Joachim Ferreau, ABB Corporate Research (Switzerland), TEMPO Summer School 2015

Embedded Quadratic Programming Using qpOASES

Outline

- Quadratic Programming (QP)
- Model Predictive Control (MPC)
- QP Formulations and Algorithms
- The Online QP Solver qpOASES
- Embedded Applications of qpOASES
- Using qpOASES

Outline

- **Quadratic Programming (QP)**
- Model Predictive Control (MPC)
- QP Formulations and Algorithms
- The Online QP Solver qpOASES
- Embedded Applications of qpOASES
- Using qpOASES

Quadratic Programming
Definition

- A **QP problem** is an optimization problem of the form:

$$\min_{z} \ \frac{1}{2} z' H z + g' z$$

$$s.t. \ \ Az \geq b$$

  - Hessian matrix $H \in \mathcal{S}^n \stackrel{\text{def}}{=} \{ M \in \mathbb{R}^{n \times n} \mid M = M' \}$
  - gradient vector $g \in \mathbb{R}^n$
  - constraint matrix $A \in \mathbb{R}^{m \times n}$
  - constraint vector $b \in \mathbb{R}^m$

- *Note:* many *equivalent* formulations exists

Quadratic Programming
Feasibility and Boundedness

- A QP problem is called **feasible** iff its feasible set

$$\mathcal{F} \stackrel{\text{def}}{=} \{\, z \in \mathbb{R}^n \mid Az \geq b \,\}$$

  is non-empty and **infeasible** otherwise.

- A QP problem is called **bounded (from below)** iff there exists a $\alpha \in \mathbb{R}$ such that

$$\alpha \leq \tfrac{1}{2} z' H z + g' z \quad \forall z \in \mathcal{F}$$

  and **unbounded** otherwise.

Quadratic Programming
Convexity

- A QP problem is called **convex** iff its Hessian matrix is symmetric positive semi-definite, i.e.

$$H \in \mathcal{S}_{+}^{n} \stackrel{\text{def}}{=} \{\, M \in \mathcal{S}^{n} \mid x'Mx \geq 0 \;\forall x \in \mathbb{R}^{n} \,\}$$

- It is called **strictly convex** iff its Hessian matrix is symmetric positive definite, i.e.

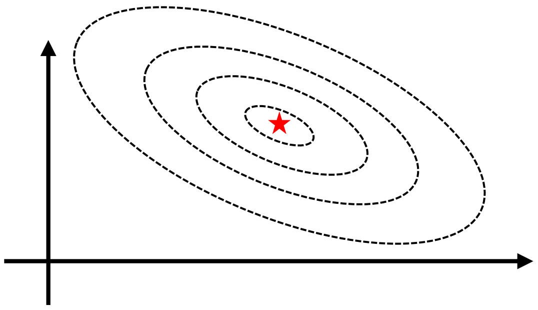
$$H \in \mathcal{S}_{++}^{n} \stackrel{\text{def}}{=} \{\, M \in \mathcal{S}^{n} \mid x'Mx > 0 \;\forall x \in \mathbb{R}^{n} \backslash \{0\} \,\}$$

- Every strictly convex QP is bounded from below.
- **Every strictly convex *and* feasible QP has a (unique) solution!**

Quadratic Programming
Constraints

- **unconstrained** QP

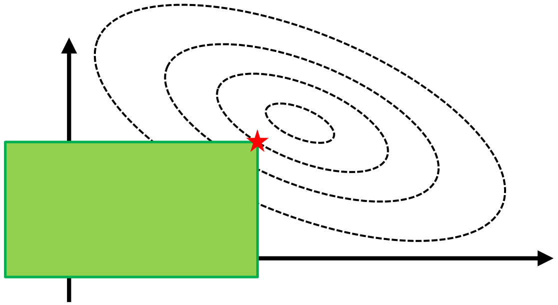
$$\min_{z} \tfrac{1}{2} z'Hz + g'z$$



- QP with **simple constraints**

$$\min_{z} \tfrac{1}{2} z'Hz + g'z$$

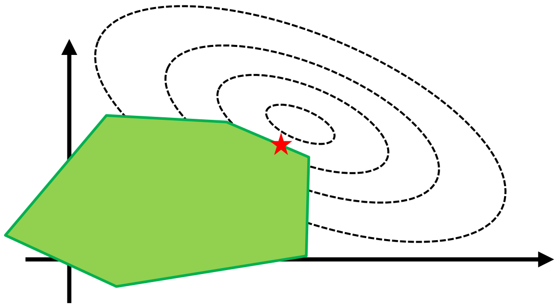
$$s.t. \ \ \underline{z} \leq z \leq \overline{z}$$



- QP with **general constraints**

$$\min_{z} \tfrac{1}{2} z'Hz + g'z$$

$$s.t. \ \ \underline{z} \leq Az \leq \overline{z}$$



Quadratic Programming
Active constraints

- Let a feasible QP problem be given. A constraint $A_i' z \geq b_i$ is called **active at $\hat{\boldsymbol{z}}$** iff $A_i' \hat{z} = b_i$ holds and inactive otherwise.

- We define the (disjoint) index sets:

$$\mathbb{A}(\hat{z}) \stackrel{\text{def}}{=} \{\, i \in \{1, \dots, m\} \mid A_i' \hat{z} = b_i \,\}$$

$$\mathbb{I}(\hat{z}) \stackrel{\text{def}}{=} \{\, i \in \{1, \dots, m\} \mid A_i' \hat{z} > b_i \,\}$$

- At any solution z^{opt} we call $\mathbb{A}(z^{opt})$ the **optimal active set**.

Quadratic Programming
Duality

- The **dual QP** can be written as:

$$\max_{z,y} \; -\frac{1}{2}z'Hz + b'y$$
$$s.t. \quad Hz + g = A'y$$
$$y \geq 0$$

- **Theorem:** *Dorn (1960)*

  Let a convex *QP* and its dual *QP*^{dual} be given, then

  - if z^{opt} if a solution to *QP*, then there exists a solution (z^{opt}, y^{opt}) to *QP*^{dual},
  - if a solution (z^{opt}, y^{opt}) to *QP*^{dual} exists, then a solution z^* to *QP* satisfying $Hz^* = Hz^{opt}$ exists,
  - in either case, the following holds:

$$\frac{1}{2}z^{opt'}Hz^{opt} + g'z^{opt} = -\frac{1}{2}z^{opt'}Hz^{opt} + b'y^{opt}$$

Quadratic Programming
Optimality conditions

- **Theorem:** *Karush (1939), Kuhn/Tucker (1951)*
  Let a strictly convex *QP* be given, then there exists a unique z^{opt}, an index set $\mathbb{A} \subseteq \mathbb{A}(z^{opt})$ and a vector y^{opt} such that:

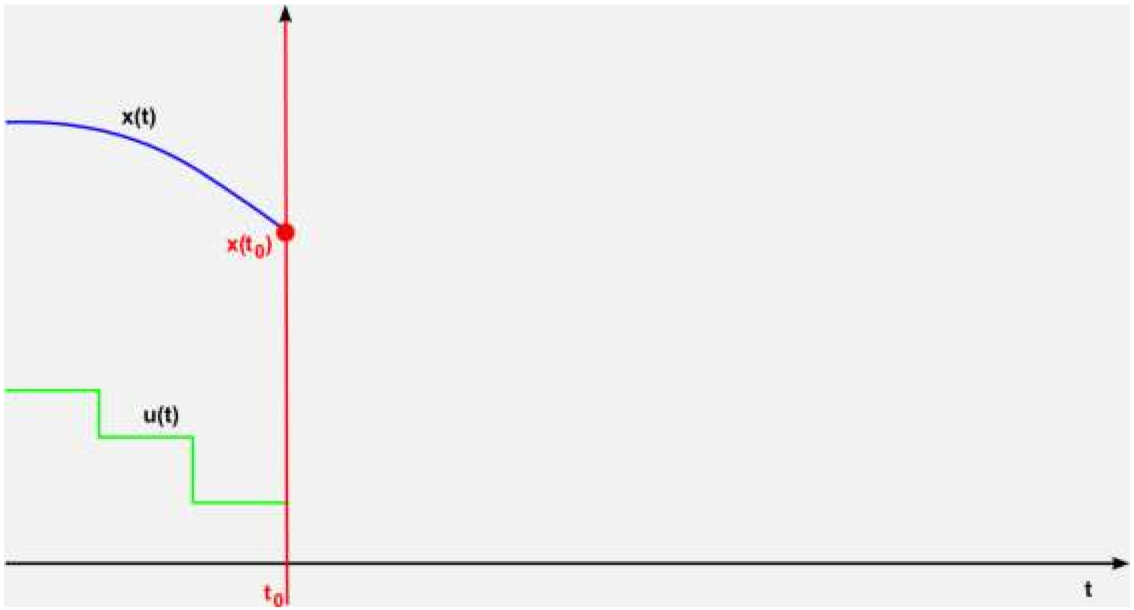
$$
\begin{aligned}
Hz^{opt} + g - A'_{\mathbb{A}} y^{opt} &= 0 \\
A_{\mathbb{A}} z^{opt} &= b_{\mathbb{A}} \\
A_{\mathbb{I}} z^{opt} &\geq b_{\mathbb{I}} \\
y_{\mathbb{A}}^{opt} &\geq 0 \\
y_{\mathbb{I}}^{opt} &= 0
\end{aligned}
$$

- Moreover,
  - z^{opt} is the unique global minimizer of *QP*,
  - (z^{opt}, y^{opt}) is an optimal solution to *QP*^{dual},
  - the optimal objective values of *QP* and *QP*^{dual} are equal.

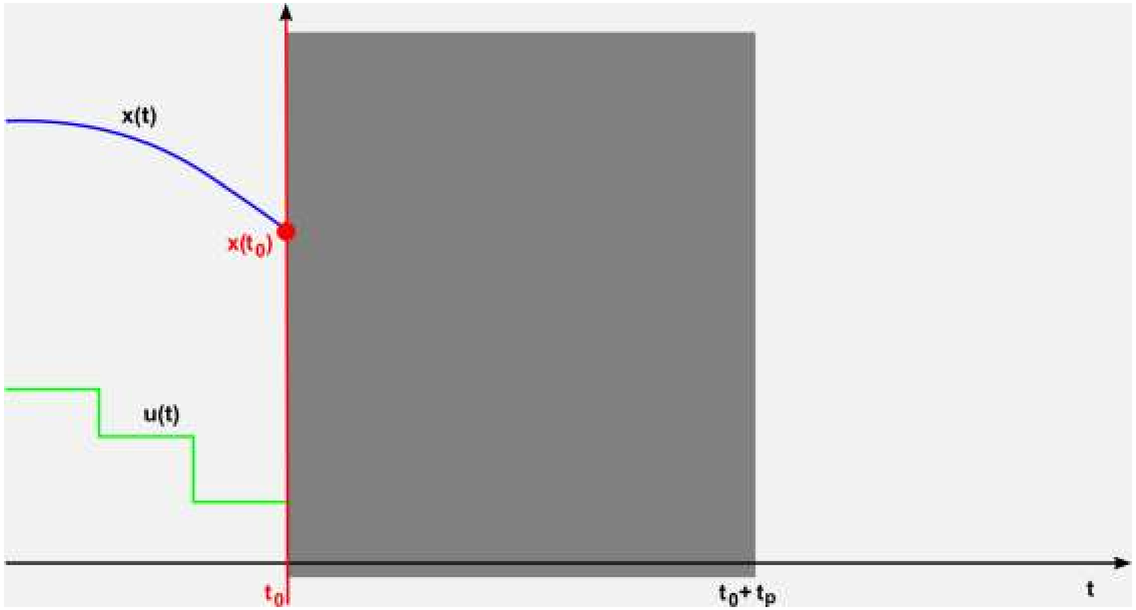
Outline

- Quadratic Programming (QP)
- **Model Predictive Control (MPC)**
- QP Formulations and Algorithms
- The Online QP Solver qpOASES
- Embedded Applications of qpOASES
- Using qpOASES

Model Predictive Control

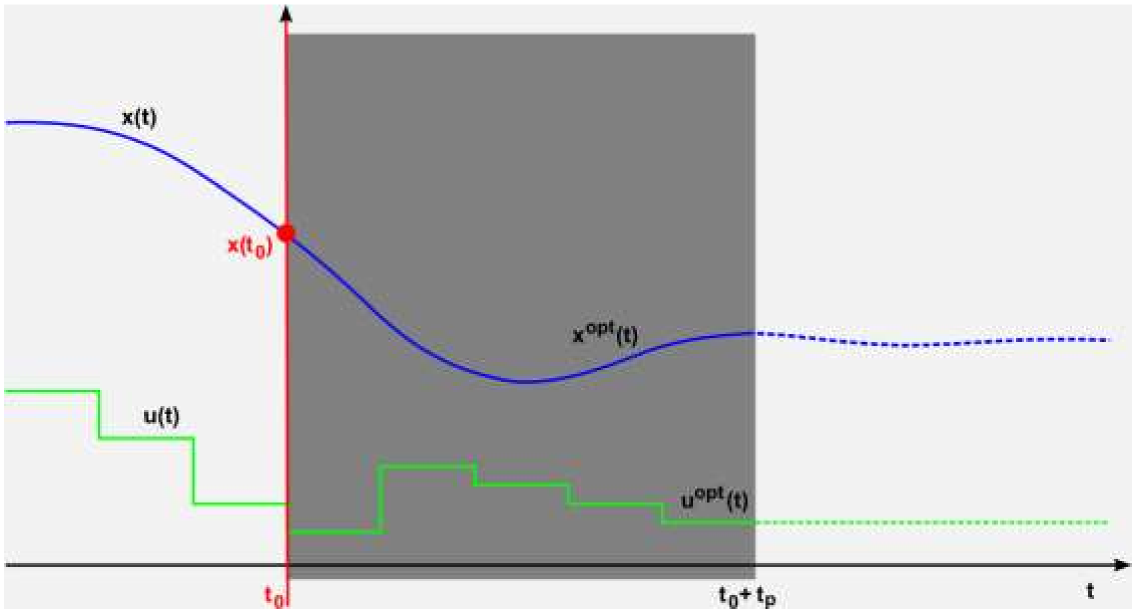


Model Predictive Control



- **Predict future behaviour** based on dynamic model …

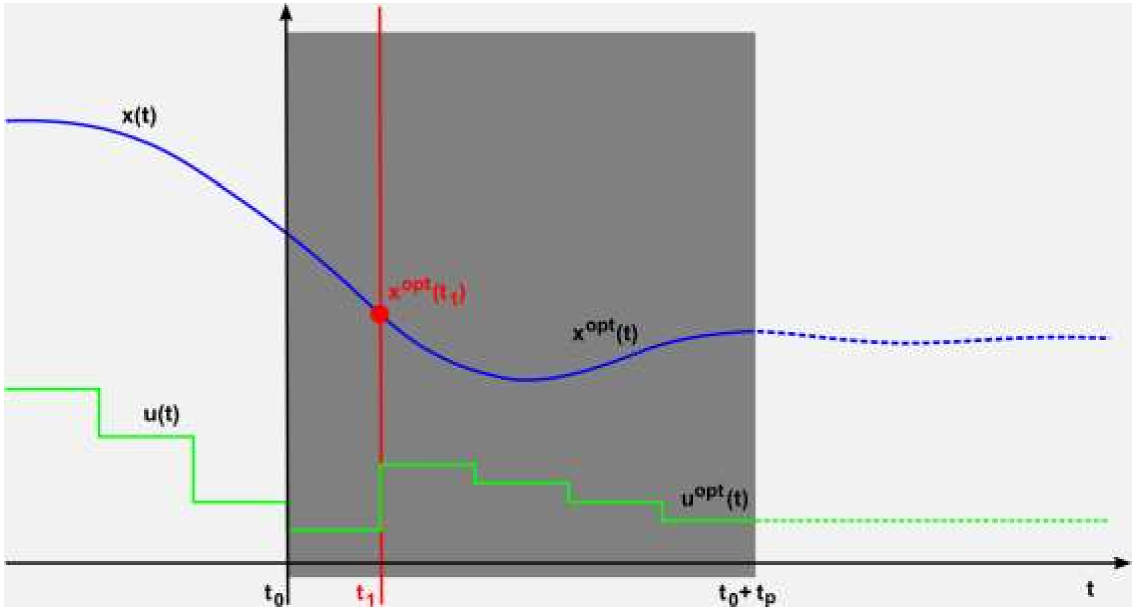
Model Predictive Control



- … and solve an **optimal control problem**:

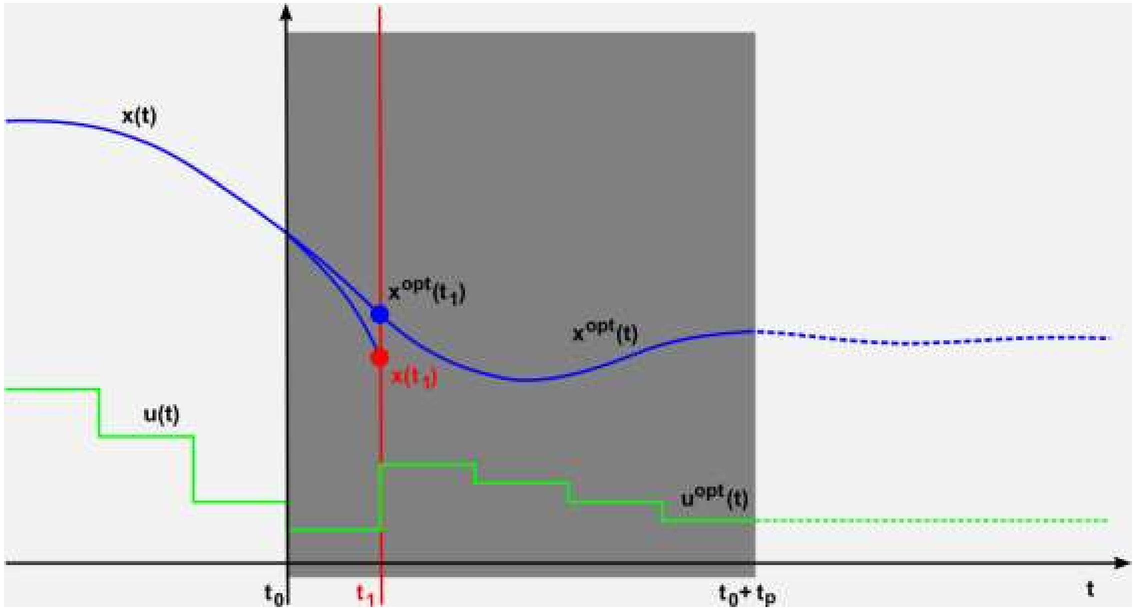
$$
\begin{aligned}
OCP(x_0):\ \min_{x(\cdot),u(\cdot)} \int_{t_0}^{t_0+t_p} & J\big(x(t),u(t)\big)\,dt + P\left(x\big(t_0+t_p\big)\right) \\
s.t.\quad x(t_0) &= \mathrm{x}_0(t_0) \\
\dot{x}(t) &= f\big(x(t),u(t)\big) \quad \forall\, t \in \big[t_0, t_0+t_p\big] \\
0 &\le c\big(x(t),u(t)\big) \quad \forall\, t \in \big[t_0, t_0+t_p\big] \\
0 &\le \tilde{c}\left(x\big(t_0+t_p\big)\right)
\end{aligned}
$$

Model Predictive Control



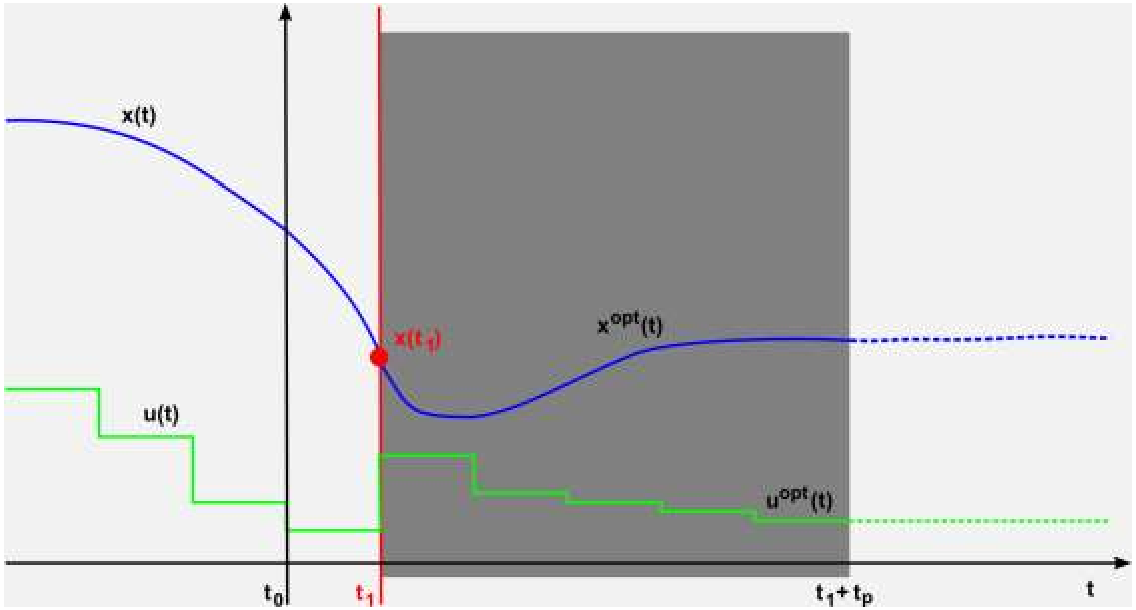
- **Apply** first piece of **optimized control input** …

Model Predictive Control



- … **obtain feedback** from real process …

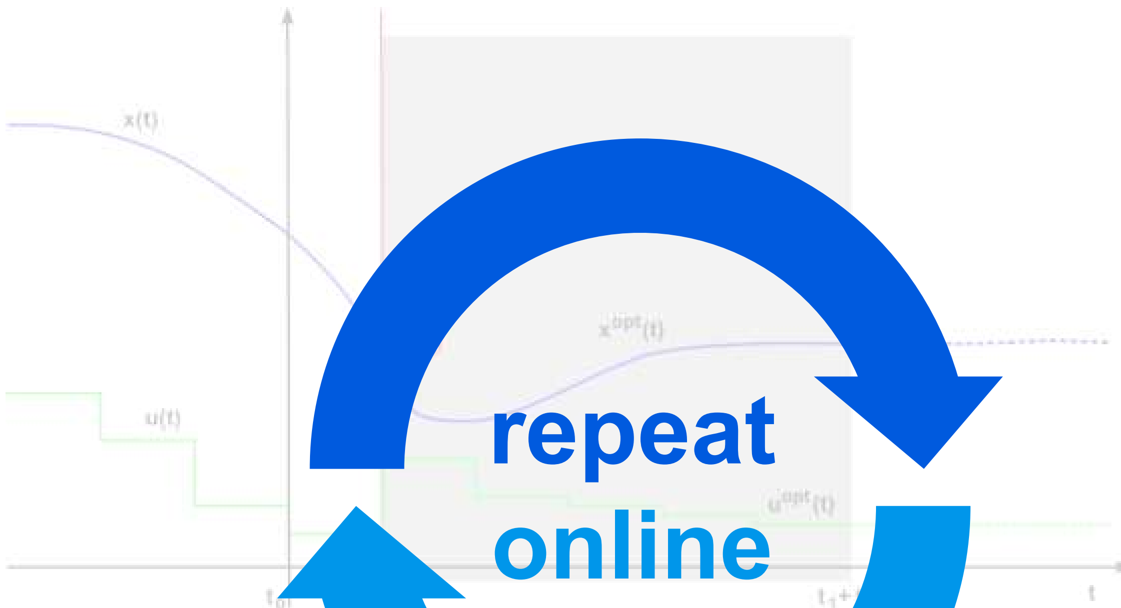
Model Predictive Control



- … and solve **updated** optimal control problem:

$$
\begin{aligned}
OCP(x_0):\ \min_{x(\cdot),u(\cdot)} & \int_{t_1}^{t_1+t_p} J\big(x(t),u(t)\big)\,dt + P\left(x\big(t_1+t_p\big)\right) \\
s.t.\ \ x(t_1) &= \mathrm{x}_0(t_1) \\
\dot{x}(t) &= f\big(x(t),u(t)\big) \quad \forall\, t \in \big[t_1, t_1+t_p\big] \\
0 &\le c\big(x(t),u(t)\big) \quad \forall\, t \in \big[t_1, t_1+t_p\big] \\
0 &\le \tilde{c}\left(x\big(t_1+t_p\big)\right)
\end{aligned}
$$

Model Predictive Control



- … and solve up… d optimal contr… blem:

$$OCP(x_0): \quad \min_{x(\cdot),u(\cdot)} \int_{t_1} \ldots)) \, dt + P\left(x(t_1+t_p)\right)$$

$$\begin{aligned} s.t. \quad x(t_1) &= x_0(t_1) \\ \dot{x}(t) &= f\left(x(t), u(t)\right) \quad \forall\, t \in \left[t_1, t_1+t_p\right] \\ 0 &\leq c\left(x(t), u(t)\right) \quad \forall\, t \in \left[t_1, t_1+t_p\right] \\ 0 &\leq \tilde{c}\left(x(t_1+t_p)\right) \end{aligned}$$

Model Predictive Control
Why solving QP problems?

- Linear (possibly time-varying) MPC leads to QP problems:

$$QP(\mathrm{x}_0): \min_{X,\,U} \; x_{k_0+N}{}^T P x_{k_0+N} + \sum_{k_0}^{k_0+N-1} x_k{}^T Q_k x_k + u_k{}^T R_k u_k$$

$$\begin{aligned} s.t. \quad x_{k_0} &= \mathrm{x}_0 \\ x_{k+1} &= A_k x_k + B_k u_k + c_k \quad && \forall\, k \in \{k_0, \dots, k_0+N-1\} \\ d_k &\leq C_k x_k + D_k u_k && \forall\, k \in \{k_0, \dots, k_0+N-1\} \\ d_{k_0+N} &\leq C_{k_0+N}\, x_{k_0+N} \end{aligned}$$

- Linearizing a nonlinear MPC problem (as done in SQP-type methods) leads to similar (convex) QP problems
- **QP solvers are at the core of most MPC implementations!**

Model Predictive Control
leads to specially structured QP problems

$$QP(\mathrm{x}_0):\ \min_{X,\,U}\ x_{k_0+N}{}^T P x_{k_0+N} + \sum_{k_0}^{k_0+N-1} x_k{}^T Q_k x_k + u_k{}^T R_k u_k$$

$$\begin{aligned} s.t.\quad x_{k_0} &= \mathrm{x}_0 \\ x_{k+1} &= A_k x_k + B_k u_k + c_k \quad \forall\, k \in \{k_0, \dots, k_0+N-1\} \\ d_k &\le C_k x_k + D_k u_k \qquad\quad \forall\, k \in \{k_0, \dots, k_0+N-1\} \\ d_{k_0+N} &\le C_{k_0+N}\, x_{k_0+N} \end{aligned}$$

- **Sparsity**
  - variables are very loosely coupled
  - effect becomes the more pronounced the longer the horizon is
- **Parametric Dependency**
  - QP problems have strong similarity
  - re-use of previous solution helps finding current one

Outline

- Quadratic Programming (QP)
- Model Predictive Control (MPC)
- **QP Formulations and Algorithms**
- The Online QP Solver qpOASES
- Embedded Applications of qpOASES
- Using qpOASES

QP Formulations
Sparsity pattern

- An MPC-QP can be written as (with $Z = \left(x_{k_0}, u_{k_0}, x_{k_0+1}, u_{k_0+1}, \ldots, x_{k_0+N}\right)$):

$$
QP(\mathrm{x}_0): \quad \min_Z \; Z' \begin{pmatrix} Q_{k_0} & & & & & \\ & R_{k_0} & & & & \\ & & \ddots & & & \\ & & & Q_{k_0+N-1} & & \\ & & & & R_{k_0+N-1} & \\ & & & & & P \end{pmatrix} Z
$$

$$
s.t. \begin{pmatrix} Id & & & & & & & \\ A_{k_0} & B_{k_0} & -Id & & & & & \\ & & A_{k_0+1} & B_{k_0+1} & -Id & & & \\ & & & \ddots & & & & \\ & & & & & A_{k_0+N-1} & B_{k_0+N-1} & -Id \end{pmatrix} Z = \begin{pmatrix} x_0 \\ -c_{k_0} \\ -c_{k_0+1} \\ \vdots \\ -c_{k_0+N-1} \end{pmatrix}
$$

$$
\begin{pmatrix} C_{k_0} & D_{k_0} & & & & & \\ & & C_{k_0+1} & D_{k_0+1} & & & \\ & & & \ddots & & & \\ & & & & C_{k_0+N-1} & D_{k_0+N-1} & \\ & & & & & & C_{k_0+N} \end{pmatrix} Z \geq \begin{pmatrix} d_{k_0} \\ d_{k_0+1} \\ \vdots \\ d_{k_0+N-1} \\ d_{k_0+N} \end{pmatrix}
$$

QP Formulations
Sparsity pattern

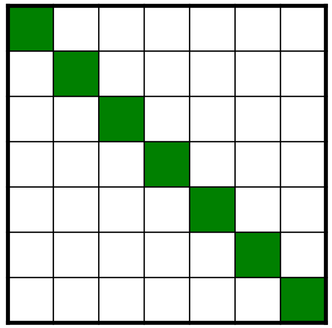
- This can be re-written as (with $Z = \left(x_{k_0}, u_{k_0}, x_{k_0+1}, u_{k_0+1}, \dots, x_{k_0+N}\right)$):

$$QP(\mathrm{x}_0): \quad \min_Z \; Z' \begin{pmatrix} Q_{k_0} & & & & & \\ & R_{k_0} & & & & \\ & & \ddots & & & \\ & & & Q_{k_0+N-1} & & \\ & & & & R_{k_0+N-1} & \\ & & & & & P \end{pmatrix} Z \qquad Z'\boldsymbol{H}Z$$

$$s.t. \quad \begin{pmatrix} Id & & & & & & \\ A_{k_0} & B_{k_0} & -Id & & & & \\ & & A_{k_0+1} & B_{k_0+1} & -Id & & \\ & & & & \ddots & & \\ & & & & A_{k_0+N-1} & B_{k_0+N-1} & -Id \end{pmatrix} Z = \begin{pmatrix} x_0 \\ -c_{k_0} \\ -c_{k_0+1} \\ \vdots \\ -c_{k_0+N-1} \end{pmatrix} \qquad \boldsymbol{A_{eq}}Z = \boldsymbol{b_{eq}(x_0)}$$

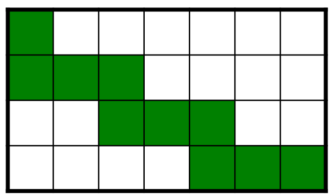
$$\begin{pmatrix} C_{k_0} & D_{k_0} & & & & \\ & & C_{k_0+1} & D_{k_0+1} & & \\ & & & \ddots & & \\ & & & C_{k_0+N-1} & D_{k_0+N-1} & \\ & & & & & C_{k_0+N} \end{pmatrix} Z \geq \begin{pmatrix} d_{k_0} \\ d_{k_0+1} \\ \vdots \\ d_{k_0+N-1} \\ d_{k_0+N} \end{pmatrix} \qquad \boldsymbol{A_{ieq}}Z \geq \boldsymbol{b_{ieq}}$$

QP Formulations
Exploiting sparsity (using sparse solver)

$\boldsymbol{H} =$ 

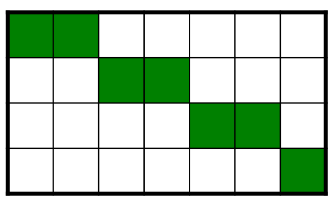
Dimension: $\left((n_x + n_u)(N-1) + n_x\right)^2$

#Nonzeros: $\left((n_x{}^2 + n_u{}^2)N + n_x{}^2\right)$

$\boldsymbol{A_{eq}} =$ 

Dimension: $\left((n_x + n_u)(N-1) + n_x\right) \cdot n_x\, N$

#Nonzeros: $\left((n_x{}^2 + n_x n_u)(N-1) + n_x N\right)$

$\boldsymbol{A_{ieq}} =$ 

Dimension: $\left((n_x + n_u)(N-1) + n_x\right)^2$

#Nonzeros: $\left((n_x + n_u)(N-1) + n_x\right)$

Assumptions: 1) Q, R, P, A, B dense, 2) input/state bounds

QP Formulations
Exploiting sparsity (using state-elimination)

- **All states** are uniquely determined by x_0 and U, thus they **can be easily eliminated from the QP (condensing):** *Bock, Plitt (1984)*

$$QP(\mathrm{x}_0): \quad \min_{Z} \; Z' \begin{pmatrix} Q_{k_0} & & & & & \\ & R_{k_0} & & & & \\ & & \ddots & & & \\ & & & Q_{k_0+N-1} & & \\ & & & & R_{k_0+N-1} & \\ & & & & & P \end{pmatrix} Z$$

$$s.t. \begin{pmatrix} Id & & & & & & & & \\ A_{k_0} & B_{k_0} & -Id & & & & & & \\ & & A_{k_0+1} & B_{k_0+1} & -Id & & & & \\ & & & & & \ddots & & & \\ & & & & & & A_{k_0+N-1} & B_{k_0+N-1} & -Id \end{pmatrix} Z = \begin{pmatrix} x_0 \\ -c_{k_0} \\ -c_{k_0+1} \\ \vdots \\ -c_{k_0+N-1} \end{pmatrix}$$

$$\begin{pmatrix} C_{k_0} & D_{k_0} & & & & & \\ & & C_{k_0+1} & D_{k_0+1} & & & \\ & & & & \ddots & & \\ & & & & C_{k_0+N-1} & D_{k_0+N-1} & \\ & & & & & & C_{k_0+N} \end{pmatrix} Z \geq \begin{pmatrix} d_{k_0} \\ d_{k_0+1} \\ \vdots \\ d_{k_0+N-1} \\ d_{k_0+N} \end{pmatrix}$$

QP Formulations
Exploiting sparsity (using state-elimination)

- **All states** are uniquely determined by w_0 and U, thus they **can be easily eliminated from the QP (condensing):**

$$QP(\mathrm{x}_0): \quad \min_{Z} \; Z' \begin{pmatrix} Q_{k_0} & & & & & \\ & R_{k_0} & & & & \\ & & \ddots & & & \\ & & & Q_{k_0+N-1} & & \\ & & & & R_{k_0+N-1} & \\ & & & & & P \end{pmatrix} Z$$

$$s.t. \quad \begin{pmatrix} Id & & & & & & \\ A_{k_0} & B_{k_0} & -Id & & & & \\ & & A_{k_0+1} & B_{k_0+1} & -Id & & \\ & & & & \ddots & & \\ & & & & A_{k_0+N-1} & B_{k_0+N-1} & -Id \end{pmatrix} Z = \begin{pmatrix} x_0 \\ -c_{k_0} \\ -c_{k_0+1} \\ \vdots \\ -c_{k_0+N-1} \end{pmatrix}$$

$$Z = \boldsymbol{E}U + \boldsymbol{f}(\boldsymbol{x}_0)$$

$$\begin{pmatrix} C_{k_0} & D_{k_0} & & & & & \\ & & C_{k_0+1} & D_{k_0+1} & & & \\ & & & & \ddots & & \\ & & & & C_{k_0+N-1} & D_{k_0+N-1} & \\ & & & & & & C_{k_0+N} \end{pmatrix} Z \geq \begin{pmatrix} d_{k_0} \\ d_{k_0+1} \\ \vdots \\ d_{k_0+N-1} \\ d_{k_0+N} \end{pmatrix}$$

QP Formulations
Exploiting sparsity (using state-elimination)

- **All states** are uniquely determined by w_0 and U, thus they **can be easily eliminated from the QP (condensing):**

$$QP(\mathrm{x}_0): \quad \min_{U} \; U' \boldsymbol{E' H E} U + 2 \cdot U' \boldsymbol{E' H f(x_0)}$$

$$s.t. \quad \boldsymbol{A_{ieq}} \boldsymbol{E} U \geq \boldsymbol{b_{ieq}} - \boldsymbol{A_{ieq} f(x_0)}$$

QP Formulations
Exploiting sparsity (using state-elimination)

- **All states** are uniquely determined by w_0 and U, thus they **can be easily eliminated from the QP (condensing):**

$$QP(\mathrm{x}_0): \quad \min_U \; U' \boldsymbol{H^c} U + U' \boldsymbol{g^c(x_0)}$$

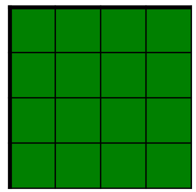
$$s.t. \quad \boldsymbol{A^c} U \geq \boldsymbol{b^c(x_0)}$$

$$\min \; Z' \begin{pmatrix} Q_{k_0} & & & & & \\ & R_{k_0} & & & & \\ & & \ddots & & & \\ & & & Q_{k_0+N-1} & & \\ & & & & R_{k_0+N-1} & \\ & & & & & P \end{pmatrix} Z$$

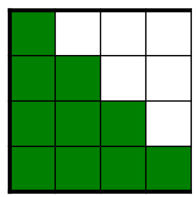
$$s.t. \quad \begin{pmatrix} Id & & & & & & \\ A_{k_0} & B_{k_0} & -Id & & & & \\ & & A_{k_0+1} & B_{k_0+1} & -Id & & \\ & & & & \ddots & & \\ & & & & A_{k_0+N-1} & B_{k_0+N-1} & -Id \end{pmatrix} Z = \begin{pmatrix} x_0 \\ -c_{k_0} \\ -c_{k_0+1} \\ \vdots \\ -c_{k_0+N-1} \end{pmatrix}$$

$$\begin{pmatrix} C_{k_0} & D_{k_0} & & & & \\ & & C_{k_0+1} & D_{k_0+1} & & \\ & & & \ddots & & \\ & & & & C_{k_0+N-1} & D_{k_0+N-1} & \\ & & & & & & C_{k_0+N} \end{pmatrix} Z \geq \begin{pmatrix} d_{k_0} \\ d_{k_0+1} \\ \vdots \\ d_{k_0+N-1} \\ d_{k_0+N} \end{pmatrix}$$

QP Formulations
Exploiting sparsity (using state-elimination)

$H^c =$ 

#Nonzeros: $(n_u N)^2$

$A^c =$ 

#Nonzeros: $n_x n_u \frac{N(N-1)}{2}$

(assumptions as before)

- Solving the MPC-QP problem:
  1. Eliminate states from QP (also called «condensing»)
  2. Solve smaller-scale QP with a **dense QP solver**
- **Linear MPC: states can be eliminated offline!**

QP Formulations
Number of nonzeros (sparse vs. dense QP)

- QP size for a MPC problem with **5 states, 2 inputs:**

N	2	5	10	20	50
#Elements	$4.1 \cdot 10^2$	$3.0 \cdot 10^3$	$1.3 \cdot 10^4$	$5.2 \cdot 10^4$	$3.3 \cdot 10^5$
#Nonzeros (**sparse QP**)	$1.4 \cdot 10^2$	$3.7 \cdot 10^2$	$7.5 \cdot 10^2$	$1.5 \cdot 10^3$	$3.8 \cdot 10^3$
#Nonzeros (**dense QP**)	$2.6 \cdot 10^1$	$2.0 \cdot 10^2$	$8.5 \cdot 10^2$	$3.5 \cdot 10^3$	$2.2 \cdot 10^4$

- QP size for a MPC problem with **30 states, 5 inputs:**

N	2	5	10	20	50
#Elements	$1.2 \cdot 10^4$	$8.3 \cdot 10^4$	$3.4 \cdot 10^5$	$1.4 \cdot 10^6$	$8.7 \cdot 10^6$
#Nonzeros (**sparse QP**)	$3.9 \cdot 10^3$	$1.0 \cdot 10^4$	$2.0 \cdot 10^4$	$4.1 \cdot 10^4$	$1.0 \cdot 10^5$
#Nonzeros (**dense QP**)	$2.5 \cdot 10^2$	$2.1 \cdot 10^3$	$9.3 \cdot 10^3$	$3.9 \cdot 10^4$	$2.5 \cdot 10^5$

(assumptions as before)

QP Formulations
Number of nonzeros (sparse vs. dense QP)

- QP size for a MPC problem with **5 states, 2 inputs:**

N	2	5	10	20	50
#Elements	$4.1 \cdot 10^2$	$3.0 \cdot 10^3$	$1.3 \cdot 10^4$	$5.2 \cdot 10^4$	$3.3 \cdot 10^5$
#Nonzeros (sparse QP)	$1.4 \cdot 10^2$	$3.7 \cdot 10^2$	$7.5 \cdot 10^2$	$1.5 \cdot 10$ [illegible]	$3.8 \cdot 10^3$
#Nonzeros (dense QP)	$2.6 \cdot 10^1$	$2.0 \cdot 10^2$	$8.5 \cdot 10^2$	$3.5 \cdot 10^3$	$2.2 \cdot 10^4$

Sparse formulation is advantageous whenever

- QP size for a MPC problem with **30 states, 5 inputs:**

N	2	5	10	20	50
#Elements	$1.2 \cdot 10^4$	$8.3 \cdot 10^4$	$3.4 \cdot 10^5$	$1.4 \cdot 10^6$	$8.7 \cdot 10^6$
#Nonzeros (sparse QP)	$3.9 \cdot 10^3$	$1.0 \cdot 10^4$	$2.0 \cdot 10^4$	$4.1 \cdot 10^4$	$1.0 \cdot 10^5$
#Nonzeros (dense QP)	$2.5 \cdot 10^2$	$2.1 \cdot 10^3$	$9.3 \cdot 10^3$	$3.9 \cdot 10^4$	$2.5 \cdot 10^5$

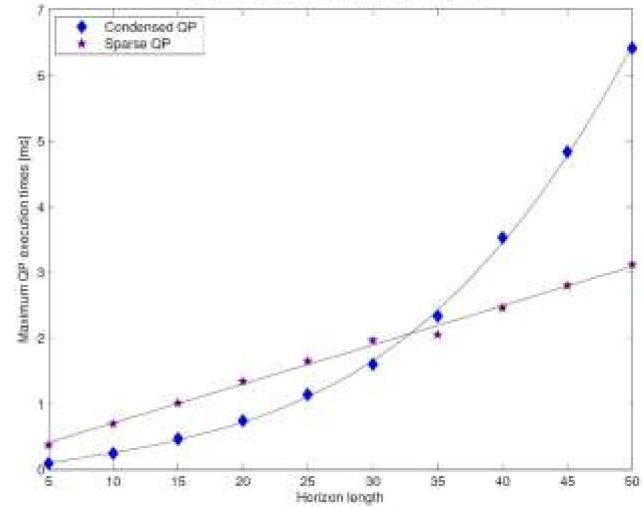
$$\frac{n_x}{n_u} \ll N$$

(assumptions as before)

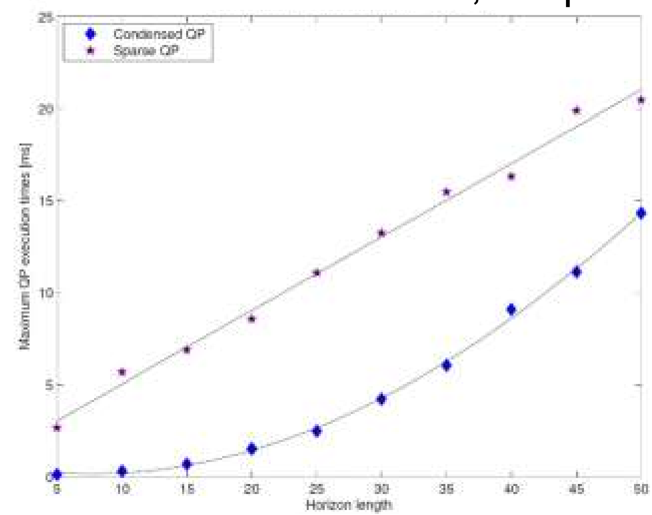
QP Formulations
An example

- Nonlinear MPC example (spring-masses toy application)
- **Red:** Time for **solving sparse QP** using an auto-generated IP method (FORCES)
- **Blue:** Time for **state-elimination and solving condensed QP** using an efficient AS method (qpOASES)
- **Remarks:**
  - worst-case execution times
  - severe disturbance, thus no QP warm-starting used

Scenario 1: 9 states, 3 inputs



Scenario 2: 21 states, 3 inputs

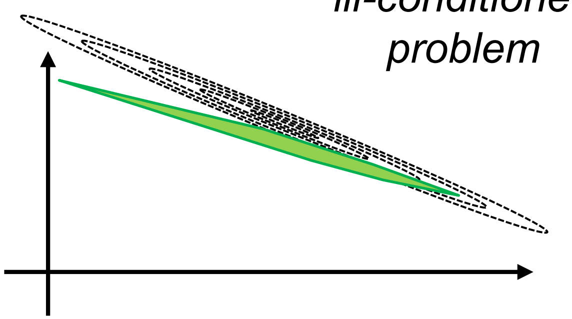


Vukov et al. (2013)

QP Algorithms
Why is there a whole zoo of them?

Fast gradient	gradient method, primal FGM, dual FGM, GPAD, FiOrdOs
Active set	quadprog (primal), QLD (dual), qpOASES (parametric)
Interior point	primal barrier, CVXGEN (primal-dual), FORCES (primal-dual), HPMPC
Others	qpDUNES (Newton-type), PQP, splitting methods (e.g. ADMM), MPT (explicit methods)

- Tailored to different problem classes
- Different numerical properties
- Amount of sorce code
- Suitability for parallelization
- Suitability for FPGA implementations



QP Algorithms
A limited and rough overview

- **Fast gradient methods:**
  - compute step towards solution of unconstrained QP
  - project to feasible set (difficult for general constraints)
- **Active-set methods:**
  - guess which inequalities hold with equality at solution
  - solve resulting equality-constrained QP (almost trivial)
  - check if guess was correct, update guess if not
- **Interior-point methods:**
  - remove inequalities, but penalize constraint violations in objective function (non-quadratic term, e.g. logarithmic)
  - solve resulting equality-constrained NLP with Newton's method
- **Explicit methods and others**

QP Algorithms
Some Pros and Cons

- **Fast gradient methods:** (e.g. FiOrdOs)
  - plenty of cheap iterations, variants for both dense/sparse QPs
  - **Pros:** simple to code (no matrix inversion), easy to parallelize
  - **Cons:** sensitive to problem formulation, limited warm-starting
- **Active set methods:** (e.g. quadprog, qpOASES)
  - many cheap iterations, most efficient for dense QPs
  - **Pros:** efficient warm-starting, can be made very reliable
  - **Cons:** difficult to parallelize, only heuristic runtime bound
- **Interior point methods:** (e.g. IPOPT, OOQP, FORCES)
  - few expensive iterations, most efficient for sparse QPs
  - **Pros:** runtime guarantee, quite easy to parallelize
  - **Cons:** limited warm-starting

Outline

- Quadratic Programming (QP)
- Model Predictive Control (MPC)
- QP Formulations and Algorithms
- **The Online QP Solver qpOASES**
- Embedded Applications of qpOASES
- Using qpOASES

Parametric Quadratic Programming
Definition

- A **parametric QP problem** is an optimization problem of the form:

$$QP(x_0): \quad \min_{z} \ \tfrac{1}{2} z'Hz + g(x_0)'z$$
$$s.t. \ \ Az \geq b(x_0)$$

  - gradient vector $g(x_0) = f + Fx_0$
  - constraint vector $b(x_0) = e + Ex_0$
  - parameter $x_0 \in \mathbb{R}^p$

- For a fixed x_0, one yields a standard QP problem

Parametric Quadratic Programming
Set of feasible parameters

- Recall the definition of the **feasible set** for $QP(x_0)$, x_0 given:

$$\mathcal{F}(x_0) \stackrel{\text{def}}{=} \{\, z \in \mathbb{R}^n \mid Az \geq b(x_0) \,\}$$

- We define the **set of feasible parameters** as follows:

$$\mathcal{P} \stackrel{\text{def}}{=} \{\, x_0 \in \mathbb{R}^p \mid \mathcal{F}(x_0) \neq \emptyset \,\}$$

- **Theorem:** *Berkelaar, Roos, Terkaly (1997)*

  The set $\mathcal{P}$ of feasible parameters is convex and closed.

Parametric Quadratic Programming
Critical regions

- Let a strictly convex $QP(x_0)$ be given. For each $x_0 \in \mathcal{P}$ let $z^{opt}(x_0)$ denote the optimal solution with corresponding optimal active set $\mathbb{A}\big(z^{opt}(x_0)\big)$.
- Then, for every index set $\mathbb{A} \subseteq \{1, \dots, m\}$, the set

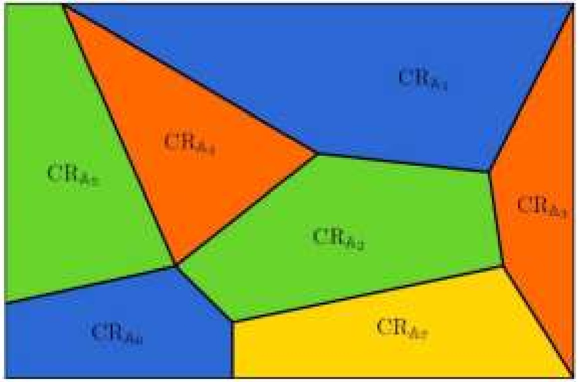
$$\mathcal{CR}_{\mathbb{A}} \stackrel{\text{def}}{=} \left\{ x_0 \in \mathcal{P} \mid \mathbb{A}\big(z^{opt}(x_0)\big) = \mathbb{A} \right\}$$

  is called a **critical region** of $\mathcal{P}$.

- A critical region contains all parameters x_0 that lead to solutions of $QP(x_0)$ with a certain optimal active set

Parametric Quadratic Programming
Critical regions (cont.)

- **Theorem:** *Bemporad, Morari, Dua, Pistikopoulos (2002)*
  For a strictly convex $QP(x_0)$ the following holds:
  - all closures of **critical regions are closed polyhedra** with pairwise disjoint interiors;
  - the set of feasible parameters $\mathcal{P}$ **can be subdivided into** a finite number of closures of **critical regions**;
  - the **optimal solution** $z^{opt}: \mathcal{P} \rightarrow \mathbb{R}^n$ is a **piecewise affine, continuous** function. *Fiacco (1983), Zafiriou (1990)*



- *Note:* explicit MPC pre-computes this partition offline!

Online Active Set Strategy
Main idea *Ferreau et al. (2008), Best (1996)*

- Let's assume we have solved the last $QP(x_0)$, with optimal solution $z^{opt}(x_0)$:

$$\min_z \ \tfrac{1}{2} z'Hz + g(x_0)'z$$
$$s.t. \ \ Az \geq b(x_0)$$

- Now we want to solve the next one, $QP(x_0^{new})$:

$$\min_z \ \tfrac{1}{2} z'Hz + g(x_0^{new})'z$$
$$s.t. \ \ Az \geq b(x_0^{new})$$

- To this aim, we introduce the following **homotopies:**

$$\widetilde{x_0}\colon [0,1] \to \mathbb{R}^q, \qquad \widetilde{x_0}(\tau) \stackrel{\text{def}}{=} x_0 + \tau(x_0^{new} - x_0)$$
$$\tilde{g}\colon [0,1] \to \mathbb{R}^n, \qquad \tilde{g}(\tau) \stackrel{\text{def}}{=} g(x_0) + \tau\big(g(x_0^{new}) - g(x_0)\big)$$
$$\tilde{b}\colon [0,1] \to \mathbb{R}^m, \qquad \tilde{b}(\tau) \stackrel{\text{def}}{=} b(x_0) + \tau\big(b(x_0^{new}) - b(x_0)\big)$$

Online Active Set Strategy
Main idea

- Let's assume we have solved the last $QP(x_0)$, with optimal solution $z^{opt}(x_0)$ and want to solve the next one, $QP(x_0^{new})$:

- To this aim, we introduce the following **homotopies:**

$$\widetilde{x_0}\colon [0,1] \to \mathbb{R}^q, \qquad \widetilde{x_0}(\tau) \stackrel{\text{def}}{=} x_0 + \tau(x_0^{new} - x_0)$$

$$\tilde{g}\colon [0,1] \to \mathbb{R}^n, \qquad \tilde{g}(\tau) \stackrel{\text{def}}{=} g(x_0) + \tau\big(g(x_0^{new}) - g(x_0)\big)$$

$$\tilde{b}\colon [0,1] \to \mathbb{R}^m, \qquad \tilde{b}(\tau) \stackrel{\text{def}}{=} b(x_0) + \tau\big(b(x_0^{new}) - b(x_0)\big)$$

- And **re-parametrize** the parametric QP:

$$QP(\tau)\colon \quad \min_z \ \tfrac{1}{2} z' H z + \tilde{g}(\tau)' z$$

$$s.t. \ \ Az \geq \tilde{b}(\tau)$$

Online Active Set Strategy
Main idea (cont.)

- We aim at **satisfying the KKT optimality conditions** at each point along the homotopy path:

$$\begin{pmatrix} H & A'_{\tilde{\mathbb{A}}(\tau)} \\ A_{\tilde{\mathbb{A}}(\tau)} & 0 \end{pmatrix} \begin{pmatrix} \tilde{z}^{opt}(\tau) \\ -\tilde{y}^{opt}_{\tilde{\mathbb{A}}(\tau)}(\tau) \end{pmatrix} = \begin{pmatrix} -\tilde{g}(\tau) \\ \tilde{b}_{\tilde{\mathbb{A}}(\tau)}(\tau) \end{pmatrix}$$

$$A_{\tilde{\mathbb{I}}(\tau)} \tilde{z}^{opt}(\tau) \geq \tilde{b}_{\tilde{\mathbb{I}}(\tau)}(\tau)$$

$$\tilde{y}^{opt}_{\tilde{\mathbb{A}}(\tau)}(\tau) \geq 0$$

$$\tilde{y}^{opt}_{\tilde{\mathbb{I}}(\tau)}(\tau) = 0$$

- Since $\tilde{z}^{opt}(\tau)$ is continuous and piecewise affine, we search for **primal-dual step directions** (valid for $\tau \in [0, \tau_{max}]$):

$$\tilde{z}^{opt}(\tau) \stackrel{\text{def}}{=} z^{opt} + \tau \cdot \Delta z^{opt}, \qquad \tilde{y}^{opt}_{\mathbb{A}}(\tau) \stackrel{\text{def}}{=} y^{opt}_{\mathbb{A}} + \tau \cdot \Delta y^{opt}_{\mathbb{A}}$$

Online Active Set Strategy
Main idea (cont.)

- This leads to the «local» KKT optimality conditions:

$$\begin{pmatrix} H & A'_{\mathbb{A}} \\ A_{\mathbb{A}} & 0 \end{pmatrix} \begin{pmatrix} \Delta z^{opt} \\ -\Delta y_{\mathbb{A}}^{opt} \end{pmatrix} = \begin{pmatrix} -g(x_0^{new}) + g(x_0) \\ b_{\mathbb{A}}(x_0^{new}) - b_{\mathbb{A}}(x_0) \end{pmatrix}$$

$$A_{\mathbb{I}}(z^{opt} + \tau \cdot \Delta z^{opt}) \geq b_{\mathbb{I}}(x_0^{new}) - b_{\mathbb{I}}(x_0)$$

$$y_{\mathbb{A}}^{opt} + \tau \cdot \Delta y_{\mathbb{A}}^{opt} \geq 0$$

$$y_{\mathbb{I}}^{opt} + \tau \cdot \Delta y_{\mathbb{I}}^{opt} = 0$$

- **Solving the linear system** yields the primal-dual step direction
- We follow this direction (i.e. **move along the homotopy path**) until any of KKT inequality conditions becomes violated

Online Active Set Strategy
Main idea (cont.)

- The **step length** τ_{max} is computed as follows:

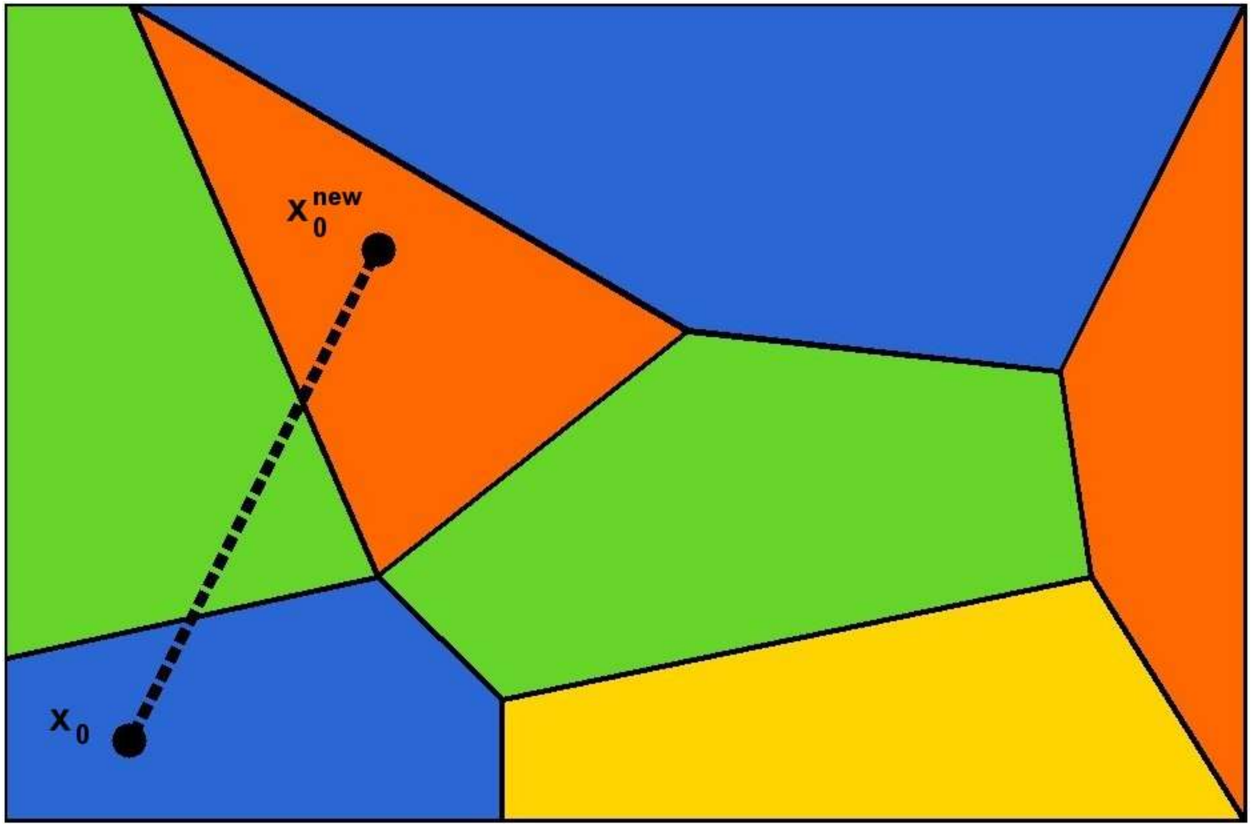
$$\tau_{max}^{prim} \stackrel{\text{def}}{=} \min_{i\in\mathbb{I}} \left\{ \frac{b_i(x_0) - A_i' z^{opt}}{A_i' \Delta z^{opt} - \Delta b_i} \mid A_i' \Delta z^{opt} < \Delta b_i \right\}$$

$$\tau_{max}^{dual} \stackrel{\text{def}}{=} \min_{i\in\mathbb{A}} \left\{ -\frac{y_i^{opt}}{\Delta y_i} \mid \Delta y_i < 0 \right\}$$

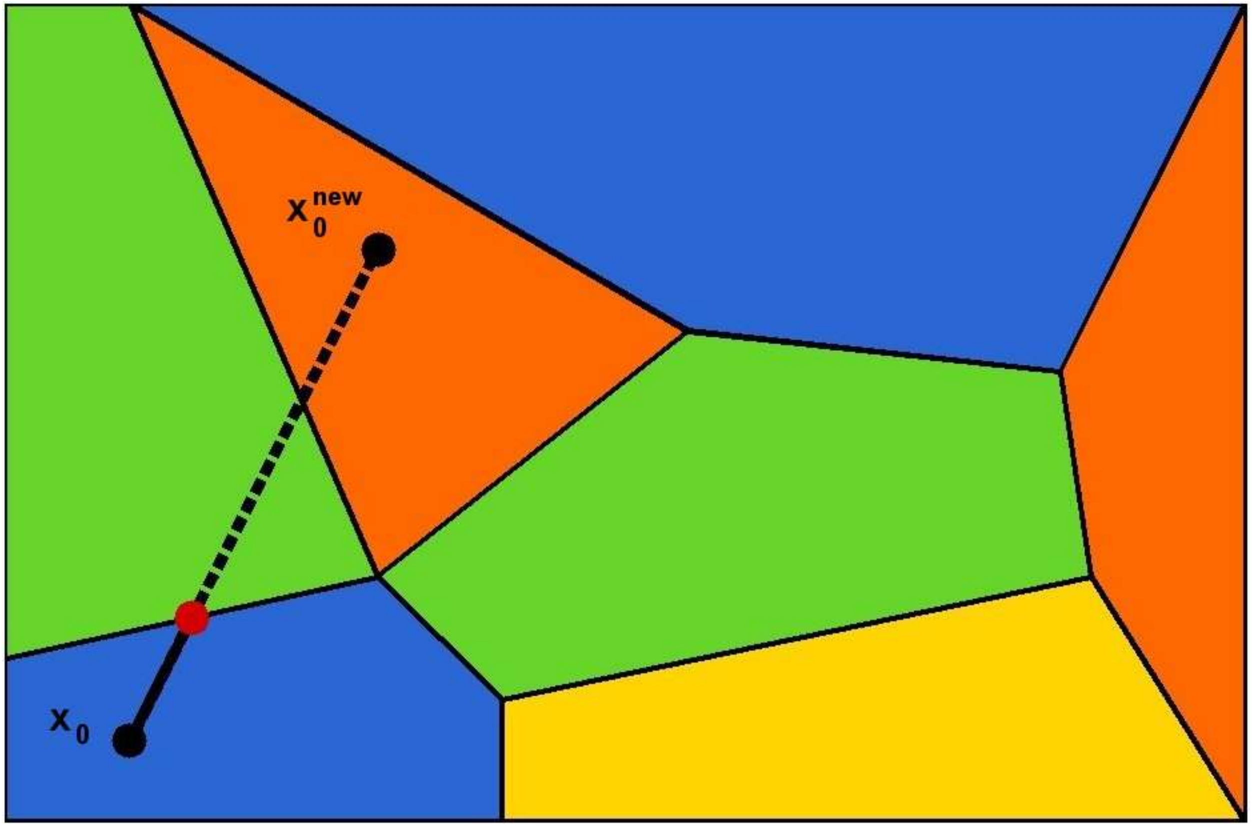
$$\tau_{max} \stackrel{\text{def}}{=} \min\{1, \tau_{max}^{prim}, \tau_{max}^{dual}\} \in [0,1]$$

- If $\tau_{max} = 1$, the optimal solution of $QP(x_0^{new})$ has been found!
- Otherwise, at $\tau = \tau_{max}$ **a constraint is added or removed** from the working set and a new primal-dual step direction is computed

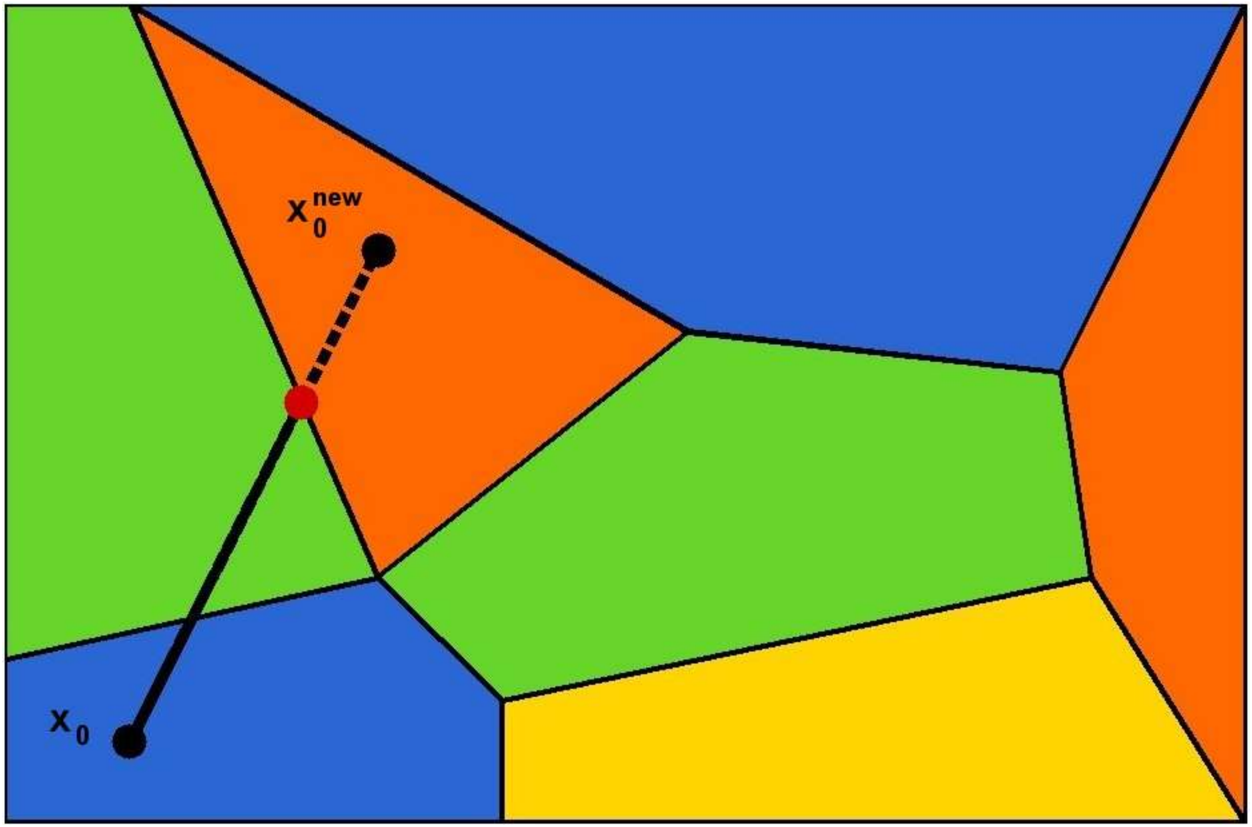
Online Active Set Strategy
Illustration



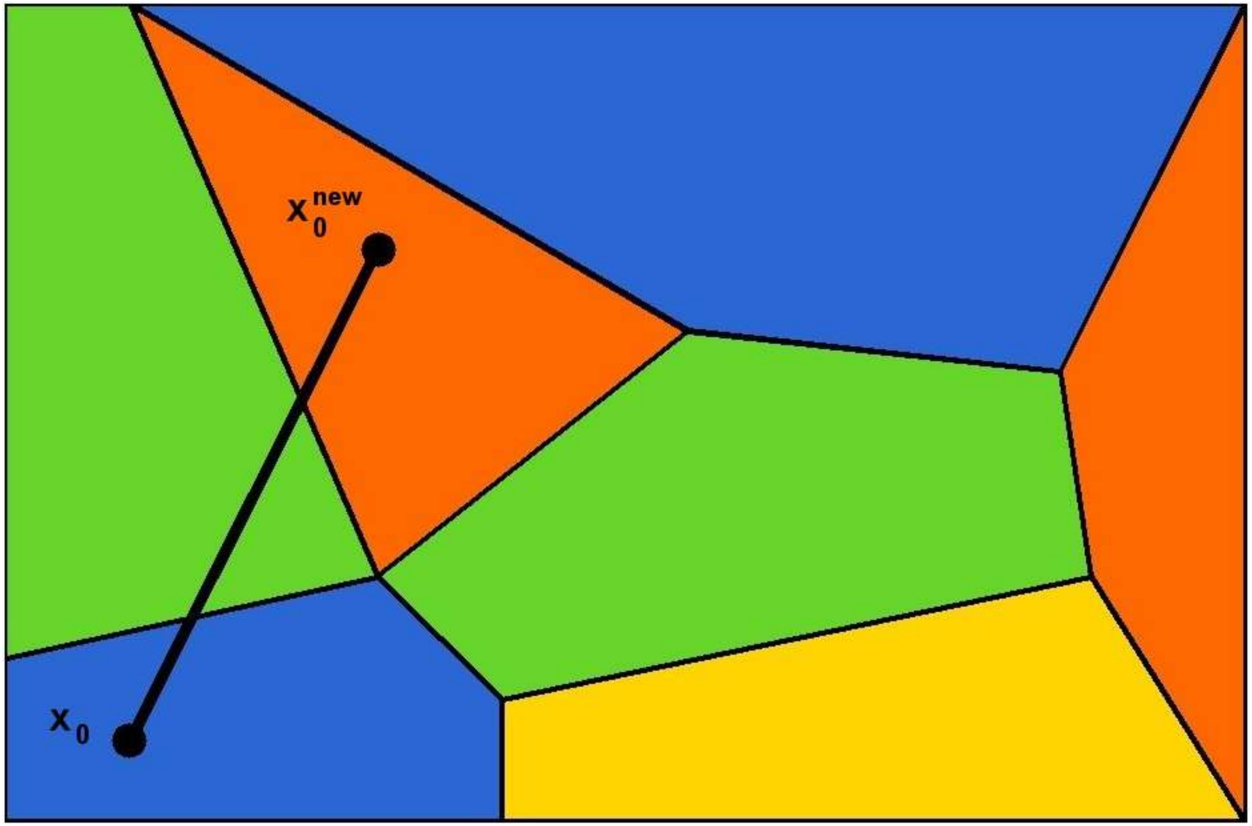
Online Active Set Strategy
Illustration



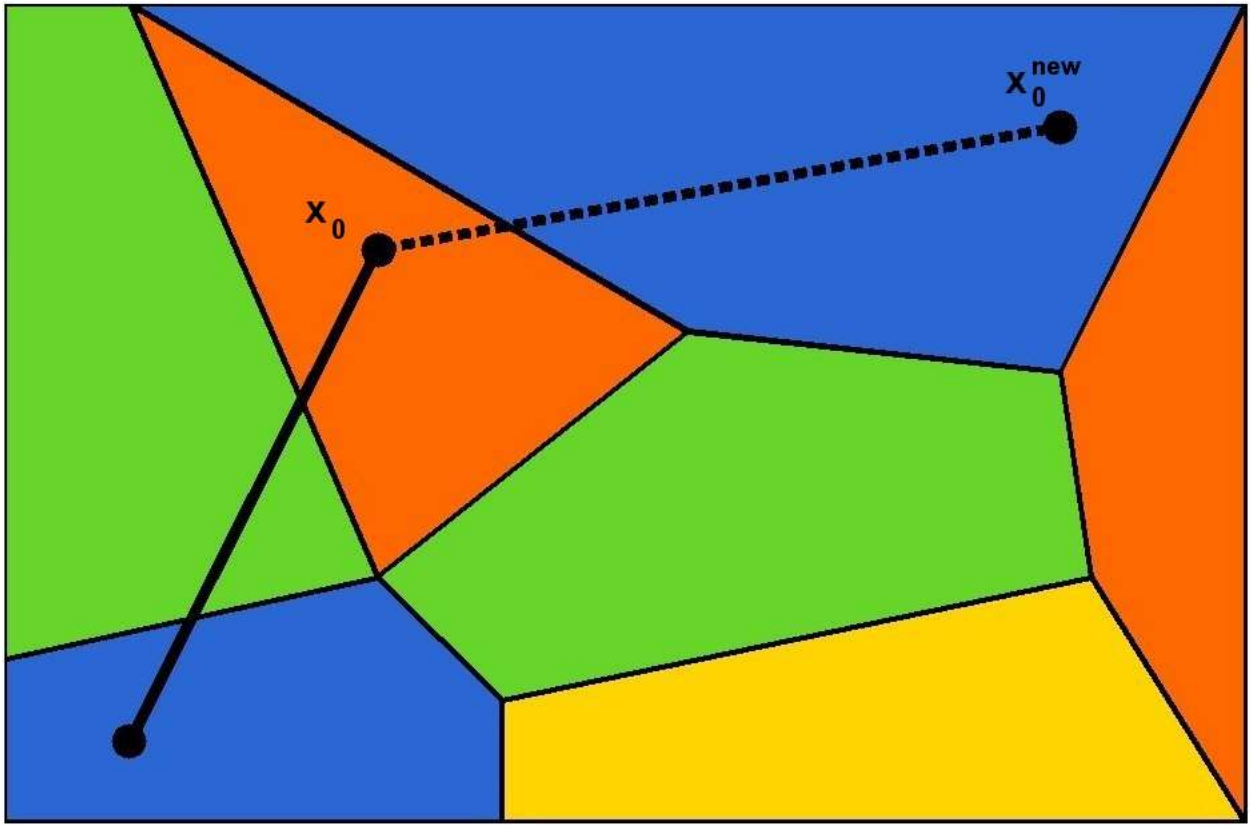
Online Active Set Strategy
Illustration



Online Active Set Strategy
Illustration

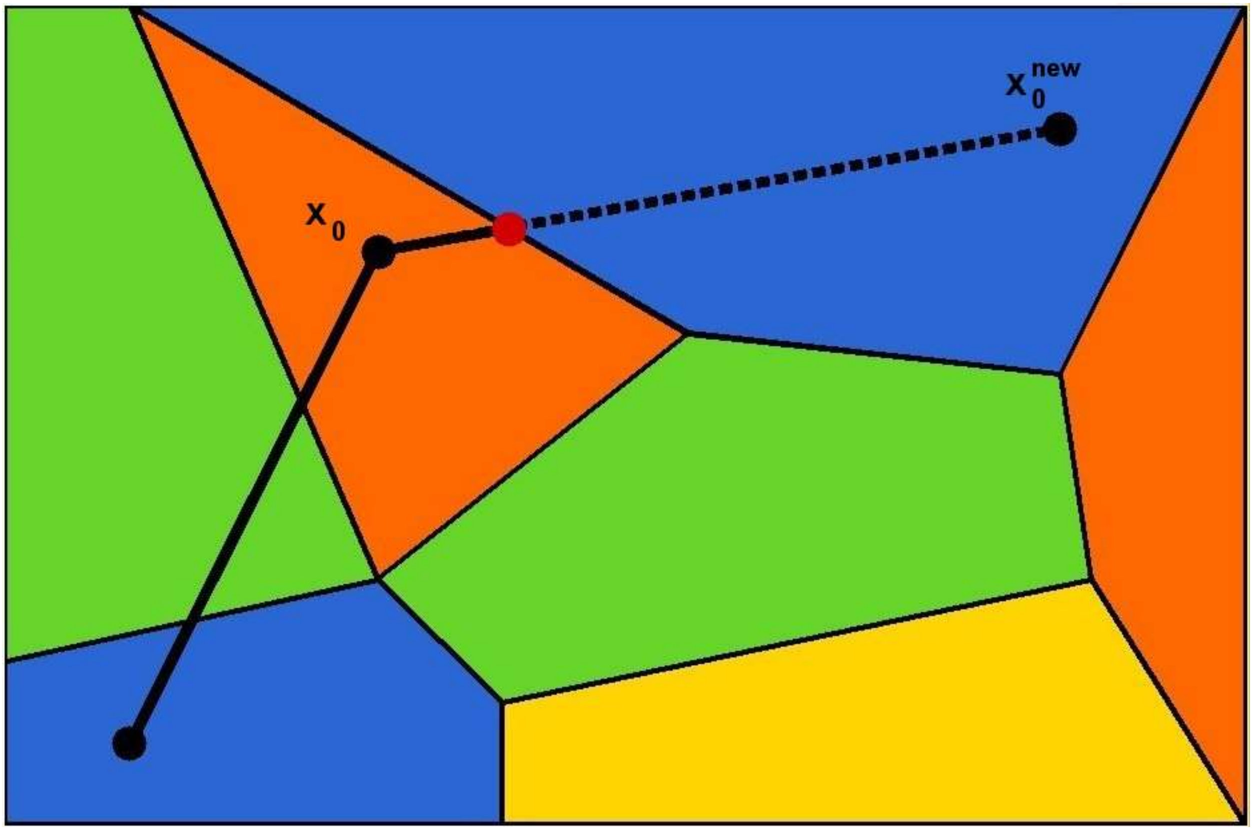


Online Active Set Strategy
Illustration

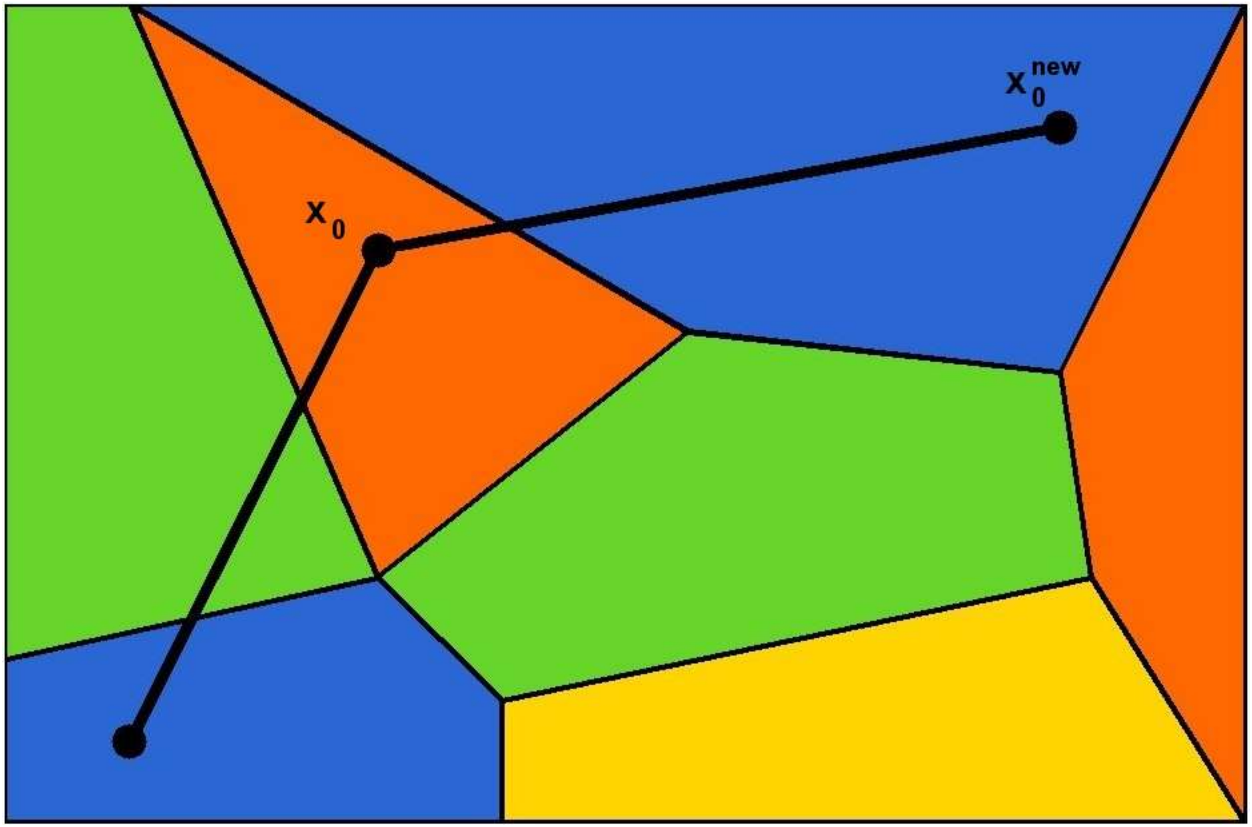


ABB

Online Active Set Strategy
Illustration



Online Active Set Strategy
Illustration

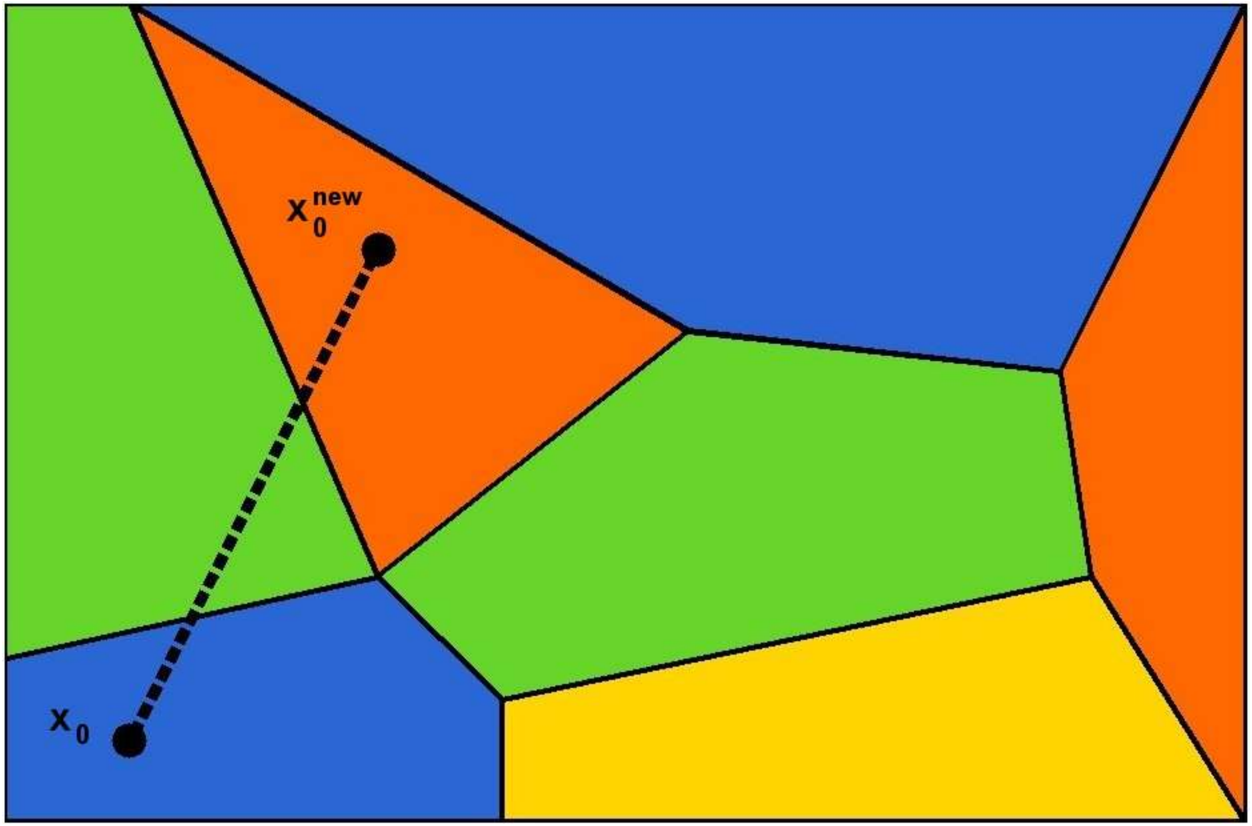


Online Active Set Strategy
Advantages and Limitations

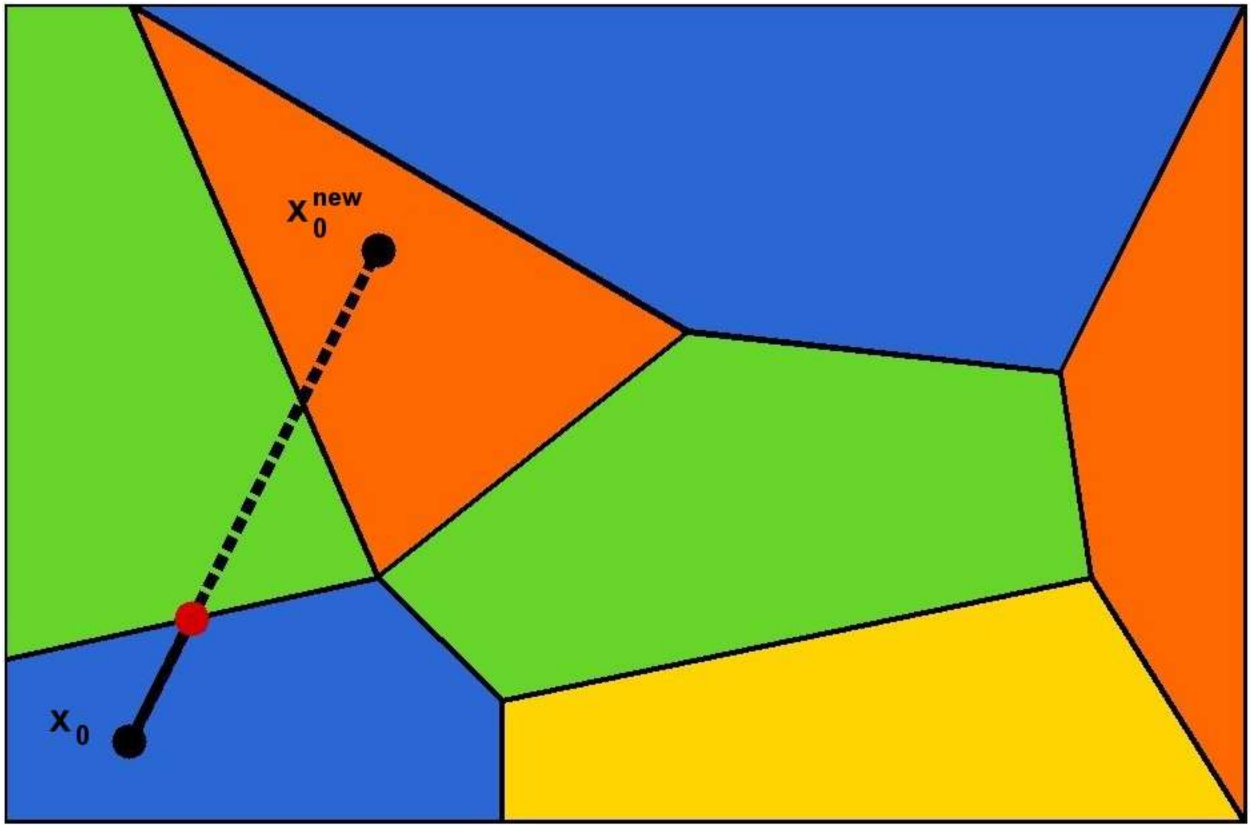
- *Advantages:*
  - Often **fewer number of iterations** by exploiting parametric nature of MPC problem
  - **Hot-starts** with full solution information of previous QP (including re-use of matrix factorizations)
  - **Real-time variant** if procedure has to stop prematurely
  - Homotopy helps to make implementation **numerically robust**

- *Limitations:*
  - Rather complex code (e.g. matrix factorizations/updates)
  - Difficult to parallelize

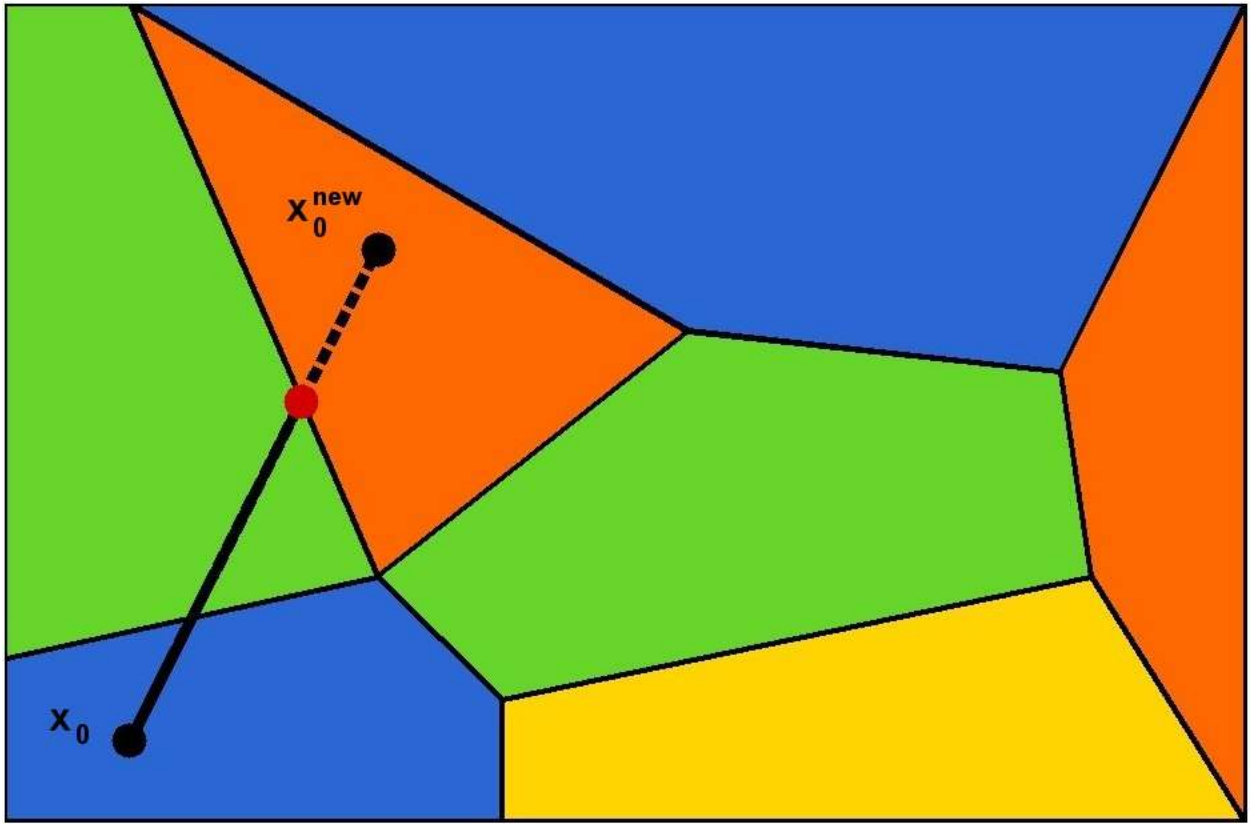
Online Active Set Strategy
Real-time variant



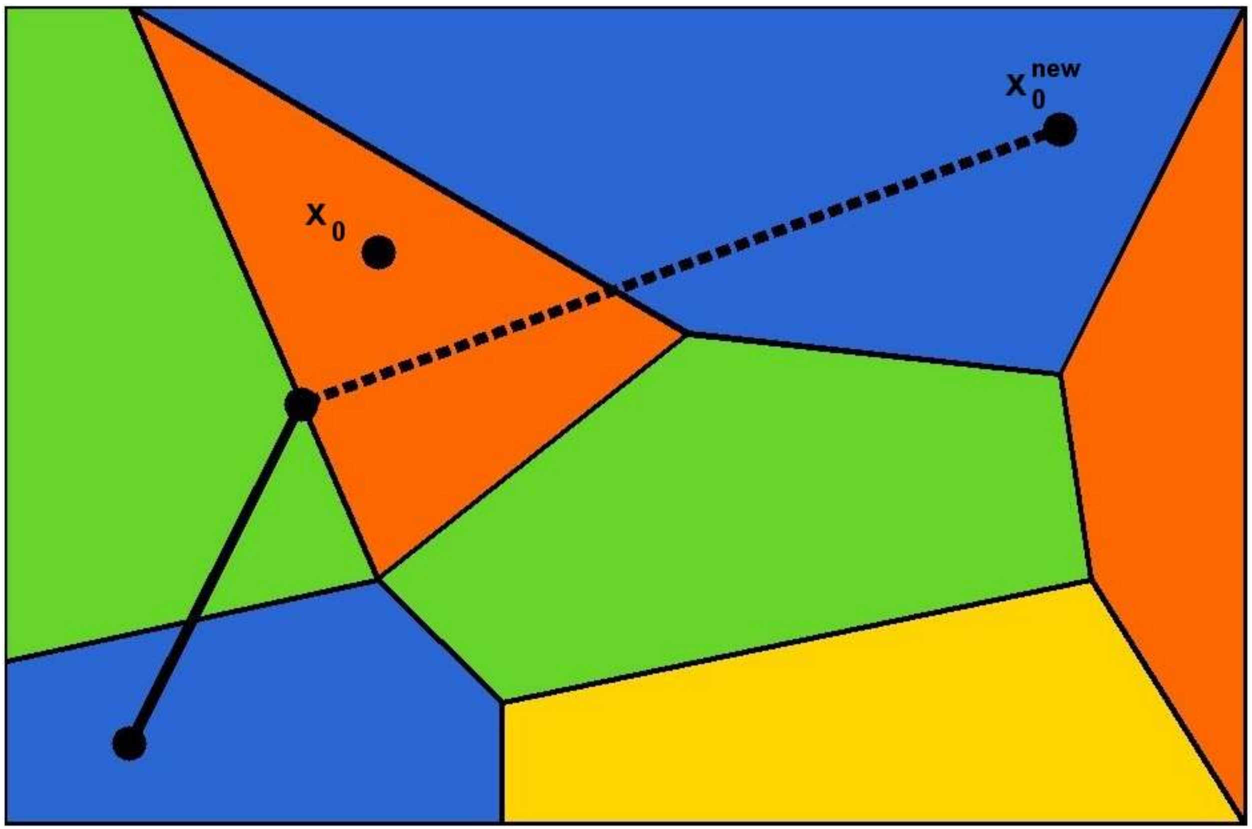
Online Active Set Strategy
Real-time variant



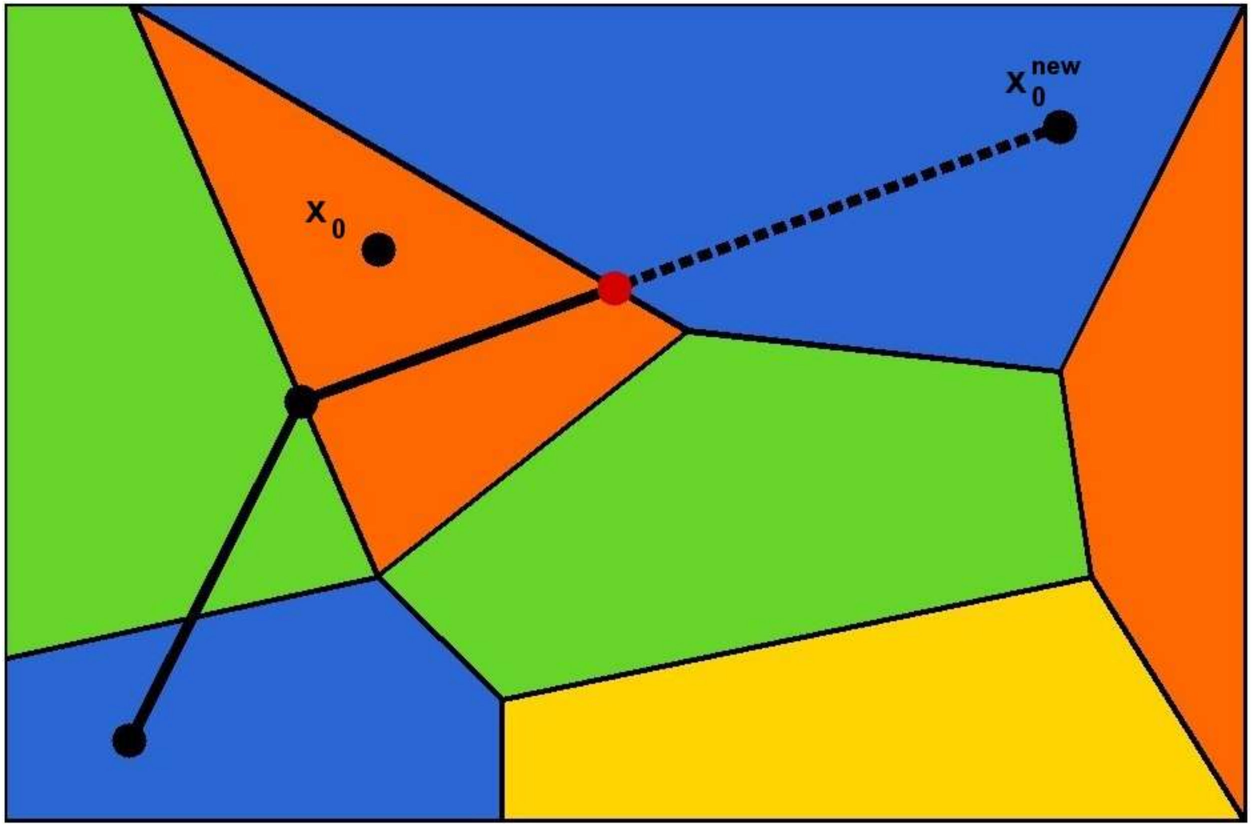
Online Active Set Strategy
Real-time variant



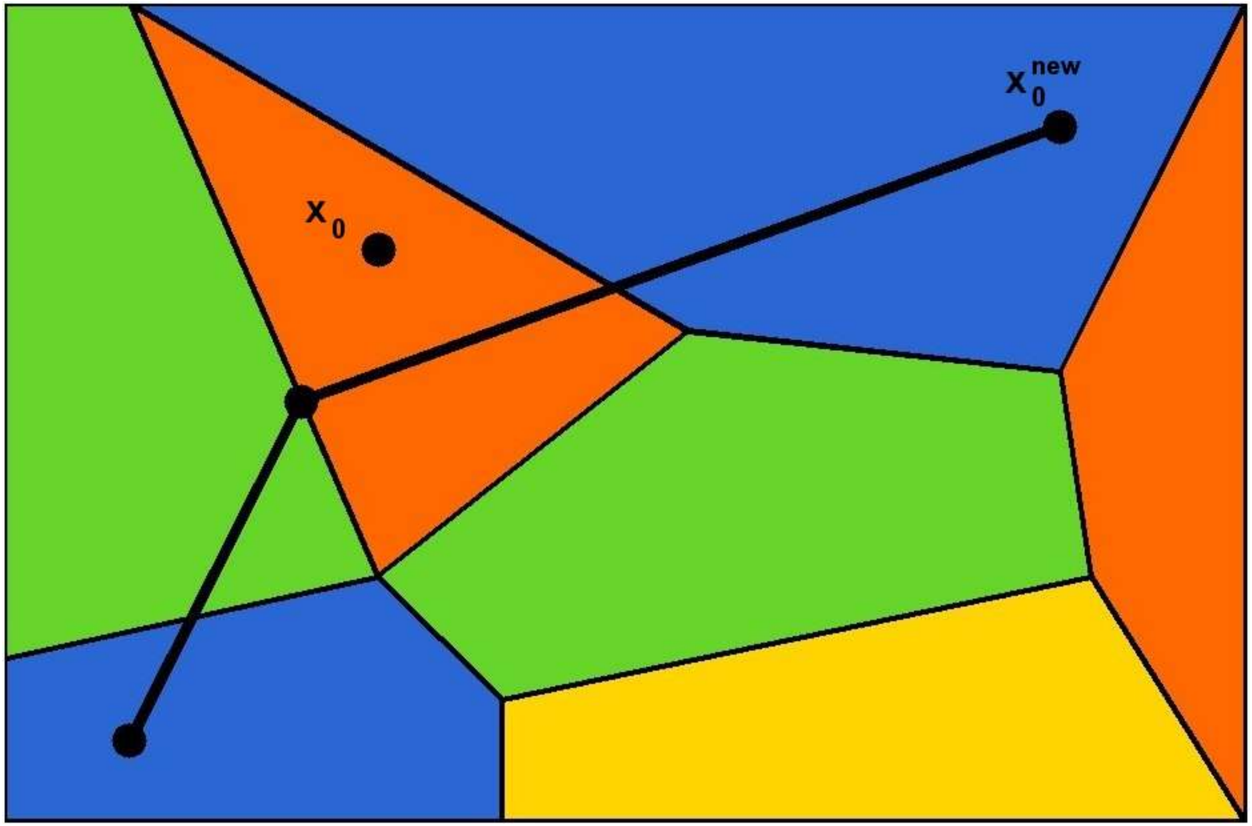
Online Active Set Strategy
Real-time variant



Online Active Set Strategy
Real-time variant



Online Active Set Strategy
Real-time variant



Online Active Set Strategy
Initialization and Degeneracy handling

- Homotopy is started from a QP problem with known solution, e.g.

$$\min_z \ \tfrac{1}{2} z' H z + \mathbf{0}' z$$
$$s.t. \ \ Az \geq -\mathbf{1}$$

- During all iterations, $A_{\mathbb{A}}$ **has to keep full row rank**, i.e. constraints in working set must be linearly independent
- This can be easily done by solving an auxiliary linear system

- **Infeasible QP problems** are easily detected while moving along the homotopy path (recall that $\mathcal{P}$ is convex!)
- Homotopy is stopped until the next feasible QP appears

qpOASES
An implementation of the **O**nline **A**ctive **SE**t **S**trategy

- qpOASES solves QP problems of the following form:

$$\min_z \ \tfrac{1}{2} z' H z + g(x_0)' z$$
$$s.t. \ \ \underline{b}(x_0) \le z \le \bar{b}(x_0)$$
$$\underline{c}(x_0) \le Az \le \bar{c}(x_0)$$

- **C/C++ implementation** with dense linear algebra
  Ferreau, Kirches, Potschka, Bock, Diehl (2014)

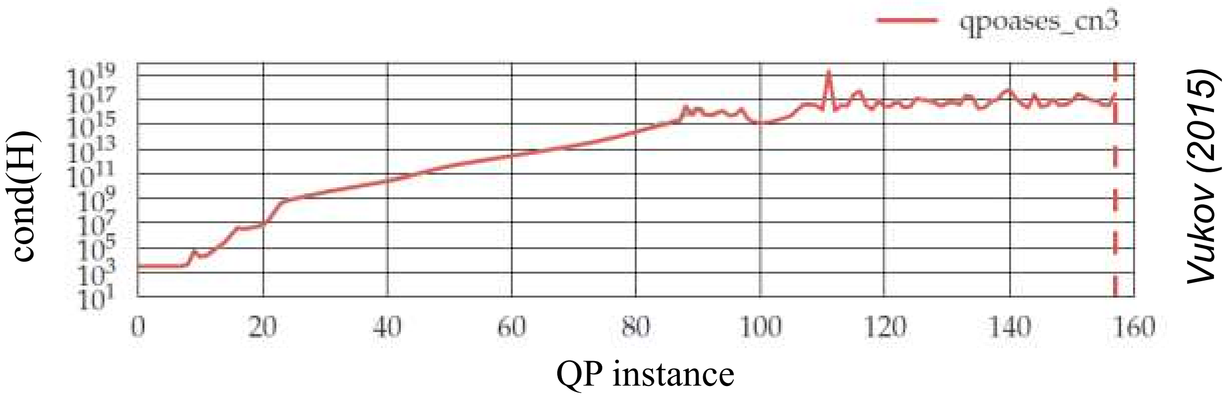
- **Reliable and efficient code** for solving small- to medium-scale QPs (states eliminated from MPC problem)

- **Self-contained code** (optionally, LAPACK/BLAS can be linked)

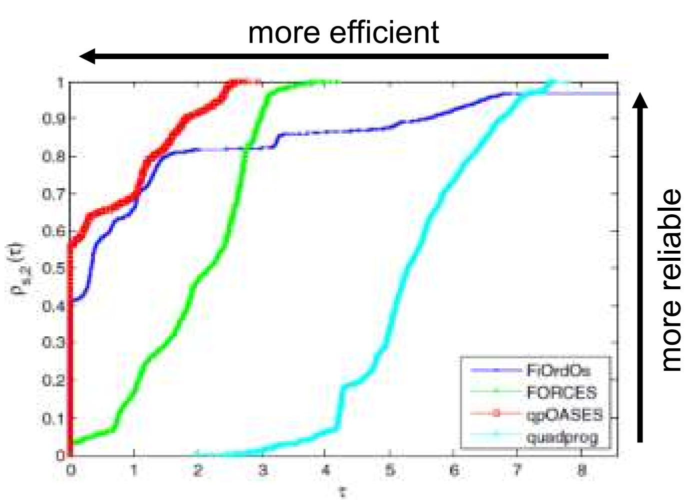
- Distributed as **open-source software** (GNU LGPL), download at: **https://projects.coin-or.org/qpOASES**

qpOASES
is reliable and efficient

- Robust against bad conditioning of Hessian matrix:



- Overall computational performance on 14 MPC benchmark examples: *Kouzoupis et al. (2015)*
  - > 2500 QP instances
  - 2-12 states
  - 1-4 control inputs
  - 3-100 intervals
  - different constraints



qpOASES
Further algorithmic features

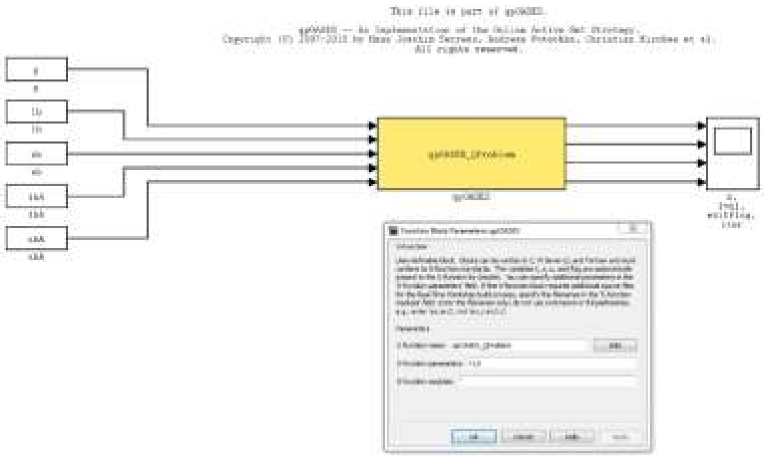
- Handles semi-definite (even indefinite) Hessian matrices
- **Structure exploitation** for various QP variants, e.g.
  - box constraints
  - varying matrices
  - limited sparsity support
- Reliable **detection of infeasible QP problems**
- Start from arbitrary initial guesses (without Phase I)
- Choose between **double and single precision arithmetic**

qpOASES
Offers various interfaces to third-party software

- Matlab / Octave / Scilab

```
[x,fval,exitflag,iter,lambda] = qpOASES( H,g,A,lb,ub,lbA,ubA )
```

- Simulink
  - dSPACE
  - xPC Target



- Python
- YALMIP / ACADO Toolkit / MUSCOD-II / CasADi

Outline

- Quadratic Programming (QP)
- Model Predictive Control (MPC)
- QP Formulations and Algorithms
- The Online QP Solver qpOASES
- **Embedded Applications of qpOASES**
- Using qpOASES

ABB

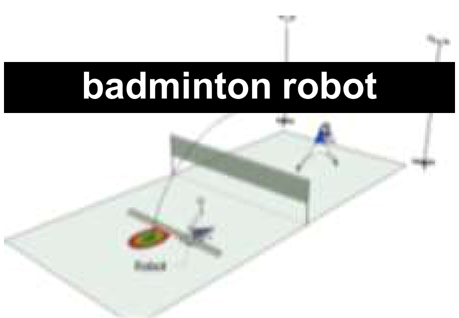
Using qpOASES as algorithmic building block

- ACADO Code Generation Tool uses qpOASES within an **SQP-type algorithm for nonlinear MPC** *Houska et al. (2011)*

- If MPC horizon becomes long, **block condensing** may be applied to adjust the sparsity level of the QP problem *Axehill (2015)*
- Recently proposed **dual Newton strategy** shows promising performance combining block condensing and qpOASES *Kouzoupis et al. (2015a), Frasch et al. (2014)*

- Optimum experimental design problems often lead to **nonconvex NLPs** with block-diagonal Hessian matrix
- A **filter line-search SQP method** using SR1/BFGS updates based on qpOASES has been proposed *Janka et al. (2015)*

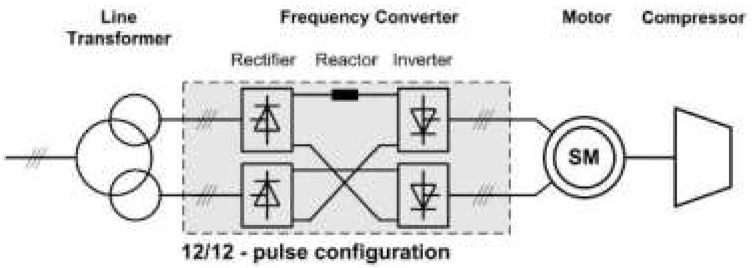
Using qpOASES for Real-World Applications



Using qpOASES
to Control a 48 Megawatt Drive!

- **Load commutated inverters** (LCIs) play an important role in powering **electrically-driven compressor stations**
- MPC can help LCIs to ride through partial loss of grid voltage
- **qpOASES** solves a small-scale QP problem **every millisecond** on embedded hardware
- Successfully tested **on a 48 MW pilot plant installation**

  Besselmann et al. (to appear)



Outline

- Quadratic Programming (QP)
- Model Predictive Control (MPC)
- QP Formulations and Algorithms
- The Online QP Solver qpOASES
- Embedded Applications of qpOASES
- **Using qpOASES**

Using qpOASES
for your own project

- **Matlab interface** offers basically the complete functionality of the C++ core
- qpOASES can be called either in **offline mode** or **online mode**

- **Offline mode:** initialize each QP problem from scratch

```
[x,fval,exitflag,iter,lambda,auxOutput] = ...
            qpOASES( H,g,A,lb,ub,lbA,ubA,options,auxInput );
```

- **Online mode:** use hotstarts to speed-up solution

```
[QP,x,fval,…] = qpOASES_sequence( 'i',H,g,A,lb,ub,… );
[x,fval,…]    = qpOASES_sequence( 'h',QP,g,lb,ub,… );
                qpOASES_sequence( 'c',QP );
```

Using qpOASES
for your own project (cont.)

- If no options are passed, **default options** are used that are typically **slower but more reliable**
- **Enable MPC options** by calling

```
myOptions = qpOASES_options( 'mpc' );
```

- Options can also be used to specify maximum number of iterations (or CPU time limit)
- Type

```
help qpOASES
help qpOASES_sequence
help qpOASES_options
help qpOASES_auxInput
```

for more information

Summary

- qpOASES is a **reliable, self-contained, open-source QP solver,** also for embedded optimization
- Efficient due to plenty of **structure-exploiting features**
- Successfully used in numerous **real-world applications**

https://projects.coin-or.org/qpOASES

(thanks to Christian Kirches, Andreas Potschka, Alexander Buchner, Manuel Kudruss, Sebastian Walter and all the other contributors)

Power and productivity
for a better world™