Interior-point Algorithms: Methods & Tools

Part I: Introduction to IPMs & FORCES Pro

Alexander Domahidi

Co-founder embotech GmbH

July 30, 2015

TEMPO Summer School on Numerical Optimal Control
University of Freiburg
Germany

Convex Optimization is the Workhorse

- Many problems can be boiled down to solving

$$\text{minimize } 0.5x^T H x + f^T x$$
$$\text{subject to } Ax = b$$
$$Gx \preceq_K h$$

Bounds, polytopes, second-order cones, 2-norm balls, exponential cones, ...

  - Linear constrained optimal control
  - Nonlinear programming: sequential quadratic programming
  - Mixed-integer problems: convex relaxations
  - Stochastic optimization: sampling
- In fact, this is what we *can* solve reliably
- In real-time control: **parametric convex problems**

Methods & Tools for Parametric Problems

EXPLICIT

Precompute controller

- Fixed complexity
- Simple to evaluate
- < µs sampling
- LPs/QPs only
- small problems only

Parametric Programming

ITERATIVE METHODS

First order methods

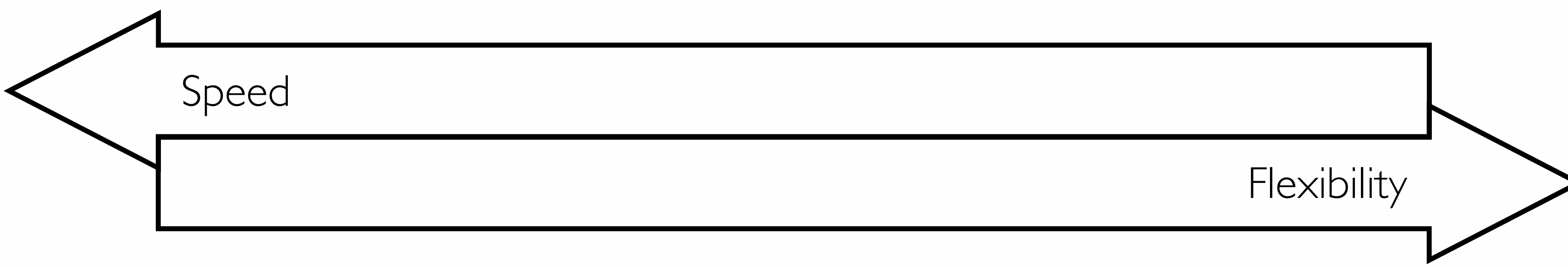
- Any size
- Very fast for problems with simple sets
- µs – ms sampling
- Certification

Gradient Methods
Operator Splitting (ADMM)

Second order methods

- Any size & problem
- Robust to conditioning
- ms sampling

Active Set Methods
Interior Point Methods



Methods & Tools for Parametric Problems

EXPLICIT	ITERATIVE METHODS	
Precompute controller	**First order methods**	**Second order methods**
• Fixed complexity • Simple to evaluate • < µs sampling • LPs/QPs only • small problems only	• Any size • Very fast for problems with simple sets • µs – ms sampling • Certification	• Any size & problem • Robust to conditioning • ms sampling

Software

Multi-Parametric Toolbox 3.0
[Kvasnica, Herceg, Jones, 2012]

- µAO-MPC
  [Zometa et al., 2013]
- FiOrdOs [Richter et al., 2012]
- QPgen [Giselsson, 2015]
- SPLIT Toolbox [Jones, 2015]

- QPC [Wills, 2008]
- qpOASES [Ferreau & Diehl, 2008]
- “Fast MPC” [Wang & Boyd 2008]
- CVXGEN [Mattingley & Boyd 2010]
- FORCES [Domahidi et al., 2012]
- ECOS [Domahidi et al., 2013]
- HPMPC [Frison et al, 2014]

embotech*

Spinoff ETH zürich

Outline of this Talk

EXPLICIT	ITERATIVE METHODS	
Precompute controller	First order methods	1 Interior-point methods
• Fixed complexity • Simple to evaluate • < µs sampling • LPs/QPs only • small problems only	• Any size • Very fast for problems with simple sets • µs – ms sampling • Certification	• Any size & problem • Robust to conditioning • ms sampling

Software

Multi-Parametric Toolbox 3.0 [Kvasnica, Herceg, Jones, 2012]	• µAO-MPC [Zometa et al., 2013] • FiOrdOs [Richter et al., 2012] • QPgen [Giselsson, 2015] • SPLIT Toolbox [Jones, 2015]	• QPC [Wills, 2008] • qpOASES [Ferreau & Diehl, 2008] • "Fast MPC" [Wang & Boyd 2008] • CVXGEN [Mattingley & Boyd 2010] 2 FORCES [Domahidi et al., 2012] 3 ECOS [Domahidi et al., 2013] • HPMPC [Frison et al, 2014]

Outline of this Talk

EXPLICIT	ITERATIVE METHODS	
Precompute controller	First order methods	1 Interior-point methods

The following slides are (C) ETH Zurich, Automatic Control Lab with the help of

Stefan Richter
Melanie Zeilinger
Colin Jones
Manfred Morari

2 FORCES [Domahidi et al., 2012]
3 ECOS [Domahidi et al., 2013]

Constrained Minimization Problem

Consider the following problem with inequality constraints

$$\min f(x)$$
$$\text{s.t. } g_i(x) \leq 0,\ i = 1, \ldots, m$$

Assumptions:

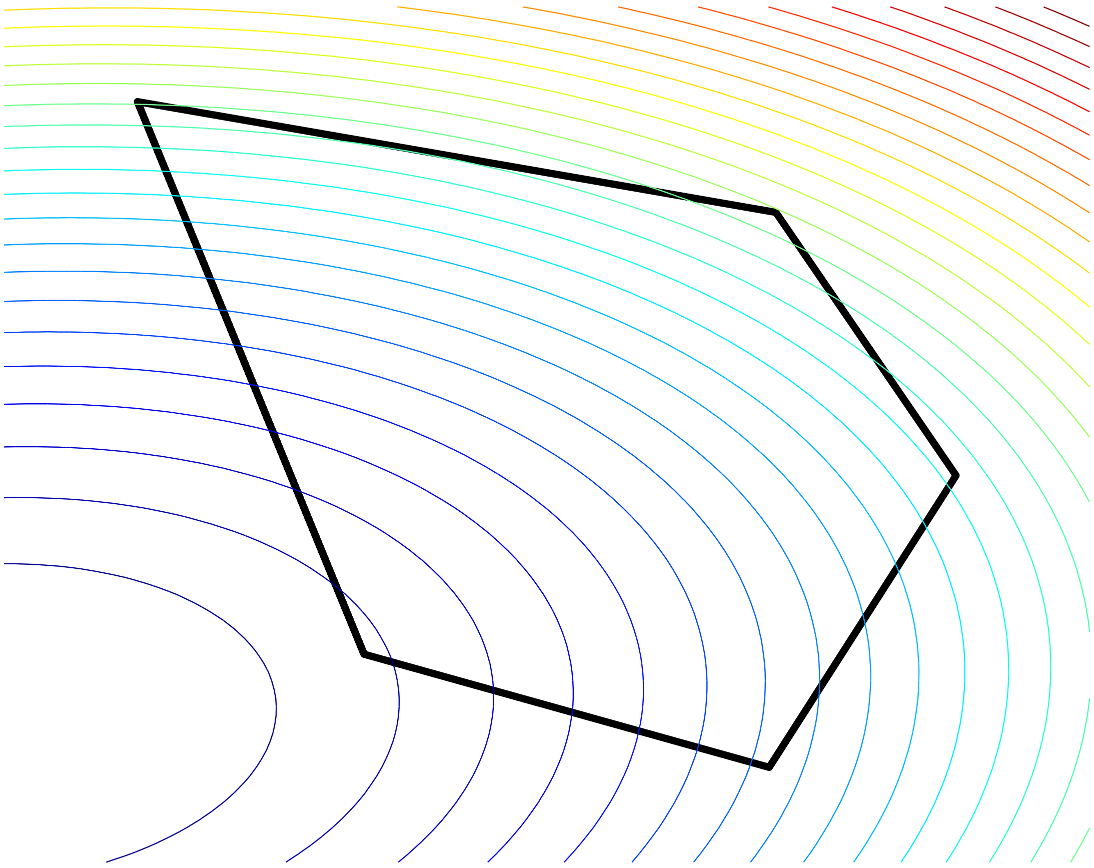
- f, g_i convex, twice continuously differentiable
- $f(x^*)$ is finite and attained
- strict feasibility: there exists a $\tilde{x}$ with

$$\tilde{x} \in \text{ dom } f, \quad g_i(\tilde{x}) < 0, i = 1, \ldots, m$$

- feasible set is closed and compact

Idea: There exist many methods for unconstrained minimization
$\Rightarrow$ Reformulate problem as an unconstrained problem

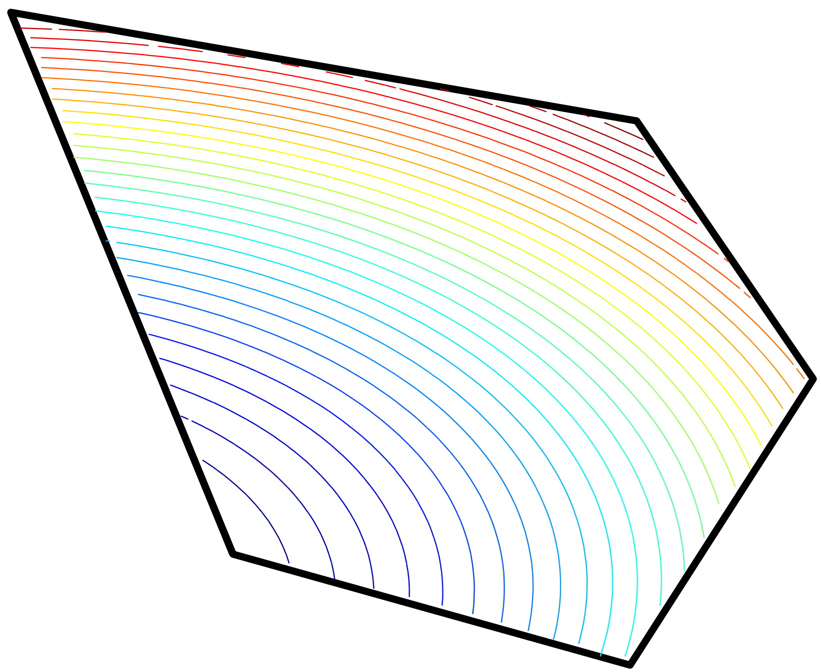
Graphical Illustration



Minimize a function over a set

Graphical Illustration

Define function as ∞ if constraints violated.



Minimize this function over $\mathbb{R}^n$

Barrier Method

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & g_i(x) \le 0,\ i = 1, \dots, m \end{array} \quad \Leftrightarrow \quad \min \quad f(x) + \kappa\phi(x)$$

Constraints have been moved to objective via *indicator function:*

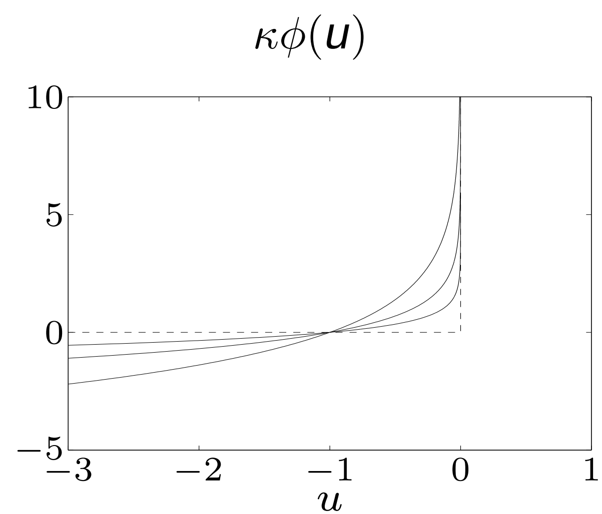
$$\phi(x) = \sum_{i=1}^{m} \mathcal{I}_{-}(g_i(x)), \quad \kappa = 1$$

where $\mathcal{I}_{-}(u) = 0$ if $u \le 0$ and $\mathcal{I}_{-} = \infty$ otherwise

- Augmented cost is not differentiable
  $\rightarrow$ approximation by *logarithmic barrier:*

$$\phi(x) = -\sum_{i=1}^{m} \log(-g_i(x))$$

- For $\kappa > 0$ smooth approximation of indicator function
- Approximation improves as $\kappa \rightarrow 0$



Logarithmic Barrier Function

$$\phi(x) = -\sum_{i=1}^{m} log(-g_i(x)), \qquad \text{dom } \phi = \{x \mid g_1(x) < 0, \dots, g_m(x) < 0\}$$

- Convex, smooth on its domain
- $\phi(x) \to \infty$ as x approaches boundary of domain
- $\arg\min_x \phi(x)$ is called **analytic center** of the set defined by inequalities $g_1 < 0, \dots, g_m < 0$
- Twice continuously differentiable with derivatives

$$\nabla \phi(x) = \sum_{i=1}^{m} \frac{1}{-g_i(x)} \nabla g_i(x)$$

$$\nabla^2 \phi(x) = \sum_{i=1}^{m} \frac{1}{g_i(x)^2} \nabla g_i(x) \nabla g_i(x)^T + \frac{1}{-g_i(x)} \nabla^2 g_i(x)$$

Central Path

- Define $x^*(\kappa)$ as the solution of

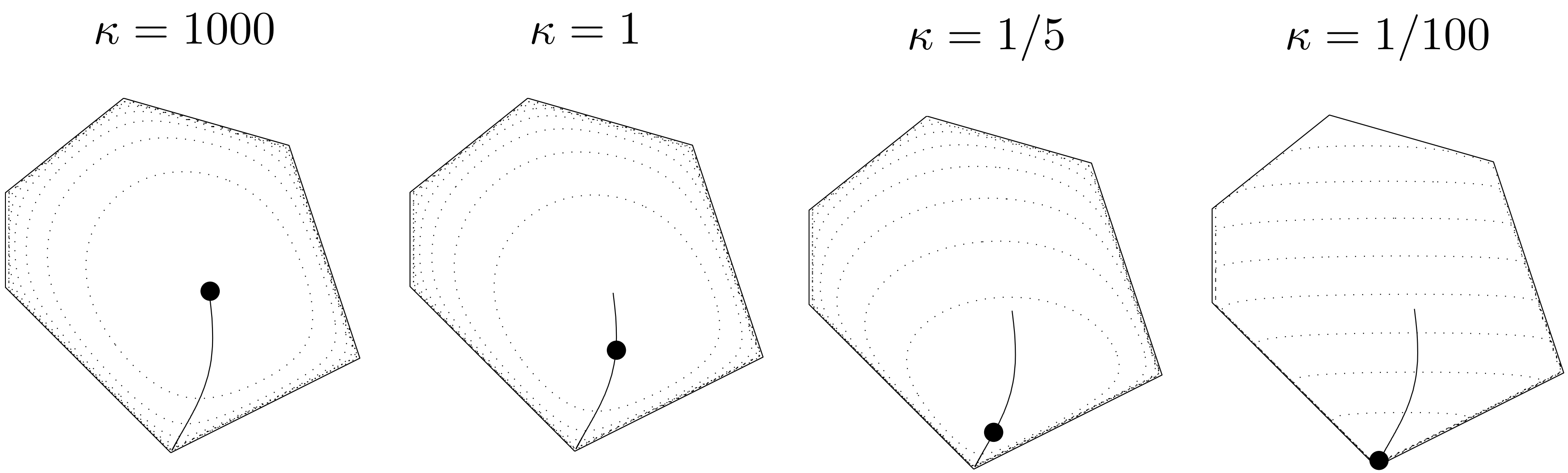
$$\min_x f(x) + \kappa\phi(x)$$

  (assume minimizer exists and is unique for each $\kappa > 0$)
- Barrier parameter κ determines relative weight between objective and barrier
- Barrier 'traps' $x(\kappa)$ in strictly feasible set
- *Central path* is defined as $\{x^*(\kappa) \mid \kappa > 0\}$
- For given κ can compute $x^*(\kappa)$ by solving smooth unconstrained minimization problem
- Intuitively $x^*(\kappa)$ converges to optimal solution as $\kappa \to 0$ (easy to prove under mild conditions)

Example: Central Path for an LP

$$\begin{array}{ll} \text{min} & c^T x \\ \text{s.t.} & a_i^T x \leq b_i, i = 1, \ldots, 6 \end{array}$$

$x \in \mathbb{R}^2$, c points upward



Path-following Methods

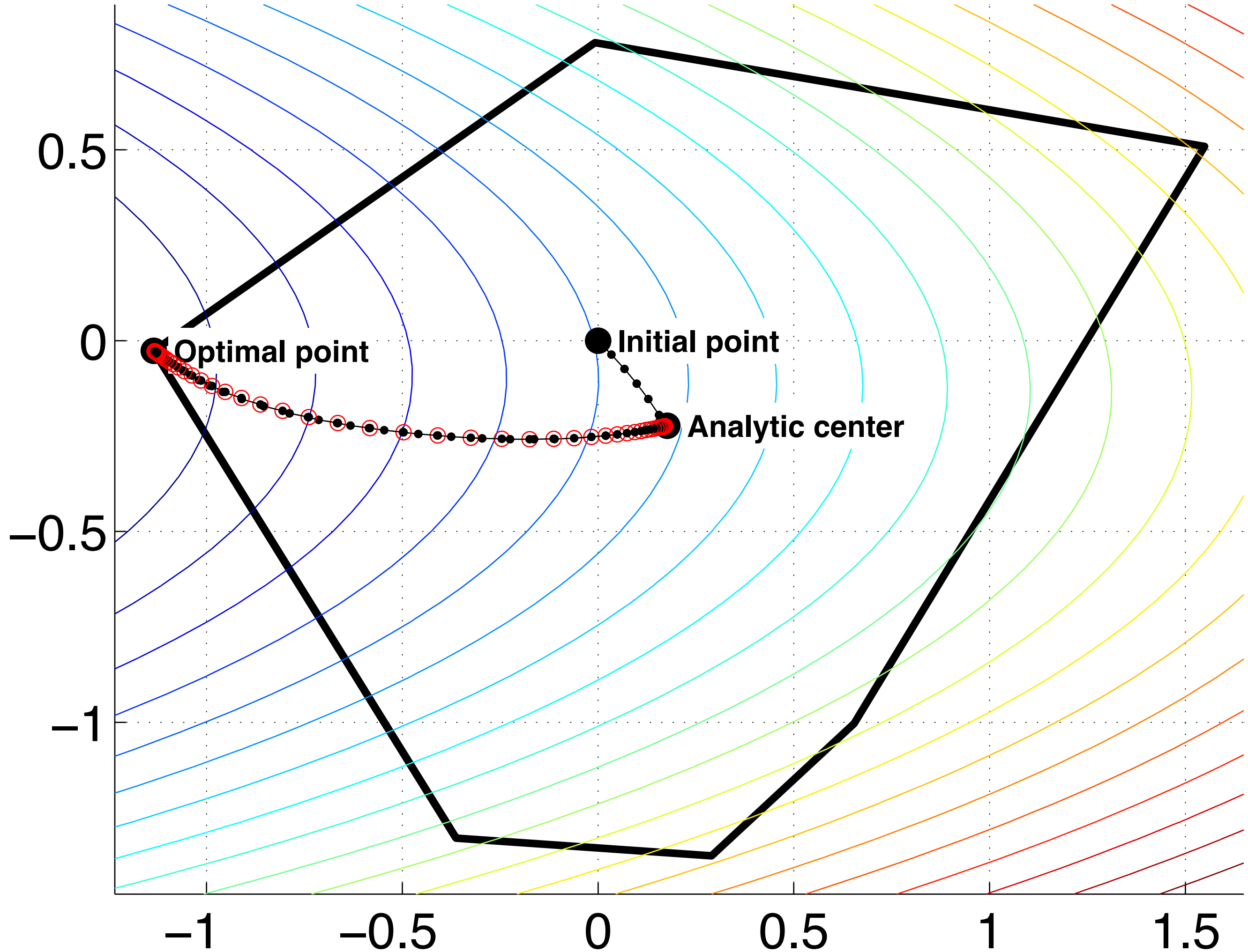
Idea: Follow central path to the optimal solution

Solve sequence of smooth unconstrained problems:

$$x^*(\kappa) = \arg\min_x f(x) + \kappa\phi(x)$$

- Assume current solution is on the central path $x_i = x^*(\kappa_i)$
- Obtain κ_{i+1} by decreasing κ_i by some amount: $\kappa_{i+1} = \kappa_i/\mu, \mu > 1$
- Solve for $x^*(\kappa_{i+1})$ starting from $x^*(\kappa_i)$ (unconstrained optimization). Called "centering step" because it computes a point on (or close to) the central path
- Method converges to the optimal solution, i.e., $x_i \rightarrow x^*$ for $\kappa \rightarrow 0$

Example: PF-IPM for Quadratic Program



embotech*

Spinoff ETH zürich

Centering Step Using Newton Method

Idea: Exploit fast local convergence of Newton's method starting at point close to optimum

- *Newton direction*: Δx_{nt} minimizes second order approximation

$$
\begin{aligned}
f(x+v) + \kappa\phi(x+v) \approx\, & f(x) + \kappa\phi(x) + \nabla f(x)^T v + \kappa \nabla\phi(x)^T v \\
& + \frac{1}{2} v^T \nabla^2 f(x) v + \frac{1}{2}\kappa v^T \nabla^2 \phi(x) v
\end{aligned}
$$

- Newton direction for barrier method is given by solution of

$$
(\nabla^2 f(x) + \kappa \nabla^2 \phi(x))\Delta x_{nt} = -\nabla f(x) - \kappa \nabla \phi(x)
$$

Line search consists of two steps:

1. For retaining feasibility, find $h_{max} = \arg\max_{h>0}\{h \mid g_1(x+h\Delta x) < 0, \ldots, g_m(z+h\Delta x) < 0\}$
2. Find $h^* = \arg\min_{h\in[0,\, h_{max}]}\{f(x+h\Delta x) + \kappa\phi(x+h\Delta x)\}$

both either solved exactly or through backtracking

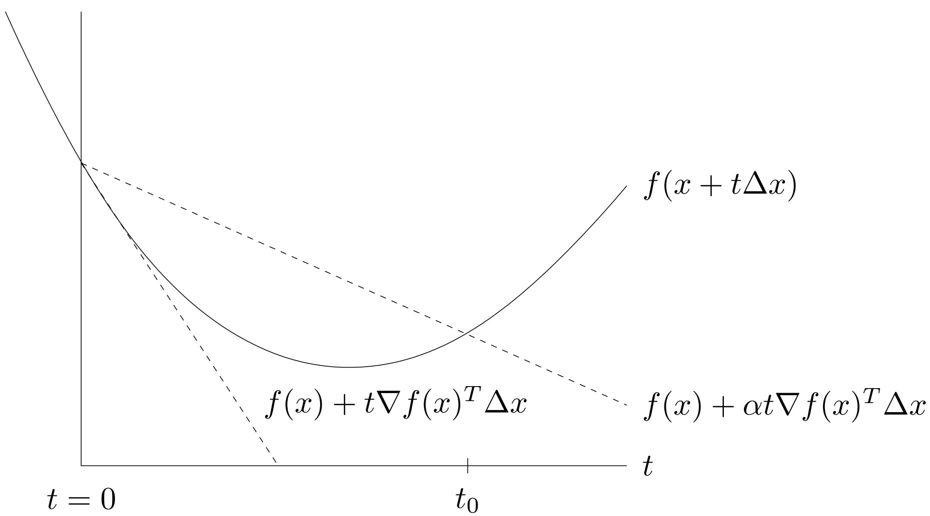
Backtracking Line Search

given a descent direction Δx for f at $x \in \mathbf{dom}\, f$, $\alpha \in (0, 0.5)$, $\beta \in (0, 1)$.
$t := 1$.
while $f(x + t\Delta x) > f(x) + \alpha t \nabla f(x)^T \Delta x, \quad t := \beta t.$

Backtracking line search terminates when *Armijo's condition* is satisfied:

$$f(x_i + h_i \Delta x_{nt}) \leq f(x_i) + \alpha h_i \nabla f(x_i)^T \Delta x_{nt}$$

This is required for convergence of the optimization algorithm.
From *[Boyd & Vandenberghe, Convex Optimization, 2004]* with $t \equiv h, x \equiv x_i$:



Centering Step with Equality Constraints

Centering Step: Compute $x^*(\kappa)$ by solving

$$\begin{array}{ll} \min & f(x) + \kappa\phi(x) \\ \text{s.t.} & Cx = d \end{array}$$

- Newton step Δx_{nt} for minimization with equality constraints is given by solution of

$$\begin{bmatrix} \nabla^2 f(x) + \kappa \nabla^2 \phi(x) & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \Delta x_{nt} \\ \nu \end{bmatrix} = - \begin{bmatrix} \nabla f(x) + \kappa \nabla \phi(x) \\ 0 \end{bmatrix}$$

- Same interpretation as Newton step for unconstrained problem: $x + \Delta x_{nt}$ minimizes second order approximation

$$\begin{array}{ll} \min & \nabla f(x)^T v + \kappa \nabla \phi(x)^T v + \frac{1}{2} v^T \nabla^2 f(x) v + \frac{1}{2} \kappa v^T \nabla^2 \phi(x) v \\ \text{s.t.} & Cv = 0 \end{array}$$

Barrier IPM with Equality Constraints

$$\min_x \{f(x) \mid g(x) \leq 0,\ Cx = d\}$$

Require: strictly feasible x_0 w.r.t. $g(x)$, κ_0, $\mu > 1$, tol. $\epsilon > 0$

repeat

1. ***Centering step:*** **Compute $x^*(\kappa_i)$ by minimizing $f(x) + \kappa_i \phi(x)$ subject to $Cx = d$ starting from x_{i-1}**
2. *Update* $x_i = x^*(\kappa_i)$
3. *Stopping criterion:* Stop if $m\kappa_i < \epsilon$
4. *Decrease barrier parameter:* $\kappa_{i+1} = \kappa_i / \mu$

- Several heuristics for choice of κ_0 and other parameters
- Terminates with $f(x_i) - f(x^*) \leq \epsilon$
- Steps 1-4 represent one outer iteration
- Step 1: Solve equality constrained minimization problem (via Newton steps)

Example: Newton Step for QPs

$$\min_x \{1/2x^T Hx \mid Gx \leq d\}$$

- Barrier method:

$$\min_x f(x) + \kappa\phi(x) = \min_x \frac{1}{2}x^T Hx - \kappa \sum_{i=1}^{m} \log(d_i - g_i x)$$

where $g_1, \ldots, g_m$ are the rows of G.

- The gradient and Hessian of the barrier function are:

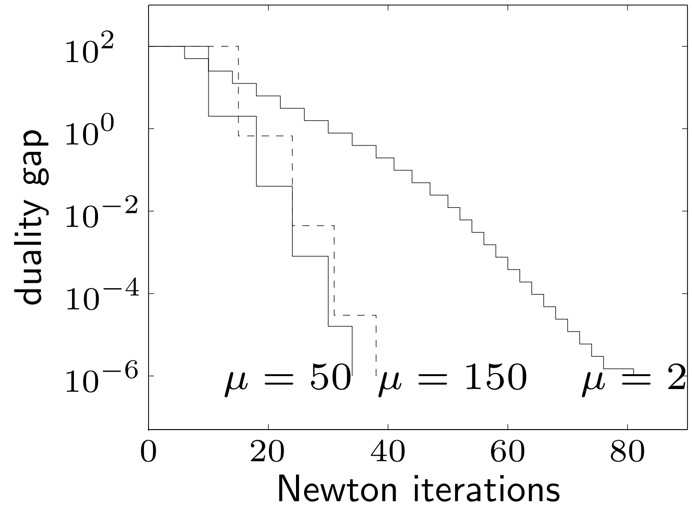
$$\nabla\phi(x) = \sum_{i=1}^{m} \frac{1}{d_i - g_i x} g_i^T, \nabla^2\phi(x) = \sum_{i=1}^{m} \frac{1}{(d_i - g_i x)^2} g_i^T g_i$$

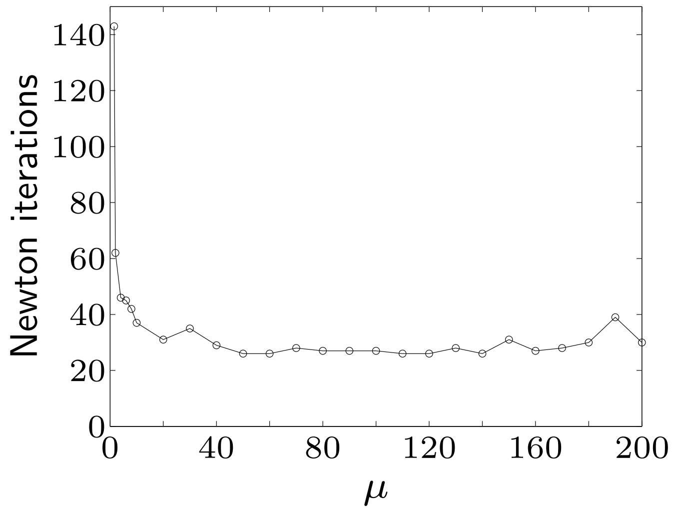
- Newton step: $(\nabla^2 f(x) + \kappa\nabla^2\phi(x))\Delta x_{nt} = -\nabla f(x) - \kappa\nabla\phi(x)$

$$(H + \kappa \sum_{i=1}^{m} \frac{1}{(d_i - g_i x)^2} g_i^T g_i)\Delta x_{nt} = -Hx - \sum_{i=1}^{m} \frac{1}{d_i - g_i x} g_i^T$$

Remarks on Barrier IPMs

- Choice of μ involves trade-off: large $\mu \Rightarrow$ few outer iterations, but more inner iterations to compute x_{i+1} from x_i (typical values $\mu = 10 - 20$)
- Good convergence properties for a wide range of parameters μ
  Example: LP with 100 inequalities, 50 variables





- Barrier method requires strictly feasible initial point
  $\rightarrow$ Phase I method, e.g., $\min_{x,s}\{s \mid g(x) \leq s \cdot \mathbf{1}\}$

Barrier IPMs and KKT Conditions

KKT conditions of the barrier problem:

$$
\begin{aligned}
Cx^* &= d \\
g_i(x^*) &\leq 0, i = 1, \dots, m \\
\nabla f(x^*) + \kappa \sum_{i=1}^{m} \frac{1}{-g_i(x^*)} \nabla g_i(x^*) + C^T \nu^* &= 0
\end{aligned}
$$

Defining

$$\lambda_i^* = \kappa \frac{1}{-g_i(x^*)} \geq 0$$

results in the standard KKT conditions where complementarity slackness is replaced by the relaxed condition

$$\lambda_i^* g_i(x^*) = -\kappa$$

Primal-Dual Interior-Point Methods

Using the result from above, the **relaxed** KKT system can be written as

$$\begin{aligned} Cx^* &= d \\ g_i(x^*) + s_i^* &= 0 \; i = 1, \dots, m \\ \nabla f(x^*) + \textstyle\sum_{i=1}^m \lambda_i^* \nabla g_i(x^*) + C^T \nu^* &= 0 \\ \lambda_i^* g_i(x^*) &= -\kappa \\ \lambda_i^*, s_i^* &\geq 0, \; i = 1, \dots, m \end{aligned} \tag{**}$$

Idea: leave dual multipliers λ_i^* as variables (before, they were implicitly defined by primal log barrier)[3]:

- Solve the primal and dual problem simultaneously via (**)
- Primal-dual central path $\triangleq \{(x, s, \lambda, \nu) \mid (**) \text{ holds}\}$
- Follow central path to solution by reducing κ to zero
- Solve (**) by Newton method (with additional "safeguards" & line search)

Primal-Dual Search Direction

At each iteration, linearize (**) and solve

$$\begin{bmatrix} H(x,\lambda) & C^T & G(x)^T & 0 \\ C & 0 & 0 & 0 \\ G(x) & 0 & 0 & I \\ 0 & 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \lambda \\ \Delta s \end{bmatrix} = - \begin{bmatrix} \nabla f(x) + C^T y + G(x)^T \lambda \\ Cx - d \\ g(x) + s \\ S\lambda - \nu \end{bmatrix}$$

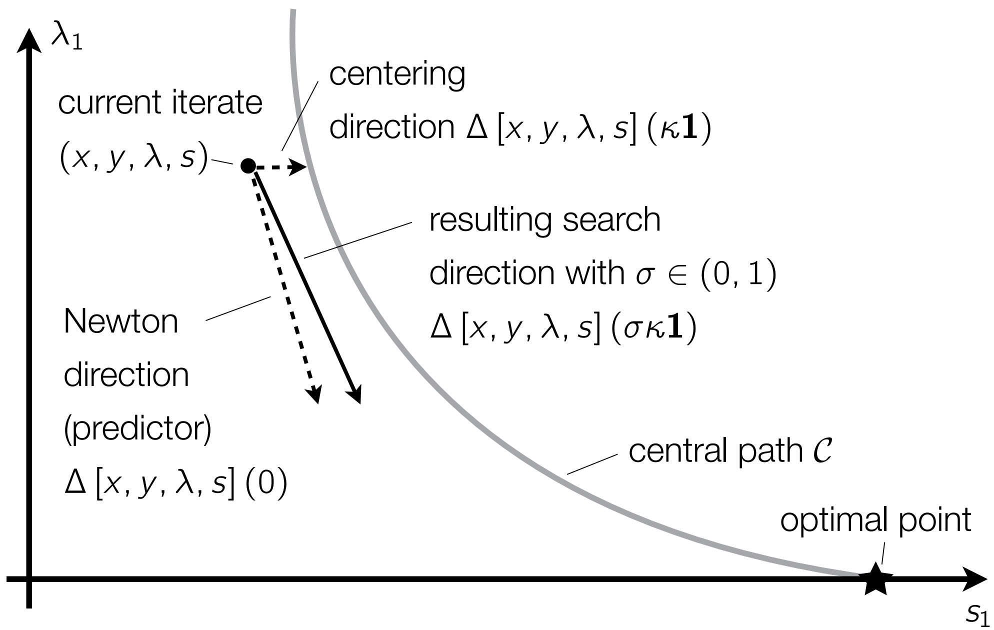
where $S \triangleq \operatorname{diag}(s_1, \ldots, s_m)$ and $\Lambda \triangleq \operatorname{diag}(\lambda_1, \ldots, \lambda_m)$, the (1,1) block in the coefficient matrix is

$$H(x,\lambda) \triangleq \nabla^2 f(x) + \sum_{i=1}^{m} \lambda_i \nabla^2 g_i(x)$$

and the vector $\nu \in \mathbb{R}^m$ allows for a modification of the right-hand side. Call resulting direction $\Delta\left[x, y, \lambda, s\right](\nu)$.

Primal-Dual Search Directions

Can generate different directions $\Delta\,[x, y, \lambda, s]\,(\nu)$ depending on ν:



- $\nu = 0$: standard Newton method for solving nonlinear equations
- $\nu = \kappa\mathbf{1}$: centering direction, next iterate is on central path

$\Rightarrow$ Using linear combination via **centering parameter** $\sigma \in (0, 1)$ ensures fast convergence in theory & practice by tracking central path to solution

Summary Interior-Point Methods

Barrier method
(also called *Sequential Unconstrained Minimization Technique, SUMT)*

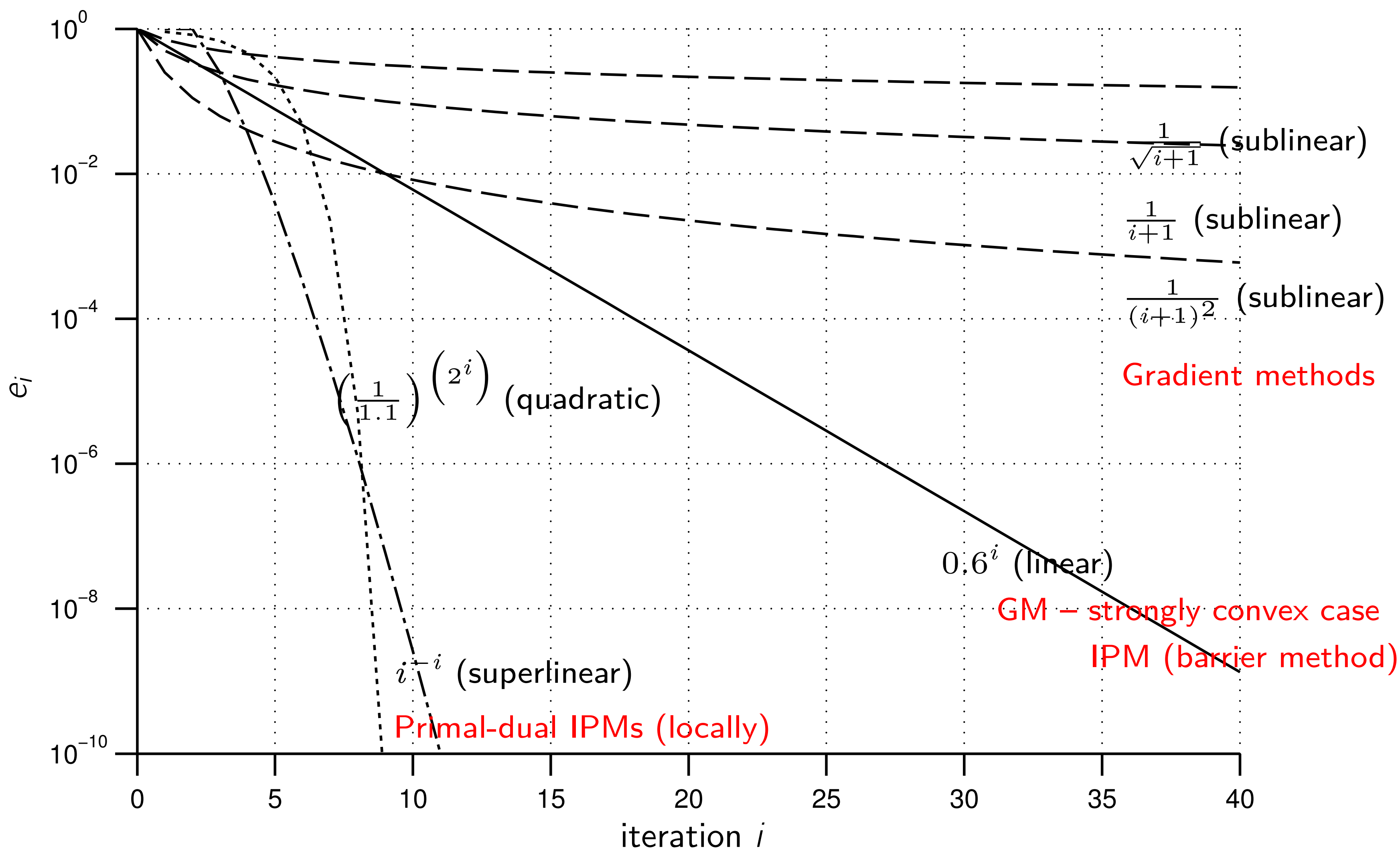
- Intuition: Follow central path to the optimal solution
- Log barrier function ensures satisfaction of inequality constraints
- Centering problems can be solved efficiently using Newton's method
- Requires strictly feasible initial point

'Modern' methods: Primal-dual methods

- Often more efficient than barrier method (superlinear convergence)
- Cost per iteration roughly the same as barrier method
- Allow for **infeasible start** (w.r.t. both equality and inequality constraints)
- Can provide **certificates of infeasibility** (using self-dual embedding)
- Most efficient in practice: Mehrotra's predictor-corrector method : <30 iterations in practice

Interior-point methods are very efficient for range of optimization problems, e.g. LPs, QPs, second-order cone and semidefinite programs.

Convergence Rates



Outline of this Talk

EXPLICIT	ITERATIVE METHODS	
Precompute controller	First order methods	1 Interior-point methods

2 FORCES [Domahidi et al., 2012]

3 ECOS [Domahidi et al., 2013]

Interior Point Methods in a Nutshell

Convex problem

$$\begin{aligned} \min_{y} \quad & f(y) \\ \text{s.t.} \quad & g(y) \leq 0 \\ & Cy + c = 0 \end{aligned}$$

Linearized KKT system

$$\begin{bmatrix} \mathcal{H}(y,\lambda) & C^T & J^T(y) & \\ C & & & \\ J(y) & & & I \\ & & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta \nu \\ \Delta \lambda \\ \Delta s \end{bmatrix} = - \begin{bmatrix} r_C \\ r_E \\ r_I \\ r_s \end{bmatrix}$$

KKT conditions

$$\begin{aligned} \nabla f(y) + \nabla g(y)^T \lambda + C^T \nu &= 0 \\ Cy + c &= 0 \\ g(y) + s &= 0 \\ \lambda^T s &= 0 \\ \lambda, s &\geq 0 \end{aligned}$$

Interior Point Method

1. Initialize $z_0 = (y_0, s_0, \lambda_0, \nu_0)$
2. Solve linearized KKT system
   → search direction Δz_i
3. Determine step size α
4. $z_{i+1} = z_i + \alpha \Delta z_i$

Interior Point Methods in a Nutshell

Convex problem

$$\begin{aligned} \min_{y} \quad & f(y) \\ \text{s.t.} \quad & g(y) \leq 0 \\ & Cy + c = 0 \end{aligned}$$

KKT conditions

$$\begin{aligned} \nabla f(y) + \nabla g(y)^T \lambda + C^T \nu &= 0 \\ Cy + c &= 0 \\ g(y) + s &= 0 \\ \lambda^T s &= 0 \\ \lambda, s &\geq 0 \end{aligned}$$

Interior Point Method

1. Initialize $z_0 = (y_0, s_0, \lambda_0, \nu_0)$
2. Solve linearized KKT system
   → search direction Δz_i
3. Determine step size α
4. $z_{i+1} = z_i + \alpha \Delta z_i$

Linearized KKT system

$$\begin{bmatrix} \mathcal{H}(y,\lambda) & C^T & J^T(y) & \\ C & & & \\ J(y) & & & I \\ & & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta \nu \\ \Delta \lambda \\ \Delta s \end{bmatrix} = - \begin{bmatrix} r_C \\ r_E \\ r_I \\ r_s \end{bmatrix}$$

Solving the linearized KKT system is ~95% of the computation

Solving Ax=b, A square

- Iterative solvers: generate sequence of iterates s.t. $Ax \approx b$
  - MINRES, conjugate gradient, Krylov subspace methods, Lanczos...
  - Useful for parallelizing
  - Matrix-vector products only, max. O(n^2) per iteration
  - Number of iterations required depends strongly on cond(A)
  - Literature in context of IPMs: inexact Newton methods

- Direct solvers: factor A & forward/backward solve
  - General A: LU factorization 4/3 n^3 flops
    - Gauss elimination with partial pivoting
  - A symmetric indefinite: LDL factorization 4/3 n^3 flops (1/2 the memory of LU)
    - Bunch-Parlett pivoting (1x1 and 2x2 blocks in D)
  - A symmetric positive definite: Cholesky 2/3 n^3 flops
    - stable without pivoting
    - needs square roots

Direct Methods Solving the KKT System

1. Unreduced form

$$\begin{bmatrix} \mathcal{H}(y,\lambda) & C^T & J^T(y) & \\ C & & & \\ J(y) & & & I \\ & & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta \nu \\ \Delta \lambda \\ \Delta s \end{bmatrix} = - \begin{bmatrix} r_C \\ r_E \\ r_I \\ r_s \end{bmatrix}$$

- Not symmetric and indefinite
- LU factorization

2. Augmented form

$$\begin{bmatrix} \Phi & C^T \\ C & \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta \nu \end{bmatrix} = - \begin{bmatrix} r_d \\ r_E \end{bmatrix}$$

$$\Phi := \mathcal{H}(y,\lambda) + J^T(y) S^{-1} \Lambda J(y)$$

- Eliminate Δλ and Δs (S and L are PD and diagonal)
- Symmetric, but indefinite
- LDL factorization
  - Requires pivoting
  - CVXGEN: regularization instead

3. Normal form

$$Y \Delta \nu = \beta$$

$$Y := C \Phi^{-1} C^T$$

- Eliminate Δy (Schur complement)
- Symmetric, positive definite
- Cholesky factorization
  - Stable without pivoting

Direct Methods Solving the KKT System

1. Unreduced form

$$\begin{bmatrix} \mathcal{H}(y,\lambda) & C^T & J^T(y) & \\ C & & & \\ J(y) & & & I \\ & & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta \nu \\ \Delta \lambda \\ \Delta s \end{bmatrix} = - \begin{bmatrix} r_C \\ r_E \\ r_I \\ r_s \end{bmatrix}$$

- Not symmetric and indefinite
- LU factorization

2. Augmented form

$$\begin{bmatrix} \Phi & C^T \\ C & \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta \nu \end{bmatrix} = - \begin{bmatrix} r_d \\ r_E \end{bmatrix}$$

$$\Phi := \mathcal{H}(y,\lambda) + J^T(y) S^{-1} \Lambda J(y)$$

- Eliminate Δλ and Δs (S and L are PD and diagonal)
- Symmetric, but indefinite
- LDL factorization
  - Requires pivoting
  - CVXGEN: regularization instead

3. Normal form

$$Y \Delta \nu = \beta$$

$$Y := C \Phi^{-1} C^T$$

- Eliminate Δy (Schur complement)
- Symmetric, positive definite
- Cholesky factorization
  - Stable without pivoting

Direct Methods Solving the KKT System

1. Unreduced form

$$\begin{bmatrix} \mathcal{H}(y,\lambda) & C^T & J^T(y) & \\ C & & & \\ J(y) & & & I \\ & & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta \nu \\ \Delta \lambda \\ \Delta s \end{bmatrix} = - \begin{bmatrix} r_C \\ r_E \\ r_I \\ r_s \end{bmatrix}$$

- Not symmetric and indefinite
- LU factorization

2. Augmented form

$$\begin{bmatrix} \Phi + \delta I & C^T \\ C & -\delta I \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta \nu \end{bmatrix} = - \begin{bmatrix} r_d \\ r_E \end{bmatrix}$$

$$\Phi := \mathcal{H}(y,\lambda) + J^T(y) S^{-1} \Lambda J(y)$$

- Eliminate Δλ and Δs
  (S and L are PD and diagonal)
- Symmetric, but indefinite
- LDL factorization
  - Requires pivoting
  - CVXGEN: regularization instead

3. Normal form

$$Y \Delta \nu = \beta$$

$$Y := C \Phi^{-1} C^T$$

- Eliminate Δy
  (Schur complement)
- Symmetric, positive definite
- Cholesky factorization
  - Stable without pivoting

Direct Methods Solving the KKT System

1. Unreduced form

$$\begin{bmatrix} \mathcal{H}(y,\lambda) & C^T & J^T(y) & \\ C & & & \\ J(y) & & & I \\ & & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta \nu \\ \Delta \lambda \\ \Delta s \end{bmatrix} = - \begin{bmatrix} r_C \\ r_E \\ r_I \\ r_s \end{bmatrix}$$

- Not symmetric and indefinite
- LU factorization

2. Augmented form

$$\begin{bmatrix} \Phi + \delta I & C^T \\ C & -\delta I \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta \nu \end{bmatrix} = - \begin{bmatrix} r_d \\ r_E \end{bmatrix}$$

$$\Phi := \mathcal{H}(y,\lambda) + J^T(y) S^{-1} \Lambda J(y)$$

- Eliminate Δλ and Δs (S and L are PD and diagonal)
- Symmetric, but indefinite
- LDL factorization
  - Requires pivoting
  - CVXGEN: regularization instead

3. Normal form

$$Y \Delta \nu = \beta$$

$$Y := C \Phi^{-1} C^T$$

- Eliminate Δy (Schur complement)
- Symmetric, positive definite
- Cholesky factorization
  - Stable without pivoting

Convex Multistage Problems

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{i=1}^{N} \frac{1}{2} z_i^T H_i z_i + f_i^T z_i \\ \text{subject to} \quad & \underline{z}_i \leq z_i \leq \bar{z}_i \\ & A_i z_i \leq b_i \\ & z_i^T Q_{i,j} z_i + l_{i,j}^T z_i \leq r_{i,j} \\ & C_i z_i + D_{i+1} z_{i+1} = c_i \end{aligned}$$

separable objective

upper/lower bounds

affine inequalities

quadratic inequalities

affine equalities, each coupling only two consecutive variables

where $H_i, Q_i \succeq 0$ and A_i has full row rank

Convex Multistage Problems

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{N} \tfrac{1}{2} z_i^T H_i z_i + f_i^T z_i \\
\text{subject to} \quad & \underline{z}_i \leq z_i \leq \bar{z}_i \\
& A_i z_i \leq b_i \\
& z_i^T Q_{i,j} z_i + l_{i,j}^T z_i \leq r_{i,j} \\
& C_i z_i + D_{i+1} z_{i+1} = c_i
\end{aligned}
$$

separable objective

upper/lower bounds

affine inequalities

quadratic inequalities

affine equalities, each coupling only two consecutive variables

where $H_i, Q_i \succeq 0$ and A_i has full row rank

Captures OCP, MPC, MHE, spline optimization, portfolio optimization, etc.

Multistage Property Induces Structure

Multistage problem

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{i=1}^{N} \frac{1}{2} z_i^T H_i z_i + f_i^T z_i \\ \text{subject to} \quad & \underline{z}_i \le z_i \le \bar{z}_i \\ & A_i z_i \le b_i \\ & z_i^T Q_{i,j} z_i + l_{i,j}^T z_i \le r_{i,j} \\ & C_i z_i + D_{i+1} z_{i+1} = c_i \end{aligned}$$

Linearized KKT system

$$\begin{bmatrix} \mathcal{H}(y,\lambda) & C^T & J^T(y) & \\ C & & & \\ J(y) & & & I \\ & & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta \nu \\ \Delta \lambda \\ \Delta s \end{bmatrix} = - \begin{bmatrix} r_C \\ r_E \\ r_I \\ r_S \end{bmatrix}$$

1. $\mathcal{H}(y,\lambda)$ and $J(y)$ are block diagonal
2. $C := \begin{bmatrix} C_0 & D_1 & 0 & \cdots & 0 \\ 0 & C_1 & D_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & C_{N-1} & D_N \end{bmatrix}$
3. Dimensions known at compile time

Multistage Property Induces Structure

Multistage problem

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} \tfrac{1}{2} z_i^T H_i z_i + f_i^T z_i \\ \text{subject to} \quad & \underline{z}_i \le z_i \le \bar{z}_i \\ & A_i z_i \le b_i \\ & z_i^T Q_{i,j} z_i + l_{i,j}^T z_i \le r_{i,j} \\ & C_i z_i + D_{i+1} z_{i+1} = c_i \end{aligned}$$

Linearized KKT system

$$\begin{bmatrix} \mathcal{H}(y,\lambda) & C^T & J^T(y) & \\ C & & & \\ J(y) & & & I \\ & & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta \nu \\ \Delta \lambda \\ \Delta s \end{bmatrix} = - \begin{bmatrix} r_C \\ r_E \\ r_I \\ r_s \end{bmatrix}$$

1. $\mathcal{H}(y,\lambda)$ and $J(y)$ are block diagonal
2. $C := \begin{bmatrix} C_0 & D_1 & 0 & \cdots & 0 \\ 0 & C_1 & D_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & C_{N-1} & D_N \end{bmatrix}$
3. Dimensions known at compile time

Goal: Exploit problem structure to speed up solution

Direct Methods Solving the KKT System

1. Unreduced form

$$\begin{bmatrix} \mathcal{H}(y,\lambda) & C^T & J^T(y) & \\ C & & & \\ J(y) & & & I \\ & & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta \nu \\ \Delta \lambda \\ \Delta s \end{bmatrix} = - \begin{bmatrix} r_C \\ r_E \\ r_I \\ r_s \end{bmatrix}$$

- Not symmetric and indefinite
- LU factorization

2. Augmented form

$$\begin{bmatrix} \Phi & C^T \\ C & \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta \nu \end{bmatrix} = - \begin{bmatrix} r_d \\ r_E \end{bmatrix}$$

$$\Phi := \mathcal{H}(y,\lambda) + J^T(y) S^{-1} \Lambda J(y)$$

- Eliminate Δλ and Δs (S and L are PD and diagonal)
- Symmetric, but indefinite
- LDL factorization
  - Requires pivoting
  - CVXGEN: regularization instead

3. Normal form

$$Y \Delta \nu = \beta$$

$$Y := C \Phi^{-1} C^T$$

- Eliminate Δy (Schur complement)
- Symmetric, positive definite
- Cholesky factorization
  - Stable without pivoting

Direct Methods Solving the KKT System

1. Unreduced form

$$\begin{bmatrix} \mathcal{H}(y,\lambda) & C^T & J^T(y) & \\ C & & & \\ J(y) & & & I \\ & & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta \nu \\ \Delta \lambda \\ \Delta s \end{bmatrix} = - \begin{bmatrix} r_C \\ r_E \\ r_I \\ r_s \end{bmatrix}$$

- Not symmetric and indefinite
- LU factorization

2. Augmented form

$$\begin{bmatrix} \Phi + \delta I & C^T \\ C & -\delta I \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta \nu \end{bmatrix} = - \begin{bmatrix} r_d \\ r_E \end{bmatrix}$$

$$\Phi := \mathcal{H}(y,\lambda) + J^T(y) S^{-1} \Lambda J(y)$$

- Eliminate Δλ and Δs (S and L are PD and diagonal)
- Symmetric, but indefinite
- LDL factorization
  - Requires pivoting
  - CVXGEN: regularization instead

3. Normal form

$$Y \Delta \nu = \beta$$

$$Y := C \Phi^{-1} C^T$$

- Eliminate Δy (Schur complement)
- Symmetric, positive definite
- Cholesky factorization
  - Stable without pivoting

Search Direction Computation

1

Set up KKT System

$$Y := C\Phi^{-1}C^T$$

$9Nr^3$ FLOPS

2

Factor coefficient matrix

$2Nr^3$ FLOPS

3

Compute search direction (forward & backward subst.)

$\mathcal{O}(Nr^2)$ FLOPS

N : number of stages, r : stage/block size

Cholesky Factorization of Y

- Want to compute L_Y such that $Y = L_Y L_Y^T$

$$Y = \begin{bmatrix} Y_{11} & Y_{12} & 0 & 0 & \cdots \\ Y_{12}^T & Y_{22} & Y_{23} & 0 & \cdots \\ 0 & Y_{23}^T & Y_{33} & Y_{34} & \cdots \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{bmatrix} \qquad L_Y = \begin{bmatrix} L_{11} & 0 & 0 & \cdots & 0 \\ L_{21} & L_{22} & 0 & \cdots & 0 \\ 0 & L_{32} & L_{33} & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & L_{N,N-1} & L_{N,N} \end{bmatrix}$$

- Sequential block-wise factorization:

$$\begin{array}{lrl} L_{11} = \mathrm{chol}(Y_{11}) & 2/3r^3 & \text{(Cholesky factorization)} \\ \text{for } i = 2 : N \text{ do} & & \\ \quad L_{i+1,i}^T = Y_{i,i+1}/L_{i,i} & r^3 & \text{(Matrix backward subst.)} \\ \quad U_i = L_{i,i-1} L_{i,i-1}^T & r^3 & \text{(Matrix matrix mult.)} \\ \quad L_{ii} = \mathrm{chol}(Y_{ii} - U_i) & 2/3r^3 & \text{(Cholesky factorization)} \end{array}$$

Search Direction Computation

1

Set up KKT System

$Y := C\Phi^{-1}C^T$

$9Nr^3$ FLOPS

2

Factor coefficient matrix

$2Nr^3$ FLOPS

3

Compute search direction (forward & backward subst.)

$\mathcal{O}(Nr^2)$ FLOPS

N : number of stages, r : stage/block size

Search Direction Computation

1

Set up KKT System

$$Y := C\Phi^{-1}C^T$$

$9Nr^3$ FLOPS

2

Factor coefficient matrix

$2Nr^3$ FLOPS

3

Compute search direction (forward & backward subst.)

$\mathcal{O}(Nr^2)$ FLOPS

- Reduce naive $O(N^3r^3)$ to $O(Nr^3)$ by banded fact.
  [Wang & Boyd 2008], [Rao, Wright & Rawlings 1998]
- 20% of total effort
- Independent of problem structure

N : number of stages, r : stage/block size

Search Direction Computation

1

Set up KKT System

$$Y := C\Phi^{-1}C^T$$

$9Nr^3$ FLOPS

2

Factor coefficient matrix

$2Nr^3$ FLOPS

3

Compute search direction (forward & backward subst.)

$\mathcal{O}(Nr^2)$ FLOPS

- 80% of total effort
- Generally ignored in the literature
- Possible to exploit structure of problem
  [Domahidi et. al. CDC 2012]

- Reduce naive $O(N^3r^3)$ to $O(Nr^3)$ by banded fact.
  [Wang & Boyd 2008], [Rao, Wright & Rawlings 1998]
- 20% of total effort
- Independent of problem structure

N : number of stages, r : stage/block size

Cost Analysis for Forming $Y = C\Phi^{-1}C^T$

Block structure due to MS property

$$Y = \begin{bmatrix} Y_{11} & Y_{12} & 0 & 0 & \cdots \\ Y_{12}^T & Y_{22} & Y_{23} & 0 & \cdots \\ 0 & Y_{23}^T & Y_{33} & Y_{34} & \cdots \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{bmatrix}$$

Block-wise computation of Y

$$Y_{ii} := C_{i-1}\Phi_{i-1}^{-1}C_{i-1}^T + D_i\Phi_i^{-1}D_i^T$$

$$Y_{ii+1} := D_i\Phi_i^{-1}C_i^T$$

- 80% of total effort
- inherently parallel

re-use already computed elements

Proposed method (saves $2r^3$ flops)

1	$L_i = chol(\Phi_i)$	$2/3r^3$	(Cholesky factorization)
2	$V_i = C_i/L_i^T$	r^3	(Matrix backward subst.)
3	$W_i = D_i/L_i^T$	r^3	
4	$Y_{i,i} = V_i^TV_i$	r^3	(Matrix-matrix products)
5	$+W_i^TW_i$	r^3	
6	$Y_{i,i+1} = W_iV_i^T$	$2r^3$	
	Total	$20/3r^3$	flops

Cost Analysis for Forming $Y = C\Phi^{-1}C^T$

Block structure due to MS property

$$Y = \begin{bmatrix} Y_{11} & Y_{12} & 0 & 0 & \cdots \\ Y_{12}^T & Y_{22} & Y_{23} & 0 & \cdots \\ 0 & Y_{23}^T & Y_{33} & Y_{34} & \cdots \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{bmatrix}$$

Block-wise computation of Y

$$Y_{ii} := C_{i-1}\Phi_{i-1}^{-1}C_{i-1}^T + D_i\Phi_i^{-1}D_i^T$$

$$Y_{ii+1} := D_i\Phi_i^{-1}C_i^T$$

- 80% of total effort
- inherently parallel

re-use already computed elements

Proposed method (saves $2r^3$ flops)

1	$L_i = chol(\Phi_i)$	$2/3r^3$	(Cholesky factorization)
2	$V_i = C_i/L_i^T$	r^3	(Matrix backward subst.)
3	$W_i = D_i/L_i^T$	r^3	
4	$Y_{i,i} = V_i^T V_i$	r^3	(Matrix-matrix products)
5	$+W_i^T W_i$	r^3	
6	$Y_{i,i+1} = W_i V_i^T$	$2r^3$	
	Total	$20/3r^3$	flops

How much can be saved if the structure of C, D, and Φ are known?

Fine-grained Structure Exploitation

- Additional structure exploitation possible for special cases (block-wise):

Theoretical speedups compared to base case		Objective		
		$c_i^T v_i$	$v_i^T Q v_i$, Q diag.	$v_i^T Q v_i$, Q dense
Constraints	$\underline{v} \leq v_i \leq \bar{v}$	9.3x	9.3x	1.4x
	$F v_i \leq f_i$	1.0x	1.0x	1.0x
	$v_i^T M v_i \leq r$, M diag.	6.7x	6.7x	1.4x
	$v_i^T M v_i \leq r$, M dense	1.4x	1.4x	1.4x (1.8x if M=Q)

Fine-grained Structure Exploitation

- Additional structure exploitation possible for special cases (block-wise):

Theoretical speedups compared to base case		Objective		
		$c_i^T v_i$	$v_i^T Q v_i$, Q diag.	$v_i^T Q v_i$, Q dense
Constraints	$\underline{v} \leq v_i \leq \bar{v}$	9.3x	9.3x	1.4x
	$F v_i \leq f_i$	1.0x	1.0x	1.0x
	$v_i^T M v_i \leq r$, M diag.	6.7x	6.7x	1.4x
	$v_i^T M v_i \leq r$, M dense	1.4x	1.4x	1.4x (1.8x if M=Q)

- Example for typical MPC problem:
  - Stages 0...N-1: Q, R diagonal
  - Stage N: P dense

$\Rightarrow$ **~75% complexity reduction**

$$
\begin{aligned}
\min \quad & x_N^T P x_N + \textstyle\sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i \\
\text{s.t.} \quad & x_0 = x, \; x_{i+1} = A x_i + B u_i \\
& \underline{x} \leq x_i \leq \bar{x}, \; \underline{u} \leq u_i \leq \bar{u}, \\
& x_N^T P x_N \leq \alpha
\end{aligned}
$$

Fine-grained Structure Exploitation

- Additional structure exploitation possible for special cases (block-wise):

Theoretical speedups compared to base case

		Objective		
		$c_i^T v_i$	$v_i^T Q v_i$, Q diag.	$v_i^T Q v_i$, Q dense
Constraints	$\underline{v} \leq v_i \leq \bar{v}$	9.3x	9.3x	1.4x
	$F v_i \leq f_i$	1.0x	1.0x	1.0x
	$v_i^T M v_i \leq r$, M diag.	6.7x	6.7x	1.4x
	$v_i^T M v_i \leq r$, M dense	1.4x	1.4x	1.4x (1.8x if M=Q)

- Example for typical MPC problem:
  - Stages 0...N-1: Q, R diagonal
  - Stage N: P dense

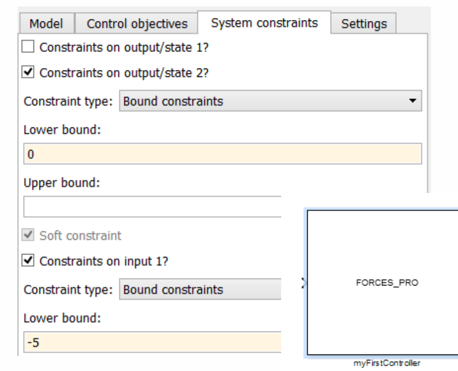
$\Rightarrow$ **~75% complexity reduction**

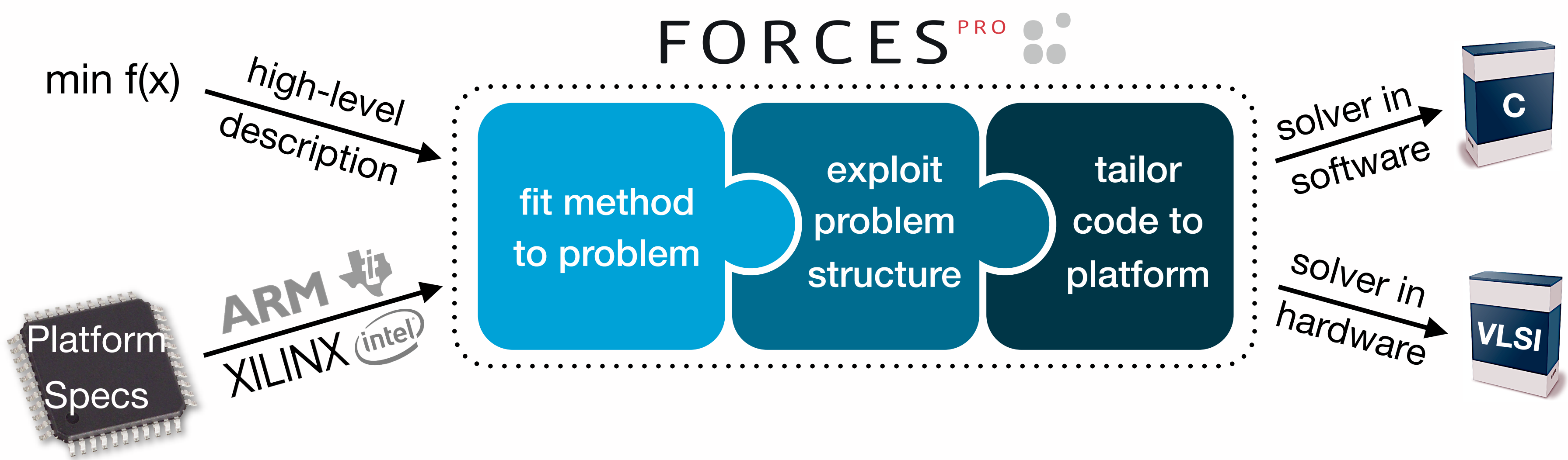
$$\min \quad x_N^T P x_N + \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i$$
$$\text{s.t.} \quad x_0 = x, \;\; x_{i+1} = A x_i + B u_i$$
$$\underline{x} \leq x_i \leq \bar{x}, \;\; \underline{u} \leq u_i \leq \bar{u},$$
$$x_N^T P x_N \leq \alpha$$

Structure exploitation can be automatized by code generation

FORCES Pro: Multi-method Autocoder

- From problem & platform specification to implementation
- Generates library-free ANSI-C code
- New: generate code directly from Simulink





Current Features

- Optimized for **parametric** multistage convex programs of the form

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{N} \frac{1}{2} z_i^T H_i z_i + f_i^T z_i \\
\text{subject to} \quad & D_1 z_1 = c_1 \\
& C_{i-1} z_{i-1} + D_i z_i = c_i, && \forall i \in \{2, 3, \ldots, N\} \\
& z_{\min} \leq z_i \leq z_{\max}, && \forall i \in \{1, 2, \ldots, N\} \\
& A_i z_i \leq b_{\max}, && \forall i \in \{1, 2, \ldots, N\} \\
& z_i^T Q_{i,k} z_i + L_{i,k} z_i \leq r_{i,k} \quad \forall k && \forall i \in \{1, 2, \ldots, N\}
\end{aligned}
$$

Interfaces
- Matlab
- Simulink
- Python
- dSpace

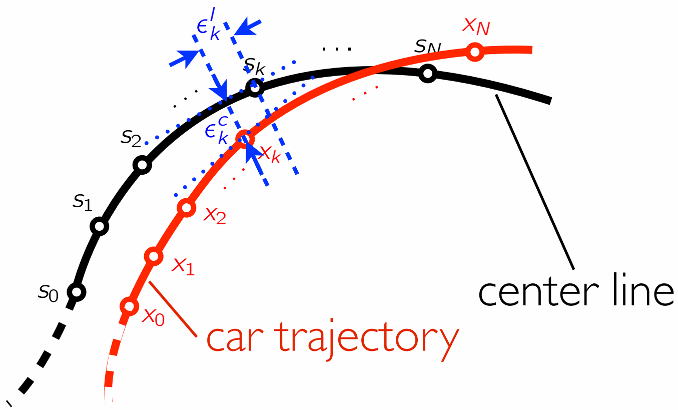
Methods
- Primal-dual interior point
- ADMM 1 & 2, custom projections
- Primal (fast) gradient
- Dual (fast) gradient 1

Platforms
- x86, x86_64
- Tricore
- PowerPC
- ARM

Example: Autonomous Racing

Autonomous Racing - Implementation

- **Idea:** reformulate "minimum time" objective as "maximum progress"
  - Progress measured by projection on center line (nonlinear operator)



$$\max\ s_N - \sum_{k=1}^{N} \gamma_c \|\epsilon_k^c\|^2 + \gamma_l \|\epsilon_k^l\|^2$$

$$\text{s.t. } s_0 = \tilde{s}\,,\ \ x_0 = \tilde{x}$$

$$s_{k+1} = s_k + v_k$$

$$0 \le v_k \le \bar{v}\,,\ \ x_k \in \mathbb{X}_k,\, u_k \in \mathbb{U}_k$$

$$x_{k+1} = A_k x_k + B_k u_k + g_k$$

$$\epsilon_k^c = E_k x_k + F_k s_k + f_k$$

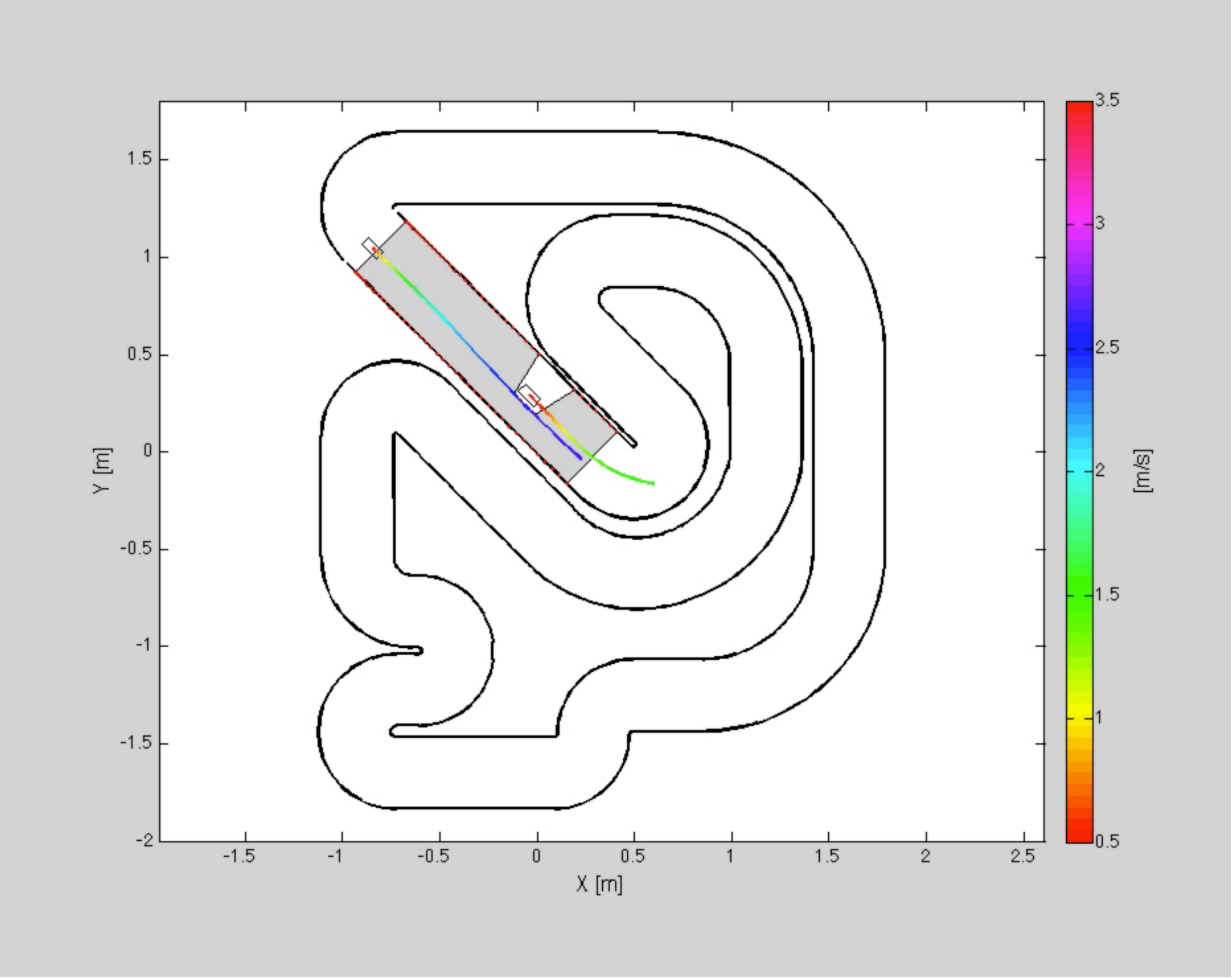
$$\epsilon_k^l = G_k x_k + H_k s_k + h_k$$

- SQP method [Diehl 2002]:
  1. **Linearize** continuous-time dynamics around trajectory
  2. **Discretize** using matrix exponential
  3. **Solve local convex** approximation (**QP**)
  4. Update trajectory & apply first input

- QP solved in 14 milliseconds (N=40, 540 variables, 680 constraints)

50 Hz sampling rate on smartphone

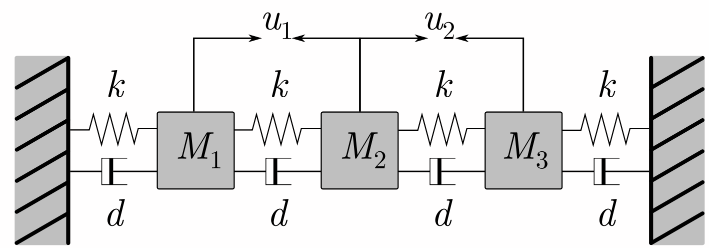
Closed-loop Simulation



[Watch online]

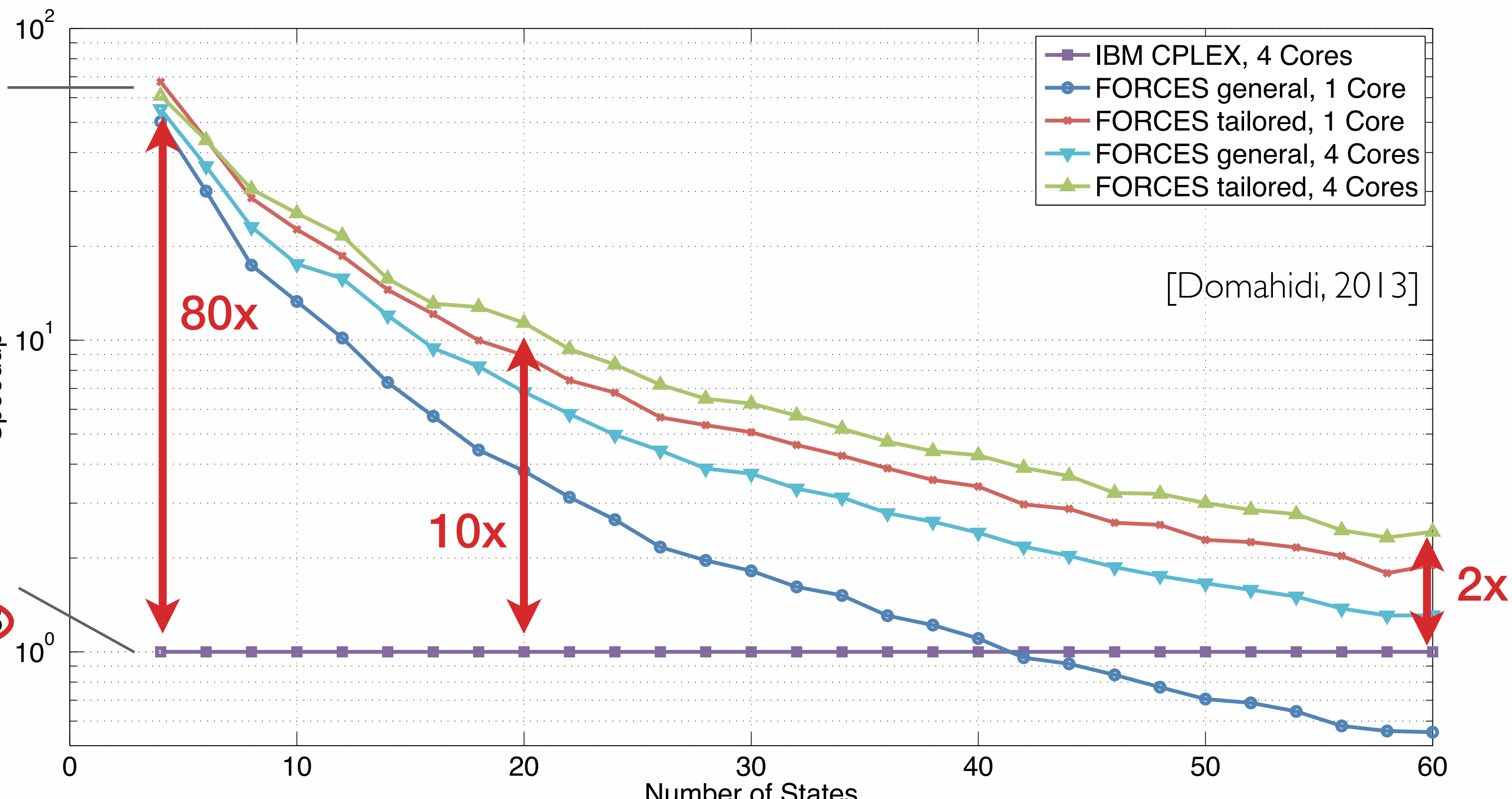
Comparison IPM to Commercial Solver

- Standard MPC problem for oscillating chain of masses (on Intel i5 @3.1 GHz)
- CPLEX N/A on embedded systems



FORCES
Solve time: 90 µs
Code size: 52 KB

IBM CPLEX
Solve time: 5470 µs
Code size: 11700 KB

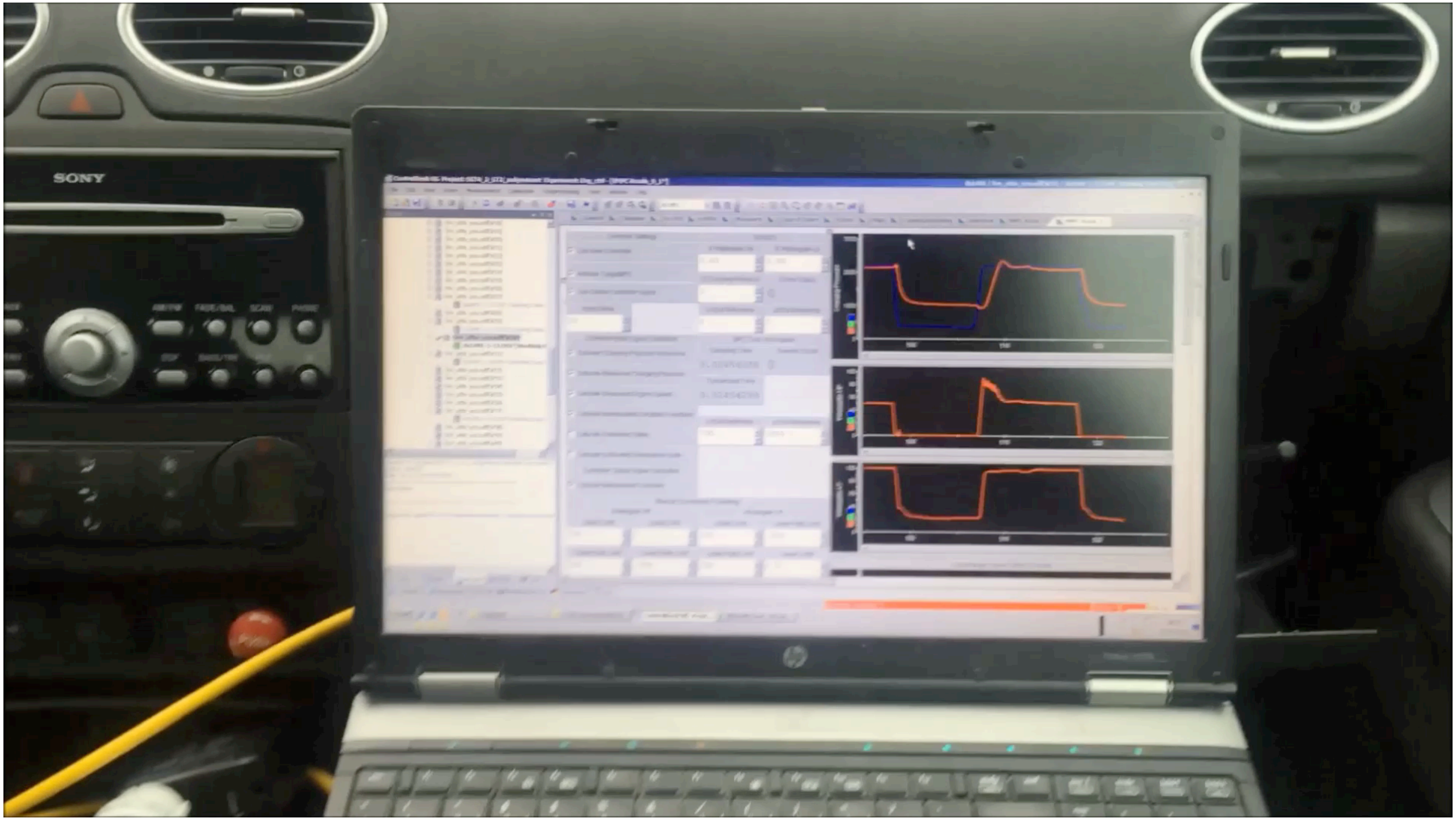


Gasoline 2-Stage Turbocharger Control

By courtesy of D. Ritter and T. Albin, Institut für Regelungstechnik, RWTH Aachen University
ACADO + FORCES Pro

50 Hz sampling rate on dSpace Autobox

Gasoline 2-Stage Turbocharger Control



By courtesy of D. Ritter and T. Albin, Institut für Regelungstechnik, RWTH Aachen University
ACADO + FORCES Pro

50 Hz sampling rate on dSpace Autobox

Non-embedded Applications: Finance

expected return

transaction costs

$$\max_{x} \quad \mu^T x - \sum_{i=1}^{N} c_i |x_i - \bar{x}_i|$$

current position

$$\text{s.t.} \quad x_{\min} \leq x \leq x_{max},$$

risk constraint

$$x^T \Sigma x \leq r$$

- Reliability is a must
- QCQP solver implemented using FORCES Pro allocating 30M$/day in NYC
- Switching to FORCES Pro allowed to reduce simulation time by 100x

Future Developments

MIXED-INTEGER PROBLEMS

- Branch-and-bound solvers
- Disjunctive programming
- **Piecewise-affine dynamics**

NONLINEAR SMOOTH PROBLEMS

- Efficient integration methods for ODEs
- Easy-to-use automatic linearisation and discretisation tools

LARGE-SCALE PROBLEMS

- Tools for power distribution grids
- Difference of convex functions programming
- Large-scale portfolio problems

Exercise Session

- Choice of 3 types of exercises:
  - Graphical optimal control design using **Simulink** + FORCES Pro
  - Matlab API of FORCES Pro (**quadratic constraints** etc.)
  - Your own **barrier interior-point method**
- For barrier IPM:
  - Problem: $\min_x \{1/2 x^T H x \mid Gx \leq d\}$
  - Centering step via Newton: $(\nabla^2 f(x) + \kappa \nabla^2 \phi(x))\Delta x_{nt} = -\nabla f(x) - \kappa \nabla \phi(x)$
  - Gradient and Hessian of Barrier function:

$$\nabla \phi(x) = \sum_{i=1}^{m} \frac{1}{d_i - g_i x} g_i^T, \nabla^2 \phi(x) = \sum_{i=1}^{m} \frac{1}{(d_i - g_i x)^2} g_i^T g_i$$

  - Newton step: $(H + \kappa \sum_{i=1}^{m} \frac{1}{(d_i - g_i x)^2} g_i^T g_i)\Delta x_{nt} = -Hx - \sum_{i=1}^{m} \frac{1}{d_i - g_i x} g_i^T$