An Active-Set Quadratic Programming Method Based On Sequential Hot-Starts

Andreas Wächter

Department of Industrial Engineering and Management Sciences
Northwestern University
waechter@iems.northwestern.edu

Computational Contact Mechanics: Advances and Frontiers in Modeling Contact
Banff International Research Station

February 17, 2014

Outline

- Solving sequences of "similar" QPs
- Goal: Use information computed for first QP to accelerate the (approximate) solution of later QPs
  - Avoid factorization of matrices
  - Avoid computation of full matrices

Outline

- Solving sequences of "similar" QPs
- Goal: Use information computed for first QP to accelerate the (approximate) solution of later QPs
  - Avoid factorization of matrices
  - Avoid computation of full matrices
- Ingredients
  - Active set method
  - Iterative refinement
  - Accelerated iterative linear solver
- Numerical results
  - Model predictive control
  - Sequence of random NLPs

Outline

- Solving sequences of "similar" QPs
- Goal: Use information computed for first QP to accelerate the (approximate) solution of later QPs
  - Avoid factorization of matrices
  - Avoid computation of full matrices
- Ingredients
  - Active set method
  - Iterative refinement
  - Accelerated iterative linear solver
- Numerical results
  - Model predictive control
  - Sequence of random NLPs

- In collaboration with:
  - Christian Kirches
  - Travis Johnson

Solve Sequence of Similar Quadratic Programs

$$QP$$

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \tfrac{1}{2} d^T W d + g^T d \\ \text{s.t.} \quad & A d + c = 0 \\ & d \geq l \end{aligned}$$

- $W \in \mathbb{R}^{n\times n}$ (pos.def.), $A \in \mathbb{R}^{m\times n}$, $g \in \mathbb{R}^n$, $c \in \mathbb{R}^m$, and $l \in \mathbb{R}^n$

Solve Sequence of Similar Quadratic Programs

QP

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \tfrac{1}{2} d^T W d + g^T d \\ \text{s.t.} \quad & A d + c = 0 \\ & d \geq l \end{aligned}$$

- $W \in \mathbb{R}^{n\times n}$ (pos.def.), $A \in \mathbb{R}^{m\times n}$, $g \in \mathbb{R}^n$, $c \in \mathbb{R}^m$, and $l \in \mathbb{R}^n$

QP_0

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \tfrac{1}{2} d^T W_0 d + g_0^T d \\ \text{s.t.} \quad & A_0 d + c_0 = 0 \\ & d \geq l_0 \end{aligned}$$

QP_1

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \tfrac{1}{2} d^T W_1 d + g_1^T d \\ \text{s.t.} \quad & A_1 d + c_1 = 0 \\ & d \geq l_1 \end{aligned}$$

QP_2

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \tfrac{1}{2} d^T W_2 d + g_2^T d \\ \text{s.t.} \quad & A_2 d + c_2 = 0 \\ & d \geq l_2 \end{aligned}$$

$\ldots$

- Solve sequence of QPs
  - Same size, with "similar" data: $W_0 \approx W_i$, $A_0 \approx A_i$, $g_0 \approx g_i$, $\ldots$
- Goal: Exploit information computed for QP_0 to solve QP_i approx.

Example: Sequential Quadratic Programming

NLP

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0 \\ & x \geq 0 \end{aligned}$$

$\longrightarrow$

QP_k

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \tfrac{1}{2} d^T W_k d + g_k^T d \\ \text{s.t.} \quad & A_k d + c_k = 0 \\ & d \geq l_k \end{aligned}$$

- Solve QP for each iterate x_k
  - $W_k \approx \nabla^2 \mathcal{L}(x_k, \lambda_k)$, $A_k = \nabla c(x_k)^T$, $g_k = \nabla f(x_k)$, $c_k = c(x_k)$, $l_k = -x_k$

Example: Sequential Quadratic Programming

NLP

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0 \\ & x \geq 0 \end{aligned}$$

$\longrightarrow$

QP_k

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \tfrac{1}{2} d^T W_k d + g_k^T d \\ \text{s.t.} \quad & A_k d + c_k = 0 \\ & d \geq l_k \end{aligned}$$

- Solve QP for each iterate x_k
  - $W_k \approx \nabla^2 \mathcal{L}(x_k, \lambda_k)$, $A_k = \nabla c(x_k)^T$, $g_k = \nabla f(x_k)$, $c_k = c(x_k)$, $l_k = -x_k$
- QP data does not change much close to solution x_*

Example: Sequential Quadratic Programming

NLP

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0 \\ & x \geq 0 \end{aligned}$$

$\longrightarrow$

QP_k

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \tfrac{1}{2} d^T W_k d + g_k^T d \\ \text{s.t.} \quad & A_k d + c_k = 0 \\ & d \geq l_k \end{aligned}$$

- Solve QP for each iterate x_k
  - $W_k \approx \nabla^2 \mathcal{L}(x_k, \lambda_k)$, $A_k = \nabla c(x_k)^T$, $g_k = \nabla f(x_k)$, $c_k = c(x_k)$, $l_k = -x_k$
- QP data does not change much close to solution x_*
- Original motivation:
  - Solve sequence of similar NLPs

Example: Sequential Quadratic Programming

NLP

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0 \\ & x \geq 0 \end{aligned}$$

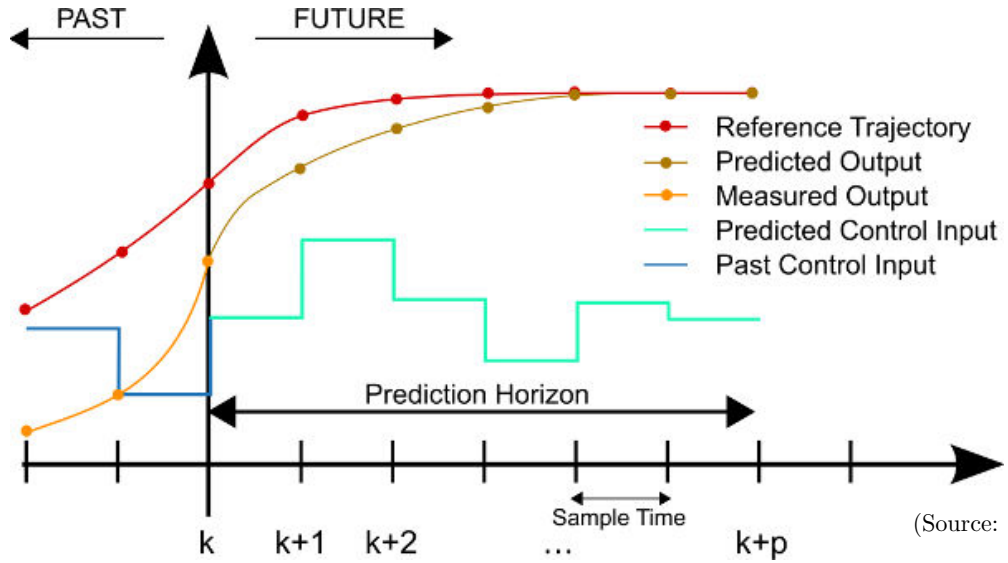
$\longrightarrow$

QP_k

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \tfrac{1}{2} d^T W_k d + g_k^T d \\ \text{s.t.} \quad & A_k d + c_k = 0 \\ & d \geq l_k \end{aligned}$$

- Solve QP for each iterate x_k
  - $W_k \approx \nabla^2 \mathcal{L}(x_k, \lambda_k)$, $A_k = \nabla c(x_k)^T$, $g_k = \nabla f(x_k)$, $c_k = c(x_k)$, $l_k = -x_k$
- QP data does not change much close to solution x_*
- Original motivation:
  - Solve sequence of similar NLPs
  - Example: Mixed-Integer Nonlinear Programming
    - ★ Solve nodes in a branch-and-bound tree
    - ★ Linear: dual simplex method can rapidly solve new instance
    - ★ Reason: an existing factorization of basis matrix can be used!

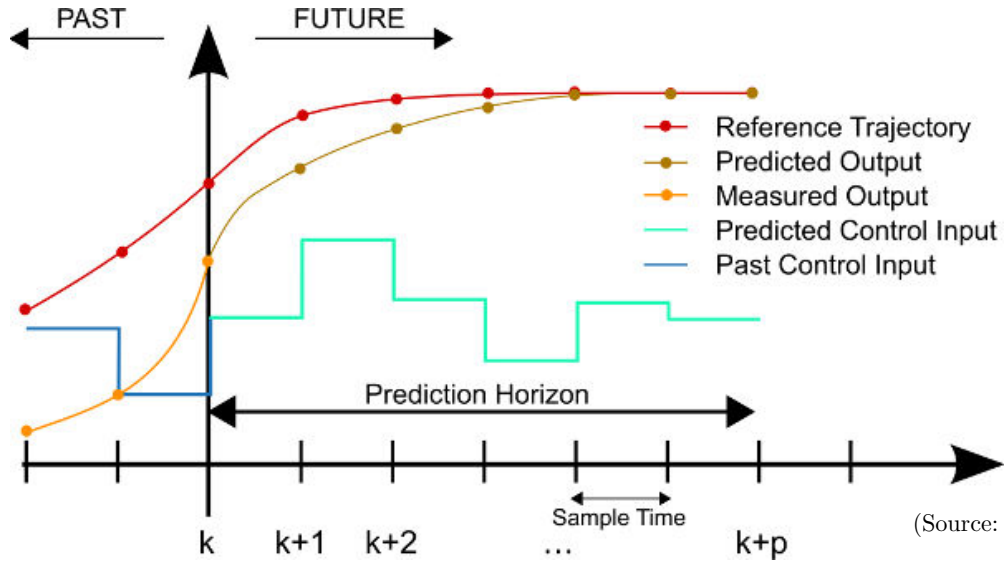
Example: Model Predictive Control



(Source: Martin Behrent)

- At time t_k:
  - Optimize control over prediction horizon (NLP or QP)
  - Implement optimal control action over time $[t_k, t_{k+1}]$

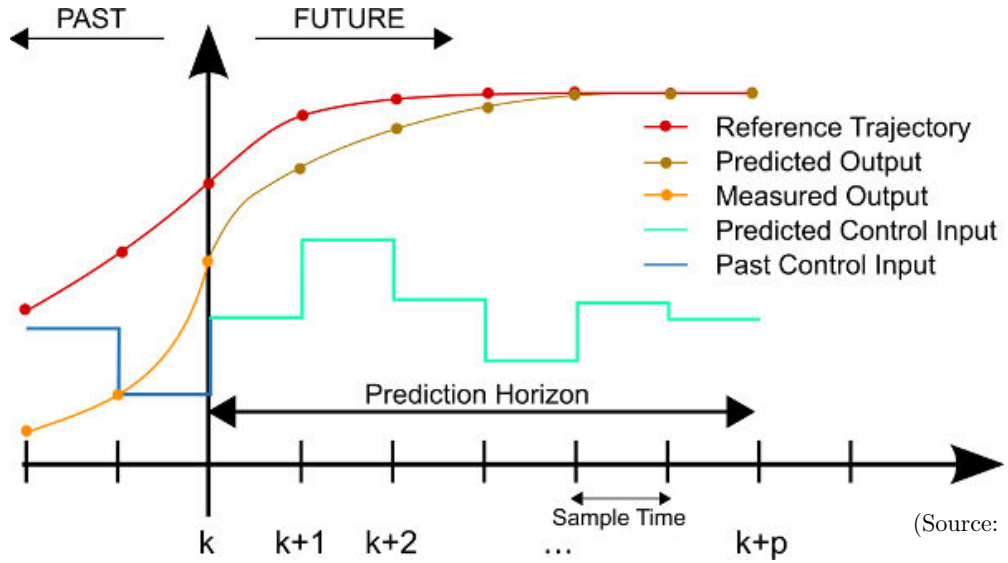
Example: Model Predictive Control



(Source: Martin Behrent)

- At time t_k:
  - Optimize control over prediction horizon (NLP or QP)
  - Implement optimal control action over time $[t_k, t_{k+1}]$
- At time t_{k+1}:
  - Remeasure actual system; repeat
  - Usually, little change in optimization problem

Example: Model Predictive Control



(Source: Martin Behrent)

- At time t_k:
  - Optimize control over prediction horizon (NLP or QP)
  - Implement optimal control action over time $[t_k, t_{k+1}]$
- At time t_{k+1}:
  - Remeasure actual system; repeat
  - Usually, little change in optimization problem
- Multiple shooting approach:
  - Integrate differential equations over subintervals $[t_k, t_{k+1}]$
  - QP data expensive
    - ★ Computation of full matrices very expensive (Av and $A^T v$ available)

Active Set QP Solver

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \tfrac{1}{2} d^T W d + g^T d \\ \text{s.t.} \quad & A d + c = 0 \end{aligned}$$

Compute solution from linear system:

$$\begin{bmatrix} W & A^T \\ A & 0 \end{bmatrix} \begin{pmatrix} d \\ \lambda \end{pmatrix} = - \begin{pmatrix} g \\ c \end{pmatrix}$$

Active Set QP Solver

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \tfrac{1}{2} d^T W d + g^T d \\ \text{s.t.} \quad & A d + c = 0 \\ & d \geq l \end{aligned}$$

$\mathcal{A} \subseteq \{1, \ldots, n\}$: Guess of the active set, i.e., $d^{\mathcal{A}} = l^{\mathcal{A}}$

Compute solution from linear system:

$$\begin{bmatrix} W & A^T & (I^{\mathcal{A}})^T \\ A & 0 & 0 \\ I^{\mathcal{A}} & 0 & 0 \end{bmatrix} \begin{pmatrix} d \\ \lambda \\ -\mu^{\mathcal{A}} \end{pmatrix} = - \begin{pmatrix} g \\ c \\ l^{\mathcal{A}} \end{pmatrix}$$

Active Set QP Solver

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \tfrac{1}{2} d^T W d + g^T d \\ \text{s.t.} \quad & A d + c = 0 \\ & d \geq l \end{aligned}$$

$\mathcal{A} \subseteq \{1, \ldots, n\}$: Guess of the active set, i.e., $d^{\mathcal{A}} = l^{\mathcal{A}}$

Compute solution from linear system:

$$\begin{bmatrix} W & A^T & (I^{\mathcal{A}})^T \\ A & 0 & 0 \\ I^{\mathcal{A}} & 0 & 0 \end{bmatrix} \begin{pmatrix} d \\ \lambda \\ -\mu^{\mathcal{A}} \end{pmatrix} = - \begin{pmatrix} g \\ c \\ l^{\mathcal{A}} \end{pmatrix}$$

$\mathcal{A}$ is correct, if $d \geq l$ and $\mu^{\mathcal{A}} \geq 0$

Warm Start vs. Hot Start

- Warm Start:

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \tfrac{1}{2} d^T \widetilde{W} d + \widetilde{g}^T d \\ \text{s.t.} \quad & \widetilde{A} d + \widetilde{c} = 0 \\ & d \geq \widetilde{l} \end{aligned} \longrightarrow \begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \tfrac{1}{2} d^T W d + g^T d \\ \text{s.t.} \quad & A d + c = 0 \\ & d \geq l \end{aligned}$$

Use active set $\mathcal{A}$ as starting guess, factorize with new matrix data

Warm Start vs. Hot Start

- Warm Start:

$$\min_{d\in\mathbb{R}^n} \ \tfrac{1}{2} d^T \widetilde{W} d + \widetilde{g}^T d \quad \text{s.t.} \ \widetilde{A} d + \widetilde{c} = 0, \ d \geq \widetilde{l} \longrightarrow \min_{d\in\mathbb{R}^n} \ \tfrac{1}{2} d^T W d + g^T d \quad \text{s.t.} \ A d + c = 0, \ d \geq l$$

Use active set $\mathcal{A}$ as starting guess, factorize with new matrix data

- Hot Start:

$$\min_{d\in\mathbb{R}^n} \ \tfrac{1}{2} d^T \widetilde{W} d + \widetilde{g}^T d \quad \text{s.t.} \ \widetilde{A} d + \widetilde{c} = 0, \ d \geq \widetilde{l} \longrightarrow \min_{d\in\mathbb{R}^n} \ \tfrac{1}{2} d^T \widetilde{W} d + g^T d \quad \text{s.t.} \ \widetilde{A} d + c = 0, \ d \geq l$$

Use $\mathcal{A}$ as starting guess, reuse factorization with same matrix data

Warm Start vs. Hot Start

- Warm Start:

$$\min_{d\in\mathbb{R}^n} \ \tfrac{1}{2} d^T \widetilde{W} d + \widetilde{g}^T d \quad \text{s.t.} \ \widetilde{A} d + \widetilde{c} = 0, \ d \geq \widetilde{l} \longrightarrow \min_{d\in\mathbb{R}^n} \ \tfrac{1}{2} d^T W d + g^T d \quad \text{s.t.} \ A d + c = 0, \ d \geq l$$

  Use active set $\mathcal{A}$ as starting guess, factorize with new matrix data

- Hot Start:

$$\min_{d\in\mathbb{R}^n} \ \tfrac{1}{2} d^T \widetilde{W} d + \widetilde{g}^T d \quad \text{s.t.} \ \widetilde{A} d + \widetilde{c} = 0, \ d \geq \widetilde{l} \longrightarrow \min_{d\in\mathbb{R}^n} \ \tfrac{1}{2} d^T \widetilde{W} d + g^T d \quad \text{s.t.} \ \widetilde{A} d + c = 0, \ d \geq l$$

  Use $\mathcal{A}$ as starting guess, reuse factorization with same matrix data

- Appropriate algorithm: Parametric QP solver

Iterative Refinement

- Want to solve linear system

$$\boxed{M\,x + b = 0}$$

- At iterate x_i:

$$M\,x_i + b = r_i \neq 0$$

- New iterate $x_{i+1} = x_i + p_i$ should satisfy

$$M(x_i + p_i) + b = 0$$

- Get step p_i from

$$M\,p_i = -\underbrace{(M\,x_i + b)}_{=r_i}$$

Iterative Refinement

- Want to solve linear system

$$\boxed{M\,x + b = 0}$$

- At iterate x_i:

$$M\,x_i + b = r_i \neq 0$$

- New iterate $x_{i+1} = x_i + p_i$ should satisfy

$$M(x_i + p_i) + b = 0$$

- Get step p_i from

$$\widetilde{M}\,p_i = -\underbrace{(M\,x_i + b)}_{=r_i}$$

- Converges if $\|I - \widetilde{M}^{-1}\,M\| < 1$.

Equality Constrained QP

$$\begin{aligned} \min_{d} \quad & \tfrac{1}{2} d^T W d + g^T d \\ \text{s.t.} \quad & A d + c = 0 \end{aligned}$$

$$\begin{bmatrix} W & A^T \\ A & 0 \end{bmatrix} \begin{pmatrix} d \\ \lambda \end{pmatrix} = - \begin{pmatrix} g \\ c \end{pmatrix}$$

Iterative refinement ($d_{i+1} = d_i + p_i$ and $\lambda_{i+1} = \lambda_i + p_i^\lambda$)

$$\begin{bmatrix} \widetilde{W} & \widetilde{A}^T \\ \widetilde{A} & 0 \end{bmatrix} \begin{pmatrix} p_i \\ p_i^\lambda \end{pmatrix} = - \left[\begin{pmatrix} W d_i + A^T \lambda_i \\ A d_i \end{pmatrix} + \begin{pmatrix} g \\ c \end{pmatrix} \right]$$

Equality Constrained QP

$$\begin{aligned} \min_d \quad & \tfrac{1}{2} d^T W d + g^T d \\ \text{s.t.} \quad & A d + c = 0 \end{aligned}$$

$$\begin{bmatrix} W & A^T \\ A & 0 \end{bmatrix} \begin{pmatrix} d \\ \lambda \end{pmatrix} = - \begin{pmatrix} g \\ c \end{pmatrix}$$

Iterative refinement ($d_{i+1} = d_i + p_i$ and $\lambda_{i+1} = \lambda_i + p_i^\lambda$)

$$\begin{bmatrix} \widetilde{W} & \widetilde{A}^T \\ \widetilde{A} & 0 \end{bmatrix} \begin{pmatrix} p_i \\ p_i^\lambda \end{pmatrix} = - \left[\begin{pmatrix} W d_i + A^T \lambda_i \\ A d_i \end{pmatrix} + \begin{pmatrix} g \\ c \end{pmatrix} \right]$$

$$\begin{aligned} \min_p \quad & \tfrac{1}{2} p^T \widetilde{W} p + \left(W d_i + A^T \lambda_i + g \right)^T p \\ \text{s.t.} \quad & \widetilde{A} p + (A d_i + c) = 0 \qquad \longrightarrow p_i^\lambda \end{aligned}$$

Inequality Constraints

Solve

$$\begin{aligned} \min_{d} \quad & \tfrac{1}{2} d^T W d + g^T d \\ \text{s.t.} \quad & A d + c = 0 \end{aligned}$$

by computing a sequence of iterative refinement steps from

$$\begin{aligned} \min_{p} \quad & \tfrac{1}{2} p^T \widetilde{W} p + \Big(W d_i + A^T \lambda_i + g \Big)^T p \\ \text{s.t.} \quad & \widetilde{A} p + (A d_i + c) = 0 \qquad \longrightarrow p_i^\lambda \end{aligned}$$

Inequality Constraints

Solve

$$\begin{aligned} \min_{d} \quad & \tfrac{1}{2} d^T W d + g^T d \\ \text{s.t.} \quad & A d + c = 0 \\ & d \geq l \end{aligned}$$

by computing a sequence of iterative refinement steps from

$$\begin{aligned} \min_{p} \quad & \tfrac{1}{2} p^T \widetilde{W} p + \Big(W d_i + A^T \lambda_i + g \Big)^T p \\ \text{s.t.} \quad & \widetilde{A} p + (A d_i + c) = 0 \qquad \longrightarrow p_i^\lambda \end{aligned}$$

Inequality Constraints

Solve

$$\begin{aligned} \min_{d} \quad & \tfrac{1}{2} d^T W d + g^T d \\ \text{s.t.} \quad & A d + c = 0 \\ & d \geq l \end{aligned}$$

by computing a sequence of iterative refinement steps from

$$\begin{aligned} \min_{p} \quad & \tfrac{1}{2} p^T \widetilde{W} p + \Big(W d_i + A^T \lambda_i + g \Big)^T p \\ \text{s.t.} \quad & \widetilde{A} p + (A d_i + c) = 0 && \longrightarrow p_i^\lambda \\ & d_i + p \geq l && \longrightarrow \mu_{i+1} \end{aligned}$$

Note: We do not need iterates for bound multipliers

Inequality Constraints and Iterative Refinement

- Uses hot-starts (starts from existing factorization)
- Uses only $W d_i$, $A d_i$, and $A^T \lambda_i$.

Inequality Constraints and Iterative Refinement

- Uses hot-starts (starts from existing factorization)
- Uses only $W d_i$, $A d_i$, and $A^T \lambda_i$.
- No need to track multipliers of bound constraints
  - True for any constraint that does not change
- Key consequence: QP solver is responsible for handling active set
  - we do not need to explicitly maintain complementarity of iterates
  - can use your favorite hot-startable QP solver
  - suitable: parametric QP solver

Inequality Constraints and Iterative Refinement

- Uses hot-starts (starts from existing factorization)
- Uses only $W d_i$, $A d_i$, and $A^T \lambda_i$.
- No need to track multipliers of bound constraints
  - True for any constraint that does not change
- Key consequence: QP solver is responsible for handling active set
  - we do not need to explicitly maintain complementarity of iterates
  - can use your favorite hot-startable QP solver
  - suitable: parametric QP solver
- Theorem [Johnson, Kirches, W 13]:
  - If iterates converge, then to optimal solution of QP
  - If SOSC and some contraction conditions hold, convergence to (d_*, λ_*) if (d_0, λ_0) sufficiently close to (d_*, λ_*)

Inequality Constraints and Iterative Refinement

- Uses hot-starts (starts from existing factorization)
- Uses only $W d_i$, $A d_i$, and $A^T \lambda_i$.
- No need to track multipliers of bound constraints
  - True for any constraint that does not change
- Key consequence: QP solver is responsible for handling active set
  - we do not need to explicitly maintain complementarity of iterates
  - can use your favorite hot-startable QP solver
  - suitable: parametric QP solver
- Theorem [Johnson, Kirches, W 13]:
  - If iterates converge, then to optimal solution of QP
  - If SOSC and some contraction conditions hold, convergence to (d_*, λ_*) if (d_0, λ_0) sufficiently close to (d_*, λ_*)
- Related method [Bock et al., 07]: Only one refinement step

Inequality Constraints and Iterative Refinement

- Uses hot-starts (starts from existing factorization)
- Uses only $W d_i$, $A d_i$, and $A^T \lambda_i$.
- No need to track multipliers of bound constraints
  - True for any constraint that does not change
- Key consequence: QP solver is responsible for handling active set
  - we do not need to explicitly maintain complementarity of iterates
  - can use your favorite hot-startable QP solver
  - suitable: parametric QP solver
- Theorem [Johnson, Kirches, W 13]:
  - If iterates converge, then to optimal solution of QP
  - If SOSC and some contraction conditions hold, convergence to (d_*, λ_*) if (d_0, λ_0) sufficiently close to (d_*, λ_*)
- Related method [Bock et al., 07]: Only one refinement step
- Disadvantage: Iterative refinement converges slowly

Accelerated Version

- Monitor active set $\mathcal{A}_i = \{j : d_i^{(j)} = l^{(j)}\}$.
- If $\mathcal{A}_i = \mathcal{A}_{i-1} =: \mathcal{A}$ during iterative refinement, set $\mathcal{F} = \mathcal{A}^C$.
- Active-set QP solver sets $d^{\mathcal{A}} = l^{\mathcal{A}}$ and solves

$$\begin{bmatrix} W^{\mathcal{F}\mathcal{F}} & (A^{\mathcal{F}})^T \\ A^{\mathcal{F}} & 0 \end{bmatrix} \begin{pmatrix} d^{\mathcal{F}} \\ \lambda \end{pmatrix} = - \begin{pmatrix} \tilde{g}^{\mathcal{F}} \\ \tilde{c} \end{pmatrix}$$

- Apply iterative linear solver (e.g., SQMR)

Accelerated Version

- Monitor active set $\mathcal{A}_i = \{j : d_i^{(j)} = l^{(j)}\}$.
- If $\mathcal{A}_i = \mathcal{A}_{i-1} =: \mathcal{A}$ during iterative refinement, set $\mathcal{F} = \mathcal{A}^C$.
- Active-set QP solver sets $d^{\mathcal{A}} = l^{\mathcal{A}}$ and solves

$$\begin{bmatrix} W^{\mathcal{F}\mathcal{F}} & (A^{\mathcal{F}})^T \\ A^{\mathcal{F}} & 0 \end{bmatrix} \begin{pmatrix} d^{\mathcal{F}} \\ \lambda \end{pmatrix} = - \begin{pmatrix} \tilde{g}^{\mathcal{F}} \\ \tilde{c} \end{pmatrix}$$

- Apply iterative linear solver (e.g., SQMR)
- Preconditioner requires solutions of

$$\begin{bmatrix} \widetilde{W}^{\mathcal{F}\mathcal{F}} & (\widetilde{A}^{\mathcal{F}})^T \\ \widetilde{A}^{\mathcal{F}} & 0 \end{bmatrix} \begin{pmatrix} z^{\mathcal{F}} \\ z^{\lambda} \end{pmatrix} = \begin{pmatrix} r_g^{\mathcal{F}} \\ r_c \end{pmatrix}$$

Preconditioner QP

$$\begin{bmatrix} W^{\mathcal{F}\mathcal{F}} & (A^{\mathcal{F}})^T \\ A^{\mathcal{F}} & 0 \end{bmatrix} \begin{pmatrix} d^{\mathcal{F}} \\ \lambda \end{pmatrix} = - \begin{pmatrix} g^{\mathcal{F}} \\ c \end{pmatrix}$$

Preconditioner

$$\begin{bmatrix} \widetilde{W}^{\mathcal{F}\mathcal{F}} & (\widetilde{A}^{\mathcal{F}})^T \\ \widetilde{A}^{\mathcal{F}} & 0 \end{bmatrix} \begin{pmatrix} z^{\mathcal{F}} \\ z^{\lambda} \end{pmatrix} = \begin{pmatrix} r_g^{\mathcal{F}} \\ r_c \end{pmatrix}$$

Could solve

$$\begin{aligned} \min_z \quad & \tfrac{1}{2} z^T \widetilde{W} z + (-r_g^{\mathcal{F}})^T z^{\mathcal{F}} \\ \text{s.t.} \quad & \widetilde{A} z - r_c = 0 \\ & 0 \le z^{\mathcal{A}} \le 0 \end{aligned}$$

Preconditioner QP

$$\begin{bmatrix} W^{\mathcal{F}\mathcal{F}} & (A^{\mathcal{F}})^T \\ A^{\mathcal{F}} & 0 \end{bmatrix} \begin{pmatrix} d^{\mathcal{F}} \\ \lambda \end{pmatrix} = - \begin{pmatrix} g^{\mathcal{F}} \\ c \end{pmatrix}$$

Preconditioner

$$\begin{bmatrix} \widetilde{W}^{\mathcal{F}\mathcal{F}} & (\widetilde{A}^{\mathcal{F}})^T \\ \widetilde{A}^{\mathcal{F}} & 0 \end{bmatrix} \begin{pmatrix} z^{\mathcal{F}} \\ z^{\lambda} \end{pmatrix} = \begin{pmatrix} r_g^{\mathcal{F}} \\ r_c \end{pmatrix}$$

Could solve

$$\begin{aligned} \min_z \quad & \tfrac{1}{2} z^T \widetilde{W} z + (-r_g^{\mathcal{F}})^T z^{\mathcal{F}} \\ \text{s.t.} \quad & \widetilde{A} z - r_c = 0 \\ & 0 \le z^{\mathcal{A}} \le 0 \end{aligned}$$

Guessed active set $\mathcal{A}$ might be too small

- As SQMR converges, we might have $d_*^{\mathcal{F}} \not\geq l^{\mathcal{F}}$
  - Stop SQMR if $d_i \not\geq l$, go back to iterative refinement.

Preconditioner QP

$$\begin{aligned} \min_z \quad & \tfrac{1}{2} z^T \widetilde{W} z + (-r_g^{\mathcal{F}})^T z^{\mathcal{F}} \\ \text{s.t.} \quad & \widetilde{A} z - r_c = 0 \\ & 0 \le z^{\mathcal{A}} \le 0 \end{aligned}$$

We solve

$$\begin{aligned} \min_z \quad & \tfrac{1}{2} z^T \tilde{W} z + (-r_g^{\mathcal{F}})^T z^{\mathcal{F}} + \left(W^{\mathcal{A},\cdot} d_i + (A^{\mathcal{A}})^T \lambda_i + g^{\mathcal{A}} \right)^T z^{\mathcal{A}} \\ \text{s.t.} \quad & \tilde{A} z - r_c = 0 \\ & 0 \le z^{\mathcal{A}} \end{aligned}$$

- Guessed active set $\mathcal{A}$ might be too large
  - Stop SQMR if $z^{(j)} > 0$ for some $j \in \mathcal{A}$

Preconditioner QP

$$\begin{aligned} \min_z \quad & \tfrac{1}{2} z^T \widetilde{W} z + (-r_g^{\mathcal{F}})^T z^{\mathcal{F}} \\ \text{s.t.} \quad & \widetilde{A} z - r_c = 0 \\ & 0 \le z^{\mathcal{A}} \le 0 \end{aligned}$$

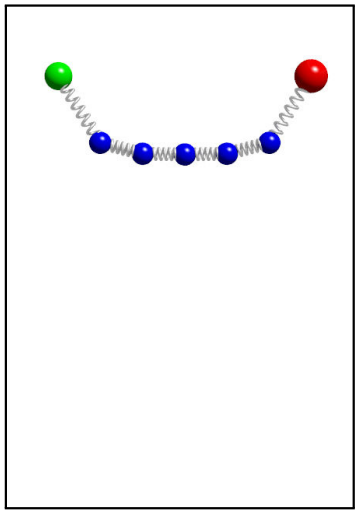
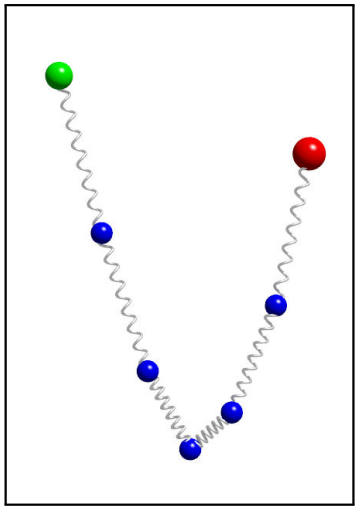
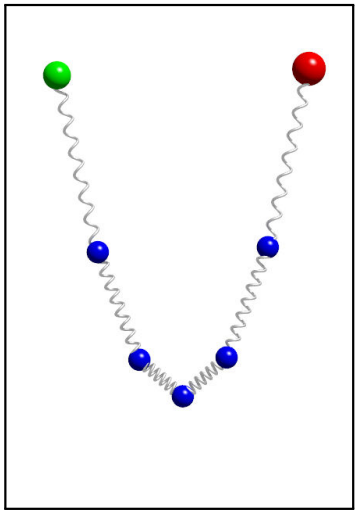
We solve

$$\begin{aligned} \min_z \quad & \tfrac{1}{2} z^T \tilde{W} z + (-r_g^{\mathcal{F}})^T z^{\mathcal{F}} + \left(W^{\mathcal{A},\cdot} d_i + (A^{\mathcal{A}})^T \lambda_i + g^{\mathcal{A}} \right)^T z^{\mathcal{A}} \\ \text{s.t.} \quad & \tilde{A} z - r_c = 0 \\ & 0 \le z^{\mathcal{A}} \end{aligned}$$

- Guessed active set $\mathcal{A}$ might be too large
  - Stop SQMR if $z^{(j)} > 0$ for some $j \in \mathcal{A}$
- Theorem [Johnson, Kirches, W 13]:
  - If iterates converge, then to optimal solution of QP_k
  - If SOSC holds, convergence to solution (d_*, λ_*) during SQMR, if (d_0, λ_0) sufficiently close to (d_*, λ_*)

Numerical Results Model Predictive Control

Controlling hanging chain

Numerical Results Model Predictive Control

Controlling hanging chain

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T w_{\mathrm{v}} \sum_{i=1}^{N_{\mathrm{PM}}} \|v_i(t)\|_2^2 + w_{\mathrm{x}} \|x_{N_{\mathrm{PM}}}(t) - x_{\mathrm{e}}\|_2^2 + w_{\mathrm{u}} \|u(t)\|_2^2 + w_{\mathrm{sl}} \|u_{\mathrm{sl}}(t)\|_2^2 \, \mathrm{d}t \\
\text{s.t.} \quad & \dot{x}_i(t) = v_i(t) && t \in [0,T],\ 1 \le i < N_{\mathrm{PM}} \\
& \dot{v}_i(t) = (F_{i+1}(t) - F_i(t)) \cdot N_{\mathrm{PM}}/m - g && t \in [0,T],\ 1 \le i < N_{\mathrm{PM}} \\
& \dot{x}_{N_{\mathrm{PM}}}(t) = u(t) && t \in [0,T] \\
& x(0) = \hat{x}_0 \\
& u(t) \in [-1,1]^3 && t \in [0,T] \\
& x_1^{\mathrm{low}} \le x_1(t) \le x_1^{\mathrm{high}}, \quad x_3^{\mathrm{low}} - u_{\mathrm{sl}}(t) \le x_3(t) && t \in [0,T], \\
& u_{\mathrm{sl}}(t) \ge 0, && t \in [0,T].
\end{aligned}
$$

- Matlab implementation of iterative QP solver (`iQP`)
  - Uses parametric QP solver `qpOASES` [Ferreau et al 08;Potschka et al 10]
  - Tight tolerance ($\epsilon = 10^{-8}$)

Numerical Results Model Predictive Control

Controlling hanging chain

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T w_{\mathrm{v}} \sum_{i=1}^{N_{\mathrm{PM}}} \|v_i(t)\|_2^2 + w_{\mathrm{x}}\|x_{N_{\mathrm{PM}}}(t) - x_{\mathrm{e}}\|_2^2 + w_{\mathrm{u}}\|u(t)\|_2^2 + w_{\mathrm{sl}}\|u_{\mathrm{sl}}(t)\|_2^2 \,\mathrm{d}t \\
\text{s.t.} \quad & \dot{x}_i(t) = v_i(t) && t \in [0,T],\ 1 \le i < N_{\mathrm{PM}} \\
& \dot{v}_i(t) = (F_{i+1}(t) - F_i(t)) \cdot N_{\mathrm{PM}}/m - g && t \in [0,T],\ 1 \le i < N_{\mathrm{PM}} \\
& \dot{x}_{N_{\mathrm{PM}}}(t) = u(t) && t \in [0,T] \\
& x(0) = \hat{x}_0 \\
& u(t) \in [-1,1]^3 && t \in [0,T] \\
& x_1^{\mathrm{low}} \le x_1(t) \le x_1^{\mathrm{high}}, \quad x_3^{\mathrm{low}} - u_{\mathrm{sl}}(t) \le x_3(t) && t \in [0,T], \\
& u_{\mathrm{sl}}(t) \ge 0, && t \in [0,T].
\end{aligned}
$$

- Matlab implementation of iterative QP solver (`iQP`)
  - Uses parametric QP solver `qpOASES` [Ferreau et al 08;Potschka et al 10]
  - Tight tolerance ($\epsilon = 10^{-8}$)
- Multiple shooting approach
  - Computation of full derivative matrices expensive
  - Here: Performance metric is number of matrix-vector products

Numerical Results NMPC

15 multiple shooting intervals

	QP Size		**Solve**	**Matrix-Vector Products**			
N_{PM}	n	m	Success	Min	Mean	Max	Full A_k
6	656	575	57	16	29.8	128	33
8	880	787	57	20	33.5	130	45
10	1,104	999	57	24	36.9	102	57
12	1,328	1,211	57	32	38.2	90	69
14	1,522	1,423	57	28	35.8	74	81

20 multiple shooting intervals

	QP Size		**Solve**	**Matrix-Vector Products**			
N_{PM}	n	m	Success	Min	Mean	Max	Full A_k
6	861	765	75	16	32.4	148	33
8	1,155	1,047	76	16	30.0	112	45
10	1,449	1,329	76	24	34.8	124	57
12	1,743	1,611	76	24	33.0	104	69
14	1,977	1,893	76	24	33.8	108	81

Numerical Results with Random NLPs

- Experiment with (exact) SQP method
  - QPs solved to tight tolerance (10^{-8} to 10^{-12})
- NLP with random perturbation (dense matrices)

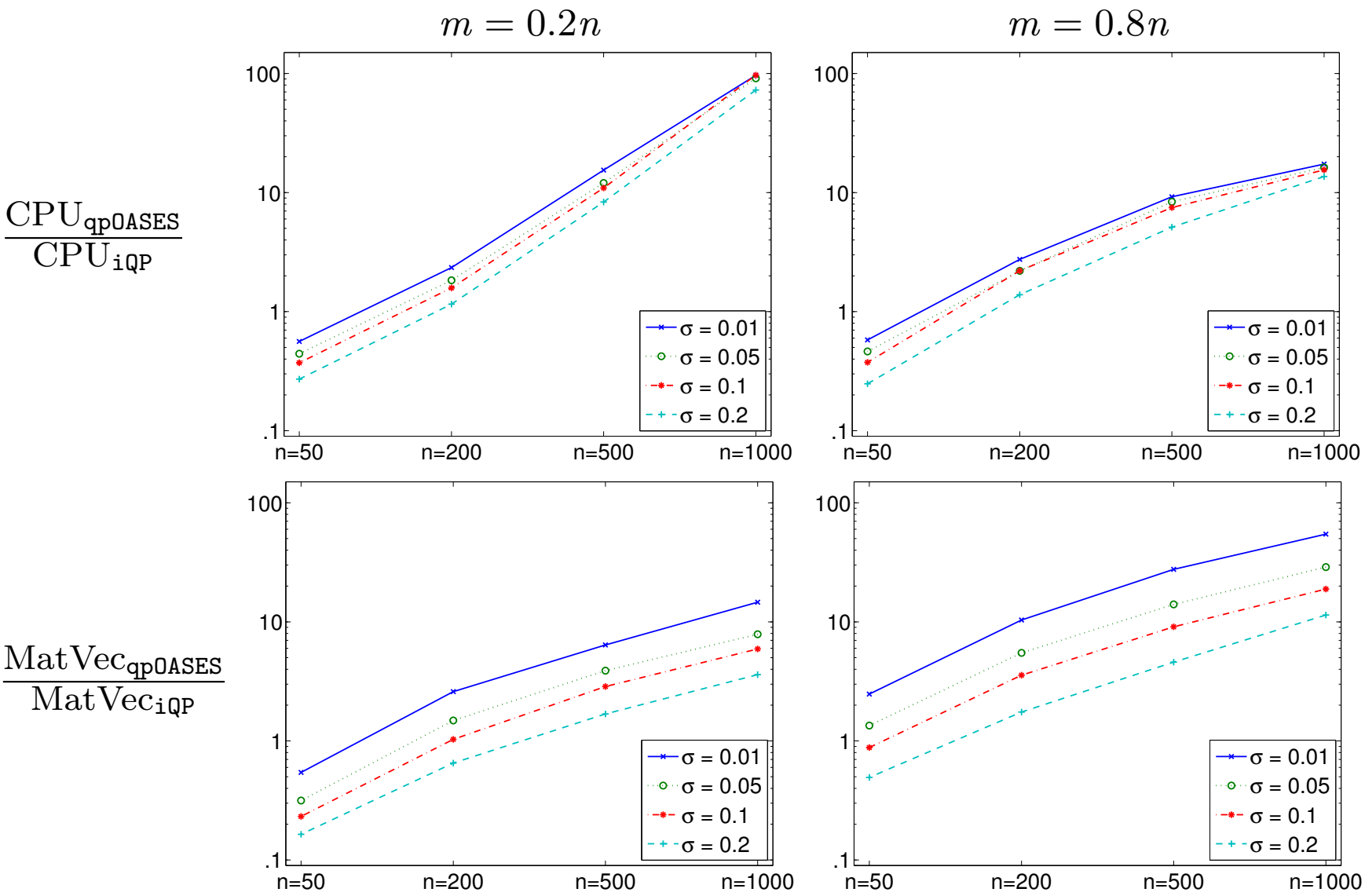
$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \tfrac{1}{2} x^T H_0 x + (q_0)^T x + r_0 \\ \text{s.t.} \quad & \tfrac{1}{2} x^T H_j x + (q_j)^T x + r_j \le 0 \qquad j = 1, \ldots, m \\ & x \ge 0 \end{aligned}$$

  - $n \in \{50, 200, 500, 1000\}$
  - $m \in \{0.2n, 0.5n, 0.8n, 1.5n\}$
  - $\sigma \in \{0.01, 0.05, 0.1, 0.2\}$ standard deviation of perturbation
- Solve sequences of 10 perturbed NLPs (3 for $n = 1000$)
- Solve first NLP, store final QP as reference QP

Hot-Started NLP Results

	Size of perturbation σ			
	0.01	0.05	0.1	0.2
Successfully solved	99.3%	99.3%	97.7%	42.9%
Avrg # `iQP` iters per SQP iter	4.9	8.5	12.5	20.6
Avrg change in # active ineq	5.3%	19.0%	25.7%	32.4%
Avrg change in # active bounds	3.4%	9.5%	13.8%	19.2%

Hot-Started NLP Results



Conclusions

- Active-set QP solver exploiting hot-starts for similar QP instances
  - Iterative refinement with inequality constraints
  - Accelerated linear solver for fixed active set
    - ★ QP as preconditioner

Conclusions

- Active-set QP solver exploiting hot-starts for similar QP instances
  - Iterative refinement with inequality constraints
  - Accelerated linear solver for fixed active set
    - ★ QP as preconditioner
  - Performance improvement
    - ★ Fewer matrix-vector products
    - ★ Less CPU time for dense instances

Conclusions

- Active-set QP solver exploiting hot-starts for similar QP instances
  - Iterative refinement with inequality constraints
  - Accelerated linear solver for fixed active set
    - ⋆ QP as preconditioner
  - Performance improvement
    - ⋆ Fewer matrix-vector products
    - ⋆ Less CPU time for dense instances
- Open questions
  - Ensure convergence?

Conclusions

- Active-set QP solver exploiting hot-starts for similar QP instances
  - Iterative refinement with inequality constraints
  - Accelerated linear solver for fixed active set
    - ⋆ QP as preconditioner
  - Performance improvement
    - ⋆ Fewer matrix-vector products
    - ⋆ Less CPU time for dense instances
- Open questions
  - Ensure convergence?
  - Use within nonlinear iteration (e.g., inexact SQP)?
    - ⋆ Solve QPs only inexactly

Conclusions

- Active-set QP solver exploiting hot-starts for similar QP instances
  - Iterative refinement with inequality constraints
  - Accelerated linear solver for fixed active set
    - ⋆ QP as preconditioner
  - Performance improvement
    - ⋆ Fewer matrix-vector products
    - ⋆ Less CPU time for dense instances
- Open questions
  - Ensure convergence?
  - Use within nonlinear iteration (e.g., inexact SQP)?
    - ⋆ Solve QPs only inexactly
  - Reference QP:
    - ⋆ Based on preconditioner instead of first QP instance?

Conclusions

- Active-set QP solver exploiting hot-starts for similar QP instances
  - Iterative refinement with inequality constraints
  - Accelerated linear solver for fixed active set
    - ★ QP as preconditioner
  - Performance improvement
    - ★ Fewer matrix-vector products
    - ★ Less CPU time for dense instances
- Open questions
  - Ensure convergence?
  - Use within nonlinear iteration (e.g., inexact SQP)?
    - ★ Solve QPs only inexactly
  - Reference QP:
    - ★ Based on preconditioner instead of first QP instance?
  - Can we handle nonconvex QPs?

Conclusions

- Active-set QP solver exploiting hot-starts for similar QP instances
  - Iterative refinement with inequality constraints
  - Accelerated linear solver for fixed active set
    - ⋆ QP as preconditioner
  - Performance improvement
    - ⋆ Fewer matrix-vector products
    - ⋆ Less CPU time for dense instances
- Open questions
  - Ensure convergence?
  - Use within nonlinear iteration (e.g., inexact SQP)?
    - ⋆ Solve QPs only inexactly
  - Reference QP:
    - ⋆ Based on preconditioner instead of first QP instance?
  - Can we handle nonconvex QPs?
  - Is this useful in the LCP context?
  - ...

THANK YOU!