Operator Splitting Methods for Fast MPC

Colin Jones

Laboratoire d'Automatique, EPFL
Some material from Stephen Boyd's lecture notes on ADMM

Introduction

Many optimization problems in control have the form

$$\begin{aligned} \min\ & f(x) \\ \text{s.t. }\ & Ax = b \\ & x \in X \end{aligned}$$

where

- $f(x)$ is **smooth**, and often **quadratic**
- X is **simple** ($\min_x \|x - \bar{x}\|,$ s.t. $x \in X$ is **easy**)

How can we use this information to develop a simple optimization scheme?

Idea : Break into sequence of **easy** problems involving f or X alone

Outline

1. Duality
2. Dual Decomposition
3. Method of Multipliers
4. Alternating Direction Method of Multipliers
5. Common Patterns in Control
6. ADMM for MPC
7. Alternating Minimization Algorithm
8. Accelerating Convergence

Duality

Primal problem:

$$\min_z \; f(z) \quad \text{s.t. } Az = b$$

Define the **Lagrangian**

$$L(z, \lambda) = f(z) + \lambda^T (Az - b)$$

and the **dual function**

$$d(\lambda) = \min_z \; L(z, \lambda)$$

The dual problem is

$$\max_\lambda \; d(\lambda)$$

Recover the primal optimal solution from

$$z^\star = \text{argmin}_z \; L(z, \lambda^\star)$$

Properties of the Dual Function

$\boxed{d(\lambda) \text{ is concave}^1}$

$$d(\lambda) = \min_z \; f(z) + \lambda^T (Az - b)$$

The dual function is the pointwise minimum of affine functions

[1] This is true whether f is convex or not

Properties of the Dual Function

$\boxed{d(\lambda) \text{ is concave}^1}$

$$d(\lambda) = \min_z \; f(z) + \lambda^T(Az - b)$$

The dual function is the pointwise minimum of affine functions

$\boxed{d(\lambda) \leq f(z) \text{ for all } \lambda \text{ and all } z \text{ such that } Az = b}$

Given a feasible $\bar{z}$ such that $A\bar{z} = b$

$$f(\bar{z}) = f(\bar{z}) + \lambda^T(A\bar{z} - b) \geq \min_z \; L(z, \lambda) = d(\lambda)$$

Dual function gives lower bounds on the optimal solution.

[1] This is true whether f is convex or not

Dual Problem

Lagrange dual problem (find the best lower bound)

$$\max_{\lambda} \; d(\lambda)$$

- Always a convex optimization problem
- $\max_\lambda d(\lambda) \leq \min_{Ax=b} f(x)$

If problem is convex, then (under mild assumptions), we have strong duality:

$$\max_{\lambda} \; d(\lambda) = \min_{z} f(z) \text{ s.t. } Az = b$$

We can solve the primal, or the dual (whichever is easier).

Dual Ascent Method

Dual problem:

$$\max_{\lambda} \; d(\lambda)$$

This is a convex, unconstrained optimization problem.

We would like to apply a gradient ascent approach:

$$\lambda^{k+1} = \lambda^k + c^k \nabla d(\lambda^k)$$

$\rightarrow$ How do we compute a gradient of the dual function?

(Note: We're making the strong assumption that the dual function is differentiable here. Similar procedure in the non-differentiable case too.)

Gradient of the Dual

Theorem:

If $\bar{z} = \text{argmin}\, L(z, \lambda)$, then $A\bar{z} - b \in \partial d(\lambda)$

Recall: g is a supergradient of a function h at x if and only if

$$h(x) - g^T(x - y) \geq h(y)$$

for all y.

$$\begin{aligned}
d(\lambda) - (A\bar{z} - b)^T(\lambda - \hat{\lambda}) &= L(\bar{z}, \lambda) - (A\bar{z} - b)^T(\lambda - \hat{\lambda}) \\
&= f(\bar{z}) + \lambda^T(A\bar{z} - b) - (A\bar{z} - b)^T(\lambda - \hat{\lambda}) \\
&= f(\bar{z}) + \hat{\lambda}^T(A\bar{z} - b) \\
&\geq d(\hat{\lambda})
\end{aligned}$$

Note: If d is differentiable, then $\partial d(\lambda) = \nabla d(\lambda)$

Dual Gradient Method

We can compute the gradient of the dual and implement the gradient method:

$$x^{k+1} = \text{argmin}_x \ L(x, \lambda^k)$$
$$\lambda^{k+1} = \lambda^k + c(Ax^{k+1} - b)$$

This works, but requires a number of strong assumptions.

Outline

1. Duality
2. Dual Decomposition
3. Method of Multipliers
4. Alternating Direction Method of Multipliers
5. Common Patterns in Control
6. ADMM for MPC
7. Alternating Minimization Algorithm
8. Accelerating Convergence

Dual Decomposition

Suppose our problem has the form:

$$\begin{aligned} &\min\ f(x) + g(y) \\ &\text{s.t. } Ax + By = d \end{aligned}$$

Computing the dual function:

$$\begin{aligned} \min_{x,y}\ L(x,y,\lambda) &= \min_{x,y}\ f(x) + g(y) + \lambda^T(Ax + By - d) \\ &= \min_{x}\ (f(x) + \lambda^T Ax) + \min_{y}\ (g(y) + \lambda^T By) - d \end{aligned}$$

The Lagrangian function is **separable** for a fixed λ!

Algorithm becomes:

$$\begin{aligned} x_{k+1} &= \text{argmin}_x\ f(x) + \lambda_k^T Ax \\ y_{k+1} &= \text{argmin}_y\ g(y) + \lambda_k^T By \\ \lambda_{k+1} &= \lambda_k + c(Ax_{k+1} + By_{k+1} - b) \end{aligned}$$

Minimizing f and g **independently** and **in parallel**

Dual Decomposition

Benefits:

- Can solve very large problems in parallel
- $\min f$ and $\min g$ may be much simpler to solve than $\min f + g$

Limitations:

- The function value converges non-monotonically to the optimal value
  Doesn't matter for control
- Slow (sub-gradient method)
- If objective is not strongly convex, then the primal iterates x^k do not necessarily converge
  - MPC objectives will almost never be strongly convex because they will include indicator functions
  - The primal iterates are the control law $\rightarrow$ these must converge!

Outline

Augmented Lagrangian

$$\begin{aligned} &\min \ f(z) \\ &\text{s.t. } Az = b \end{aligned}$$

Augmented Lagrangian

$$\begin{aligned} &\min\ f(z) \\ &\text{s.t. } Az = b \end{aligned}$$

Add a penalty term to the cost function to make it strongly convex:

$$\begin{aligned} &\min\ f(z) + \frac{\rho}{2}\|Az - b\|^2 \\ &\text{s.t. } Az = b \end{aligned}$$

Note that this doesn't change the solution!

Recall: $f(z)$ is strongly convex if

$$\forall z_1, z_2, \forall t \in (0,1) \quad f(tz_1 + (1-t)z_2) < tf(z_1) + (1-t)f(z_2)$$

Dual Function

The (augmented) Lagrangian is:

$$L_\rho(z,\lambda) = f(z) + \lambda^T(Az-b) + \frac{\rho}{2}\|Az-b\|^2$$

The dual function is

$$d(\lambda) = \min_z \; L_\rho(z,\lambda)$$

Note that the dual function has changed

Theorem: Convex Analysis, Rockafellar (1970)

If a convex program has a strongly convex objective, it has a unique solution and its Lagrangian dual function is differentiable.

- Convergence of the iterates
- Differentiable function $\rightarrow$ faster gradient method, rather than sub-gradient

Augmented Lagrangian Method

$$
\begin{aligned}
x_{k+1} &= \text{argmin}_x \;\; f(x) + \lambda_k^T(Ax - b) + \frac{\rho}{2}\|Ax - b\|^2 \\
\lambda_{k=1} &= \lambda_k + \rho(Ax_{k+1} - b)
\end{aligned}
$$

We want to apply this to problems of the form

$$
\begin{aligned}
&\min \; f(x) + g(y) \\
&\text{s.t. } Ax + By = d
\end{aligned}
$$

Problem: Augmented Lagrangian doesn't decompose

$$
\min_{x,y} \;\; f(x) + g(y) + \lambda_k^T(Ax + By - b) + \frac{\rho}{2}\underbrace{\|Ax + By - b\|^2}_{\text{Couples } x \text{ and } y}
$$

Augmented Lagrangian Method

Positive:

- Converges under extremely loose conditions:
  non-differentiable functions, unbounded functions / indicator functions, etc

Negative:

- Does not decompose / parallelize due to the quadratic term

Outline

Alternating Direction Method of Multipliers

$$\begin{aligned}\min\ & f(x) + g(y)\\ \text{s.t. } & Ax + By = b\end{aligned}$$

$$L_\rho(x, y, \lambda) = f(x) + g(y) + \lambda^T(Ax + By - b) + \frac{\rho}{2}\|Ax + By - b\|^2$$

ADMM:

$$\begin{aligned}x^{k+1} &= \text{argmin}_x\ \ L_\rho(x, y^k, \lambda^k)\\ y^{k+1} &= \text{argmin}_y\ \ L_\rho(x^{k+1}, y, \lambda^k)\\ \lambda^{k+1} &= \lambda^k + \rho(Ax^{k+1} + By^{k+1} - b)\end{aligned}$$

Idea: Approximate the computation of the dual (sort of) via one step of Gauss-Seidel / block coordinate descent.

ADMM: A Cleaner Formulation

Combine linear and quadratic terms:

$$L_\rho(x, y, \lambda) = f(x) + g(y) + \lambda^T(Ax + By - b) + \frac{\rho}{2}\|Ax + By - b\|^2 \tag{1}$$

$$= f(x) + g(y) + \frac{\rho}{2}\|Ax + By - b + \mu\|^2 \tag{2}$$

where $\mu = \frac{1}{\rho}\lambda$

ADMM (scaled form):

$$x^{k+1} = \text{argmin}_x \;\; f(x) + \frac{\rho}{2}\|Ax + By^k - b + \mu^k\|^2$$

$$y^{k+1} = \text{argmin}_y \;\; g(y) + \frac{\rho}{2}\|Ax^{k+1} + By - b + \mu^k\|^2$$

$$\mu^{k+1} = \mu^k + Ax^{k+1} + By^{k+1} - b$$

Convergence of ADMM

If

- f, g convex, closed, proper
- L_ρ has a saddle point (i.e., an optimal solution exists).
  Note that this requires that the problem is feasible!

then

- iterates approach feasibility $Ax^k + By^k - b \to 0$
- objective approaches optimal value
  $f(x^k) + g(y^k) \to \min_{x,y} f(x) + g(y)$ s.t. $Ax + By = c$

Outline

1. Duality
2. Dual Decomposition
3. Method of Multipliers
4. Alternating Direction Method of Multipliers
5. Common Patterns in Control
6. ADMM for MPC
7. Alternating Minimization Algorithm
8. Accelerating Convergence

Easily Evaluated Updates

Given a function $f(x)$, we need to compute

$$x^{+} = \operatorname{argmin}_x f(x) + \frac{\rho}{2}\|Ax - v\|^2$$

If $A = I$ (common), then this is called the **proximal operator** of f

$$\operatorname{prox}_{f,\rho}(v) = \operatorname{argmin}_x f(x) + \frac{\rho}{2}\|x - v\|^2$$

For which types of functions f can we evaluate this easily?

Common Patterns in Control

▷ Upper / lower bounds

$$f(x) = \begin{cases} \infty & x \leq l \\ 0 & l \leq x \leq u \\ \infty & x \geq u \end{cases}$$

$$\begin{aligned} \mathrm{prox}_{f,\rho}(v) &= \mathrm{argmin}_x \;\; f(x) + \frac{\rho}{2}\|x - v\|^2 \\ &= \;\mathrm{argmin}_x \; \|x - v\|^2 \\ &\qquad\qquad \text{s.t. } l \leq x \leq u \\ &= \begin{cases} l & v \leq l \\ v & l \leq v \leq u \\ u & v \geq u \end{cases} \end{aligned}$$

Evaluation of the proximal operator is trivial!

Common Patterns in Control

▷ Polytopic constraints

$$\begin{aligned} \min\ & f(x) \\ \text{s.t. } & Hx \leq h \end{aligned}$$

Re-write using slack variables:

$$\begin{aligned} \min\ & f(x) + g(s) \\ \text{s.t. } & Hx + s = h \end{aligned}$$

where $g(s)$ is the indicator function for the positive orthant

$$g(s) = \begin{cases} 0 & s \geq 0 \\ \infty & \text{otherwise} \end{cases}$$

$$\text{prox}_{g,\rho}(s) = \max\{s, 0\}$$

Common Patterns in Control

- Quadratic function

$$f(x) = \frac{1}{2}x^T Qx + c^T x$$

$$\mathrm{prox}_{f,\rho}(v) = \mathrm{argmin}_x \ \ \frac{1}{2}x^T Qx + c^T x + \frac{\rho}{2}(x-v)^T(x-v)$$

Take gradient, set to zero

$$\begin{aligned} Qx + c + \rho(x-v) &= 0 \\ \mathrm{prox}_{f,\rho}(v) &= (Q+\rho I)^{-1}(\rho v - c) \end{aligned}$$

Other Common Patterns

Many other common constraints and functions have nice proximal operators

- Ellipsoidal / ball-constraints
- Vector norms: $1-$, $2-$, $\inf -$norms
- Matrix norms: Frobenius-norm, $2-$norm, Nuclear-norm
- Standard convex cones: second-order cone, semi-definite cone, positive orthant, etc

Distributed Optimization

Separable cost function with shared variables

$$\begin{aligned} \min \ & \sum f_i(x_i) \\ \text{s.t. } & x_i = z \text{ for all } i \end{aligned}$$

Note that $g(z) = 0$.

Augmented Lagrangian is:

$$L(x_0, \ldots, x_n, \mu) = \sum f_i(x_i) + \sum \frac{\rho}{2} \|x_i - z + \mu_i\|^2$$

ADMM:

$$\begin{aligned} x_i^{k+1} &= \text{argmin}_{x_i} \ \ f_i(x_i) + \frac{\rho}{2}\|x_i - z^k + \mu_i^k\|^2 && \text{Parallel} \\ z^{k+1} &= \text{argmin}_z \ \sum \frac{\rho}{2}\|x_i^{k+1} - z + \mu_i^k\|^2 && \\ &= \frac{1}{n}\sum x_i^{k+1} + \mu_i^k && \text{Consensus} \\ \mu_i^{k+1} &= \mu_i^k + x_i^{k+1} - z^{k+1} && \text{Parallel} \end{aligned}$$

Outline

1. Duality
2. Dual Decomposition
3. Method of Multipliers
4. Alternating Direction Method of Multipliers
5. Common Patterns in Control
6. ADMM for MPC
7. Alternating Minimization Algorithm
8. Accelerating Convergence

Linear Quadratic Predictive Control

- Linear dynamics
- Quadratic stage costs
- Simple stage constraints

$$\min_{x,u} \sum_{i=0}^{N-1} x_i' Q x_i + u_i' R u_i$$
$$\text{s.t. } x_{i+1} = A x_i + B u_i$$
$$x_i \in X, \ u_i \in U$$

Assumption: Prox operators for X and U are simple
(Also possible for more complex sets)

How to define functions f and g?

Sequential Convex Program

$$\min_{x,u} \sum_{i=0}^{N-1} x_i' Q x_i + u_i' R u_i$$

$$\text{s.t. } x_{i+1} = A x_i + B u_i$$

$$\boxed{\bar{x}_i = x_i\,,\ \ \bar{u}_i = u_i}$$

$$\bar{x}_i \in X\,,\ \ \bar{u}_i \in U$$

Make a 'copy' of states and inputs

Sequential Convex Program

$$
\begin{aligned}
\min_{x,u} \; & \sum_{i=0}^{N-1} x_i' Q x_i + u_i' R u_i \\
\text{s.t. } & x_{i+1} = A x_i + B u_i \\
& \bar{x}_i = x_i, \; \bar{u}_i = u_i \\
& \bar{x}_i \in X, \; \bar{u}_i \in U
\end{aligned}
$$

Sequential Convex Program

$$
\begin{aligned}
\min_{x,u} \; & \sum_{i=0}^{N-1} x_i' Q x_i + u_i' R u_i \\
\text{s.t. } & x_{i+1} = A x_i + B u_i \\
& \bar{x}_i = x_i, \;\; \bar{u}_i = u_i \\
& \bar{x}_i \in X, \;\; \bar{u}_i \in U
\end{aligned}
$$

$f(x,u)$

Linear quadratic regulator

Sequential Convex Program

$$\min_{x,u} \sum_{i=0}^{N-1} x_i' Q x_i + u_i' R u_i$$

$$\text{s.t. } x_{i+1} = A x_i + B u_i$$

$$\bar{x}_i = x_i, \ \bar{u}_i = u_i$$

$$\bar{x}_i \in X, \ \bar{u}_i \in U$$

$f(x,u)$

Linear quadratic regulator

Linear coupling constraints

$\bar{x}_i, \bar{u}_i \ \leftrightarrow \ x_i, u_i$

Sequential Convex Program

$$
\begin{aligned}
\min_{x,u} \quad & \sum_{i=0}^{N-1} x_i' Q x_i + u_i' R u_i \\
\text{s.t.} \quad & x_{i+1} = A x_i + B u_i \\
& \bar{x}_i = x_i, \ \ \bar{u}_i = u_i \\
& \bar{x}_i \in X, \ \ \bar{u}_i \in U
\end{aligned}
$$

$f(x,u)$

Linear quadratic regulator

Linear coupling constraints

$\bar{x}_i, \bar{u}_i \ \leftrightarrow \ x_i, u_i$

$g(\bar{x}_i, \bar{u}_i)$

Simple constraints

Sequential Convex Program – Proximal Operators

1. LQR (Linearly constrained least-squares)

$$\text{prox}_{\rho f}(\tilde{x}, \tilde{u}) = \min_{x,u} \sum_{i=0}^{N-1} x_i' Q x_i + u_i' R u_i + \frac{\rho}{2}\|x_i - \tilde{x}_i\|_2^2 + \frac{\rho}{2}\|u_i - \tilde{u}_i\|_2^2 = M \begin{bmatrix} \tilde{x} \\ \tilde{u} \end{bmatrix}$$

$$\text{s.t. } x_{i+1} = A x_i + B u_i$$

2. Stage constraints

- Box (upper/lower bounds) → Clipping
- Sphere → Scaling

Also possible with additional scaling:

- Polyhedron
- Ellipse

Putting it Together: Sequential Convex Programming

$$\min_{x,u} \sum_{i=0}^{N-1} x_i' Q x_i + u_i' R u_i$$
$$\text{s.t. } x_{i+1} = A x_i + B u_i$$
$$\bar{x}_i = x_i, \;\; \bar{u}_i = u_i$$
$$\bar{x}_i \in X, \;\; \bar{u}_i \in U$$

$$\min \; f(x) + g(y)$$
$$\text{s.t. } Ax + By = c$$

$$\begin{bmatrix} x^{k+1} \\ u^{k+1} \end{bmatrix} = M \begin{bmatrix} \bar{x}^k + \lambda^k \\ \bar{u}^k + \nu^k \end{bmatrix}$$

Multiplication

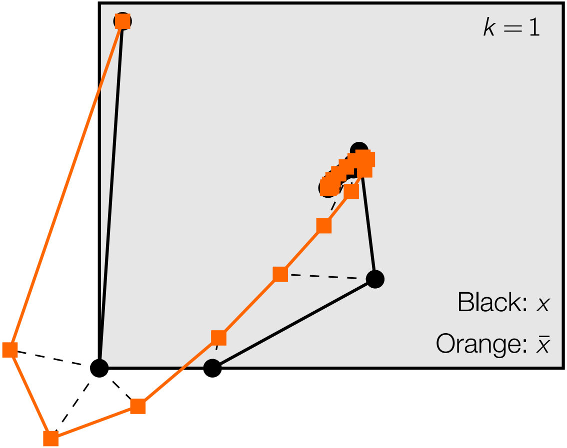
$$\bar{x}^{k+1} = \pi_X(x^{k+1}) \qquad \bar{u}^{k+1} = \pi_U(u^{k+1})$$

Clipping

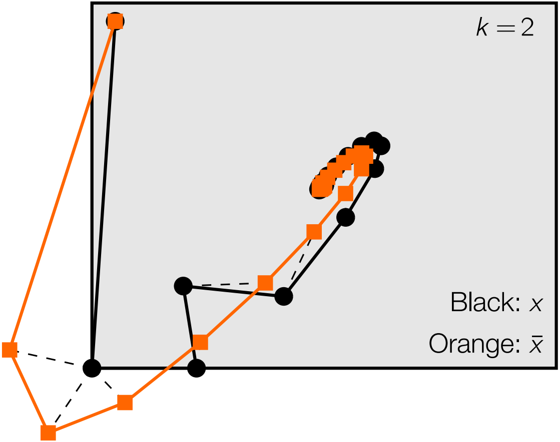
$$\lambda^{k+1} = \lambda^k + x^{k+1} - \bar{x}^{k+1}$$
$$\nu^{k+1} = \nu^k + u^{k+1} - \bar{u}^{k+1}$$

Addition

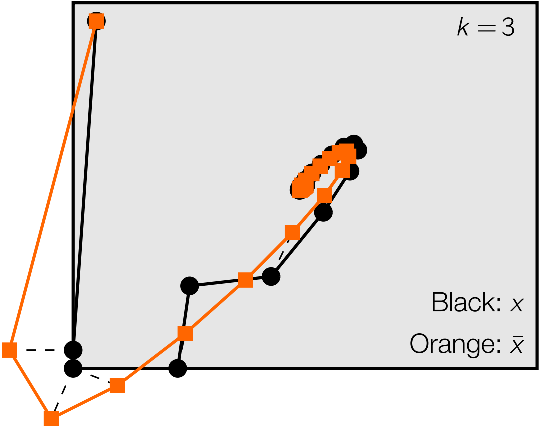
Second-Order Example



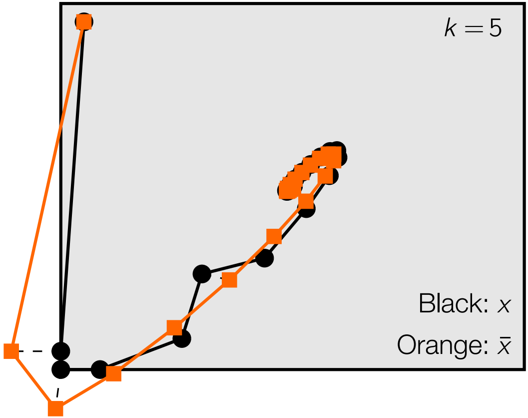
Second-Order Example



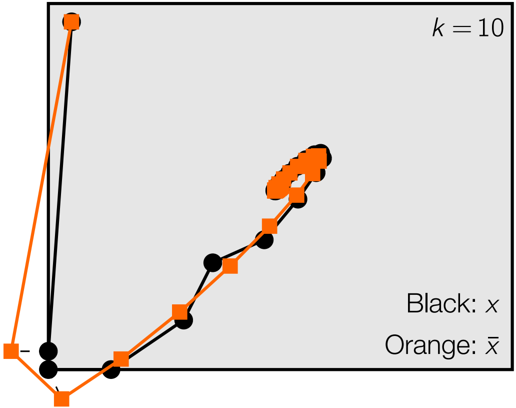
Second-Order Example



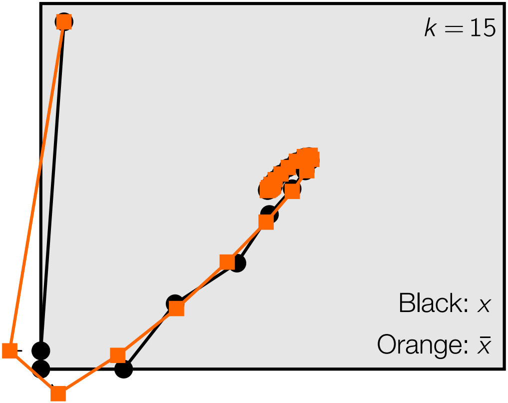
Second-Order Example



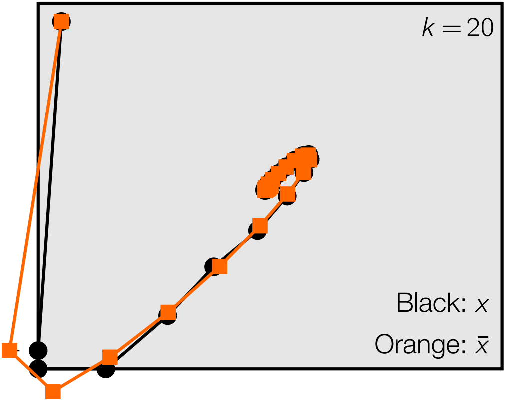
Second-Order Example



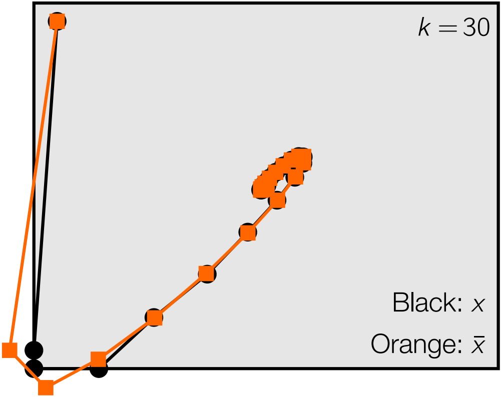
Second-Order Example



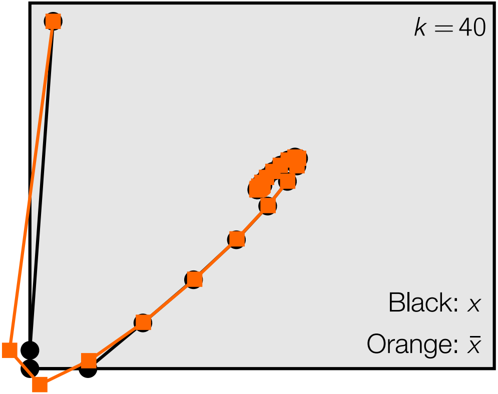
Second-Order Example



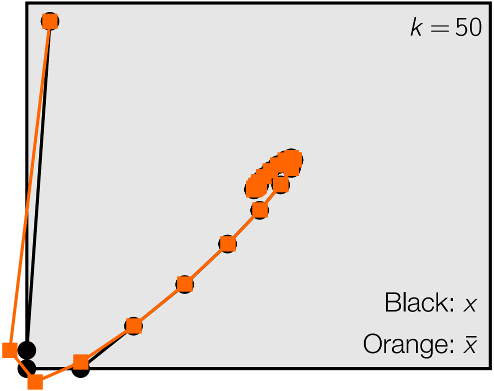
Second-Order Example



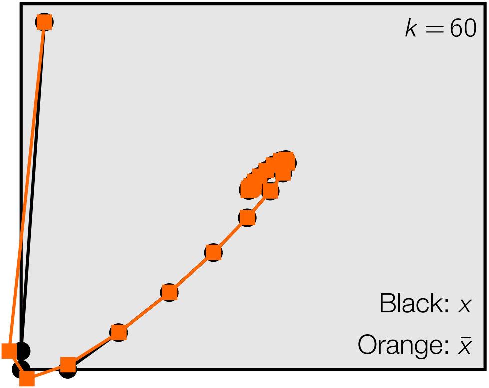
Second-Order Example



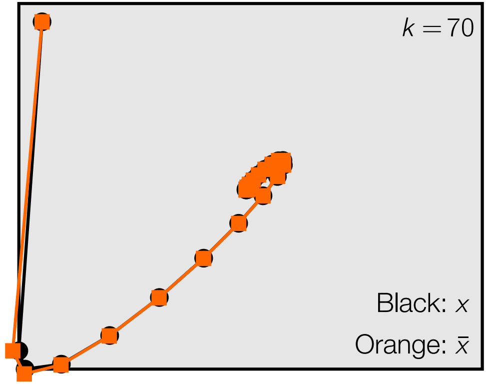
Second-Order Example



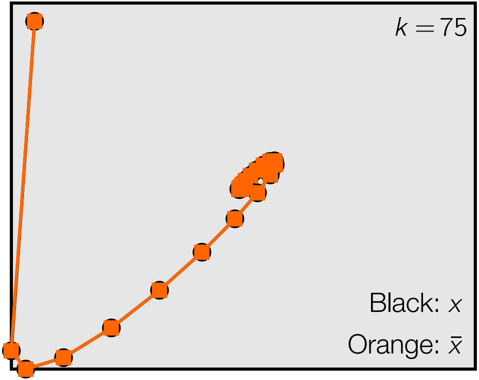
Second-Order Example



Second-Order Example

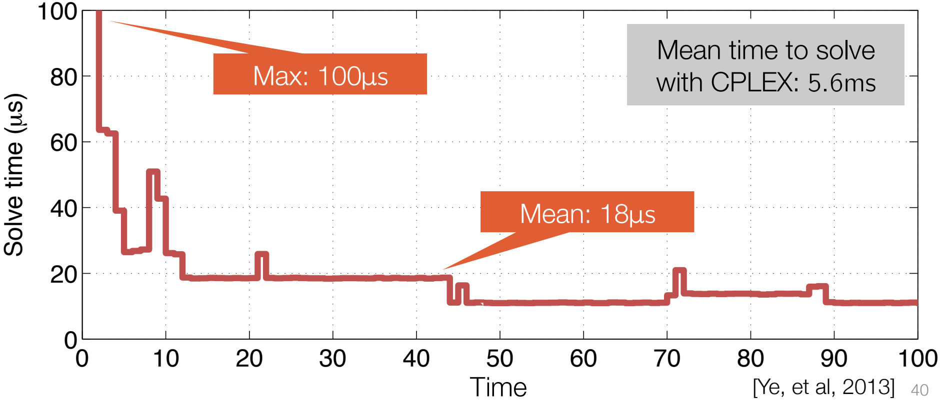


Second-Order Example



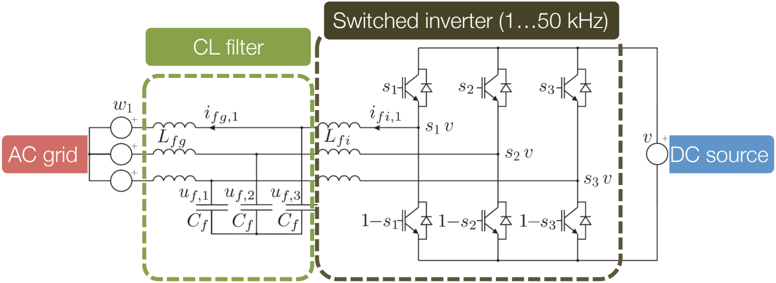
Example: Quad-Copter

- 7 states, 4 inputs, horizon 8
- Upper/lower bounds on states and inputs
- Ellipsoidal terminal set
- Algorithm: Fast AMA



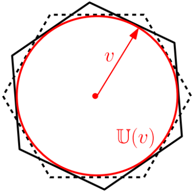
[Ye, et al, 2013]

Example: AC/DC Converter

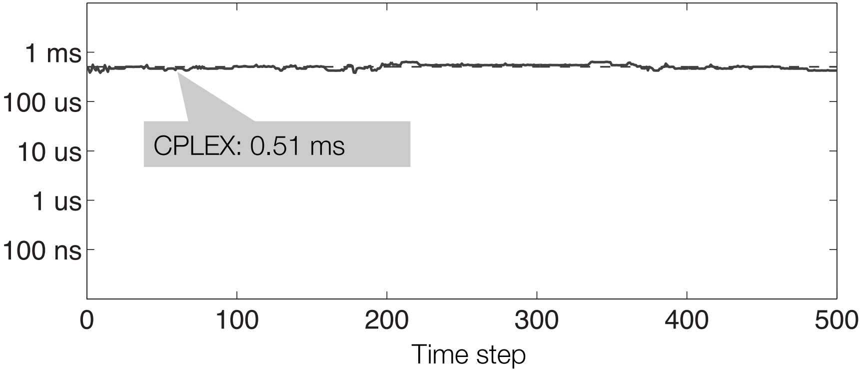


$$\min \ \frac{1}{2} u^T H u + g(x, x_{ss}, u_{ss}, w)^T u$$

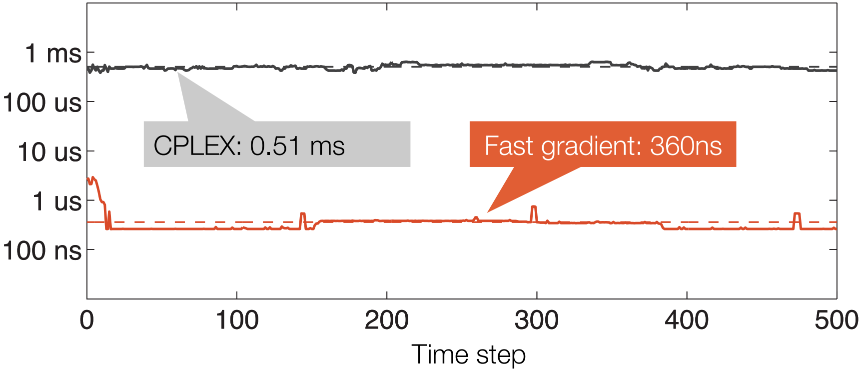
$$\text{s.t.} \ u_k \in \mathbb{U}(\nu) - u_{ss} \ , \ k = 0, \ldots, N-1$$



Performance of Auto-Tuned FGM on 2.5GHz PC



Performance of Auto-Tuned FGM on 2.5GHz PC



On average 1400x faster than CPLEX

[Richter, et al, 2010]

Subspace Identification with Rank Regularization

$$
\begin{aligned}
\min \quad & \overbrace{\|\bar{Y} - H\bar{U}\|_F^2}^{f(H)} + \overbrace{\tau\|\mathcal{H}\|_\star}^{g(\mathcal{H})} \\
\text{s.t. } & \mathcal{L}(\mathcal{H}) = H
\end{aligned}
$$

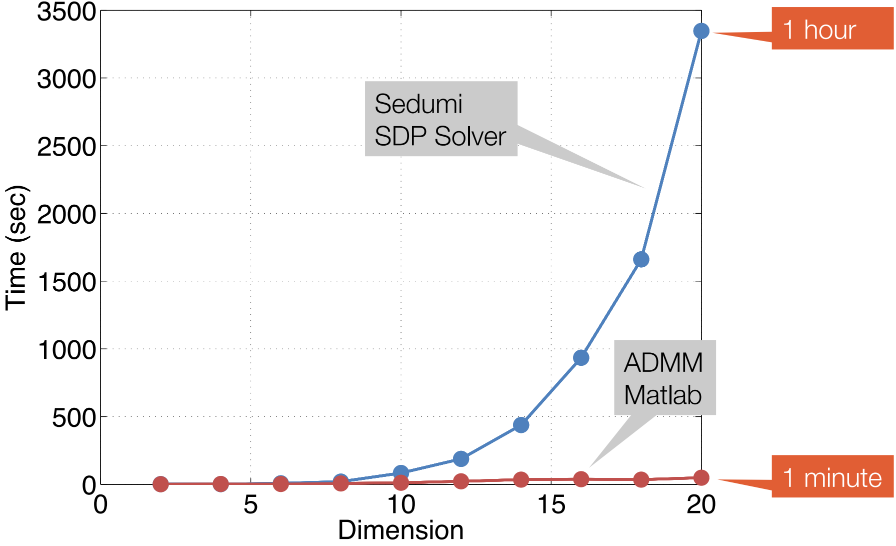
Impose Hankel structure

Step 1: Matrix multiply
- Prox of Frobenius norm is unconstrained least-squares
- Compute off-line

Step 2: Singular-value decomposition
- Prox of nuclear norm => Clip singular values that are too big

Procedure:
1. Matrix-matrix multiply
2. Singular-value decomposition
3. Matrix-vector multiply and addition

Subspace Identification - Example



Predictive Control

Infinite-Horizon Optimal Control

$$u^\star(x) := \operatorname{argmin} \sum_{i=0}^{\infty} x_i^T Q x_i + u_i^T R u_i$$
$$\text{s.t.}\ \ x_{i+1} = Ax_i + Bu_i$$
$$x_i \in X \quad u_i \in U$$
$$x_0 = x$$

Properties of optimal controller:

- Stabilizing
- Invariant (satisfies constraints)
- Maximizes region of attraction
- 'Optimal' performance

Stabilizing Predictive Control

$$\min \sum_{i=0}^{N} x_i^T Q x_i + u_i^T R u_i + V_f(x_N)$$
$$\text{s.t.}\ \ x_{i+1} = Ax_i + Bu_i$$
$$x_i \in X \quad u_i \in U$$
$$x_0 = x$$
$$x_N \in X_N$$

Additional cost / constraints enforce

- Stability
- Constraint satisfaction

Downsides:

- X_N calculable for limited systems
- $\underbrace{\text{Small region of attraction}}_{\text{Reason never used in practice}}$

Operator Splitting on Hilbert Spaces

$$\min \; f(\mathbf{z}) + g(\boldsymbol{\sigma}) \qquad \text{s.t. } \mathcal{A}\mathbf{z} - \boldsymbol{\sigma} = \mathbf{b}$$

$\mathbf{z} = (z_i)_{i\in\mathbb{N}}$, $z_i = (x_i, u_i)$ are elements of a real Hilbert space

$$\begin{aligned} \min \quad & \sum_{i=0}^{\infty} x_i^T Q x_i + u_i^T R u_i \\ \text{s.t.} \quad & x_{i+1} = A x_i + B u_i, \; x_0 = x \\ & C x_i + D u_i - \sigma_i = b \\ & \sigma_i \geq 0 \end{aligned}$$

$$f(\mathbf{z}) = \sum_{i=0}^{\infty} x_i^\top Q x_i + u_i^\top R u_i + \delta_D(\mathbf{z})$$

$$(\mathcal{A}\mathbf{z})_i = C x_i + D u_i = \bar{C} z_i$$

$$g(\boldsymbol{\sigma}) = \begin{cases} 0 & \sigma_i \geq 0 \; \forall i \in \mathbb{N} \\ \infty & \text{otherwise} \end{cases}$$

Theorem: Optimal solution can be computed with a finite amount of memory and computation

[Stathopolous, Korda, Jones, IFAC 2014]

Outline

Alternating Direction Method of Multipliers

$$\begin{aligned} \min\ & f(x) + g(y) \\ \text{s.t. } & Ax + By = b \end{aligned}$$

Lagrangian $\quad L(x, y, \lambda) = f(x) + g(y) + \lambda^T(Ax + By - b)$

Augmented Lagrangian $\quad L_\rho(x, y, \lambda) = f(x) + g(y) + \lambda^T(Ax + By - b) + \frac{\rho}{2}\|Ax + By - b\|^2$

ADMM:

$$x^{k+1} = \text{argmin}_x \ L_\rho(x, y^k, \lambda^k) \qquad \text{Augmented Lagrangian}$$

$$y^{k+1} = \text{argmin}_y \ L_\rho(x^{k+1}, y, \lambda^k) \qquad \text{Augmented Lagrangian}$$

$$\lambda^{k+1} = \lambda^k + \rho(Ax^{k+1} + By^{k+1} - b)$$

Requirement : f and g convex

Alternating Minimization Algorithm (AMA)

$$\min \; f(x) + g(y)$$
$$\text{s.t. } Ax + By = b$$

Lagrangian	$L(x, y, \lambda) = f(x) + g(y) + \lambda^T(Ax + By - b)$
Augmented Lagrangian	$L_\rho(x, y, \lambda) = f(x) + g(y) + \lambda^T(Ax + By - b) + \frac{\rho}{2}\|Ax + By - b\|^2$

AMA:

$$x^{k+1} = \text{argmin}_x \;\; L(x, y^k, \lambda^k) \qquad \text{Lagrangian}$$
$$y^{k+1} = \text{argmin}_y \;\; L_\rho(x^{k+1}, y, \lambda^k) \qquad \text{Augmented Lagrangian}$$
$$\lambda^{k+1} = \lambda^k + \rho(Ax^{k+1} + By^{k+1} - b)$$

Requirement : f **strongly convex**, g convex

AMA : A cleaner formulation

Problem prototype

$$\min\ f(x) + g(y)$$
$$\text{s.t. } Ax + b = y$$

where f is strongly convex

AMA:

$$\begin{aligned}
x^{k+1} &= \text{argmin}_x\ \ f(x) + \langle \lambda^k, Ax \rangle \\
y^{k+1} &= \text{prox}_{g,\rho}\left(Ax^{k+1} + b + \lambda^k/\rho\right) \\
\lambda^{k+1} &= \lambda^k + \rho(Ax^{k+1} + b - y^{k+1})
\end{aligned}$$

AMA vs ADMM

- ADMM has weaker assumptions (f must be strongly convex for AMA)
  - f is strongly convex for many control problems
- AMA better when minimizing Lagrangian is simpler than augmented form
- Theoretically stronger results for AMA
  - Acceleration
  - Pre-conditioning
- Tuning easier for AMA
  - Any stepsize ρ works for ADMM, limited for AMA
  - Optimal stepsize relates to properties of the functions being optimized

Note : The theoretical derivation of AMA and ADMM are very different

Outline

1. Duality
2. Dual Decomposition
3. Method of Multipliers
4. Alternating Direction Method of Multipliers
5. Common Patterns in Control
6. ADMM for MPC
7. Alternating Minimization Algorithm
8. Accelerating Convergence

Convergence Rates

Splitting methods are generally **slow** $O(1/k)$

$$f(z^k) - f(z^\star) \leq \frac{M}{k}$$

Key factors in speeding them up:

- Acceleration
- Pre-conditioning

Other factors to consider:

- Stepsize selection
- How to split / formulate problem?
- Which algorithm to use (many variants)?

See *[Stathopoulos et al, soon to be submitted]* for details

Accelerated Variants

$$\min \; f(z)$$

Gradient method

$$z^{k+1} = z^k - \rho^k \nabla f(z^k)$$

Heavy-ball method [Polyak, 1964]

$$\begin{aligned} \hat{z}^k &= z^k + \alpha^k (z^k - z^{k-1}) \\ z^{k+1} &= \hat{z}^k - \rho^k \nabla f(z^k) \end{aligned}$$

Nesterov acceleration [Nesterov, 1983]

$$\left.\begin{aligned} \alpha^k &= \left(1 + \sqrt{4(\alpha^{k-1})^2 + 1}\right)/2 \\ \hat{z}^k &= z^k + \frac{\alpha^{k-1} - 1}{\alpha^k}(z^k - z^{k-1}) \\ z^{k+1} &= \hat{z}^k - \rho^k \nabla f(\hat{z}^k) \end{aligned}\right\} \quad \text{Optimal convergence } O(1/k^2)$$

Fast Alternating Minimization Algorithm : FAMA

Acceleration can be directly applied to the dual sequence λ^k

$$
\begin{aligned}
x^{k+1} &= \text{argmin}_x \;\; f(x) + \langle \hat{\lambda}^k, Ax \rangle \\
y^{k+1} &= \text{prox}_{g,\rho}\left(Ax^{k+1} + b + \hat{\lambda}^k/\rho \right) \\
\lambda^k &= \hat{\lambda}^k + \rho(Ax^{k+1} + b - y^{k+1}) \\
\hat{\lambda}^{k+1} &= \lambda^k + ((\alpha^k - 1)/\alpha^{k+1})(\lambda^k - \lambda^{k-1})
\end{aligned}
$$

$\rightarrow$ Result : $O(1/k^2)$ convergence rate

- ADMM can be accelerated, but in a heuristic fashion (no guarantee that it goes faster)

Pre-conditioning

Unlike Newton methods, first-order methods are very sensitive to conditioning.

Consider the problem

$$\begin{aligned}\min\ & f(x) + g(y) \\ \text{s.t. } & Ax + b = y\end{aligned}$$

Define the new coordinate system

$$\begin{aligned}x_p &= Dx && \text{Primal pre-conditioning} \\ y_d &= Ey && \text{Dual pre-conditioning}\end{aligned}$$

The problem now becomes

$$\begin{aligned}\min\ & f(D^{-1}x_p) + g(E^{-1}y_d) \\ \text{s.t. } & A_d x_p + b_d = y_d\end{aligned}$$

where $A_d = EAD^{-1}$, $b_d = Eb$

Pre-conditioned AMA

AMA with pre-conditioning:

$$
\begin{aligned}
x^{k+1} &= \text{argmin}_x \;\; f(x) + \langle \lambda^k, A_d x \rangle \\
y^{k+1} &= E \, \text{prox}_{g,\rho}^{P}(E^{-1}(A_d x^{k+1} + b + \lambda^k/\rho) \\
\lambda^{k+1} &= \lambda^k + \rho(A_d x^{k+1} + b_d - y^{k+1})
\end{aligned}
$$

where

$$
\text{prox}_{g,\rho}^{P}(\bar{z}) = g(z) + \frac{1}{2}\|x - \bar{z}\|_P^2
$$

and

$$
\|x - \bar{z}\|_P^2 = (x - \bar{z})^T P (x - \bar{z})
$$

Requirements

- P must be positive definite and diagonal
- E and D selected to maintain sparsity structure of A

The question : **How to choose a good pre-conditioner?**

How to Choose a Preconditioning Matrix?

The main idea:

- Derive an expression for the convergence rate in terms of E and D
- Solve optimization problem to maximize rate of convergence
  - $\rightarrow$ Often requires solution of an SDP
  - $\rightarrow$ Many heuristic approaches available

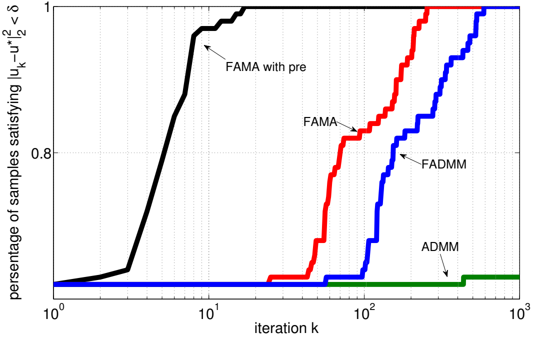
For details see

- *[P. Giselsson and S. Boyd, 2015]*
  `http://stanford.edu/~boyd/papers/pdf/metric_select_fdfbs.pdf`
- *[Stathopoulos et al, soon to be submitted]*

Impact of Pre-conditioning and Acceleration

- 7 states, 4 inputs, horizon 8
- Upper/lower bounds on states and inputs
- Ellipsoidal terminal set
- Algorithm : Fast AMA

Pre-conditioned via new invariance-maintaining approach for SOCPs[2]



2 *[Y. Pu, M. Zeilinger and C.N. Jones, under review]*

Splitting Methods for Control

Main idea:

- Separate complex optimization into a sequence of simpler operations
- Use dual to push the individual problems into consensus

Key properties

- Centralized optimization : Each iteration is extremely cheap
- Parallel optimization : Sub-problems can be solved in parallel

Major downside

- Number of iterations may be very high and sensitive to the current parameter / state (Lots of ongoing research to deal with this issue)

Toolbox for Deployment of Embedded Optimization

SPLIT

“CVX”-like syntax

```
% Optimization variables
x = splitvar(n, N);
u = splitvar(m, N-1);
x(:,1) = parameter(n,1);

% Objective and dynamics
obj = 0;
for i = 1:N-1
  x(:,i+1) == A*x(:,i) + B*u(:,i);
  obj = obj + x(:,i)'*Q*x(:,i) + ...
          u(:,i)'*R*u(:,i);
end
obj = obj + x(:,end)'*x(:,end);

% set up constraints
-5 <= x <= 5;
-1 <= u <= 1;
norm(x(2,:), 2) + x(:)'*x(:,) <= 4;

minimize(obj);
```

Standard form for splitting algorithms

Least-squares

Simple prox operators

$$\begin{aligned} \min_x \quad & x'Qx + f'x + \sum p_i(L_i x + l_i) \\ \text{s.t.} \quad & Ax = b \end{aligned}$$

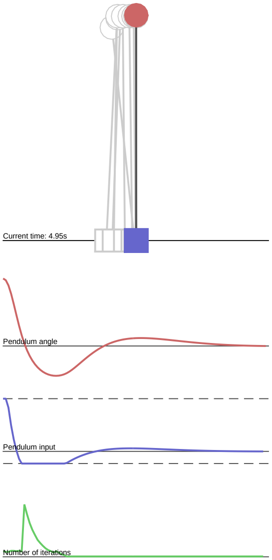
Solve in Matlab

Export to C

Wide range of algorithms implemented

Release date: Very soon ;)

Exercise : Revisit the inverted pendulum



Tasks:

1. Implement the AMA algorithm for the linearized pendulum model
2. Implement the accelerated version FAMA
3. Implement pre-conditioning