An Active-Set Trust-Region Algorithm for Nonlinear Optimization

Richard Waltz
Northwestern University

with

Richard Byrd, Univ. of Colorado
Nick Gould, Rutherford Appleton Laboratory, UK
Jorge Nocedal, Northwestern Univ.
Motivation

- Develop \textit{active-set} approach which solves larger problems than existing active-set methods (e.g., SQP).
  
  - Demand for solving larger nonlinear problems
  
  - Active-set methods have good properties (E.g., Warm starts, good active-set and sensitivity info, more stable)
Existing Active-set Approaches

- Sequential Linear Programming (SLP)
- Gradient Projection
- Sequential Linearly Constrained (SLC)
- Sequential Quadratic Programming (SQP)

**SLP-EQP**
- Fletcher, Sainz de la Maza (1989)
- Chin, Fletcher (1999)
- **SLIQUE** (Byrd, Gould, Nocedal, W., 2003)
SQP drawbacks

- Inefficient when reduced space is large
- Must form and factor a (dense) reduced Hessian matrix
- QP subproblems often too expensive.
- How to handle indefinite case?
- Not always able to effectively make use of second derivatives.


Advantages of SLPEQQP

- Solve LPs and EQPs rather than general QPs
- EQPs solved using iterative approach (PCG)
- Do not need to form/factorize Hessian
- Not restricted to problems with a small reduced space
- Easily makes use of exact 2nd derivatives
Problem

\[ \begin{align*}
&\text{min} \quad f(x) \\
&\text{s.t.} \quad h_i(x) = 0, \quad i \in E \\
&\quad \quad g_i(x) \geq 0, \quad i \in I \\
&\quad x \in \mathbb{R}^n
\end{align*} \]

Functions twice continuously differentiable
SLPEQP Overview

0. Given: $x$

1. Solve LP to get working set $\mathcal{W}$.

2. Compute a step, $d$, by solving an equality constrained QP using constraints in $\mathcal{W}$.

3. Set: $x_T = x + d$. 


Estimate active set by solving an LP

$$\min_{d} \nabla f(x)^{T}d$$

s.t. $$h_i(x) + \nabla h_i(x)^{T}d = 0, \; i \in E$$

$$g_i(x) + \nabla g_i(x)^{T}d \geq 0, \; i \in I$$

$$\|d\|_{\infty} \leq \Delta_{LP}$$

$$W(x) = \{i \in E \mid h_i(x) + \nabla h_i(x)^{T}d^{LP} = 0\} \cup \{i \in I \mid g_i(x) + \nabla g_i(x)^{T}d^{LP} = 0\}$$
$l(d) = \nabla f(x)^T d$

\[ + \nu \sum_{i \in E} \left| h_i(x) + \nabla h_i(x)^T d \right| \]

\[ + \nu \sum_{i \in I} \max(\ 0, -g_i(x) - \nabla g_i(x)^T d ) \]

\[
\min_d \quad l(d)
\]

s.t. \quad \|d\|_\infty \leq \Delta^{LP}
Global convergence

\[
\min_{\alpha} m(\alpha d^{LP}), \quad \alpha \in [0, 1]
\]

\[
m(d) = l(d) + \frac{1}{2} d^T H d
\]

\[
x^C = x + \alpha^{LP} d^{LP}
\]
EQP Solution

Solve using Projected Conjugate Gradient method (GLTR)

\[
\min_d \quad \nabla f(x)^T d + \frac{1}{2} d^T Hd \\
\text{s.t.} \quad h_i(x) + \nabla h_i(x)^T d = 0, \quad i \in \mathcal{W} \cap E \\
\quad g_i(x) + \nabla g_i(x)^T d = 0, \quad i \in \mathcal{W} \cap I \\
\quad \|d\|_2 \leq \Delta
\]
Trial Step

\[ x_T = x_C + \alpha_2 (x_{EQP} - x_C) \]
Good News

1. SLIQUE seems to do a good job of identifying the active-set quickly

2. SLIQUE can solve many problems which have lots of degrees of freedom (DOF)

<table>
<thead>
<tr>
<th>DOF</th>
<th>#</th>
<th>SLIQUE</th>
<th>KNITRO</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000+</td>
<td>57</td>
<td>42</td>
<td>48</td>
</tr>
</tbody>
</table>

Problems with reduced space > 2000 at the solution.
## Bad News

**SLIQUE inefficient on large-scale problems**

<table>
<thead>
<tr>
<th>Size</th>
<th>#</th>
<th>SLIQUE</th>
<th>KNITRO</th>
</tr>
</thead>
<tbody>
<tr>
<td>VS</td>
<td>266</td>
<td>251 / 94%</td>
<td>251 / 94%</td>
</tr>
<tr>
<td>S</td>
<td>76</td>
<td>52 / 68%</td>
<td>67 / 88%</td>
</tr>
<tr>
<td>M</td>
<td>118</td>
<td>90 / 76%</td>
<td>103 / 87%</td>
</tr>
<tr>
<td>L</td>
<td>100</td>
<td>63 / 63%</td>
<td>85 / 85%</td>
</tr>
<tr>
<td>TOT</td>
<td>560</td>
<td>456</td>
<td>506</td>
</tr>
</tbody>
</table>

**VS:** \( n + m < 100 \)  \[ M: \] 1000 \( \leq n + m < 10000 \)

**S:** \( 100 \leq n + m < 1000 \)  \[ L: \] 10000 \( \leq n + m \)
Why?

SLIQUE timing Statistics

<table>
<thead>
<tr>
<th>Size</th>
<th>%LP</th>
<th>%EQP</th>
<th>%Fact</th>
<th>%Eval</th>
</tr>
</thead>
<tbody>
<tr>
<td>VS</td>
<td>10</td>
<td>16</td>
<td>5</td>
<td>48</td>
</tr>
<tr>
<td>S</td>
<td>36</td>
<td>16</td>
<td>13</td>
<td>19</td>
</tr>
<tr>
<td>M</td>
<td>43</td>
<td>33</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>L</td>
<td>49</td>
<td>35</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>TOT</td>
<td>41</td>
<td>30</td>
<td>7</td>
<td>13</td>
</tr>
</tbody>
</table>

- LP too expensive for large-scale problems.
- Relative LP cost grows as problem size grows
Why is LP so costly?

Warms starts inefficient:

1. Typically many LP trust-region constraints active (even near the solution).

2. Near solution active problem constraints stabilize, but active TR constraints may not!
Why is LP so costly?

Statistics:

1. On average about 45% of total LP iterations are to sort out trust-region constraints
   - Terminate LP early

2. On average active-set does not change over last 27% of outer iterations
   - Skip LP as we approach solution
Approximate LP

LP problem

\[
\min_{d} \quad l(d)
\]

s.t.
\[
\|d\|_{\infty} \leq \Delta_{LP}
\]

Truncate LP solution when

\[
l(0) - l(d) \geq \eta \left[ l(0) - l(d^{LP}) \right]
\]
Approximate LP

Condition

\[ l(0) - l(d) \geq \eta [l(0) - l(d_{LP})] \]

is satisfied when

\[ l(0) - l(d) \geq \frac{\eta}{(1 - \eta)} \text{gap}(d) \]

\[ \text{gap}(d) = \sum_{j \in S} |rc_j(2\Delta_{LP})| \]
# Approximate LP Results

Tested on 568 CUTEr problems

<table>
<thead>
<tr>
<th>Type</th>
<th>#</th>
<th>Ratio LP iters (inexact/exact)</th>
<th>Ratio Outer iters (inexact/exact)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC</td>
<td>110</td>
<td>0.33</td>
<td>4.74</td>
</tr>
<tr>
<td>QP</td>
<td>119</td>
<td>0.81</td>
<td>1.85</td>
</tr>
<tr>
<td>GC</td>
<td>339</td>
<td>0.81</td>
<td>2.52</td>
</tr>
<tr>
<td>TOT</td>
<td>568</td>
<td>0.70</td>
<td>2.87</td>
</tr>
</tbody>
</table>
Total LP iterations

Total LP Iterations

- exact LP
- inexact LP ($\eta=0.5$)
- inexact LP ($\eta=0.1$)
Total outer iterations

Total Outer Iterations

- **exact LP**
- **inexact LP ($\eta=0.5$)**
- **inexact LP ($\eta=0.1$)**
Skip LP

- Skip LP solve when the active set seems to have stabilized.

- Move into SEQP mode.

- How do you know when active-set has stabilized?

- How to recover if wrong active-set?
Summary/Conclusion

- SLIQUE robust overall
- Efficient for small/medium problems
- Better able to handle problems with large-reduced space
- Less efficient and robust for large-scale problems (vs. IP) because of warm start inefficiencies…
- But progress being made on using approximate LP solutions

PART 2: R. Byrd, Wed. 9:30 (306/34)