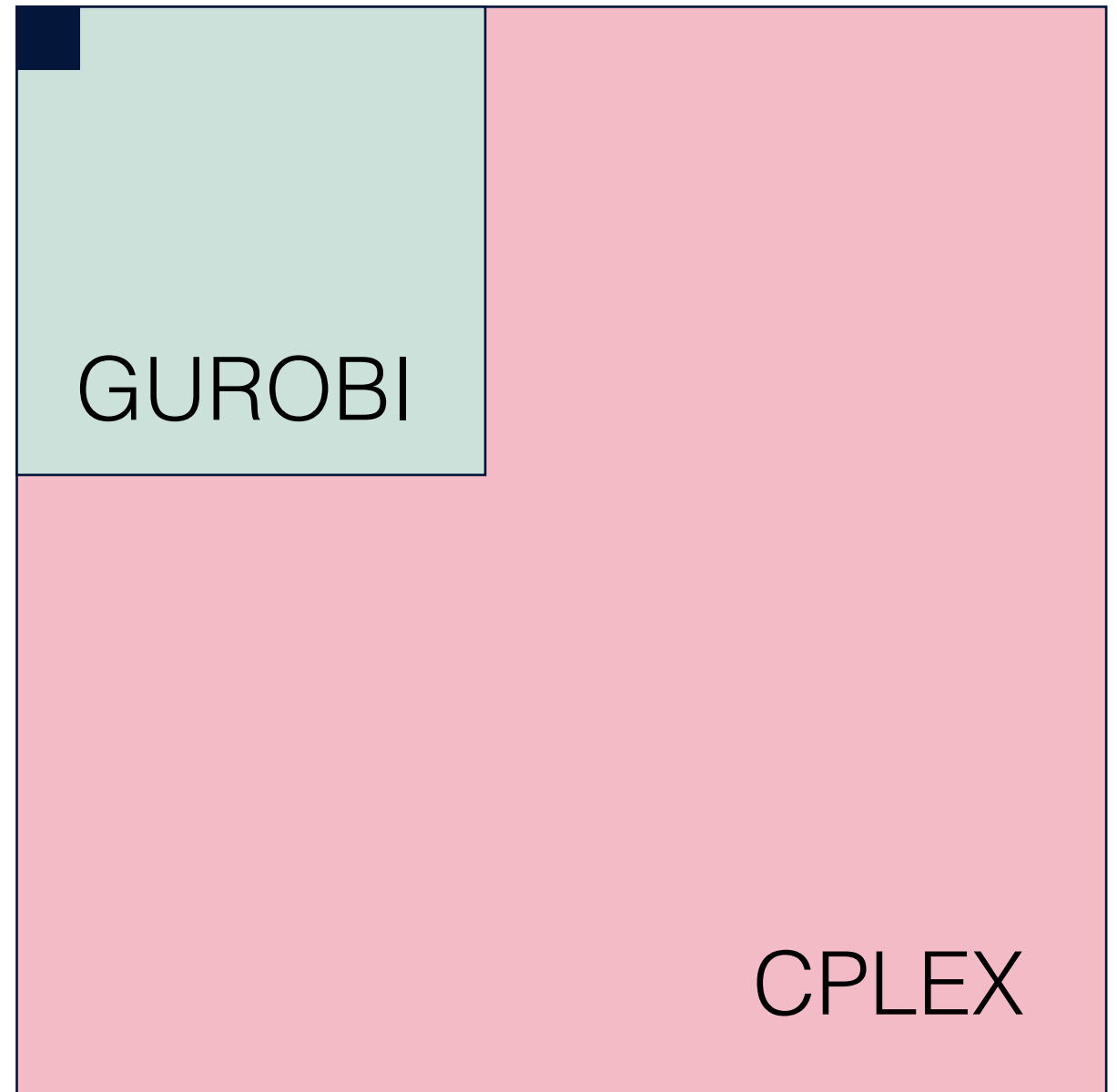


Embedded
hardware



Compiled code size ~80kb

OSQP



300x
Reduction!

Mixed-Integer Quadratic Programs

$$\begin{array}{ll}\text{minimize} & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & l \leq Ax \leq u \\ & x_i \in \mathbf{Z} \quad \forall i \in \mathbf{I}\end{array}$$

Mixed-Integer Quadratic Programs

$$\begin{array}{ll}\text{minimize} & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & l \leq A x \leq u \\ & x_i \in \mathbf{Z} \quad \forall i \in \mathbf{I}\end{array}$$

Integer
constraints

Mixed-Integer Quadratic Programs

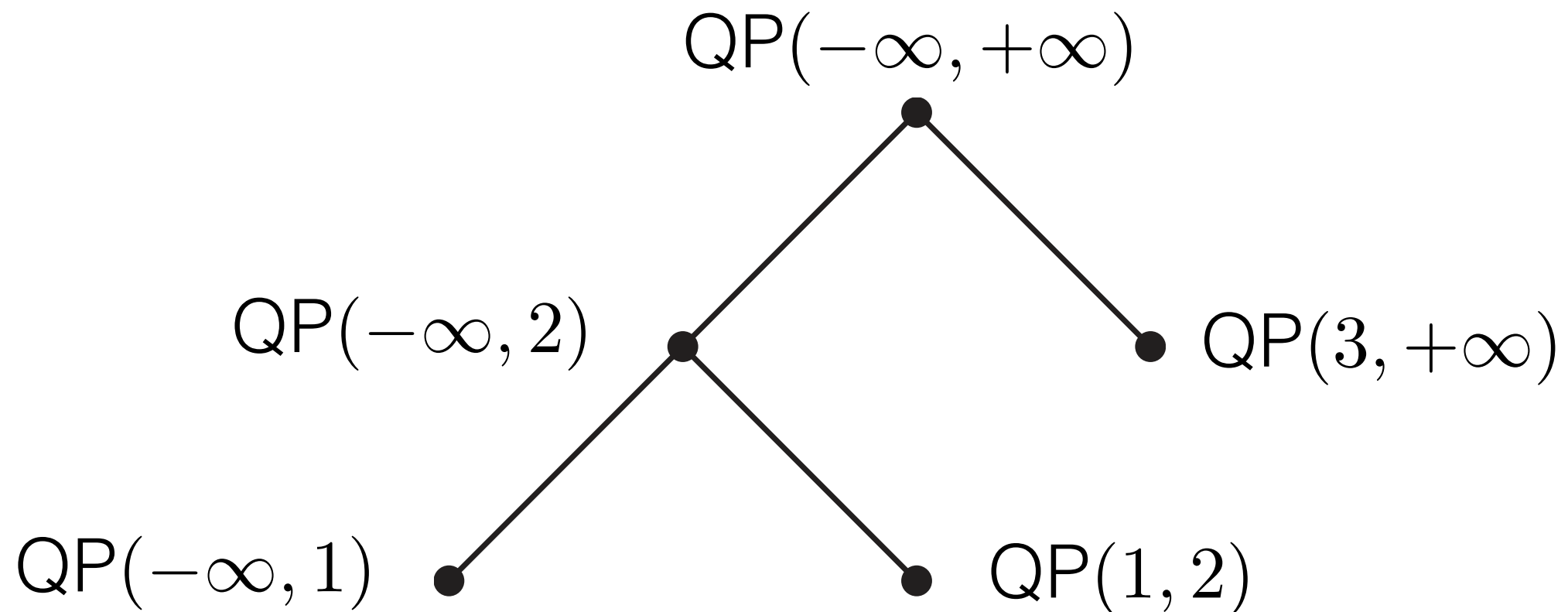
$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & l \leq A x \leq u \\ & x_i \in \mathbf{Z} \quad \forall i \in \mathbf{I} \end{array} \quad f(x)$$

Integer
constraints

Mixed-Integer Quadratic Programs

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x \quad f(x) \\ \text{subject to} & l \leq Ax \leq u \\ & x_i \in \mathbf{Z} \quad \forall i \in \mathbf{I} \end{array}$$

Integer
constraints



Inner QPs

$$\text{QP}(\underline{x}, \bar{x})$$

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & \underline{l} \leq A x \leq u \\ & \underline{x}_i \leq x_i \leq \bar{x}_i \quad \forall i \in \mathbf{I} \end{array}$$

Inner QPs

QP($\underline{x}, \bar{x}$)

minimize $\frac{1}{2}x^T P x + q^T x$
subject to $l \leq Ax \leq u$

$$\underline{x}_i \leq x_i \leq \bar{x}_i \quad \forall i \in \mathbf{I}$$

Changing
bounds



Reusing
factorization

Branch-and-bound

```
initialize  $U \leftarrow \infty, \mathcal{H} \leftarrow \text{QP}(-\infty, \infty)$   
while  $\mathcal{H} \neq \emptyset$  do  
     $\tilde{x}, f(\tilde{x}) \leftarrow$  pick  $\text{QP}(\underline{x}, \overline{x})$  from  $\mathcal{H}$  and solve  
    if  $\text{QP}(\underline{x}, \overline{x})$  is infeasible or  $f(\tilde{x}) > U$  then  
        prune  
    else if  $\tilde{x}$  is integer feasible then  
         $U \leftarrow f(\tilde{x}), x^* \leftarrow \tilde{x}$ , prune  
    else  
        branch  
    end if  
end while
```

Branch-and-bound

initialize $U \leftarrow \infty$, $\mathcal{H} \leftarrow \text{QP}(-\infty, \infty)$

while $\mathcal{H} \neq \emptyset$ do

$\tilde{x}, f(\tilde{x}) \leftarrow$ pick $\text{QP}(\underline{x}, \bar{x})$ from $\mathcal{H}$ and solve

if $\text{QP}(\underline{x}, \bar{x})$ is infeasible or $f(\tilde{x}) > U$ then

prune

else if $\tilde{x}$ is integer feasible then

$U \leftarrow f(\tilde{x})$, $x^* \leftarrow \tilde{x}$, prune

else

branch

end if

end while

Bad node

Branch-and-bound

initialize $U \leftarrow \infty$, $\mathcal{H} \leftarrow \text{QP}(-\infty, \infty)$

while $\mathcal{H} \neq \emptyset$ do

$\tilde{x}, f(\tilde{x}) \leftarrow$ pick $\text{QP}(\underline{x}, \bar{x})$ from $\mathcal{H}$ and solve

if $\text{QP}(\underline{x}, \bar{x})$ is infeasible or $f(\tilde{x}) > U$ then
prune

Bad node

else if $\tilde{x}$ is integer feasible then

$U \leftarrow f(\tilde{x})$, $x^* \leftarrow \tilde{x}$, prune

New solution

else

branch

end if

end while

Branch-and-bound

initialize $U \leftarrow \infty$, $\mathcal{H} \leftarrow \text{QP}(-\infty, \infty)$

while $\mathcal{H} \neq \emptyset$ do

$\tilde{x}, f(\tilde{x}) \leftarrow$ pick $\text{QP}(\underline{x}, \bar{x})$ from $\mathcal{H}$ and solve

if $\text{QP}(\underline{x}, \bar{x})$ is infeasible or $f(\tilde{x}) > U$ then
prune

Bad node

else if $\tilde{x}$ is integer feasible then

$U \leftarrow f(\tilde{x})$, $x^* \leftarrow \tilde{x}$, prune

New solution

else

branch

Continue

end if

end while

Saving computations

Factorization
caching

+

Warm
starting

ADMM

$\text{QP}(\underline{x}, \bar{x})$

Repeated
MIQPs

Numerical example

Hybrid Model Predictive Control

$$\begin{aligned} &\text{minimize} && \sum_{k=0}^{N-1} \gamma^k l(x(k)) + \gamma^N V(x(N)) \\ &\text{subject to} && x(k+1) = Ax(k) + Bu(k) \\ &&& x(0) = x_0 \\ &&& x(k) \in \mathcal{X}, u(k) \in \{-1, 0, 1\}^3 \end{aligned}$$

Numerical example

Hybrid Model Predictive Control

minimize
$$\sum_{k=0}^{N-1} \gamma^k l(x(k)) + \gamma^N V(x(N))$$

subject to $x(k+1) = Ax(k) + Bu(k)$ Dynamics

$$x(0) = x_0$$

$$x(k) \in \mathcal{X}, u(k) \in \{-1, 0, 1\}^3$$

Numerical example

Hybrid Model Predictive Control

minimize
$$\sum_{k=0}^{N-1} \gamma^k l(x(k)) + \gamma^N V(x(N))$$

subject to
$$x(k+1) = Ax(k) + Bu(k)$$
 Dynamics

$$x(0) = x_0$$
 Initial state

$$x(k) \in \mathcal{X}, u(k) \in \{-1, 0, 1\}^3$$

Numerical example

Hybrid Model Predictive Control

minimize
$$\sum_{k=0}^{N-1} \gamma^k l(x(k)) + \gamma^N V(x(N))$$

subject to

$$x(k+1) = Ax(k) + Bu(k)$$

Dynamics

$$x(0) = x_0$$

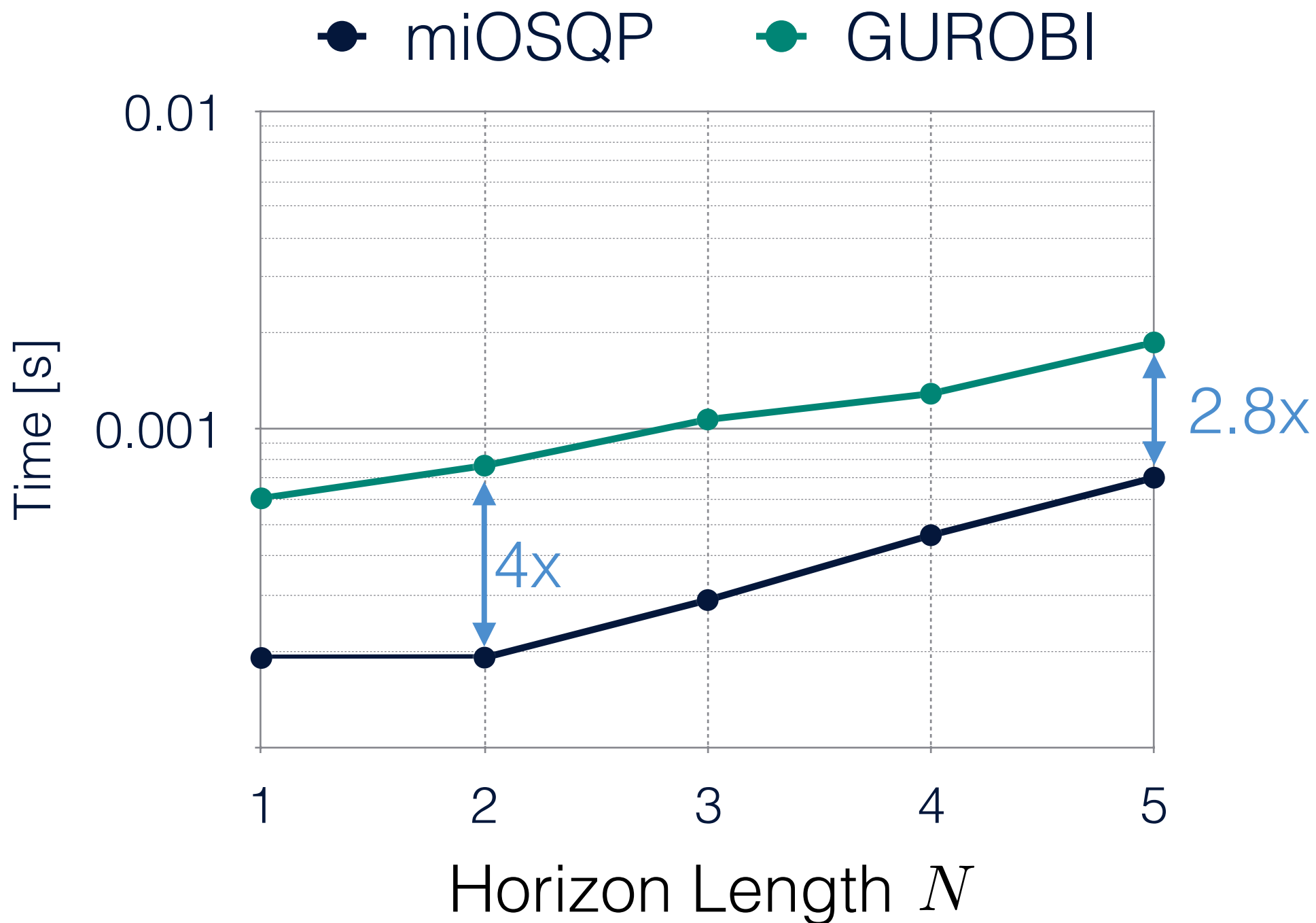
Initial state

$$x(k) \in \mathcal{X}, u(k) \in \{-1, 0, 1\}^3$$

State/input constraints

MIQP Timings

Hybrid Model Predictive Control



Simple
Python
Implementation!

Conclusions

Acknowledgements



Goran
Banjac

Oxford



Nicholas
Moehle

Stanford



Paul
Goulart

Oxford



Alberto
Bemporad

IMT Lucca



Stephen
Boyd

Stanford

Final remarks

OSQP

Simple

Robust

Embeddable

Warm-starting

Precise

Detects infeasibility

Future work

Semidefinite
programs

“Meta-algorithms”

Mixed-Integer

SQP

References

B. Stellato, G. Banjac, P. Goulart, A. Bemporad and S. Boyd. *OSQP: An Operator Splitting Solver for Quadratic Programs. (Coming soon!)*

G. Banjac, P. Goulart, B. Stellato, and S. Boyd. *Infeasibility detection in the alternating direction method of multipliers for convex optimization.* optimization-online.org, 2017

G. Banjac, B. Stellato, N. Moehle, P. Goulart, A. Bemporad and S. Boyd. *Embedded code generation using the OSQP solver. IEEE Conference on Decision and Control (CDC) (submitted), 2017*

B. Stellato, V. Naik, A. Bemporad, P. Goulart, and S. Boyd. *Embedded mixed-integer quadratic optimization using the OSQP solver. IEEE Conference on Decision and Control (CDC) (submitted), 2017*