AWP15-ENR-01/JSI-02

EUROfusion

Fast Model Predictive Control for Magnetic Plasma Control

MPC using fast online 1st-order QP methods

Matija Perne

Jožef Stefan Institute

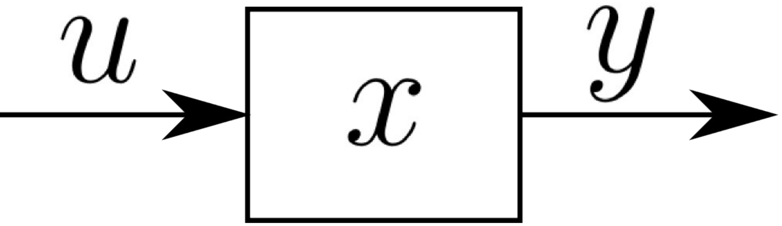
CREATE PPCC team

cosylab CONTROL SYSTEM LABORATORY

SLOVENIAN RESEARCH AGENCY

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

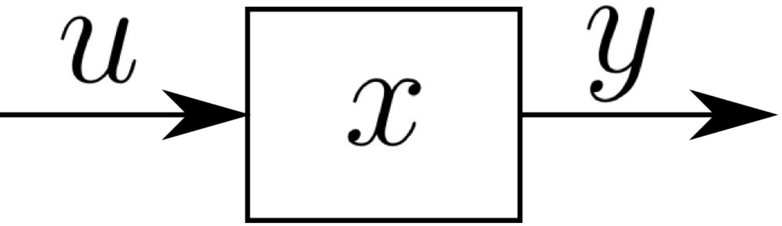
Linear dynamic system, optimum



- Discrete-time model:

$$x_{k+1} = Ax_k + Bu_k$$
$$y_k = Cx_k + Du_k$$

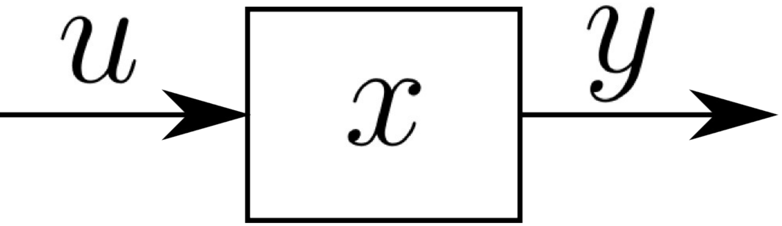
Linear dynamic system, optimum



- Discrete-time model:

$$x_{k+1} = Ax_k + Bu_k$$
$$y_k = Cx_k$$

Linear dynamic system, optimum



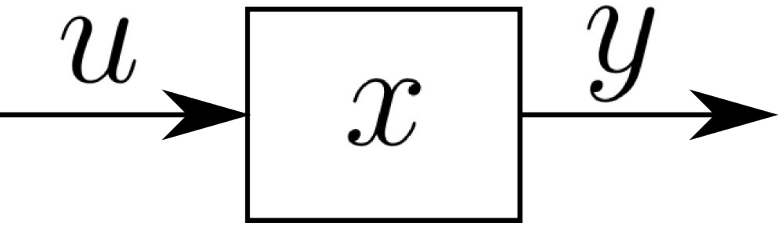
- Discrete-time model:

$$x_{k+1} = Ax_k + Bu_k$$
$$y_k = Cx_k$$

- Cost minimization:

$$J_t = \frac{1}{2}(x - x_r)' Q_t (x - x_r) + \frac{1}{2}(u - u_r)' R_t (u - u_r)$$

Linear dynamic system, optimum



- Discrete-time model:

$$x_{k+1} = Ax_k + Bu_k$$
$$y_k = Cx_k$$

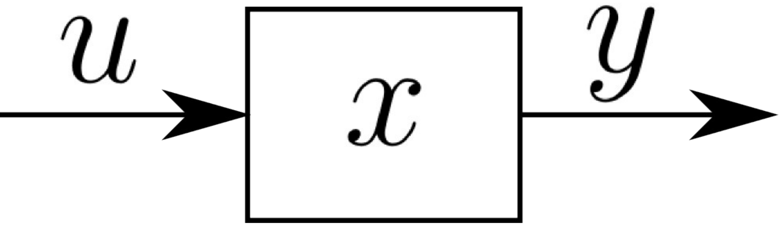
- Cost minimization:

$$J_t = \frac{1}{2}(x - x_r)' Q_t (x - x_r) + \frac{1}{2}(u - u_r)' R_t (u - u_r)$$

- Constraints:

$$u \in \mathcal{U} \qquad u_{min} \leq u \leq u_{max}$$
$$x \in \mathcal{X} \qquad x_{min} \leq x \leq x_{max}$$
$$y \in \mathcal{Y} \qquad y_{min} \leq y \leq y_{max}$$

Linear dynamic system, optimum



- Discrete-time model:

$$x_{k+1} = Ax_k + Bu_k$$
$$y_k = Cx_k$$

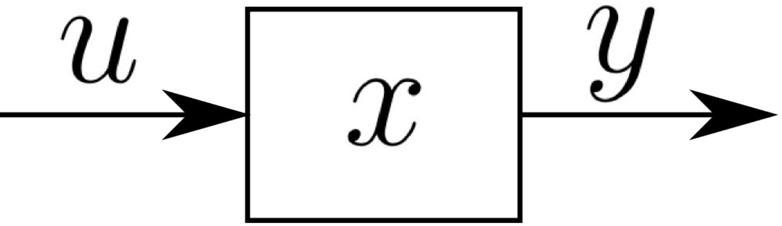
- Cost minimization:

$$J_t = \frac{1}{2}(x - x_r)' Q_t (x - x_r) + \frac{1}{2}(u - u_r)' R_t (u - u_r)$$

- Constraints:

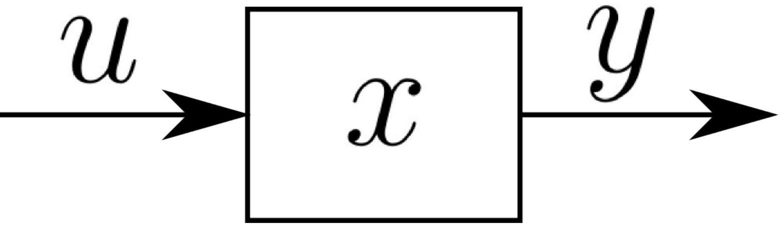
$$u_{min} \leq Uu \leq u_{max}$$
$$x_{min} \leq Xx \leq x_{max}$$
$$y_{min} \leq Yy \leq y_{max}$$

Planning the future



$$J = \sum_{k=0}^{N-1} \frac{1}{2} x'_k Q x_k + g'_x x_k + \frac{1}{2} u'_k R u_k + g'_u u_k$$

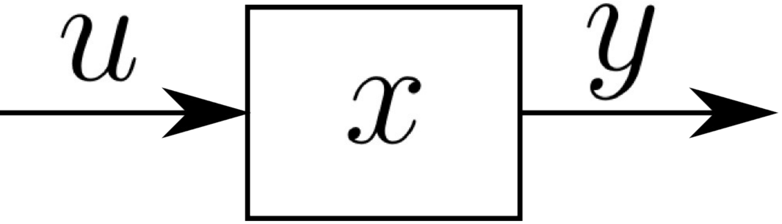
Planning the future



$$J = \sum_{k=0}^{N-1} \frac{1}{2} x_k' Q x_k + g_x' x_k + \frac{1}{2} u_k' R u_k + g_u' u_k$$

$$u^* = \arg\min_u J$$

Planning the future



$$J = \sum_{k=0}^{N-1} \frac{1}{2} x_k' Q x_k + g_x' x_k + \frac{1}{2} u_k' R u_k + g_u' u_k$$

$$u^* = \arg\min_u J \qquad \text{s.t.} \qquad \begin{aligned} x_{k+1} &= A x_k + B u_k \\ x_{k+1} &\in \mathcal{X} \\ u_k &\in \mathcal{U} \end{aligned}$$

Quadratic program

$$z = \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \\ u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$J = z' H z + g' z$$

$$A_c z = b_c$$

$$z \in \mathcal{Z}$$

Quadratic program

$$z = \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \\ u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$J = z' H z + g' z$$

Hessian

$$A_c z = b_c$$

$$z \in \mathcal{Z}$$

Quadratic program

$$z = \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \\ u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$J = z' H z + g' z$$

Hessian

Gradient

$$A_c z = b_c$$

$$z \in \mathcal{Z}$$

Quadratic program

$$z = \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \\ u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$J = z'Hz + g'z$$

Hessian

Gradient

Affine set

$$A_c z = b_c$$

$$z \in \mathcal{Z}$$

Quadratic program

$$z = \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \\ u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$J = z' H z + g' z$$

Hessian

Gradient

Affine set

$$A_c z = b_c$$

$$z \in \mathcal{Z}$$

Polyhedron

Quadratic program

$$z = \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \\ u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$J = z' H z + g' z$$

Hessian

Gradient

Affine set

$$A_c z = b_c$$

$$z \in \mathcal{Z}$$

Polyhedron

Dual method

Solving the QP

Solving the QP

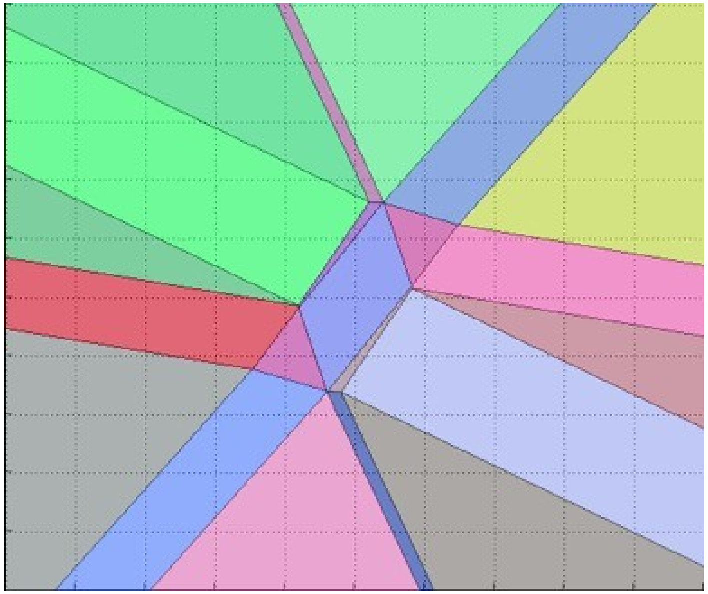
- Explicit MPC: solving the problem in advance

Solving the QP

- Explicit MPC: solving the problem in advance
  - Solution piecewise linear

Solving the QP

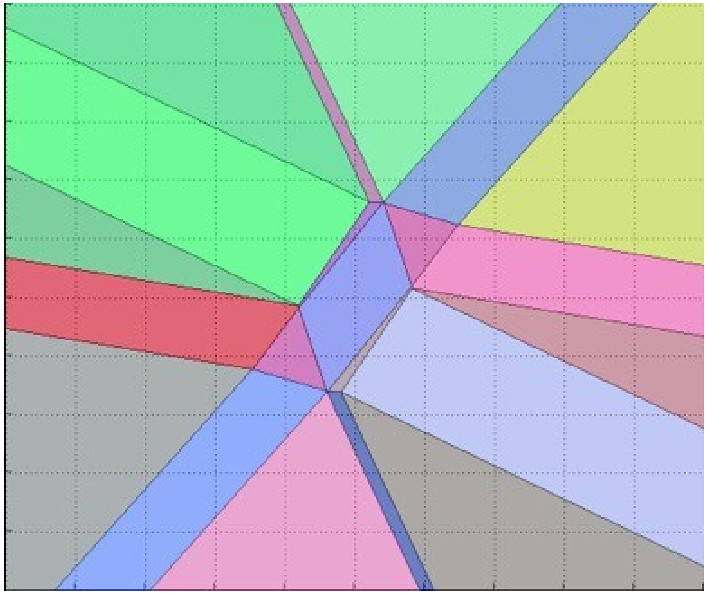
- Explicit MPC: solving the problem in advance
  - Solution piecewise linear



(http://www.seas.upenn.edu/~ese680/papers/explicit_linear_mpc.pdf)

Solving the QP

- Explicit MPC: solving the problem in advance
  - Solution piecewise linear
  - Active and inactive constraints:

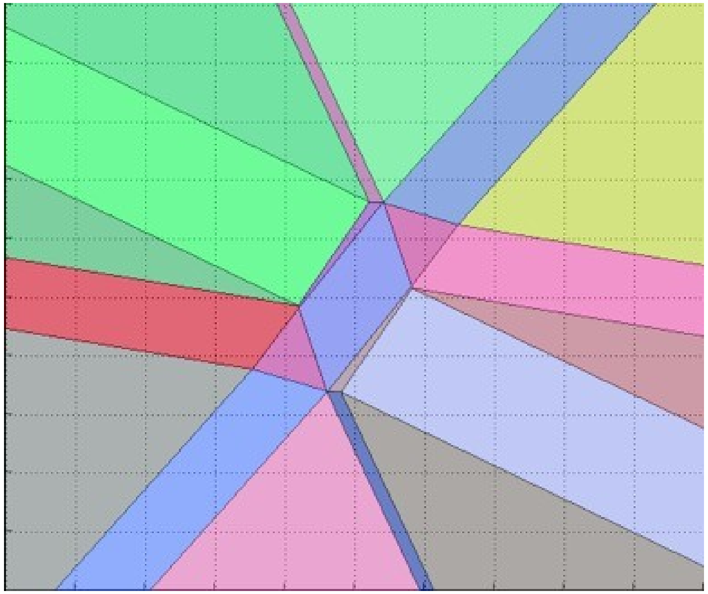


$$a \leq b$$

(http://www.seas.upenn.edu/~ese680/papers/explicit_linear_mpc.pdf)

Solving the QP

- Explicit MPC: solving the problem in advance
  - Solution piecewise linear
  - Active and inactive constraints:



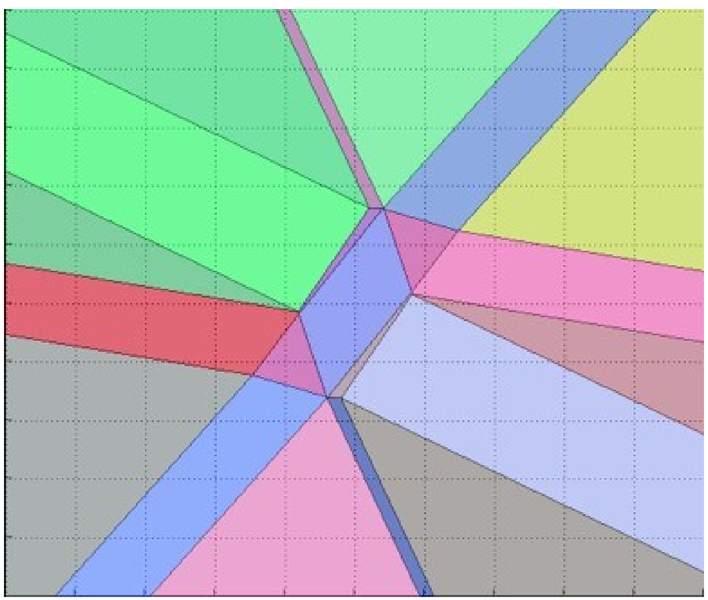
$$a \leq b$$

$$a < b$$

(http://www.seas.upenn.edu/~ese680/papers/explicit_linear_mpc.pdf)

Solving the QP

- Explicit MPC: solving the problem in advance
  - Solution piecewise linear
  - Active and inactive constraints:



$$a \leq b$$

$$a < b \qquad a = b$$

(http://www.seas.upenn.edu/~ese680/papers/explicit_linear_mpc.pdf)

Solving the QP

- Explicit MPC: solving the problem in advance
  - Solution piecewise linear
  - Active and inactive constraints:

$$a \leq b$$

$$a < b \qquad a = b$$

- Active set methods

Solving the QP

- Explicit MPC: solving the problem in advance
  - Solution piecewise linear
  - Active and inactive constraints:

$$
\begin{tikzcd}[column sep=small]
 & a \leq b \arrow[dl] \arrow[dr] & \\
a < b & & a = b
\end{tikzcd}
$$

- Active set methods
  - Online search for active constraints

Solving the QP

- Explicit MPC: solving the problem in advance
  - Solution piecewise linear
  - Active and inactive constraints:

$$a \leq b$$

$$a < b \qquad a = b$$

- Active set methods
  - Online search for active constraints
  - Iterative

Solving the QP

- Interior point methods

Solving the QP

- Interior point methods
  - Cost is modified with a steep smooth function close to polyhedral constraints

Solving the QP

- Interior point methods
  - Cost is modified with a steep smooth function close to polyhedral constraints
  - Modified problem solved with unconstrained optimization method

Solving the QP

- Interior point methods
  - Cost is modified with a steep smooth function close to polyhedral constraints
  - Modified problem solved with unconstrained optimization method
  - Modification decreased iteratively

Solving the QP

- Interior point methods
  - Cost is modified with a steep smooth function close to polyhedral constraints
  - Modified problem solved with unconstrained optimization method
  - Modification decreased iteratively
- First order methods

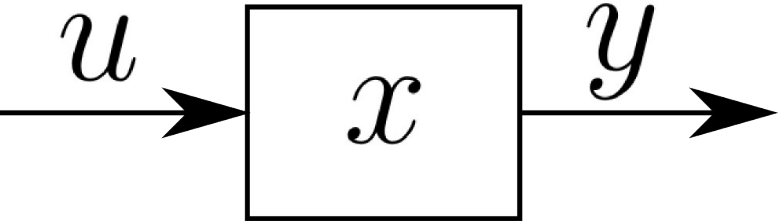
Solving the QP

- Interior point methods
  - Cost is modified with a steep smooth function close to polyhedral constraints
  - Modified problem solved with unconstrained optimization method
  - Modification decreased iteratively
- First order methods
  - Inspired by gradient method

Solving the QP

- Interior point methods
  - Cost is modified with a steep smooth function close to polyhedral constraints
  - Modified problem solved with unconstrained optimization method
  - Modification decreased iteratively
- First order methods
  - Inspired by gradient method
  - Simple, low order of convergence

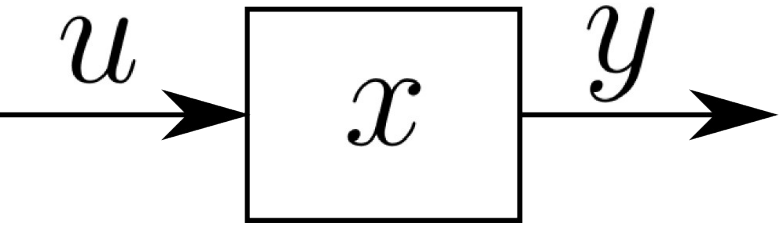
Existence of a solution



$$J = \sum_{k=0}^{N-1} \frac{1}{2} x_k' Q x_k + g_x' x_k + \frac{1}{2} u_k' R u_k + g_u' u_k$$

$$u^* = \arg\min_u J \quad \text{s.t.} \quad \begin{aligned} x_{k+1} &= A x_k + B u_k \\ x_{k+1} &\in \mathcal{X} \\ u_k &\in \mathcal{U} \end{aligned}$$

Existence of a solution



$$J = \sum_{k=0}^{N-1} \frac{1}{2} x_k' Q x_k + g_x' x_k + \frac{1}{2} u_k' R u_k + g_u' u_k$$

$$u^* = \arg\min_u J \qquad \text{s.t.} \qquad \begin{array}{c} x_{k+1} = A x_k + B u_k \\ x_{k+1} \in \mathcal{X} \\ u_k \in \mathcal{U} \end{array}$$

Existence of a solution

$$z = \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \\ u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$J = z' H z + g' z$$

$$A_c z = b_c$$

$$z \in \mathcal{Z}$$

Existence of a solution

$$z = \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \\ u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$J = z' H z + g' z$$

$$A_c z = b_c$$

$$z \in \mathcal{Z}$$

Soft constraints

Making the QP convenient

$$z = \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \\ u_0 \\ \vdots \\ u_{N-1} \end{bmatrix} \quad z = \begin{bmatrix} x_0 \\ u_0 \\ x_1 \\ u_1 \\ \vdots \\ x_{N-1} \\ u_{N-1} \end{bmatrix} \quad z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$z' = P \cdot z$$

Simplifying the QP

Move blocking, constraint reduction

$$z = \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \\ u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$J = z' H z + g' z$$

$$A_c z = b_c$$

$$z \in \mathcal{Z}$$

Move blocking, constraint reduction

$$J = z'Hz + g'z$$

$$z = \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \\ u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$A_c z = b_c$$

$$z \in \mathcal{Z}$$

Move blocking, constraint reduction

$$J = z' H z + g' z$$

$$z = \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \\ u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$A_c z = b_c$$

$$z \in \mathcal{Z}$$

$$J = \sum_{k=0}^{N-1} \frac{1}{2} x_k' Q x_k + g_x' x_k + \frac{1}{2} u_k' R u_k + g_u' u_k$$

Move blocking, constraint reduction

$$J = z'Hz + g'z$$

$$A_c z = b_c$$

$$z \in \mathcal{Z}$$

$$z = \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \\ u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$J = \sum_{k=0}^{N-1} \frac{1}{2} x_k' Q x_k + g_x' x_k + \frac{1}{2} u_k' R u_k + g_u' u_k$$

Move blocking, constraint reduction

$$J = z'Hz + g'z$$

$$A_c z = b_c$$

$$z \in \mathcal{Z}$$

$$z = \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \\ u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$J = \sum_{k=0}^{N-1} \frac{1}{2} x_k' Q x_k + g_x' x_k + \frac{1}{2} u_k' R u_k + g_u' u_k$$

Move blocking, constraint reduction

$$J = z'Hz + g'z$$

$$A_c z = b_c$$

$$z \in \mathcal{Z}$$

$$z = \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \\ u_0 \\ \vdots \\ u_{N-1} \end{bmatrix} \quad x_{k+1} = Ax_k + Bu_k$$

$$x_{k+1} \in \mathcal{X}$$

$$u_k \in \mathcal{U}$$

$$J = \sum_{k=0}^{N-1} \frac{1}{2} x_k' Q x_k + g_x' x_k + \frac{1}{2} u_k' R u_k + g_u' u_k$$

Cat

(http://animalswalls.blogspot.si/2011/08/cat-wallpapers.html)

Gradient method (ordinary)

(F. Borrelli, A. Bemporad, M. Morari, Predictive Control for linear and hybrid systems, 2015.)

unconstrained

Algorithm 4.1 Gradient method for smooth convex optimization
Input: Initial iterate $z^0 \in \mathbb{R}^s$, Lipschitz constant L of ∇f
Output: Approximate minimizer

repeat
$z^{k+1} = z^k - \frac{1}{L} \nabla f(z^k)$
until stopping criterion is satisfied

constrained

Algorithm 4.6 Gradient method for constrained smooth convex optimization
Input: Initial iterate $z^0 \in S$, Lipschitz constant L of ∇f
Output: Approximate minimizer

repeat
$z^{k+1} = \pi_S \left(z^k - \frac{1}{L} \nabla f(z^k) \right)$
until stopping criterion is satisfied

Lipschitz constant

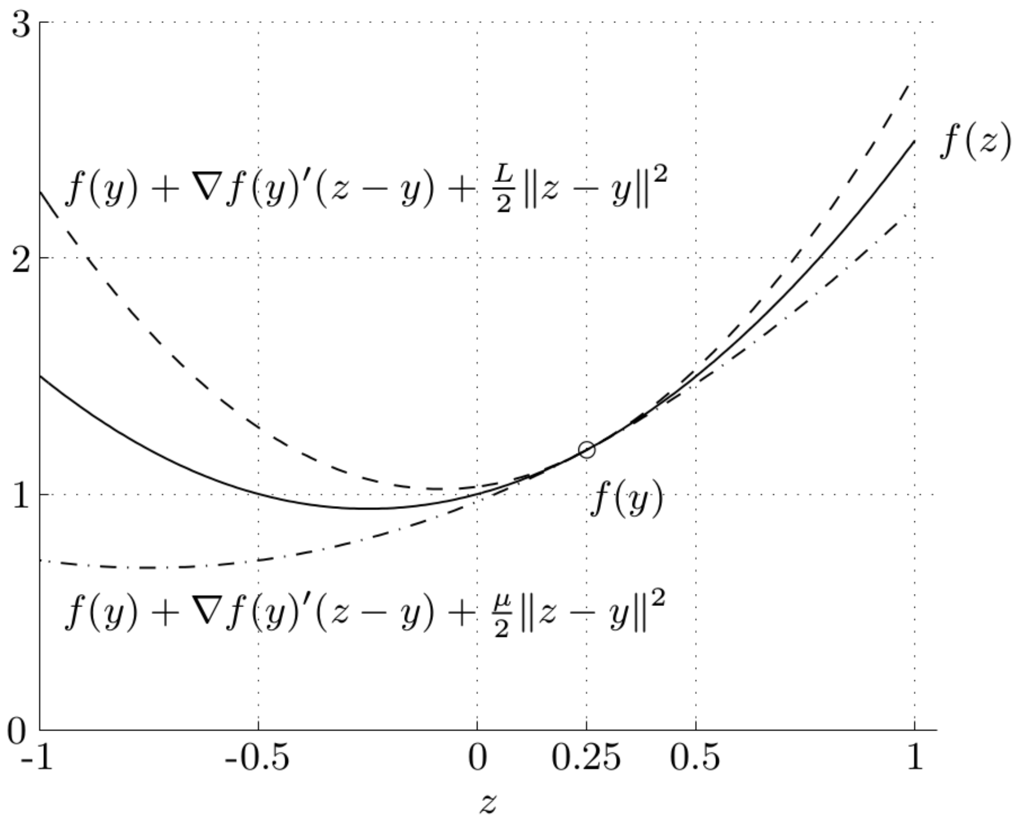


Figure 4.1 Upper and lower bounds on functions: The quadratic upper bound stems from L-smoothness of f (Definition 4.2), while the quadratic lower bound is obtained from strong convexity of f (Theorem 4.1). In this example, $f(z) = z^2 + \frac{1}{2}z + 1$ and we have chosen $L = 3$ and $\mu = 1$ for illustration.

Gradient method (ordinary)

(F. Borrelli, A. Bemporad, M. Morari, Predictive Control for linear and hybrid systems, 2015.)

unconstrained

Algorithm 4.1 Gradient method for smooth convex optimization
Input: Initial iterate $z^0 \in \mathbb{R}^s$, Lipschitz constant L of ∇f
Output: Approximate minimizer

repeat
$z^{k+1} = z^k - \frac{1}{L}\nabla f(z^k)$
until stopping criterion is satisfied

constrained

Algorithm 4.6 Gradient method for constrained smooth convex optimization
Input: Initial iterate $z^0 \in S$, Lipschitz constant L of ∇f
Output: Approximate minimizer

repeat
$z^{k+1} = \pi_S\left(z^k - \frac{1}{L}\nabla f(z^k)\right)$
until stopping criterion is satisfied

Gradient method (fast)

(F. Borrelli, A. Bemporad, M. Morari, Predictive Control for linear and hybrid systems, 2015.)

unconstrained

Algorithm 4.2 Fast gradient method for smooth convex optimization
Input: Initial iterates $z^0 \in \mathbb{R}^s$, $y^0 = z^0$; $\alpha^0 = \frac{1}{2}(\sqrt{5}-1)$, Lipschitz constant L of ∇f
Output: Approximate minimizer

repeat

$$z^{k+1} = y^k - \tfrac{1}{L}\nabla f(y^k)$$
$$\alpha^{k+1} = \frac{\alpha^k}{2}\left(\sqrt{\alpha^{k^2}+4} - \alpha^k\right)$$
$$\beta^k = \frac{\alpha^k(1-\alpha^k)}{\alpha^{k^2}+\alpha^{k+1}}$$
$$y^{k+1} = z^{k+1} + \beta^k(z^{k+1} - z^k)$$

until stopping criterion is satisfied

constrained

Algorithm 4.7 Fast gradient method for constrained smooth strongly convex optimization
Input: Initial iterates $z^0 \in S$, $y^0 = z^0$; $0 < \sqrt{\mu/L} \leq \alpha^0 < 1$, Lipschitz constant L of ∇f, strong convexity parameter μ of f
Output: Approximate minimizer

repeat

$$z^{k+1} = \pi_S\left(y^k - \tfrac{1}{L}\nabla f(y^k)\right)$$
Compute $\alpha^{k+1} \in (0,1)$: $\alpha^{{k+1}^2} = (1-\alpha^{k+1})\alpha^{k^2} + \frac{\mu\alpha^{k+1}}{L}$
$$\beta^k = \frac{\alpha^k(1-\alpha^k)}{\alpha^{k^2}+\alpha^{k+1}}$$
$$y^{k+1} = z^{k+1} + \beta^k(z^{k+1} - z^k)$$

until stopping criterion is satisfied

Lagrange multipliers

("Lagrange multiplier." Wikipedia, The Free Encyclopedia, 2015.)

$$\max_x f(x) \quad \text{s.t.} \quad g(x) = 0$$

Lagrange multipliers

("Lagrange multiplier." Wikipedia, The Free Encyclopedia, 2015.)

$$\max_{x} f(x) \quad \text{s.t.} \quad g(x) = 0$$

$$\mathcal{L}(x, \lambda) \coloneqq f(x) + \lambda \cdot g(x)$$

Lagrange multipliers

("Lagrange multiplier." Wikipedia, The Free Encyclopedia, 2015.)

$$\max_{x} f(x) \quad \text{s.t.} \quad g(x) = 0$$

$$\mathcal{L}(x,\lambda) \coloneqq f(x) + \lambda \cdot g(x)$$

$$\nabla_{x,\lambda}\mathcal{L}(x,\lambda) = 0$$

Lagrange and dual functions

(Stephen Boyd & Lieven Vandenberghe, Convex Optimization, Cambridge University Press 2004)

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p, \end{array}$$

Lagrange and dual functions

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p, \end{array}$$

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x),$$

Lagrange and dual functions

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \nu) = \inf_{x \in \mathcal{D}} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right)$$

Lagrange and dual functions

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \nu) = \inf_{x \in \mathcal{D}} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right)$$

$$g(\lambda, \nu) \leq p^\star \qquad \lambda \succeq 0$$

Lagrange and dual functions

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \nu) = \inf_{x \in \mathcal{D}} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right)$$

$$g(\lambda, \nu) \le p^\star \qquad \lambda \succeq 0$$

$$\begin{array}{ll} \text{maximize} & g(\lambda, \nu) \\ \text{subject to} & \lambda \succeq 0 \end{array}$$

Lagrange and dual functions

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \nu) = \inf_{x \in \mathcal{D}} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right)$$

$$g(\lambda, \nu) \leq p^\star \qquad \lambda \succeq 0 \qquad \begin{array}{ll} \text{maximize} & g(\lambda, \nu) \\ \text{subject to} & \lambda \succeq 0 \end{array}$$

We mentioned at the beginning of §5.5.3 that if strong duality holds and a dual optimal solution $(\lambda^\star, \nu^\star)$ exists, then any primal optimal point is also a minimizer of $L(x, \lambda^\star, \nu^\star)$. This fact sometimes allows us to compute a primal optimal solution from a dual optimal solution.

FGMdual

(generalized fast dual gradient method)

Improving Fast Dual Ascent for MPC - Part II: The Embedded Case ⋆

Pontus Giselsson *

* *Electrical Engineering, Stanford University (e-mail: pontusg@stanford.edu).*

(generalized fast dual gradient method)

Improving Fast Dual Ascent for MPC - Part II: The Embedded Case ⋆

Pontus Giselsson *

* *Electrical Engineering, Stanford University (e-mail: pontusg@stanford.edu).*

$$
\begin{array}{ll}
\text{minimize} & \frac{1}{2} y^T H y \\
\text{subject to} & Ay = b\bar{x} \\
 & By = v \\
 & \underline{d} \leq v \leq \bar{d}
\end{array}
$$

Letting $H_A = AH^{-1}A^T$, the algorithm becomes

$$y^k = H^{-1}(A^T H_A^{-1}(AH^{-1}B^T v^k + b\bar{x}) - B^T v^k) \tag{47}$$

$$\mu^k = \text{prox}_{g^\star}^{\mathbf{L}_\mu}(v^k + \mathbf{L}_\mu^{-1} B y^k) \tag{48}$$

$$t^{k+1} = \frac{1+\sqrt{1+4(t^k)^2}}{2} \tag{49}$$

$$v^{k+1} = \mu^k + \left(\frac{t^k - 1}{t^{k+1}}\right)(\mu^k - \mu^{k-1}) \tag{50}$$

FGMdual

(generalized fast dual gradient method)

Improving Fast Dual Ascent for MPC - Part II: The Embedded Case ⋆

Pontus Giselsson *

* *Electrical Engineering, Stanford University (e-mail: pontusg@stanford.edu).*

$$\begin{array}{ll} \text{minimize} & \frac{1}{2} y^T H y \\ \text{subject to} & Ay = b\bar{x} \\ & By = v \\ & \underline{d} \leq v \leq \bar{d} \end{array}$$

Letting $H_A = AH^{-1}A^T$, the algorithm becomes

$$y^k = H^{-1}(A^T H_A^{-1}(AH^{-1}B^T v^k + b\bar{x}) - B^T v^k) \quad (47)$$

$$\mu^k = \mathrm{prox}_{g^\star}^{\mathbf{L}_\mu}(v^k + \mathbf{L}_\mu^{-1} B y^k) \quad (48)$$

$$t^{k+1} = \frac{1+\sqrt{1+4(t^k)^2}}{2} \quad (49)$$

$$v^{k+1} = \mu^k + \left(\frac{t^k - 1}{t^{k+1}}\right)(\mu^k - \mu^{k-1}) \quad (50)$$

$$\mathrm{prox}_\psi^{\mathbf{L}}(x) := \arg\min_y \left\{ \psi(y) + \frac{1}{2}\|y - x\|_{\mathbf{L}}^2 \right\}$$

$$\mathrm{prox}_{g^\star}^{\mathbf{L}}(x) + \mathbf{L}^{-1}\mathrm{prox}_g^{\mathbf{L}^{-1}}(\mathbf{L}x) = x$$

QPgen

QPgen

- Code generator:

QPgen

- Code generator:
  - Problem described in MATLAB

QPgen

- Code generator:
  - Problem described in MATLAB
  - MATLAB does the offline calculations

QPgen

- Code generator:
  - Problem described in MATLAB
  - MATLAB does the offline calculations
  - MATLAB outputs the algorithm code in C

QPgen

- Code generator:
  - Problem described in MATLAB
  - MATLAB does the offline calculations
  - MATLAB outputs the algorithm code in C
  - Code compiled, ran, result returned to MATLAB

QPgen

- Code generator:
  - Problem described in MATLAB
  - MATLAB does the offline calculations
  - MATLAB outputs the algorithm code in C
  - Code compiled, ran, result returned to MATLAB
  - FGMdual one of the supported algorithms

QPgen

- Code generator:
  - Problem described in MATLAB
  - MATLAB does the offline calculations
  - MATLAB outputs the algorithm code in C
  - Code compiled, ran, result returned to MATLAB
  - FGMdual one of the supported algorithms
- Knowing the code:

QPgen

- Code generator:
  - Problem described in MATLAB
  - MATLAB does the offline calculations
  - MATLAB outputs the algorithm code in C
  - Code compiled, ran, result returned to MATLAB
  - FGMdual one of the supported algorithms
- Knowing the code:
  - MATLAB code is long

QPgen

- Code generator:
  - Problem described in MATLAB
  - MATLAB does the offline calculations
  - MATLAB outputs the algorithm code in C
  - Code compiled, ran, result returned to MATLAB
  - FGMdual one of the supported algorithms
- Knowing the code:
  - MATLAB code is long
  - Generated code is confusing

```c
while ((jj < *max_iter) && (cond < 0)) {

jj++;

copy_vec_part_negate(v,tmp_var_p,40);

mat_vec_mult_sparse(&CT,tmp_var_p,tmp_var_n);

vec_sub(tmp_var_n,q1,tmp_var_n,60);

stack_vec(tmp_var_n,q2,rhs,60,40);

perm_fwdsolve(&L,p,rhs,tmp_var_nm);

mat_vec_mult_sparse(&Dinv,tmp_var_nm,tmp_var_nm2);

backsolve_perm(&LT,p,tmp_var_nm2,tmp_var_nm);

copy_vec_part(tmp_var_nm,x,60);

mat_vec_mult_sparse(&C,x,tmp_var_p);

vec_add(v,tmp_var_p,tmp_var_p,40);

copy_vec_part(tmp_var_p,arg_prox_h,40);

clip_soft(tmp_var_p,l,u,(double *) &soft,40);

mat_vec_mult_diag(&Einv,tmp_var_p,y);

copy_vec_part(lambda,lambda_old,40);

vec_sub(arg_prox_h,tmp_var_p,lambda,40);

vec_sub(lambda,lambda_old,tmp_var_p,40);

theta_old = theta;

theta = (1+sqrt(1+4*pow(theta_old,2)))/2;

scalar_mult((theta_old-1)/theta,tmp_var_p,40);

copy_vec_part(v,v_old,40);

vec_add(tmp_var_p,lambda,v,40);

if (mod(jj,10) == 0) {
 cond = check_stop_cond_FGM(&Einv,lambda,lambda_old,tmp_var_p,tm
 }

restart(lambda,lambda_old,v,v_old,tmp_var_p,tmp_var_p2,40);

}
```

ponavljaj

$$tmp\_var\_p = -v^k$$

$$tmp\_var\_n = \mathsf{B}^{\mathrm{T}} tmp\_var\_p \left(= -\mathsf{B}^{\mathrm{T}} v^k\right)$$

$$tmp\_var\_n = tmp\_var\_n - g \left(= -\mathsf{B}^{\mathrm{T}} v^k - g\right)$$

$$rhs = \begin{bmatrix} tmp\_var\_n \\ b\bar{x} \end{bmatrix} \left(= \begin{bmatrix} -\mathsf{B}^{\mathrm{T}} v^k - g \\ b\bar{x} \end{bmatrix}\right)$$

$$\mathsf{L} tmp\_var\_nm = rhs, \quad tmp\_var\_nm = \mathsf{L}^{-1} rhs \qquad \left(\mathsf{K} = \mathsf{L}\mathsf{D}\mathsf{L}^{\mathrm{T}}\right)$$

$$tmp\_var\_nm2 = \mathsf{D}^{-1} tmp\_var\_nm \left(= \mathsf{D}^{-1}\mathsf{L}^{-1} rhs\right)$$

$$\mathsf{L}^{\mathrm{T}} tmp\_var\_nm = tmp\_var\_nm2, \quad tmp\_var\_nm = \left(\mathsf{L}^{\mathrm{T}}\right)^{-1} \mathsf{D}^{-1}\mathsf{L}^{-1} rhs$$

$$\begin{bmatrix} y \\ \xi \end{bmatrix} = tmp\_var\_nm$$

$$tmp\_var\_p = \mathsf{B} y$$

$$tmp\_var\_p = tmp\_var\_p + v^k \left(= \mathsf{B} y + v^k\right)$$

$$arg\_prox\_h = tmp\_var\_p \left(= \mathsf{B} y + v^k\right)$$

$$tmp\_var\_p = max\left(min\left(tmp\_var\_p, u\right), l\right)$$

$$\mu^{k-1} = \mu^k$$

$$\mu^k = arg\_prox\_h - tmp\_var\_p$$

$$tmp\_var\_p = \mu^k - \mu^{k-1}$$

$$t^k = t^{k+1}$$

$$t^{k+1} = \frac{1 + \sqrt{1 + 4\left(t^k\right)^2}}{2}$$

$$tmp\_var\_p = \left(\frac{t^k - 1}{t^{k+1}}\right) tmp\_var\_p \left(= \left(\frac{t^k - 1}{t^{k+1}}\right)\left(\mu^k - \mu^{k-1}\right)\right)$$

$$v^{k-1} = v^k$$

$$v^k = tmp\_var\_p + \mu^k \left(= \mu^k + \left(\frac{t^k - 1}{t^{k+1}}\right)\left(\mu^k - \mu^{k-1}\right)\right)$$

dokler pogoji niso izpolnjeni

$$\begin{array}{ll} \text{minimize} & \frac{1}{2} y^T H y \\ \text{subject to} & Ay = b\bar{x} \\ & By = v \\ & \underline{d} \leq v \leq \bar{d} \end{array}$$

Letting $H_A = AH^{-1}A^T$, the algorithm be

$$y^k = H^{-1}(A^T H_A^{-1}(AH^{-1}B^T v^k + b\bar{x}) - B^T v^k)$$

$$\mu^k = \operatorname{prox}_{g^\star}^{\mathbf{L}_\mu}(v^k + \mathbf{L}_\mu^{-1} B y^k)$$

$$t^{k+1} = \frac{1+\sqrt{1+4(t^k)^2}}{2}$$

$$v^{k+1} = \mu^k + \left(\frac{t^k - 1}{t^{k+1}}\right)(\mu^k - \mu^{k-1})$$

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} y^k \\ \xi \end{bmatrix} = \begin{bmatrix} -B^T v^k \\ b\bar{x} \end{bmatrix}$$

$$\operatorname{prox}_{g^\star}^{\mathbf{L}}(x) + \mathbf{L}^{-1} \operatorname{prox}_{g}^{\mathbf{L}^{-1}}(\mathbf{L}x) = x$$

$$\operatorname{prox}_{\psi}^{\mathbf{L}}(x) := \arg\min_y \left\{ \psi(y) + \tfrac{1}{2}\|y - x\|_{\mathbf{L}}^2 \right\}$$

ponavljaj

$$tmp\_var\_p = -v^k$$

$$tmp\_var\_n = \mathsf{B}^{\mathrm{T}} tmp\_var\_p \left(= -\mathsf{B}^{\mathrm{T}} v^k\right)$$

$$tmp\_var\_n = tmp\_var\_n - g \left(= -\mathsf{B}^{\mathrm{T}} v^k - g\right)$$

$$rhs = \begin{bmatrix} tmp\_var\_n \\ b\bar{x} \end{bmatrix} \left(= \begin{bmatrix} -\mathsf{B}^{\mathrm{T}} v^k - g \\ b\bar{x} \end{bmatrix} \right)$$

$$\mathsf{L} tmp\_var\_nm = rhs, \quad tmp\_var\_nm = \mathsf{L}^{-1} rhs \quad (\mathsf{K} = \mathsf{LDL}^{\mathrm{T}})$$

$$tmp\_var\_nm2 = \mathsf{D}^{-1} tmp\_var\_nm \left(= \mathsf{D}^{-1}\mathsf{L}^{-1} rhs\right)$$

$$\mathsf{L}^{\mathrm{T}} tmp\_var\_nm = tmp\_var\_nm2, \quad tmp\_var\_nm = \left(\mathsf{L}^{\mathrm{T}}\right)^{-1} \mathsf{D}^{-1}\mathsf{L}^{-1} rhs$$

$$\begin{bmatrix} y \\ \xi \end{bmatrix} = tmp\_var\_nm$$

$$tmp\_var\_p = \mathsf{B}y$$

$$tmp\_var\_p = tmp\_var\_p + v^k \left(= \mathsf{B}y + v^k\right)$$

$$arg\_prox\_h = tmp\_var\_p \left(= \mathsf{B}y + v^k\right)$$

$$tmp\_var\_p = max\left(min\left(tmp\_var\_p, u\right), l\right)$$

$$\mu^{k-1} = \mu^k$$

$$\mu^k = arg\_prox\_h - tmp\_var\_p$$

$$tmp\_var\_p = \mu^k - \mu^{k-1}$$

$$t^k = t^{k+1}$$

$$t^{k+1} = \frac{1 + \sqrt{1 + 4\left(t^k\right)^2}}{2}$$

$$tmp\_var\_p = \left(\frac{t^k - 1}{t^{k+1}}\right) tmp\_var\_p \left(= \left(\frac{t^k - 1}{t^{k+1}}\right)\left(\mu^k - \mu^{k-1}\right)\right)$$

$$v^{k-1} = v^k$$

$$v^k = tmp\_var\_p + \mu^k \left(= \mu^k + \left(\frac{t^k - 1}{t^{k+1}}\right)\left(\mu^k - \mu^{k-1}\right)\right)$$

dokler pogoji niso izpolnjeni

$\mathtt{v} \rightarrow v^k$

$\mathtt{\&CT} \rightarrow \mathsf{B}^{\mathrm{T}}$

$\mathtt{q1} \rightarrow g$

$\mathtt{q2} \rightarrow b\bar{x}$

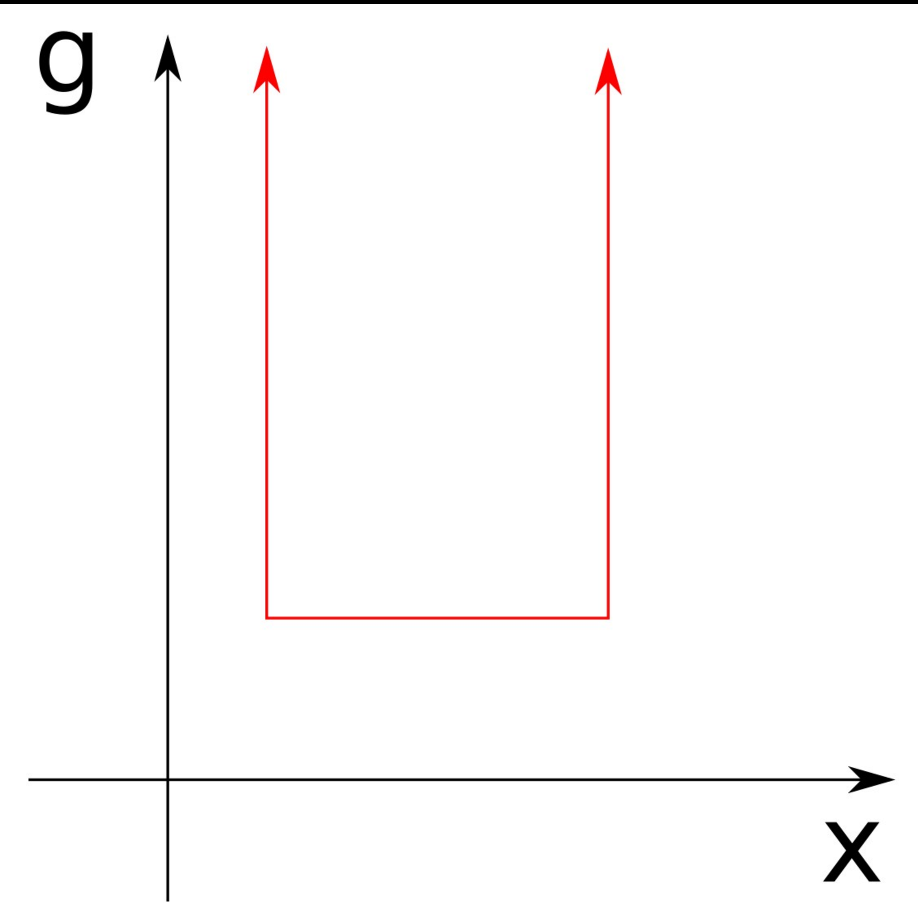
$\mathtt{\&L} \rightarrow \mathsf{L}$

$\mathtt{\&Dinv} \rightarrow \mathsf{D}^{-1}$

$\mathtt{\&LT} \rightarrow \mathsf{L}^{\mathrm{T}}$

$\mathtt{x} \rightarrow y$

$\mathtt{\&C} \rightarrow \mathsf{B}$



ponavljaj

$$tmp\_var\_p = -v^k$$
$$tmp\_var\_n = \mathsf{B}^{\mathrm{T}} tmp\_var\_p \left(= -\mathsf{B}^{\mathrm{T}} v^k\right)$$
$$tmp\_var\_n = tmp\_var\_n - g \left(= -\mathsf{B}^{\mathrm{T}} v^k - g\right)$$
$$rhs = \begin{bmatrix} tmp\_var\_n \\ b\bar{x} \end{bmatrix} \left(= \begin{bmatrix} -\mathsf{B}^{\mathrm{T}} v^k - g \\ b\bar{x} \end{bmatrix}\right)$$
$$\mathsf{L} tmp\_var\_nm = rhs, \quad tmp\_var\_nm = \mathsf{L}^{-1} rhs \quad (\mathsf{K} = \mathsf{LDL}^{\mathrm{T}})$$
$$tmp\_var\_nm2 = \mathsf{D}^{-1} tmp\_var\_nm \left(= \mathsf{D}^{-1}\mathsf{L}^{-1} rhs\right)$$
$$\mathsf{L}^{\mathrm{T}} tmp\_var\_nm = tmp\_var\_nm2, \quad tmp\_var\_nm = \left(\mathsf{L}^{\mathrm{T}}\right)^{-1} \mathsf{D}^{-1}\mathsf{L}^{-1} rhs$$
$$\begin{bmatrix} y \\ \xi \end{bmatrix} = tmp\_var\_nm$$
$$tmp\_var\_p = \mathsf{B} y$$
$$tmp\_var\_p = tmp\_var\_p + v^k \left(= \mathsf{B} y + v^k\right)$$
$$arg\_prox\_h = tmp\_var\_p \left(= \mathsf{B} y + v^k\right)$$
$$\boxed{tmp\_var\_p = max\left(min\left(tmp\_var\_p, u\right), l\right)}$$
$$\mu^{k-1} = \mu^k$$
$$\mu^k = arg\_prox\_h - tmp\_var\_p$$
$$tmp\_var\_p = \mu^k - \mu^{k-1}$$
$$t^k = t^{k+1}$$
$$t^{k+1} = \frac{1 + \sqrt{1 + 4\left(t^k\right)^2}}{2}$$
$$tmp\_var\_p = \left(\frac{t^k - 1}{t^{k+1}}\right) tmp\_var\_p \left(= \left(\frac{t^k - 1}{t^{k+1}}\right)\left(\mu^k - \mu^{k-1}\right)\right)$$
$$v^{k-1} = v^k$$
$$v^k = tmp\_var\_p + \mu^k \left(= \mu^k + \left(\frac{t^k - 1}{t^{k+1}}\right)\left(\mu^k - \mu^{k-1}\right)\right)$$

dokler pogoji niso izpolnjeni

$\mathtt{v} \to v^k$

$\&\mathtt{CT} \to \mathsf{B}^{\mathrm{T}}$

$\mathtt{q1} \to g$

$\mathtt{q2} \to b\bar{x}$

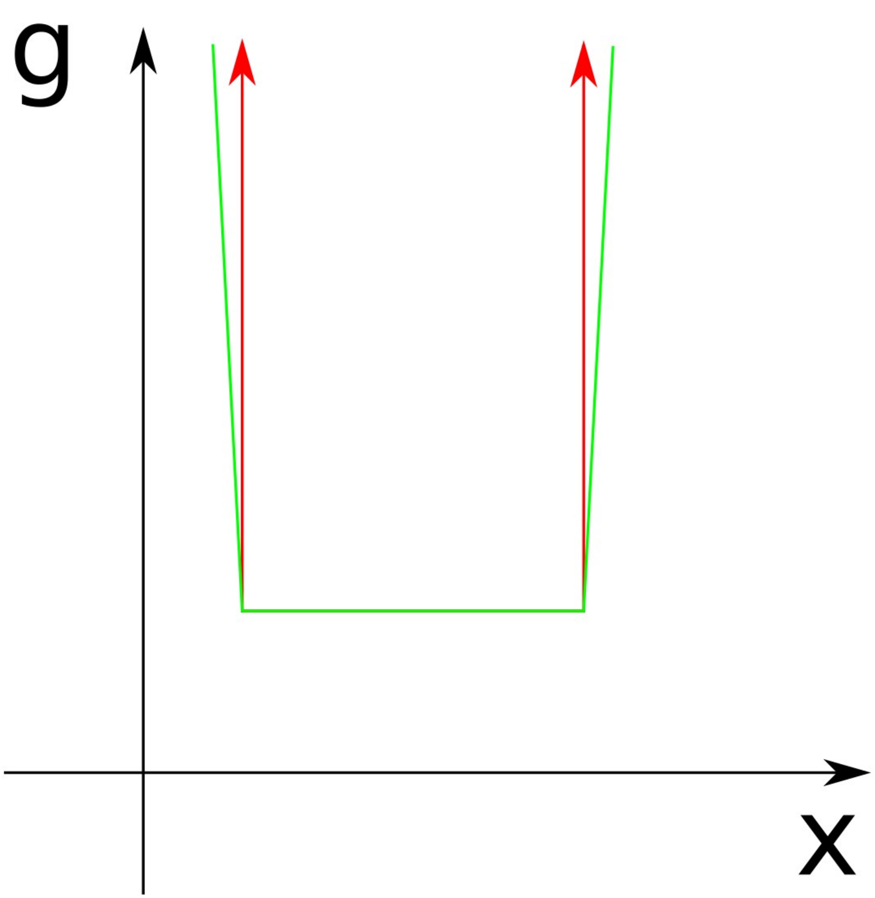
$\&\mathtt{L} \to \mathsf{L}$

$\&\mathtt{Dinv} \to \mathsf{D}^{-1}$

$\&\mathtt{LT} \to \mathsf{L}^{\mathrm{T}}$

$\mathtt{x} \to y$

$\&\mathtt{C} \to \mathsf{B}$



ponavljaj

$$tmp\_var\_p = -v^k$$
$$tmp\_var\_n = \mathsf{B}^{\mathrm{T}} tmp\_var\_p \left(= -\mathsf{B}^{\mathrm{T}} v^k\right)$$
$$tmp\_var\_n = tmp\_var\_n - g \left(= -\mathsf{B}^{\mathrm{T}} v^k - g\right)$$
$$rhs = \begin{bmatrix} tmp\_var\_n \\ b\bar{x} \end{bmatrix} \left(= \begin{bmatrix} -\mathsf{B}^{\mathrm{T}} v^k - g \\ b\bar{x} \end{bmatrix}\right)$$
$$\mathsf{L} tmp\_var\_nm = rhs, \quad tmp\_var\_nm = \mathsf{L}^{-1} rhs \quad \left(\mathsf{K} = \mathsf{L}\mathsf{D}\mathsf{L}^{\mathrm{T}}\right)$$
$$tmp\_var\_nm2 = \mathsf{D}^{-1} tmp\_var\_nm \left(= \mathsf{D}^{-1}\mathsf{L}^{-1} rhs\right)$$
$$\mathsf{L}^{\mathrm{T}} tmp\_var\_nm = tmp\_var\_nm2, \quad tmp\_var\_nm = \left(\mathsf{L}^{\mathrm{T}}\right)^{-1} \mathsf{D}^{-1}\mathsf{L}^{-1} rhs$$
$$\begin{bmatrix} y \\ \xi \end{bmatrix} = tmp\_var\_nm$$
$$tmp\_var\_p = \mathsf{B} y$$
$$tmp\_var\_p = tmp\_var\_p + v^k \left(= \mathsf{B} y + v^k\right)$$
$$arg\_prox\_h = tmp\_var\_p \left(= \mathsf{B} y + v^k\right)$$
$$\boxed{tmp\_var\_p = max\left(min\left(tmp\_var\_p, u\right), l\right)}$$
$$\mu^{k-1} = \mu^k$$
$$\mu^k = arg\_prox\_h - tmp\_var\_p$$
$$tmp\_var\_p = \mu^k - \mu^{k-1}$$
$$t^k = t^{k+1}$$
$$t^{k+1} = \frac{1 + \sqrt{1 + 4\left(t^k\right)^2}}{2}$$
$$tmp\_var\_p = \left(\frac{t^k - 1}{t^{k+1}}\right) tmp\_var\_p \left(= \left(\frac{t^k - 1}{t^{k+1}}\right)\left(\mu^k - \mu^{k-1}\right)\right)$$
$$v^{k-1} = v^k$$
$$v^k = tmp\_var\_p + \mu^k \left(= \mu^k + \left(\frac{t^k - 1}{t^{k+1}}\right)\left(\mu^k - \mu^{k-1}\right)\right)$$

dokler pogoji niso izpolnjeni

$\mathtt{v} \to v^k$

$\mathtt{\&CT} \to \mathsf{B}^{\mathrm{T}}$

$\mathtt{q1} \to g$

$\mathtt{q2} \to b\bar{x}$

$\mathtt{\&L} \to \mathsf{L}$

$\mathtt{\&Dinv} \to \mathsf{D}^{-1}$

$\mathtt{\&LT} \to \mathsf{L}^{\mathrm{T}}$

$\mathtt{x} \to y$

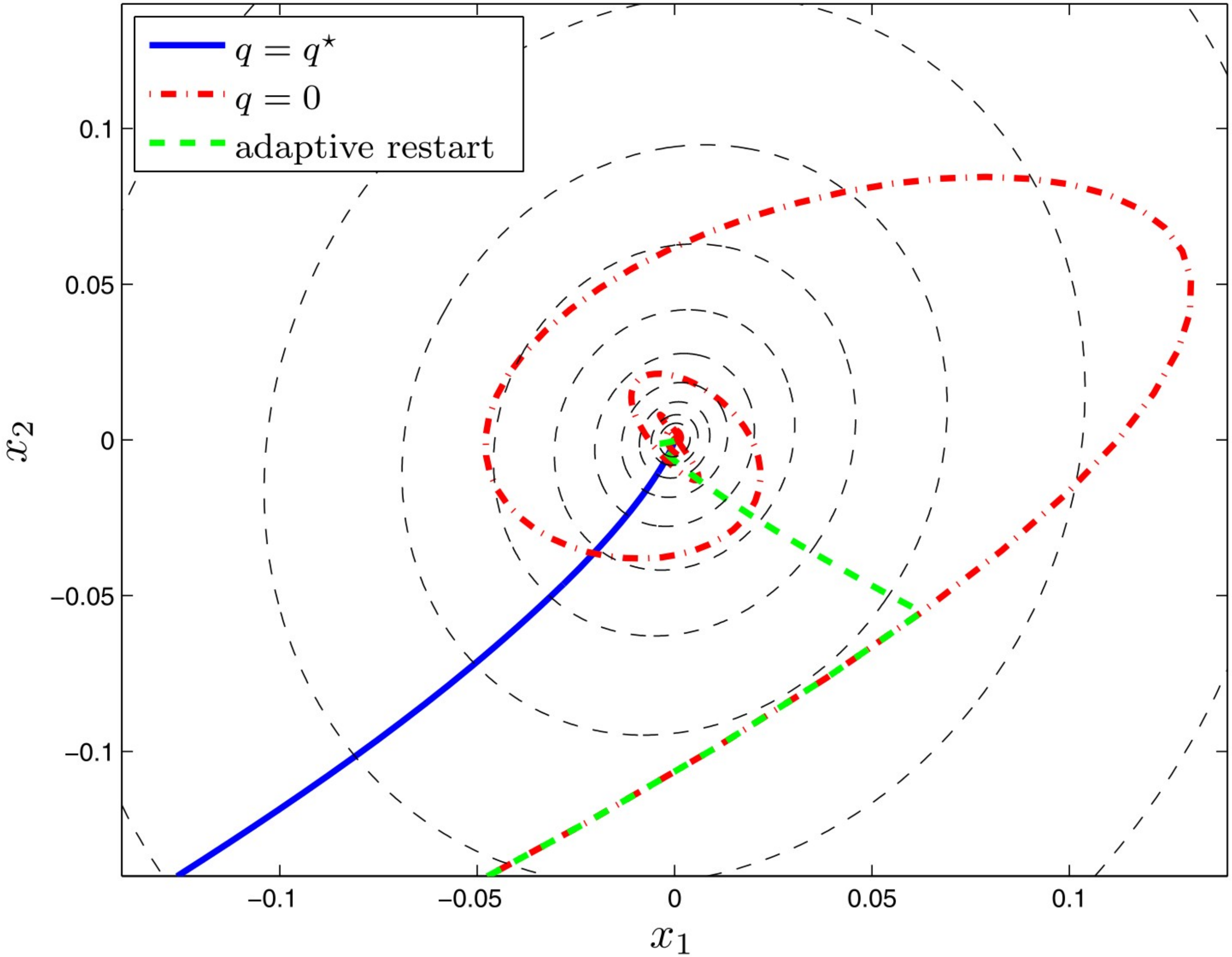
$\mathtt{\&C} \to \mathsf{B}$

```c
while ((jj < *max_iter) && (cond < 0)) {
 jj++;
 copy_vec_part_negate(v,tmp_var_p,40);
 mat_vec_mult_sparse(&CT,tmp_var_p,tmp_var_n);
 vec_sub(tmp_var_n,q1,tmp_var_n,60);
 stack_vec(tmp_var_n,q2,rhs,60,40);
 perm_fwdsolve(&L,p,rhs,tmp_var_nm);
 mat_vec_mult_sparse(&Dinv,tmp_var_nm,tmp_var_nm2);
 backsolve_perm(&LT,p,tmp_var_nm2,tmp_var_nm);
 copy_vec_part(tmp_var_nm,x,60);
 mat_vec_mult_sparse(&C,x,tmp_var_p);
 vec_add(v,tmp_var_p,tmp_var_p,40);
 copy_vec_part(tmp_var_p,arg_prox_h,40);
 clip_soft(tmp_var_p,l,u,(double *) &soft,40);
 mat_vec_mult_diag(&Einv,tmp_var_p,y);
 copy_vec_part(lambda,lambda_old,40);
 vec_sub(arg_prox_h,tmp_var_p,lambda,40);
 vec_sub(lambda,lambda_old,tmp_var_p,40);
 theta_old = theta;
 theta = (1+sqrt(1+4*pow(theta_old,2)))/2;
 scalar_mult((theta_old-1)/theta,tmp_var_p,40);
 copy_vec_part(v,v_old,40);
 vec_add(tmp_var_p,lambda,v,40);
 if (mod(jj,10) == 0) {
  cond = check_stop_cond_FGM(&Einv,lambda,lambda_old,tmp_var_p,tmp_var_p2,40,1e-09);
 }
 restart(lambda,lambda_old,v,v_old,tmp_var_p,tmp_var_p2,40);
}
```

```c
static void restart(double *x, double *x_old, double *y, double *y_old, double *tmp_v
 double test;
 vec_sub(y_old,x,tmp_var_p,n);
 vec_sub(x,x_old,tmp_var_p2,n);
 test = scalar_prod(tmp_var_p,tmp_var_p2,n);
 if (test > 0) {
 copy_vec_part(x_old,y,n);
 copy_vec_part(x_old,x,n);
 }
 }
```

Restart

(B. O'Donoghue, E. Candès, Adaptive Restart for Accelerated Gradient Schemes, Found Comput Math 2013.)



Complexity certification

(S. Richter, Computational Complexity Certification of Gradient Methods for Real-Time Model Predictive Control, ETH Zurich 2012)

- Upper bound on the required number of floating point operations that in turn stems from an upper bound on the iteration count

Complexity certification

- Upper bound on the required number of floating point operations that in turn stems from an upper bound on the iteration count
- accuracy => no. of iterations => no. of operations => computing time

Complexity certification

- Explicit: fixed complexity

Complexity certification

- Explicit: fixed complexity
- Active set:

Complexity certification

- Explicit: fixed complexity
- Active set:
  - Finite termination

Complexity certification

- Explicit: fixed complexity
- Active set:
  - Finite termination
  - Worst case number of iterations huge

Complexity certification

- Explicit: fixed complexity
- Active set:
  - Finite termination
  - Worst case number of iterations huge
  - Early termination unexplored

Complexity certification

- Explicit: fixed complexity
- Active set:
  - Finite termination
  - Worst case number of iterations huge
  - Early termination unexplored
- Interior point:

Complexity certification

- Explicit: fixed complexity
- Active set:
  - Finite termination
  - Worst case number of iterations huge
  - Early termination unexplored
- Interior point:
  - Certificates very conservative

Complexity certification

- Explicit: fixed complexity
- Active set:
  - Finite termination
  - Worst case number of iterations huge
  - Early termination unexplored
- Interior point:
  - Certificates very conservative
- First order:

Complexity certification

- Explicit: fixed complexity
- Active set:
  - Finite termination
  - Worst case number of iterations huge
  - Early termination unexplored
- Interior point:
  - Certificates very conservative
- First order:
  - Certificates within few orders of magnitude

Complexity certification

- State of the art for first order methods:

Complexity certification

- State of the art for first order methods:
  - Derived for certain (simpler) algorithms

Complexity certification

- State of the art for first order methods:
  - Derived for certain (simpler) algorithms

    (not generalized, no restarting)

Complexity certification

- State of the art for first order methods:
  - Derived for certain (simpler) algorithms

    (not generalized, no restarting)
  - Derived for certain (simpler) constraints

Complexity certification

- State of the art for first order methods:
  - Derived for certain (simpler) algorithms

    (not generalized, no restarting)
  - Derived for certain (simpler) constraints

Nesterov

(Y. Nesterov, Introductory lectures on convex optimization : a basic course, Springer 2004)

- Convex optimization

Nesterov

(Y. Nesterov, Introductory lectures on convex optimization : a basic course, Springer 2004)

- Convex optimization
- Does not mention duality

Richter

(S. Richter, Computational Complexity Certification of Gradient Methods for Real-Time Model Predictive Control, ETH Zurich 2012)

- MPC with some QP

Richter

(S. Richter, Computational Complexity Certification of Gradient Methods for Real-Time Model Predictive Control, ETH Zurich 2012)

- MPC with some QP
- Duality: partial Lagrange relaxation (simple state constraints), shows that complete is worse, but complete is what we need

Richter

(S. Richter, Computational Complexity Certification of Gradient Methods for Real-Time Model Predictive Control, ETH Zurich 2012)

- MPC with some QP
- Duality: partial Lagrange relaxation (simple state constraints), shows that complete is worse, but complete is what we need
- Duality: dual problem not strongly concave

Temporary conclusion

- What to do? Transform the MPC-obtained QP into dual space

Temporary conclusion

- What to do? Transform the MPC-obtained QP into dual space
- Follow either Richter or Nesterov with dual QP

Temporary conclusion

- What to do? Transform the MPC-obtained QP into dual space
- Follow either Richter or Nesterov with dual QP
- Preconditioning?
- Restarting?