Scalable Dynamic Optimization

Mihai Anitescu
*Department of Statistics*
*University of Chicago*

*With: Victor Zavala*
Real-Time Optimization is Pervasive in Energy: Estimation, Management, Control Requires Extreme-Scale NLP Solvers: Model Size and Short Time Scales
Need for MPC

- Traditional control approximates the model based on output (mostly) ignoring its physical structure.
- High variability in forcing and nonlinearity requires a physical model-based approach.
- Far more computationally intensive bottleneck is optimization problem.
Fundamental Limitations of Off-The-Shelf Optimization

Example DO:

$$\min_{x(t)} \frac{1}{2} (x(t) - \eta(t))^2 + \frac{1}{2} x(t)^2 \cdot \eta(t)$$

---

**Off-the-Shelf**: Solve to Given Accuracy (Neglect Dynamics)

$$e^j(t) = \| \nabla_x f(x^j(t), \eta(t)) \| \leq \delta_e$$

**Real-Time (Z & A)**: One SQP Iteration per step
Outline of the Talk

1. Generalized Equation / “Incomplete Optimization”

2. Exact Differentiable Penalty Approach for Accuracy and Reduced Latency

3. Numerical Case Studies

4. Conclusions and Future Work
1. Generalized Equation / “Incomplete Optimization”
**Context:** Parametric NLP

\[ \min_{x \in X} f(x, t), \ s.t. \ c(x, t) = 0 \]

**KKT system for QP**

\[ \min \ \nabla_x f(x_{i0}^*, t_0)^T \Delta x + \frac{1}{2} \Delta x^T \nabla_{xx} f(x_{i0}^*, t_0) \Delta x \]
\[ \text{s.t.} \ c(x_{i0}^*, t) + \nabla c(x_{i0}^*, t_0)^T \Delta x = 0 \]
\[ \Delta x \geq -x_{i0}^* \]

**Time linearization of Optimality Conditions:** Find \( \bar{w}_t = [\bar{x}_t \ \bar{\lambda}_t] \)

\[ 0 \in F(w_{t_0}^*, t) + \nabla_w F(w_{t_0}^*, t_0)(w - w_{t_0}^*) + N_W(w) \]

**Exact Solution Satisfies:**

\[ \delta \in F(w_{t_0}^*, t_0) + \nabla_w F(w_{t_0}^*, t_0)(w - w_{t_0}^*) + N_W(w) \]

\[ \delta = F(w_{t_0}^*, t_0) - F(w_{t_0}^*, t) \]

**From Lipschitz Continuity of strongly regular GE:**

\[ \|w_t^* - \bar{w}_t\| \leq \Delta t^2 \]

**Optimal Solution**

\[ r(w_t^*, t) \]

**QP Solution**

\[ \bar{w}_t \]

**Linearization Point**

\[ w_{t_0}^* \]

- **Strong Regularity Requires SSOC and LICQ**
- **NLP Error is Bounded by LGE Perturbation**
- **One QP solution from exact manifold is second-order accurate**
One-QP per step stabilizes

But for linearized DO I am never EXACTLY on the manifold: What then?

Theorem (elucidating an issue posed by Diehl et al.)

- A: LGE is Strongly Regular at ALL $w^*_{t_k}$ e.g. NLP satisfies LICQ and SOSC everywhere

Then: For sufficiently small $\Delta t$, we can track the manifold stably, solving 1 QP per step

$$\| \bar{w}_{t_k} - w^*_{t_k} \| \leq L_\psi \delta_r \Rightarrow \| \bar{w}_{t_{k+1}} - w^*_{t_{k+1}} \| \leq L_\psi \delta_r$$

Moreover: Stability Holds Even if QP Solved to $O(\Delta t^2)$ accuracy. Can use iterative methods.

Much less effort per step and better chances for real-time performance!
Need for more features of DO solvers

- One QP per step may still be too much
- Moreover I may need also good global and fast local convergence properties as well, it is not all about asymptotics!
- Sometimes one switch regimes, the optimal point moves far away, and you still want to be able to track well. – MPC algorithm must exhibit global convergence and fast local convergence (i.e. Newton)!
- Also, power grid problems can be huge (US ~ 1 – 100 Billion Variables). Need scalable solvers.

Control of Polymerization Reactor
2. Exact Differentiable Penalty Approach for Accuracy and Reduced Latency
Technical Problem

$$\min_x f(x, t) \quad \text{s.t.} \quad h(x, t) = 0, \quad (\lambda) \quad x \geq 0.$$

$$w^T = [x^T, \lambda^T]$$

Solution forms Time-Moving and Non-Smooth Manifold

- Challenge is to Track Manifold Accurately (Classical Optimization) AND Stably (Latency Conscious: A good Step, Computer Fast)
Technical Problem

- Challenge is to Track Manifold Accurately AND Stably (Get Good Step with Minimum Latency)

- This requires NLP Solvers with the Following Features:
  - A) Classical Optimization Oriented:
    1) Superlinear Convergence (Newton-Based)
    2) Scalable Step Computation (Iterative Linear Algebra)
  - B) Latency Conscious:
    3) Asymptotic Monotonicity of Minor Iterations (Makes Progress in O(N))
    4) Active-Set Detection and Warm-Start

- Existing Solvers Tend to Fail at Least One Feature
  - Interior Point: 4, and to some extent, 2,3
  - Augmented Lagrangian: 1
  - SQP: 2
Consider Transformation using Squared Slacks

\[
\begin{align*}
\min_x & \quad f(x) \\
\text{s.t.} & \quad h(x) = 0 \\
& \quad x \geq 0
\end{align*}
\]

Equivalent To:

\[
\begin{align*}
\min_{z^2} & \quad f(z^2) \\
\text{s.t.} & \quad h(z^2) = 0
\end{align*}
\]

\[\mathcal{L}(z^2, \lambda) = f(z^2) + \lambda^T h(z^2)\]

\[
\nabla_z \mathcal{L}(z^2, \lambda) = 2 \cdot Z \cdot (\nabla f(z^2) + \nabla h(z^2) \lambda) \]
\[
= 2 \cdot X^{1/2} \nabla_x \mathcal{L}(x, \lambda)
\]

Apply DiPillo and Grippo’s Penalty Function \textit{DiPillo,Grippo, 1979, Bertsekas, 1982}

\[
P(x, \lambda, \alpha, \beta) = \mathcal{L}(x, \lambda) + \frac{1}{2} \alpha c(x)^T c(x) + 2 \beta \nabla_x \mathcal{L}(x, \lambda)^T X \nabla_x \mathcal{L}(x, \lambda)
\]

Solve NLP Indirectly Through EDPF Problem:

\[
\begin{align*}
\min_{x,\lambda} & \quad P(x, \lambda, \alpha, \beta) \\
\text{s.t.} & \quad x \geq 0
\end{align*}
\]
Exact Differentiable Penalty Functions with Bound Constraints

\[
\min_x f(x) \quad \text{s.t.} \quad h(x) = 0, \quad x \geq 0
\]

\[
\min_{x, \lambda} P(x, \lambda, \alpha, \beta) \quad \text{s.t.} \quad x \geq 0
\]

\[
P(x, \lambda, \alpha, \beta) = \mathcal{L}(x, \lambda) + \frac{1}{2} \alpha h(x)^T h(x) + 2\beta \nabla_x \mathcal{L}(x, \lambda)^T x \nabla_x \mathcal{L}(x, \lambda)
\]

Advantages
- EDPF Differentiable Everywhere
- Unconstrained Problem with Box Constraints, scalable, superlinear, warm-start
- Makes Progress at Each Iteration (latency)

Questions
- Under What Conditions Do Minimizers of EDPF and NLP Coincide?
- How to Deal with Nonconvexity?
  - Detect and Exploit Negative Curvature
- Can We Enable Scalability AND NOT NEED THIRD DERIVATIVE?
  - First and Second Derivatives
  - Iterative Linear Algebra
The big picture

- Combine Bertsekas bound constrained EDPF with Lin-More trust region.

- Superlinear convergence w/o Maratos from EDPF
- Matrix free from Lin-More

- Improvement in Order N from EDPF
- Warm-Start and active set detection from Lin-More

- And maybe this will help optimization proper ....

- Our contributions:
  - Formalizing bound constrained EDPF properties
  - Using trust-region to get rid of the third derivative while preserving both global convergence of EPF and superlinear convergence of Newton.
  - Demonstrating that the approach scales well.
Derivatives and Minimizers of EDPF

\[ P(x, \lambda, \alpha, \beta) = \mathcal{L}(x, \lambda) + \frac{1}{2} \alpha h(x)^T h(x) + 2\beta \nabla_x \mathcal{L}(x, \lambda)^T X \nabla_x \mathcal{L}(x, \lambda) \]

In Compact Form

\[ P_{\alpha, \beta}(w) = \mathcal{L}(w) + \frac{1}{2} \nabla_w \mathcal{L}(w)^T K_{\alpha, \beta}(w) \nabla_w \mathcal{L}(w) \]

\[ K_{\alpha, \beta}(w) = \begin{bmatrix} 4\beta X \\ \alpha I_m \end{bmatrix} \]

First Derivative

\[ \nabla P = \nabla \mathcal{L} + \nabla^2 \mathcal{L} K \nabla \mathcal{L} + \frac{1}{2} \Gamma \text{diag}(\nabla \mathcal{L}) \nabla \mathcal{L} \]

Is KKT Point of EDPF a KKT Point of NLP?

\[ \sqrt{X} \nabla_x P = 0 \quad \rightarrow \quad \sqrt{X} \nabla_x \mathcal{L}(x, \lambda) = 0 \]
\[ \nabla_\lambda P = 0 \quad \rightarrow \quad \nabla_\lambda \mathcal{L}(x, \lambda) = 0 \]

Theorem:
Under LICQ and SC there exist \( \alpha, \beta \), such that KKT Point of EDPF is KKT point of NLP.

Proof:

\[ \begin{bmatrix} I_{n \times n} + 4\beta \sqrt{X} \nabla_x \mathcal{L}(w^*) \sqrt{X} + 2\beta \text{diag}(\nabla_x \mathcal{L}(w^*)) \alpha \sqrt{X} \nabla_x h(x^*)^T \\ 4\beta \nabla_x h(x^*) \sqrt{X} \end{bmatrix} \begin{bmatrix} \sqrt{X} \nabla_x \mathcal{L}(w^*) \\ h(x^*) \end{bmatrix} = \begin{bmatrix} 0_n \\ 0_m \end{bmatrix} \]

Matrix on LHS is PD For sufficient large \( \alpha \) and sufficiently small \( \beta \).
Derivatives and Minimizers of EDPF

Second Derivative

\[ \nabla^2 P \cdot u = \nabla^2 L \cdot u + \nabla^2 L K \nabla^2 L \cdot u + \nabla^2 L \text{diag}(\nabla L) \Gamma \cdot u + \Gamma \text{diag}(\nabla L) \nabla^2 L \cdot u + \nabla (\nabla^2 L \cdot u) K \nabla L \]

High-Order Term Vanishes at KKT Point Because \( K \nabla L = 0 \).

Is Strict Minimizer of EDPF a Strict Minimizer of NLP?

Theorem:

i) If KKT Point satisfies SSOC for NLP then there exist \( \alpha, \beta \), such that it satisfies SSOC of EDPF.

ii) If KKT Point does not satisfy SSOC for NLP then there exist \( \alpha, \beta \), such that this is not a strict local minimizer of EDPF.

Proof: Relies on Analysis of Projected Hessian where \( N \) is null-space matrix.

\[
\begin{align*}
\nu^T N^T \nabla^2 P N \nu \\
= \begin{bmatrix} \nu_x^T N_x^T & \nu_x^T \\ \nu_x^T N_x & \nu_x \end{bmatrix} \begin{bmatrix} H & A^T \\ A & N_x \nu_x \end{bmatrix} + \begin{bmatrix} \nu_x^T N_x^T & \nu_x^T \\ \nu_x^T N_x & \nu_x \end{bmatrix} \begin{bmatrix} 4 \beta X & 0 \\ 0 & \alpha \mathbb{I}_m \end{bmatrix} \begin{bmatrix} H & A^T \\ A & N_x \nu_x \end{bmatrix}.
\end{align*}
\]
Derivatives and Minimizers of EDPF

A “Strong” Dennis-More Condition

Exact Hessian

\[ \nabla^2 P \cdot u = \nabla^2 \mathcal{L} \cdot u + \nabla^2 \mathcal{L} K \nabla^2 \mathcal{L} \cdot u + \nabla^2 \mathcal{L} \text{diag}(\nabla \mathcal{L}) \Gamma \cdot u + \Gamma \text{diag}(\nabla \mathcal{L}) \nabla^2 \mathcal{L} \cdot u + \nabla(\nabla^2 \mathcal{L} \cdot u) K \nabla \mathcal{L}. \]

Approximate Hessian

\[ Q \cdot u = \nabla^2 \mathcal{L} \cdot u + \nabla^2 \mathcal{L} K \nabla^2 \mathcal{L} \cdot u + \nabla^2 \mathcal{L} \text{diag}(\nabla \mathcal{L}) \Gamma \cdot u + \Gamma \text{diag}(\nabla \mathcal{L}) \nabla^2 \mathcal{L} \cdot u \]

Approximate Hessian is Asymptotically Convergent

\[ (Q(w) - \nabla^2 P(w)) \cdot u = \nabla(\nabla^2 \mathcal{L}(w) \cdot u) K(w) \nabla \mathcal{L}(w) \]

\[ = o(u)O(\|w - w^*\|), \quad \text{because} \quad K(w^*) \nabla \mathcal{L}(w^*) = 0 \quad \text{as} \quad w \to w^* \to 0. \]

Implication:

- We can drop third-order terms and derive quasi-Newton algorithms that retain superlinear convergence.
- Much easier implementation.
Trust-Region Newton

\[ \min_{x, \lambda} P_{\alpha, \beta}(w) \text{ s.t. } w \in \Omega \]

- Need to detect and exploit directions of negative curvature

- Use Trust-Region Newton Framework of Lin and More (TRON)

1) Determine Activity Using Cauchy Point

\[ [w^c, A^c] = \text{Proj}[w - \alpha^c \nabla P(w)] \]

2) Compute Search Step by Solving Trust-Region QP using Steihaug’s Preconditioned Conjugate Gradient Approach (PCG)

\[ \min_{\Delta w} \nabla P(w)^T \Delta w + \frac{1}{2} \Delta w^T Q(w) \Delta w \]

s.t. \[ \Delta w_i = 0, \quad i \in A^c \]

\[ \|\Delta w\| \leq \Delta \]

3) Check Progress Over Cauchy Step and Update Trust Region Radius

- Approach Converges to Strict Local Minimizers of NLP Globally and Superlinearly
- Requires \( \alpha, \beta \) to Satisfy Conditions of Previous Theorems
Computational Scalability

Derivatives
- EDPF Hessian Can be Assembled using Hessian and Jacobian Vector Products

\[ \nabla^2 \mathcal{L} \cdot \nu = \begin{bmatrix} H & A^T \\ A & \nu_x \end{bmatrix} \begin{bmatrix} \nu_x \\ \nu_\lambda \end{bmatrix} = \begin{bmatrix} H \cdot \nu_x + A^T \cdot \nu_\lambda \\ A \cdot \nu_x \end{bmatrix}. \quad \text{Kernel} \]

\[ Q \cdot u = \nabla^2 \mathcal{L} \cdot u + \nabla^2 \mathcal{L} K \nabla^2 \mathcal{L} \cdot u + \nabla^2 \mathcal{L} \text{diag}(\nabla \mathcal{L}) \Gamma \cdot u + \Gamma \text{diag}(\nabla \mathcal{L}) \nabla^2 \mathcal{L} \cdot u \]

Requires 2 Unique Kernels

PCG

\[
\begin{aligned}
\min_{s_d^k} & \quad g^k_T N^k s_d^k + \frac{1}{2} s_d^k T (N^k)^T Q^k N^k s_d^k \\
\text{s.t.} & \quad \|D^k N_j^k s_d^k\| \leq \Delta^k.
\end{aligned}
\]

- Does Not Require Assembling Reduced Hessian
- Requires Action of Inverse Preconditioner \((D^k)^{-1} \cdot r\)
- Incomplete Cholesky, PARDISO, Algebraic Multigrid
- Inertia Detected Externally (Not by Linear Solver)
3. Numerical Results
**Numerical Examples**

**Algorithmic Behavior**

\[
\begin{align*}
\min & \quad (x_1 - 1)^2 + (x_2 - 2)^2 + (x_3 - 3)^2 + x_1 x_4 \\
\text{s.t.} & \quad x_1 x_4 + x_1 x_2 + x_3 = 4, \quad (\lambda) \\
& \quad x_1, x_2, x_3, x_4 \geq 0.
\end{align*}
\]

<table>
<thead>
<tr>
<th>$k$</th>
<th>$P^k$</th>
<th>$g_{\text{Proj}}^k$</th>
<th>$\rho^k$</th>
<th>$|s^k|$</th>
<th>$|\Delta^k|$</th>
<th>$|Q^k - H^k|$</th>
<th>$\lambda(Q^k_d)$</th>
<th>$\lambda(H^k_d)$</th>
<th>$\text{card}(A^k_P)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>25.150</td>
<td>2.0e+2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-2.48</td>
<td>-22.67</td>
</tr>
<tr>
<td>1</td>
<td>3.449</td>
<td>5.9e+1</td>
<td>+3.26</td>
<td>2.5e-1</td>
<td>261.9</td>
<td>2.0e+2</td>
<td></td>
<td>-2.48</td>
<td>-22.67</td>
</tr>
<tr>
<td>2</td>
<td>3.449</td>
<td>5.9e+1</td>
<td>-0.70</td>
<td>0.0e+0</td>
<td>523.9</td>
<td>5.8e+1</td>
<td></td>
<td>-2.48</td>
<td>-22.67</td>
</tr>
<tr>
<td>3</td>
<td>3.449</td>
<td>5.9e+1</td>
<td>-0.62</td>
<td>0.0e+0</td>
<td>131.0</td>
<td>5.8e+1</td>
<td></td>
<td>-2.48</td>
<td>-22.67</td>
</tr>
<tr>
<td>4</td>
<td>3.449</td>
<td>5.9e+1</td>
<td>-0.33</td>
<td>0.0e+0</td>
<td>32.0</td>
<td>5.8e+1</td>
<td></td>
<td>-2.48</td>
<td>-22.67</td>
</tr>
<tr>
<td>5</td>
<td>3.449</td>
<td>5.9e+1</td>
<td>-0.28</td>
<td>0.0e+0</td>
<td>8.0</td>
<td>5.8e+1</td>
<td></td>
<td>-2.48</td>
<td>-22.67</td>
</tr>
<tr>
<td>6</td>
<td>1.533</td>
<td>2.5e+1</td>
<td>+0.37</td>
<td>2.0e+0</td>
<td>2.0</td>
<td>5.8e+1</td>
<td></td>
<td>-2.48</td>
<td>-22.67</td>
</tr>
<tr>
<td>7</td>
<td>0.945</td>
<td>1.6e+0</td>
<td>+0.52</td>
<td>1.9e-1</td>
<td>2.0</td>
<td>2.9e+1</td>
<td>+0.15</td>
<td>-0.39</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0.944</td>
<td>4.9e-1</td>
<td>+0.48</td>
<td>2.6e-3</td>
<td>4.0</td>
<td>1.9e+0</td>
<td>+0.19</td>
<td>+0.37</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0.943</td>
<td>4.5e-1</td>
<td>+0.93</td>
<td>1.4e-3</td>
<td>4.0</td>
<td>4.0e-1</td>
<td>+0.19</td>
<td>+0.25</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0.909</td>
<td>2.3e-1</td>
<td>+0.94</td>
<td>1.8e-1</td>
<td>8.0</td>
<td>3.4e-1</td>
<td>+0.40</td>
<td>+0.40</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0.908</td>
<td>1.7e-6</td>
<td>+0.99</td>
<td>8.7e-3</td>
<td>3.1e-6</td>
<td>+0.38</td>
<td>+0.38</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

- Trust Region Management Critical  - Line Search Solvers Fail (IPOPT)
- High Nonlinearity at Beginning of Search (Third order term induces it)
Numerical Examples

Optimal Control Problem

\[
\begin{align*}
\min & \quad \int_0^T \left( \alpha_c \cdot (c(\tau) - \bar{c})^2 + \alpha_t \cdot (t(\tau) - \bar{t})^2 + \alpha_u \cdot (u(t) - \bar{u})^2 \right) d\tau \\
\text{s.t.} & \quad \dot{c}(\tau) = \frac{1 - c(\tau)}{\theta} - p_k \cdot \exp \left( - \frac{pE}{t(\tau)} \right) \cdot c(t) \\
& \quad \dot{t}(\tau) = \frac{t_f - t(\tau)}{\theta} + p_k \cdot \exp \left( - \frac{pE}{t(\tau)} \right) \cdot c(\tau) - p\alpha \cdot u(\tau) \cdot (t(\tau) - t_c) \\
& \quad c(\tau), t(\tau), u(\tau) \geq 0, \quad \tau \in [0, T] \\
& \quad c(0) = c(\tau_{sys}), \quad t(0) = t(\tau_{sys}).
\end{align*}
\]

<table>
<thead>
<tr>
<th>$N$</th>
<th>$n$</th>
<th>$m$</th>
<th>$n_w$</th>
<th>nnz(\nabla^2 \mathcal{L})</th>
<th>nnz(Q)</th>
<th>%dens(\nabla^2 \mathcal{L})</th>
<th>%dens(Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>1,500</td>
<td>1,000</td>
<td>2,500</td>
<td>10,486</td>
<td>26,492</td>
<td>2.0e−1</td>
<td>4.0e−1</td>
</tr>
<tr>
<td>1,000</td>
<td>3,000</td>
<td>2,000</td>
<td>5,000</td>
<td>20,996</td>
<td>52,972</td>
<td>8.4e−2</td>
<td>2.0e−1</td>
</tr>
<tr>
<td>5,000</td>
<td>15,000</td>
<td>10,000</td>
<td>25,000</td>
<td>104,996</td>
<td>264,972</td>
<td>1.6e−2</td>
<td>4.0e−2</td>
</tr>
<tr>
<td>10,000</td>
<td>30,000</td>
<td>20,000</td>
<td>50,000</td>
<td>209,996</td>
<td>529,972</td>
<td>8.3e−3</td>
<td>2.1e−2</td>
</tr>
</tbody>
</table>

- Discretize and Scale Problem Up by Increasing Horizon $N$
- Sparsity of Augmented System Retained in Hessian of EDPF
- Drop Tolerance Incomplete Cholesky of 1e-4
## Numerical Examples

### Scalability

<table>
<thead>
<tr>
<th>$n_w$</th>
<th>$i_{pcg}$</th>
<th>$\theta_{ichol,pcg}$</th>
<th>$\theta_{ichol,act}$</th>
<th>$\theta_{ichol,tot}$</th>
<th>$\theta_{chol,pcg}$</th>
<th>$\theta_{chol,act}$</th>
<th>$\theta_{chol,tot}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,250</td>
<td>17</td>
<td>8.5e−2</td>
<td>3.1e−2</td>
<td>1.1e−1</td>
<td>2.7e−2</td>
<td>3.3e−2</td>
<td>6.0e−2</td>
</tr>
<tr>
<td>2,500</td>
<td>24</td>
<td>4.9e−1</td>
<td>1.3e−1</td>
<td>6.2e−1</td>
<td>1.1e−1</td>
<td>1.5e−1</td>
<td>2.6e−1</td>
</tr>
<tr>
<td>5,000</td>
<td>29</td>
<td>1.7e+0</td>
<td>4.4e−1</td>
<td>2.2e+0</td>
<td>5.7e−1</td>
<td>8.5e−1</td>
<td>1.4e+0</td>
</tr>
<tr>
<td>12,500</td>
<td>31</td>
<td>9.0e+0</td>
<td>1.8e+0</td>
<td>1.1e+1</td>
<td>3.8e+0</td>
<td>8.4e+0</td>
<td>1.2e+1</td>
</tr>
<tr>
<td>25,000</td>
<td>31</td>
<td>1.8e+1</td>
<td>5.5e+0</td>
<td>2.4e+1</td>
<td>2.5e+1</td>
<td>5.4e+1</td>
<td>7.8e+1</td>
</tr>
<tr>
<td>50,000</td>
<td>31</td>
<td>3.7e+1</td>
<td>1.8e+1</td>
<td>5.5e+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>125,000</td>
<td>31</td>
<td>9.4e+1</td>
<td>1.1e+2</td>
<td>2.0e+2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>250,000</td>
<td>31</td>
<td>1.9e+2</td>
<td>4.9e+2</td>
<td>6.8e+2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- Scalability of Full Cholesky Not Competitive
- Incomplete Cholesky Gives High Flexibility
  - Can Specify Drop Tolerance to Reduce Latency
- PCG Iterations Scale Well
- Largest Problem Has 250,000 Variables
Numerical Examples

Active-Set Identification for the 2500 dimension case

<table>
<thead>
<tr>
<th>$k$</th>
<th>$P_k^k$</th>
<th>$g_{Proj}^k$</th>
<th>$A_P(w^k)$</th>
<th>$n_{PCG}^k$</th>
<th>$P_k^k$</th>
<th>$g_{Proj}^k$</th>
<th>$A_P(w^k)$</th>
<th>$n_{PCG}^k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4.05e+3</td>
<td>4.52e+3</td>
<td>44</td>
<td>-</td>
<td>1.21e+4</td>
<td>2.43e+5</td>
<td>173</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>1.14e+2</td>
<td>4.70e+3</td>
<td>44</td>
<td>41</td>
<td>4.96e+2</td>
<td>5.76e+4</td>
<td>0</td>
<td>132</td>
</tr>
<tr>
<td>2</td>
<td>1.83e+1</td>
<td>3.72e+3</td>
<td>119</td>
<td>32</td>
<td>9.48e+1</td>
<td>1.86e+3</td>
<td>0</td>
<td>45</td>
</tr>
<tr>
<td>3</td>
<td>1.83e+1</td>
<td>1.55e+2</td>
<td>170</td>
<td>27</td>
<td>5.57e+0</td>
<td>3.27e+4</td>
<td>26</td>
<td>37</td>
</tr>
<tr>
<td>4</td>
<td>1.83e+1</td>
<td>5.59e−6</td>
<td>173</td>
<td>17</td>
<td>3.98e+0</td>
<td>1.11e+3</td>
<td>43</td>
<td>26</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3.98e+0</td>
<td>8.50e−6</td>
<td>44</td>
<td>13</td>
</tr>
</tbody>
</table>

- Case 1) Has 173 variables active at solution and initialized with 44
- Case 2) Has 44 variables active at solution and initialized with 173
- Cauchy Search Efficient at Detecting Activity (Allows for Large Changes Between Iterates)
- Number of PCG Iterations Do Not Degrade as Solution Approached (Compare with IP)
Numerical Examples

Early Termination on problem with N=100

- Run MPC Problem Terminating After 2 Major Iterations and 20 PCG iterations
- Reduced Latency by A Factor of 4 (Four)
- Convergence to Equilibrium Point (Warm-Starting Effective)
4. Conclusions and Future Work

- We derived NLP algorithms that enable:
  1) Superlinear Convergence (Newton-Based)
  2) Scalable Step Computation (Enable Iterative Linear Algebra)
  3) Asymptotic Monotonicity of Minor Iterations (Makes Progress)
  4) Active-Set Detection and Warm-Start

- Critical in “Fast” Real-Time Environments

- Proposed Approach: EDPF + Trust-Region Newton + PCG
  1) Newton-Based in Primal/Dual Space with Convergent Approximate Hessian
  2) Steihaug’s PCG to Detect and Exploit Negative Curvature
  3) PCG Improvement on EDPF Function
  4) Cauchy

- ToDo:
  - More Robust Implementation (Scaling, Trust-Region Update Rules, Ill-Conditioning)
  - Alternative Penalty Functions Requiring Only One Parameter
  - Preconditioning
  - Exploiting Special Structures
  - Comparison with Other NLP Solvers