Recent Developments on (Practical) Optimization Methods for Convex and Nonconvex Optimization

Georgia Tech

November 10, 2022

Yinyu Ye
Stanford University and CUHKSZ (Sabbatical Leave)
(Currently Visiting IEDA HKUST)
Today’s Talk

• New developments of ADMM-based interior point (ABIP) Method

• Optimal Diagonal Preconditioner and HDSDP

• A Dimension Reduced Trust-Region Method

• A Homogeneous Second-Order Descent Method
ABIP (Lin, Ma, Zhang and Y, 2021)

- An ADMM (Glowinski and Marroco 75, He et al. 12, Monteiro and Svaiter 13) based interior point method solver for LP problems

\[
\begin{align*}
\text{min} & \quad c^T x \\
\text{s.t.} & \quad A x = b \\
x & \geq 0
\end{align*}
\]

\[
\begin{align*}
\text{max} & \quad b^T y \\
\text{s.t.} & \quad A^T y + s = c \\
s & \geq 0
\end{align*}
\]

- Consider homogeneous and self-dual (HSD) LP here!

\[
\begin{align*}
\text{min} & \quad \beta(n + 1)\theta + 1(r = 0) + 1(\xi = -n - 1) \\
\text{s.t.} & \quad Qu = v, \\
y & \text{free, } x \geq 0, \tau \geq 0, \theta \text{ free, } s \geq 0, \kappa \geq 0
\end{align*}
\]

where

\[
Q = \begin{bmatrix}
0 & A & -b & \bar{b} \\
-A^T & 0 & c & -\bar{c} \\
b^T & -c^T & 0 & \bar{z} \\
-\bar{b}^T & \bar{c}^T & -\bar{z} & 0
\end{bmatrix}, \quad u = \begin{bmatrix} y \\ x \\ \tau \\ \theta \end{bmatrix}, \quad v = \begin{bmatrix} r \\ s \\ \kappa \\ \xi \end{bmatrix}, \quad \bar{b} = b - Ae, \quad \bar{c} = c - e, \quad \bar{z} = c^T e + 1
\]
ABIP – Subproblem

• Introduce log-barrier function for HSD LP

\[
\min \quad B(u, v, \mu) \\
\text{s.t.} \quad Qu = v
\]

where \( B(u, v, \mu) \) barrier function

• Traditional IPM, one uses Newton’s method to solve the KKT system of the above problem, the cost is too expensive when problem is large!

• Now we apply ADMM to solve it inexactly

\[
\min \quad 1(Q\tilde{u} = \tilde{v}) + B(u, v, \mu^k) \\
\text{s.t.} \quad (\tilde{u}, \tilde{v}) = (u, v)
\]

The augmented Lagrangian function: only need to factorize a matrix once or find good diagonal preconditioners once

\[
\mathcal{L}_\beta(\tilde{u}, \tilde{v}, u, v, \mu^k, p, q) := 1(Q\tilde{u} = \tilde{v}) + B(u, v, \mu^k) - \langle \beta(p, q), (\tilde{u}, \tilde{v}) - (u, v) \rangle + \frac{\beta}{2} \| (\tilde{u}, \tilde{v}) - (u, v) \|^2
\]
ADMM Based Interior-Point (ABIP)+ Method (Deng et al. 2022)

- Different strategies/parameters may be significantly different among problems being solved
- An integration strategy based on decision tree is integrated into ABIP

\[
\begin{align*}
\min & \quad c^T x \\
\text{s.t.} & \quad Ax = b \\
& \quad x \geq 0
\end{align*}
\]

- A simple feature-to-strategy mapping is derived from a machine learning model
- For generalization limit the number of strategies (2 or 3 types)
ABIP – Restart Strategy I

- ABIP tends to induce a spiral trajectory
ABIP – Restart Strategy II

• After restart, ABIP moves more aggressively and converges faster (reduce almost 70% ADMM iterations)!
ABIP – Netlib

- Selected 105 Netlib instances

- $\epsilon = 10^{-6}$, use the direct method, $10^6$ max ADMM iterations

<table>
<thead>
<tr>
<th>Method</th>
<th># Solved</th>
<th># IPM</th>
<th># ADMM</th>
<th>Avg. Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABIP</td>
<td>65</td>
<td>74</td>
<td>265418</td>
<td>87.07</td>
</tr>
<tr>
<td>+ restart</td>
<td>68</td>
<td>74</td>
<td>88257</td>
<td>23.63</td>
</tr>
<tr>
<td>+ rescale</td>
<td>84</td>
<td>72</td>
<td>77925</td>
<td>20.44</td>
</tr>
<tr>
<td>+ hybrid $\mu$ (=ABIP+)</td>
<td>86</td>
<td>22</td>
<td>73738</td>
<td>14.97</td>
</tr>
</tbody>
</table>

- Hybrid $\mu$: If $\mu > \epsilon$ use the aggressive strategy, otherwise use another strategy

- ABIP+ decreases both # IPM iterations and # ADMM iterations significantly
ABIP – MIP2017

• 240 MIP2017 instances

• $\epsilon = 10^{-4}$, presolved by PaPILo, use the direct method, $10^6$ max ADMM iterations

<table>
<thead>
<tr>
<th>Method</th>
<th># Solved</th>
<th>SGM</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPT</td>
<td>240</td>
<td>1</td>
</tr>
<tr>
<td>PDLP (Julia)</td>
<td>202</td>
<td>17.4</td>
</tr>
<tr>
<td>ABIP</td>
<td>192</td>
<td>34.8</td>
</tr>
<tr>
<td>ABIP3+ Integration</td>
<td>212</td>
<td>16.7</td>
</tr>
</tbody>
</table>

• PDLP (Lu et al. 2021) is a practical first-order method (i.e., the primal-dual hybrid gradient (PDHG) method) for linear programming, and it enhances PDHG by a few implementation tricks.

• SGM stands for Shifted Geometric Mean, a standard measurement of solvers’ performance
ABIP – PageRank

• 117 instances, generated from sparse matrix datasets: DIMACS10, Gleich, Newman and SNAP. Second order methods in commercial solver fail in most of these instances.

• $\epsilon = 10^{-4}$, use the indirect method, 5000 max ADMM iterations.

<table>
<thead>
<tr>
<th>Method</th>
<th># Solved</th>
<th>SGM</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDLP(Julia)</td>
<td>122</td>
<td>1</td>
</tr>
<tr>
<td>ABIP3+</td>
<td>119</td>
<td>1.31</td>
</tr>
</tbody>
</table>

• Examples:

<table>
<thead>
<tr>
<th>Instance</th>
<th># nodes</th>
<th>PDLP (Julia)</th>
<th>ABIP3+</th>
</tr>
</thead>
<tbody>
<tr>
<td>amazon0601</td>
<td>403394</td>
<td>117.54</td>
<td>71.15</td>
</tr>
<tr>
<td>coAuthorsDBLP</td>
<td>299067</td>
<td>51.66</td>
<td>24.70</td>
</tr>
<tr>
<td>web-BerkStan</td>
<td>685230</td>
<td>447.68</td>
<td>139.75</td>
</tr>
<tr>
<td>web-Google</td>
<td>916428</td>
<td>293.01</td>
<td>148.18</td>
</tr>
</tbody>
</table>
ABIP – PageRank

• Generated by Google code

• When # nodes equals to # edges, the generated instance is a staircase matrix. For example,

```
-1.0000  0.1980  0  0  0  0  0  0  0  0  0
0.9900 -1.0000  0.4950  0.9900  0.4950  0.4950  0  0  0  0  0
0  0.1980 -1.0000  0  0  0  0  0  0  0  0
0  0.1980  0  -1.0000  0  0  0  0  0  0  0
0  0.1980  0  0  -1.0000  0  0  0.9900  0  0  0
0  0.1980  0  0  0  -1.0000  0  0  0.9900  0  0
0  0  0.4950  0  0  0  -1.0000  0  0  0.9900  0  0
0  0  0  0  0  0.4950  0  0  -1.0000  0  0  0
0  0  0  0  0  0  0.4950  0  0  -1.0000  0  0
0  0  0  0  0  0  0  0.4950  0  0  -1.0000  0
```

Staircase matrix instance (# nodes = 10)

• In this case, ABIP+ is significantly faster than PDLP!

<table>
<thead>
<tr>
<th># nodes</th>
<th>PDLP (Julia)</th>
<th>ABIP+</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^4$</td>
<td>8.60</td>
<td>0.93</td>
</tr>
<tr>
<td>$10^5$</td>
<td>135.67</td>
<td>10.36</td>
</tr>
<tr>
<td>$10^6$</td>
<td>2248.40</td>
<td>60.32</td>
</tr>
</tbody>
</table>
ABIP – Extension to Conic Linear Program

ABIP iteration remains valid for general conic linear program

\[
\min c^T x \\
\text{s.t. } Ax = b \\
x \in \mathcal{K}
\]

- ABIP-subproblem requires to solve a proximal mapping \( x^+ = \arg\min \lambda F(x) + \frac{1}{2} \|x - c\|^2 \) with respect to the log-barrier functions \( F(x) \) in \( B(u, v, \mu^k) \)

Positive orthant

- \( F(x) = -\log(x) \)
- \( x = \arg\min \lambda F(x) + \frac{1}{2} |x - c|^2 \)
- \( x = \frac{c + \sqrt{c^2 + 4\lambda}}{2} \)

Second-order cone

- \( F(x) = -\log(t^2 - \|x\|^2), x = (t; x) \)
- Can be solved by finding the root of quadratic functions

Positive semidefinite cone

- \( F(x) = -\log(\det x) \)
- Equivalent to solve \(-\lambda x^{-1} - c + x = 0\)
- Can be solved by eigen decomposition

- The total IPM and ADMM iteration complexities of ABIP for conic linear program are respectively:

\[
T_{IPM} = O \left( \log \left( \frac{1}{\varepsilon} \right) \right), \quad T_{ADMM} = O \left( \frac{1}{\varepsilon} \log \left( \frac{1}{\varepsilon} \right) \right)
\]
ABIP – Numerical results for large sparse SDPs (Joachim Dahl et al. 2022)

- Large sparse SDP problems from Mittelmann's library
- Relative tolerance $\epsilon = 10^{-6}$ used for stopping criteria

<table>
<thead>
<tr>
<th>Name</th>
<th>cone dim</th>
<th># constraints</th>
<th># iterations</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>theta12</td>
<td>600</td>
<td>17979</td>
<td>151</td>
<td>7s</td>
</tr>
<tr>
<td>theta102</td>
<td>500</td>
<td>37467</td>
<td>139</td>
<td>6s</td>
</tr>
<tr>
<td>theta123</td>
<td>600</td>
<td>90020</td>
<td>125</td>
<td>7s</td>
</tr>
<tr>
<td>hamming_8_3_4</td>
<td>256</td>
<td>16384</td>
<td>103</td>
<td>1s</td>
</tr>
<tr>
<td>hamming_9_5_6</td>
<td>512</td>
<td>53761</td>
<td>150</td>
<td>8s</td>
</tr>
<tr>
<td>fap09</td>
<td>174</td>
<td>30276</td>
<td>191</td>
<td>79s</td>
</tr>
</tbody>
</table>

(Performance on an AMD Ryzen 9 5900X Linux computer)
Summary

ABIP is

• a general purpose LP solver
• using ADMM to solve the subproblem
• developed with heuristics and intuitions from various strategies
• equipped with several new computational tricks
• Smart dual updates?
Today’s Talk

• New developments of ADMM-based interior point (ABIP) Method
• Optimal Diagonal Preconditioner and HDSDP
• A Dimension Reduced Trust-Region Method
• A Homogeneous Second-Order Descent Method
Interior point method for SDPs

SDP is solvable in polynomial time using the interior point methods

- Take Newton step towards the perturbed KKT system

$$\mathcal{A}X = b \quad \mathcal{A}X = b \quad \mathcal{A}\Delta X = -R_p$$
$$\mathcal{A}^* y + S = C \quad \mathcal{A}^* y + S = C \quad \mathcal{A}^* \Delta y + \Delta S = -R_D$$
$$XS = 0 \quad XS = \mu l \quad H_P(X \Delta S + \Delta XS) = -R_\mu$$

- Efficient numerical solvers have been developed

  COPT, Mosek, SDPT3, SDPA, DSDP…

- Most IPM solvers adopt primal-dual path-following IPMs except DSDP

  DSDP (Dual-scaling SDP) implements a dual potential reduction method
Homogeneous dual-scaling algorithm

From arbitrary starting dual solution \((y, S > 0, \tau > 0)\) with dual residual \(R\)

\[
\begin{align*}
\mathcal{A}X - b\tau &= 0 \\
-\mathcal{A}^*y + C\tau - S &= 0 \\
b^T y - \langle C, X \rangle - \kappa &= 0 \\
X = \mu S^{-1} &\quad \kappa = \mu \tau^{-1}
\end{align*}
\]

\[
\begin{align*}
\mathcal{A}(X + \Delta X) - b(\tau + \Delta \tau) &= 0 \\
-\mathcal{A}^*(y + \Delta y) + C(\tau + \Delta \tau) - (S + \Delta S) &= 0 \\
\mu S^{-1} \Delta S S^{-1} + \Delta X &= \mu S^{-1} - X \\
\mu \tau^{-2} \Delta \tau + \Delta \kappa &= \mu \tau^{-1} - \kappa
\end{align*}
\]

\[
\begin{pmatrix}
\mu M & -b - \mu A S^{-1} C S^{-1} \\
-b + \mu A S^{-1} C S^{-1} & -\mu(\langle C, S^{-1} C S^{-1} \rangle + \tau^{-2})
\end{pmatrix}
\begin{pmatrix}
\Delta y \\
\Delta \tau
\end{pmatrix} =
\begin{pmatrix}
b\tau \\
b^T y - \mu \tau^{-1}
\end{pmatrix} - \mu\begin{pmatrix}
\langle A S^{-1} \\
\langle C, S^{-1} \rangle
\end{pmatrix} + \mu\begin{pmatrix}
\langle A S^{-1} R S^{-1} \\
\langle C, S^{-1} R S^{-1} \rangle
\end{pmatrix}
\]

- Primal iterations can still be fully eliminated

- \(S = -\mathcal{A}^*y + C\tau - R\) inherits sparsity pattern of data
  
  Less memory and since \(X\) is generally dense

- Infeasibility or an early feasible solution can be detected via the embedding

New strategies are tailored for the method
Computational aspects for HDSDP Solver

To enhance performance, HDSDP (written in ANSI C) is equipped with

- Pre-solving that detects special structure and dependency
- Line-searches over barrier to balance optimality & centrality
- Heuristics to update the barrier parameter $\mu$
- Corrector strategy to reuse the Schur matrix
- A complete dual-scaling algorithm from DSDP5.8
- More delicate strategies for the Schur system
Computational results

• HDSDP is tuned and tested for many benchmark datasets
• Good performance on problems with both low-rank structure and sparsity
• Solve around 70/75 Mittelmann’s benchmark problems
• Solve 90/92 SDPLIB problems

<table>
<thead>
<tr>
<th>Instance</th>
<th>DSDP5.8</th>
<th>HDSDP</th>
<th>Mosek v9</th>
<th>SDPT3</th>
<th>COPT v5</th>
</tr>
</thead>
<tbody>
<tr>
<td>G40_mb</td>
<td>18</td>
<td>7</td>
<td>174</td>
<td>25</td>
<td>18</td>
</tr>
<tr>
<td>G48_mb</td>
<td>36</td>
<td>8</td>
<td>191</td>
<td>49</td>
<td>35</td>
</tr>
<tr>
<td>G48mc</td>
<td>11</td>
<td>2</td>
<td>71</td>
<td>24</td>
<td>18</td>
</tr>
<tr>
<td>G55mc</td>
<td>200</td>
<td>179</td>
<td>679</td>
<td>191</td>
<td>301</td>
</tr>
<tr>
<td>G59mc</td>
<td>347</td>
<td>246</td>
<td>646</td>
<td>256</td>
<td>442</td>
</tr>
<tr>
<td>G60_mb</td>
<td>700</td>
<td>213</td>
<td>7979</td>
<td>592</td>
<td>714</td>
</tr>
<tr>
<td>G60mc</td>
<td>712</td>
<td>212</td>
<td>8005</td>
<td>590</td>
<td>713</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance</th>
<th>DSDP5.8</th>
<th>HDSDP</th>
<th>Mosek v9</th>
<th>SDPT3</th>
<th>COPT v5</th>
</tr>
</thead>
<tbody>
<tr>
<td>checker1.5</td>
<td>87</td>
<td>41</td>
<td>72</td>
<td>71</td>
<td>81</td>
</tr>
<tr>
<td>foot</td>
<td>28</td>
<td>14</td>
<td>533</td>
<td>32</td>
<td>234</td>
</tr>
<tr>
<td>hand</td>
<td>4</td>
<td>2</td>
<td>76</td>
<td>8</td>
<td>40</td>
</tr>
<tr>
<td>ice_2.0</td>
<td>833</td>
<td>369</td>
<td>4584</td>
<td>484</td>
<td>1044</td>
</tr>
<tr>
<td>p_auss2</td>
<td>832</td>
<td>419</td>
<td>5948</td>
<td>640</td>
<td>721</td>
</tr>
<tr>
<td>r1_2000</td>
<td>17</td>
<td>8</td>
<td>333</td>
<td>20</td>
<td>187</td>
</tr>
<tr>
<td>torusg3-15</td>
<td>101</td>
<td>22</td>
<td>219</td>
<td>61</td>
<td>84</td>
</tr>
</tbody>
</table>

Selected Mittelmann’s benchmark problems where HDSDP is fastest (all the constraints are rank-one)
Optimal Diagonal Pre-Conditioner [QGHYZ 20]

Given matrix $M = X^T X > 0$, iterative method (e.g., CG) is often applied to solve

$$Mx = b$$

- Convergence of iterative methods depends on the condition number $\kappa(M)$
- Good performance needs pre-conditioning and we solve $P^{-1/2}MP^{-1/2}x' = b$
  - A good pre-conditioner reduces $\kappa(P^{-1/2}MP^{-1/2})$
- Diagonal $P = D$ is called diagonal pre-conditioner

More generally, we wish to find $D$ (or $E$) such that $\kappa(D \cdot X \cdot E)$ is minimized?

Is it possible to find optimal $D^*$ and $E^*$?

SDP works!
Finding the optimal diagonal pre-conditioner is an SDP
Two SDP blocks and sparse coefficient matrices
Trivial dual interior-feasible solution
An ideal formulation for dual SDP methods $D = \sum d_i e_i e_i^T$

What about two-sided?
Two-Sided Pre-Conditioner

\[
\min_{D_1 \succeq 0, D_2 \succeq 0} \kappa(D_1XD_2)
\]

- Common in practice and popular heuristics exist
e.g. Ruiz-scaling, matrix equilibration & balancing
- Not directly solvable using SDP
- Can be solved by *iteratively* fixing \(D_1(D_2)\) and optimizing the other side
  - Solving a sequence of SDPs
- Answer a question: how far can diagonal pre-conditioners go
Computational Results: Solving for the Optimal Pre-Conditioner

\[
\begin{align*}
\min_{D,\kappa} \quad & \kappa \\
\text{subject to} \quad & D \preceq M \\
\text{and} \quad & \kappa D \succeq M
\end{align*}
\] 

\[
\begin{align*}
\max_{\delta,d} \quad & \delta \\
\text{subject to} \quad & D - M \preceq 0 \\
\text{and} \quad & \delta M - D \succeq 0
\end{align*}
\]

SDP from optimal drag pre-conditioning problem

- Perfectly in the dual form
- Trivial dual feasible interior point solution
- 1 is an upper-bound for the optimal objective value

HDSDP

- A dual SDP algorithm (successor of DSDP5.8 by Benson)
- Support initial dual solution
- Customization for the diagonal pre-conditioner

<table>
<thead>
<tr>
<th>$n$</th>
<th>Sparsity</th>
<th>HDSDP (start from $(-10^6,0)$)</th>
<th>COPT</th>
<th>Mosek</th>
<th>SDPT3</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>0.05</td>
<td>7.1</td>
<td>6.8</td>
<td>9.1</td>
<td>18.0</td>
</tr>
<tr>
<td>1000</td>
<td>0.09</td>
<td>44.5</td>
<td>53.9</td>
<td>54.2</td>
<td>327.0</td>
</tr>
<tr>
<td>2000</td>
<td>0.002</td>
<td>34.3</td>
<td>307.1</td>
<td>374.7</td>
<td>572.3</td>
</tr>
<tr>
<td>5000</td>
<td>0.0002</td>
<td>64.3</td>
<td>&gt;1200</td>
<td>&gt;1200</td>
<td>&gt;1200</td>
</tr>
</tbody>
</table>
Computational results: Randomized preconditioner

• Many matrices result from statistical datasets
• \( M = X^T X \) estimates the covariance matrix
• It suffices to use a few samples to approximate

How few? As few as \( O(\log(\text{sample})) \)!

Experiment over regression datasets shows that

• It generally takes 1% to 5% of the samples to approximate well
• Scales well with dimension and saves much time for matrix-matrix multiplication
Computational Results: Optimal Diagonal Pre-Conditioner

- Test over 491 Suite Sparse Matrices of fewer than 1000 columns

<table>
<thead>
<tr>
<th>Reduction</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;80%</td>
<td>121</td>
</tr>
<tr>
<td>&gt;50%</td>
<td>190</td>
</tr>
<tr>
<td>&gt;20%</td>
<td>261</td>
</tr>
</tbody>
</table>

- Average reduction: 49.7%
- Better than diagonal: 36.0%
- Average time: 1.29

LIBSVM datasets

<table>
<thead>
<tr>
<th>Mat</th>
<th>Size</th>
<th>Chef</th>
<th>Caft</th>
<th>Reduce</th>
</tr>
</thead>
<tbody>
<tr>
<td>YearPredictionMSD</td>
<td>90</td>
<td>5233000.00</td>
<td>470.20</td>
<td>0.999910</td>
</tr>
<tr>
<td>YearPredictionMSD.t</td>
<td>90</td>
<td>5521000.00</td>
<td>359900.00</td>
<td>0.934816</td>
</tr>
<tr>
<td>abalone.scale.txt</td>
<td>8</td>
<td>2419.00</td>
<td>2038.00</td>
<td>0.157291</td>
</tr>
<tr>
<td>bodyfat.scale.txt</td>
<td>14</td>
<td>1281.00</td>
<td>669.10</td>
<td>0.477475</td>
</tr>
<tr>
<td>cadata.txt</td>
<td>8</td>
<td>8982000.00</td>
<td>7632.00</td>
<td>0.999150</td>
</tr>
<tr>
<td>cpusmall.scale.txt</td>
<td>12</td>
<td>20000.00</td>
<td>6325.00</td>
<td>0.683813</td>
</tr>
<tr>
<td>eunite2001.t</td>
<td>16</td>
<td>52450000.00</td>
<td>8530.00</td>
<td>0.999837</td>
</tr>
<tr>
<td>eunite2001.txt</td>
<td>16</td>
<td>67300000.00</td>
<td>3591.00</td>
<td>0.999947</td>
</tr>
<tr>
<td>housing.scale.txt</td>
<td>13</td>
<td>153.90</td>
<td>83.22</td>
<td>0.459371</td>
</tr>
<tr>
<td>mg.scale.txt</td>
<td>6</td>
<td>10.67</td>
<td>10.03</td>
<td>0.059988</td>
</tr>
<tr>
<td>mpg_scale.txt</td>
<td>7</td>
<td>142.50</td>
<td>107.20</td>
<td>0.247842</td>
</tr>
<tr>
<td>pyrim.scale.txt</td>
<td>27</td>
<td>49100000.00</td>
<td>3307.00</td>
<td>0.999933</td>
</tr>
<tr>
<td>space_ga_scale.txt</td>
<td>6</td>
<td>1061.00</td>
<td>729.60</td>
<td>0.312041</td>
</tr>
<tr>
<td>triazines.scale.txt</td>
<td>60</td>
<td>24580000.00</td>
<td>15460000.00</td>
<td>0.371034</td>
</tr>
</tbody>
</table>

Distribution of condition number reduction (Factor of improvement)
Summary

HDSDP is

• a general purpose SDP solver
• using dual-scaling and simplified HSD
• developed with heuristics and intuitions from DSDP
• equipped with several new computational tricks
• more iterative methods for solving subproblems
Today’s Talk

• New developments of ADMM-based interior point (ABIP) Method
• Optimal Diagonal Preconditioner and HDSDP
• A Dimension Reduced Trust-Region Method
• A Homogeneous Second-Order Descent Method
Early Complexity Analyses for Nonconvex Optimization

\[ \min f(x), x \in X \text{ in } \mathbb{R}^n, \]

- where \( f \) is nonconvex and twice-differentiable,

\[
g_k = \nabla f(x_k), H_k = \nabla^2 f(x_k)\]

- Goal: find \( x_k \) such that:

\[
\| \nabla f(x_k) \| \leq \epsilon \quad \text{(primary, first-order condition)}
\]

\[
\lambda_{\min}(H_k) \geq -\sqrt{\epsilon} \quad \text{(in active subspace, secondary, second-order condition)}
\]

- For the ball-constrained nonconvex QP: \( \min \ c^T x + 0.5 x^T Q x \ s.t. \| x \|_2 \leq 1 \)

\( O(\log\log(\epsilon^{-1})) \); see Y (1989,93), Vavasis&Zippel (1990)

- For nonconvex QP with polyhedral constraints: \( O(\epsilon^{-1}) \); see Y (1998), Vavasis (2001)
First-order Method (FOM): Gradient-Type Methods

• Assume $f$ has $L$-Lipschitz cont. gradient
• Global convergence by, e.g., linear-search (LS)
• No guarantee for the second-order condition
• Worst-case complexity, $O(\epsilon^{-2})$; see the textbook by Nesterov (2004)

Each iteration requires $O(n^2)$ operations
Second-order Method (SOM): Hessian-Type Methods

• Assume \( f \) has \( M \)-Lipschitz cont. Hessian
• Global convergence by, e.g., linear-search (LS), Trust-region (TR), or Cubic Regularization
• Convergence to second-order points
• No better than \( O(\epsilon^{-2}) \), for traditional methods (steepest descent and Newton); according to Cartis et al. (2010).

Each iteration requires \( O(n^3) \) operations
Analyses of SOM for general nonconvex optimization since 2000

Variants of SOM

• Trust-region with the fixed-radius strategy, $O(\epsilon^{-3/2})$, see the lecture notes by Y since 2005

• Cubic regularization, $O(\epsilon^{-3/2})$, see Nesterov and Polyak (2006), Cartis, Gould, and Toint (2011)

• A new trust-region framework, $O(\epsilon^{-3/2})$, Curtis, Robinson, and Samadi (2017)

With “slight” modification, complexity of SOM reduces from $O(\epsilon^{-2})$ to $O(\epsilon^{-3/2})$
Motivation from multi-directional FOM

• Two-directional FOM, with $d_k$ being the momentum direction $(x_k - x_{k-1})$

$$x_{k+1} = x_k - \alpha_k^1 \nabla f(x_k) + \alpha_k^2 d_k = x_k + d_{k+1}$$

where step-sizes are constructed; including CG, PT, AGD, Polyak, ADAM and many others.

• In SOM, a method typically minimizes a full dimensional quadratic Taylor expansion to obtain direction vector $d_{k+1}$. For example, one TR step solves for $d_{k+1}$ from

$$\min_d \ (g_k)^T d + 0.5d^T H_k d \quad s.t. \ ||d||_2 \leq \Delta_k$$

where $\Delta_k$ is the trust-region radius.

• DRSOM: Dimension Reduced Second-Order Method

Motivation: using few directions in SOM
The DRSOM in general uses $m$-independent directions

$$d(\alpha) = D_k \alpha, D_k \in \mathbb{R}^{nm}, \alpha \in \mathbb{R}^m$$

Plug the expression into the full-dimension TR quadratic minimization problem, we minimize a $m$-dimension trust-region subproblem to decide “$m$ stepsizes”:

$$\min m_k^\alpha(\alpha) := (c_k)^T \alpha + \frac{1}{2} \alpha^T Q_k \alpha$$

$$||\alpha||_{G_k} \leq \Delta_k$$

$$G_k = D_k^T D_k, Q_k = D_k^T H_k D_k, c_k = (g_k)^T D_k$$

How to choose $D_k$? How great would $m$ be? Rank of $H_k$? (Randomized) rank reduction of a symmetric matrix to log($n$) (So et al. 08)?
DRSOM II

• In following, as an example, DRSOM adopts two FOM directions

\[ d = -\alpha^1 \nabla f(x_k) + \alpha^2 d_k := d(\alpha) \]

where \( g_k = \nabla f(x_k) \), \( H_k = \nabla^2 f(x^k) \), \( d_k = x_k - x_{k-1} \)

• Then we minimize a 2-D trust-region problem to decide “two step-sizes”:

\[
\min m_k^\alpha (\alpha) := f(x_k) + (c_k)^T \alpha + \frac{1}{2} \alpha^T Q_k \alpha \\
\|\alpha\|_{G_k} \leq \Delta_k \]

\[ G_k = \begin{bmatrix} g_k^T g_k & -g_k^T d_k \\ -g_k^T d_k & d_k^T d_k \end{bmatrix}, \\ Q_k = \begin{bmatrix} g_k^T H_k g_k & -g_k^T H_k d_k \\ -g_k^T H_k d_k & d_k^T H_k d_k \end{bmatrix}, \\ c_k = \begin{bmatrix} -\|g_k\|^2 \\ g_k^T d_k \end{bmatrix} \]
DRSOM can be seen as:

- “Adaptive” **Accelerated Gradient Method** (Polyak’s momentum 60)
- A second-order method minimizing quadratic model in the reduced 2-D

\[
m_k(d) = f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2 f(x_k) d, \quad d \in \text{span}\{-g_k, d_k\}
\]

compare to, e.g., Dogleg method, 2-D Newton **Trust-Region Method**

\[
d \in \text{span}\{g_k, [H(x_k)]^{-1} g_k\} \quad \text{(e.g., Powell 70)}
\]

- A conjugate direction method for convex optimization exploring the **Krylov Subspace** (e.g., Yuan&Stoer 95)

- For convex quadratic programming with no radius limit, terminates in n steps
Computing Hessian-Vector Product in DRSOM is the Key

In the DRSOM with two directions:

\[ Q_k = \begin{bmatrix} \langle g_k^T H_k g_k, d_k \rangle & -\langle g_k^T H_k d_k, d_k \rangle \\ -\langle g_k^T H_k d_k, d_k \rangle & \langle d_k^T H_k d_k, d_k \rangle \end{bmatrix}, c_k = \begin{bmatrix} -||g_k||^2 \\ g_k^T d_k \end{bmatrix} \]

How to cheaply obtain Q? Compute \( H_k g_k, H_k d_k \) first.

• Finite difference:

\[ H_k \cdot v \approx \frac{1}{\epsilon} \left[ g(x_k + \epsilon \cdot v) - g_k \right], \]

• Analytic approach to fit modern automatic differentiation,

\[ H_k g_k = \nabla \left( \frac{1}{2} g_k^T g_k \right), H_k d_k = \nabla \left( d_k^T g_k \right), \]

• or use Hessian if readily available!
DRSOM: key assumptions and theoretical results (Zhang at al. SHUFE)

**Assumption.** (a) $f$ has Lipschitz continuous Hessian. (b) DRSOM iterates with a fixed-radius strategy: $\Delta_k = \epsilon/\beta$ \hspace{1cm} (c) If the Lagrangian multiplier $\lambda_k < \sqrt{\epsilon}$, assume
\[
\| (H_k - \tilde{H}_k) d_{k+1} \| \leq C \| d_{k+1} \|^2 \quad \text{(Cartis et al.)},
\]
where $\tilde{H}_k$ is the projected Hessian in the subspace (commonly adopted for approximate Hessian).

**Theorem 1.** If we apply DRSOM to QP, then the algorithm terminates in at most $n$ steps to find a first-order stationary point.

**Theorem 2.** (Global convergence rate) For $f$ with second-order Lipschitz condition, DRSOM terminates in $O(\epsilon^{-3/2})$ iterations. Furthermore, the iterate $x_k$ satisfies the first-order condition, and the Hessian is positive semi-definite in the subspace spanned by the gradient and momentum.

**Theorem 3.** (Local convergence rate) If the iterate $x_k$ converges to a strict local optimum $x^*$ such that $H(x^*) > 0$, and if **Assumption (c)** is satisfied as soon as $\lambda_k \leq C_\lambda \| d_{k+1} \|$, then DRSOM has a local superlinear (quadratic) speed of convergence, namely:
\[
\| x_{k+1} - x^* \| = O(\| x_k - x^* \|^2)
\]
Sensor Network Location (SNL)

• Consider Sensor Network Location (SNL)

\[ N_x = \{(i, j) : \|x_i - x_j\| = d_{ij} \leq r_d\}, \quad N_a = \{(i, k) : \|x_i - a_k\| = d_{ik} \leq r_d\} \]

where \( r_d \) is a fixed parameter known as the radio range. The SNL problem considers the following QCQP feasibility problem,

\[ \|x_i - x_j\|^2 = d_{ij}^2, \forall (i, j) \in N_x \]

\[ \|x_i - a_k\|^2 = d_{ik}^2, \forall (i, k) \in N_a \]

• We can solve SNL by the nonconvex nonlinear least square (NLS) problem

\[
\min_X \sum_{(i<j,j) \in N_x} (\|x_i - x_j\|^2 - d_{ij}^2)^2 + \sum_{(k,j) \in N_a} (\|a_k - x_j\|^2 - d_{k,j}^2)^2.
\]
Sensor Network Location (SNL)

- Graphical results using SDP relaxation to initialize the NLS
- $n = 80$, $m = 5$ (anchors), radio range = 0.5, degree = 25, noise factor = 0.05
- Both Gradient Descent and DRSOM can find good solutions!
Sensor Network Location (SNL)

- Graphical results without SDP relaxation
- DRSOM can still converge to optimal solutions
Neural Networks and Deep Learning

To use DRSOM in machine learning problems

- We apply the mini-batch strategy to a vanilla DRSOM
- Use Automatic Differentiation to compute gradients
- Train ResNet18 Model with CIFAR 10
- Set Adam with initial learning rate 1e-3
Neural Networks and Deep Learning

Training results for ResNet18 with DRSOM and Adam

Pros

- DRSOM has rapid convergence (30 epochs)
- DRSOM needs little tuning

Cons

- DRSOM may overfit the models
- Needs 4~5x time than Adam to run same number of epoch

Test results for ResNet18 with DRSOM and Adam

Good potential to be a standard optimizer for deep learning!
TRPO attempts to optimize a surrogate function (based on the current iterate) of the objective function while keep a KL divergence constraint

\[
\begin{align*}
\max_\theta & \quad L_{\theta_k}(\theta) \\
\text{s.t.} & \quad \text{KL} \left( \text{Pr}_{\mu}^{\pi_{\theta_k}} \parallel \text{Pr}_{\mu}^{\pi_{\theta}} \right) \leq \delta
\end{align*}
\]

In practice, it linearizes the surrogate function, quadratizes the KL constraint, and obtain

\[
\begin{align*}
\max_\theta & \quad g_k^T (\theta - \theta_k) \\
\text{s.t.} & \quad \frac{1}{2} (\theta - \theta_k)^T F_k (\theta - \theta_k) \leq \delta
\end{align*}
\]

where $F_k$ is the Hessian of the KL divergence.
DRSOM/TRPO Preliminary Results I

- Although we only maintain the linear approximation of the surrogate function, surprisingly the algorithm works well in some RL environments.
DRSOM/TRPO Preliminary Results II

- Sometimes even better than TRPO!
DRSOM for LP Potential Reduction (Gao et al. SHUFE)

We consider a simplex-constrained QP model

\[
\begin{align*}
\min_x & \quad \frac{1}{2} \|Ax\|^2 =: f(x) \\
\text{subject to} & \quad e^\top x = 1 \\
& \quad x \geq 0
\end{align*}
\]

We wish to solve a standard LP (and its dual)

\[
\begin{align*}
\min_x & \quad c^\top x \\
\text{subject to} & \quad Ax = b \\
& \quad x \geq 0
\end{align*}
\]

subject to

\[
\begin{align*}
\frac{\nu}{2} b^\top y - c^\top x &= 0 \\
A^\top y + s &= c \\
s &\geq 0
\end{align*}
\]

The self-dual embedding builds a bridge

- The homogeneous QP seems so restrictive!
- How to solve much more general LPs?

The self-dual embedding builds a bridge and apply DRSOM to it

\[
\phi(x) := \rho \log(f(x)) - \sum_{i=1}^n \log x_i
\]

\[
\nabla \phi(x) = \frac{\rho \nabla f(x)}{f(x)} - X^{-1} e
\]

Combined with scaled gradient(Hessian) projection, the method solves LPs
DR-Potential Reduction: Preliminary Results

One feature of the DR-Potential reduction is the use of negative curvature of

\[
\nabla^2 \phi(x) = -\frac{\rho \nabla f(x) \nabla f(x)^T}{f(x)^2} + \rho \frac{A^T A}{f(x)} + X^{-2}
\]

- Computable using Lanczos iteration
- Getting LPs to high accuracy \(10^{-6} \sim 10^{-8}\) if negative curvature is efficiently computed
- Now solving small and medium Netlib instances in 10 seconds within 1000 iterations
- In MATLAB and getting transferred into C for acceleration

<table>
<thead>
<tr>
<th>Problem</th>
<th>Plnfeas</th>
<th>Dlnfeas</th>
<th>Compl.</th>
<th>Problem</th>
<th>Plnfeas</th>
<th>Dlnfeas</th>
<th>Compl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADLITLLE</td>
<td>1.347e-10</td>
<td>2.308e-10</td>
<td>2.960e-09</td>
<td>KB2</td>
<td>5.455e-11</td>
<td>6.417e-10</td>
<td>7.562e-11</td>
</tr>
<tr>
<td>AFIRO</td>
<td>7.641e-11</td>
<td>7.375e-11</td>
<td>3.130e-10</td>
<td>LOTFI</td>
<td>2.164e-09</td>
<td>4.155e-09</td>
<td>8.663e-08</td>
</tr>
<tr>
<td>AGG2</td>
<td>3.374e-08</td>
<td>4.895e-08</td>
<td>6.286e-07</td>
<td>MODSZK1</td>
<td>1.527e-06</td>
<td>5.415e-05</td>
<td>2.597e-04</td>
</tr>
<tr>
<td>AGG3</td>
<td>2.248e-05</td>
<td>1.151e-06</td>
<td>1.518e-05</td>
<td>RECIPELP</td>
<td>5.868e-08</td>
<td>6.300e-08</td>
<td>1.285e-07</td>
</tr>
<tr>
<td>BANDM</td>
<td>2.444e-09</td>
<td>4.886e-09</td>
<td>3.769e-08</td>
<td>SC105</td>
<td>7.315e-11</td>
<td>5.970e-11</td>
<td>2.435e-10</td>
</tr>
<tr>
<td>BEACONFD</td>
<td>5.765e-12</td>
<td>9.853e-12</td>
<td>1.022e-10</td>
<td>SC205</td>
<td>6.392e-11</td>
<td>5.710e-11</td>
<td>2.650e-10</td>
</tr>
<tr>
<td>BLEND</td>
<td>2.018e-10</td>
<td>3.729e-10</td>
<td>1.179e-09</td>
<td>SC50A</td>
<td>1.078e-05</td>
<td>6.098e-06</td>
<td>4.279e-05</td>
</tr>
<tr>
<td>BOEING2</td>
<td>1.144e-07</td>
<td>1.110e-08</td>
<td>2.307e-07</td>
<td>SC50B</td>
<td>4.647e-11</td>
<td>3.269e-11</td>
<td>1.747e-10</td>
</tr>
<tr>
<td>BORE3D</td>
<td>2.389e-08</td>
<td>5.013e-08</td>
<td>1.165e-07</td>
<td>SCAGR25</td>
<td>1.048e-07</td>
<td>5.298e-08</td>
<td>1.289e-06</td>
</tr>
<tr>
<td>BRANDY</td>
<td>2.702e-05</td>
<td>7.818e-06</td>
<td>1.849e-05</td>
<td>SCAGR7</td>
<td>1.087e-07</td>
<td>1.173e-08</td>
<td>2.601e-07</td>
</tr>
<tr>
<td>CAPRI</td>
<td>7.575e-05</td>
<td>4.488e-05</td>
<td>4.880e-05</td>
<td>SCFXM1</td>
<td>4.323e-06</td>
<td>5.244e-06</td>
<td>8.681e-06</td>
</tr>
<tr>
<td>E226</td>
<td>2.656e-06</td>
<td>4.742e-06</td>
<td>2.512e-05</td>
<td>SCORPION</td>
<td>1.674e-09</td>
<td>1.892e-09</td>
<td>1.737e-08</td>
</tr>
<tr>
<td>FINNIS</td>
<td>8.577e-07</td>
<td>8.367e-07</td>
<td>1.001e-05</td>
<td>SCTAP1</td>
<td>5.567e-07</td>
<td>8.430e-07</td>
<td>5.081e-06</td>
</tr>
<tr>
<td>FORPLAN</td>
<td>5.874e-07</td>
<td>2.084e-07</td>
<td>4.979e-06</td>
<td>SEBA</td>
<td>2.919e-11</td>
<td>5.729e-11</td>
<td>1.448e-10</td>
</tr>
<tr>
<td>GFRD-PNC</td>
<td>4.558e-05</td>
<td>1.052e-05</td>
<td>4.363e-05</td>
<td>SHARE1B</td>
<td>3.367e-07</td>
<td>1.339e-06</td>
<td>3.578e-06</td>
</tr>
<tr>
<td>ISRAEL</td>
<td>1.422e-06</td>
<td>1.336e-06</td>
<td>1.404e-05</td>
<td>STAIR</td>
<td>5.549e-04</td>
<td>8.566e-06</td>
<td>2.861e-05</td>
</tr>
<tr>
<td>STANDATA</td>
<td>5.645e-08</td>
<td>2.735e-07</td>
<td>5.130e-06</td>
<td>STANDGUB</td>
<td>2.934e-08</td>
<td>1.467e-07</td>
<td>2.753e-06</td>
</tr>
<tr>
<td>STOCFO1</td>
<td>6.633e-09</td>
<td>9.701e-09</td>
<td>4.811e-08</td>
<td>VTP-BASE</td>
<td>1.349e-10</td>
<td>5.098e-11</td>
<td>2.342e-10</td>
</tr>
</tbody>
</table>
DRSOM for Riemannian Optimization (Tang et al. NUS)

\[
\min_{x \in \mathcal{M}} f(x)
\]

(ROP)

- $\mathcal{M}$ is a Riemannian manifold embedded in Euclidean space $\mathbb{R}^n$.
- $f : \mathbb{R}^n \to \mathbb{R}$ is a second-order continuously differentiable function that is lower bounded in $\mathcal{M}$.

**R-DRSOM:** Choose an initial point $x_0 \in \mathcal{M}$, set $k = 0$, $p_{-1} = 0$;
for $k = 0, 1, \ldots, T$ do

**Step 1.** Compute $g_k = \nabla f(x_k)$, $d_k = T_{x_k \leftarrow x_{k-1}}(p_{k-1})$, $H_k g_k = \text{Hess} f(x_k)[g_k]$ and $H_k d_k = \text{Hess} f(x_k)[d_k]$;

**Step 2.** Compute the vector $c_k = \begin{bmatrix} -\langle g_k, g_k \rangle_{x_k} \\ \langle g_k, d_k \rangle_{x_k} \end{bmatrix}$ and the following matrices

\[
Q_k = \begin{bmatrix}
\langle g_k, H_k g_k \rangle_{x_k} & \langle -d_k, H_k g_k \rangle_{x_k} \\
\langle -d_k, H_k d_k \rangle_{x_k} & \langle d_k, H_k d_k \rangle_{x_k}
\end{bmatrix}, \quad G_k := \begin{bmatrix}
\langle g_k, g_k \rangle_{x_k} & -\langle d_k, g_k \rangle_{x_k} \\
-\langle d_k, g_k \rangle_{x_k} & \langle d_k, d_k \rangle_{x_k}
\end{bmatrix}.
\]

**Step 3.** Solve the following 2 by 2 trust region subproblem with radius $\triangle_k > 0$

\[
\alpha_k := \arg \min_{\|\alpha\|_{G_k} \leq \triangle_k} f(x_k) + c_k^\top \alpha + \frac{1}{2} \alpha^\top Q_k \alpha;
\]

**Step 4.** $x_{k+1} := \mathcal{R}_{x_k}(x_k - \alpha_k^1 g_k + \alpha_k^2 d_k)$;

end

Return $x_k$. 
Max-CUT SDP

**Max-Cut:** \( \min \left\{ -\langle L, X \rangle : \text{diag}(X) = e, \ X \in \mathbb{S}_+^n \right\} \). \hspace{1cm} (1)

\[
\min \left\{ -\left\langle L, RR^\top \right\rangle : \text{diag}(RR^\top) = e, \ R \in \mathbb{R}^{n \times r} \right\}.
\] \hspace{1cm} (2)

<table>
<thead>
<tr>
<th>g67</th>
<th>Fval</th>
<th>Residue</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>n=10000</td>
<td>-30977.7</td>
<td>131.0</td>
<td>177.8</td>
</tr>
<tr>
<td>m=20000</td>
<td>-30977.7</td>
<td>1371.4</td>
<td>114.4</td>
</tr>
<tr>
<td>g70</td>
<td>Fval</td>
<td>Residue</td>
<td>Time [s]</td>
</tr>
<tr>
<td>n=10000</td>
<td>-39446.1</td>
<td>2.2e-10</td>
<td>36.2</td>
</tr>
<tr>
<td>m=9999</td>
<td>-39446.1</td>
<td>1371.4</td>
<td>114.4</td>
</tr>
<tr>
<td>g72</td>
<td>Fval</td>
<td>Residue</td>
<td>Time [s]</td>
</tr>
<tr>
<td>n=10000</td>
<td>-31234.2</td>
<td>8.2e-11</td>
<td>110.4</td>
</tr>
<tr>
<td>m=20000</td>
<td>-31234.2</td>
<td>881.2</td>
<td>907.5</td>
</tr>
<tr>
<td>g77</td>
<td>Fval</td>
<td>Residue</td>
<td>Time [s]</td>
</tr>
<tr>
<td>n=14000</td>
<td>-44182.7</td>
<td>7.8e-11</td>
<td>268.3</td>
</tr>
<tr>
<td>m=28000</td>
<td>-44182.7</td>
<td>1576.9</td>
<td>2402.6</td>
</tr>
<tr>
<td>g81</td>
<td>Fval</td>
<td>Residue</td>
<td>Time [s]</td>
</tr>
<tr>
<td>n=20000</td>
<td>-62624.8</td>
<td>4.6e-11</td>
<td>650.1</td>
</tr>
<tr>
<td>m=40000</td>
<td>-62624.8</td>
<td>4283.9</td>
<td>6087.4</td>
</tr>
</tbody>
</table>
1D-Kohn-Sham Equation

$$\min \left\{ \frac{1}{2} \text{tr}(R^\top LR) + \frac{\alpha}{4} \text{diag}(RR^\top) \right\} L^{-1} \text{diag}(RR^\top) : \ R^\top R = I_p, \ R \in \mathbb{R}^{n \times r} \right\}, \quad (3)$$

where $L$ is a tri-diagonal matrix with 2 on its diagonal and -1 on its subdiagonal and $\alpha > 0$ is a parameter. We terminate algorithms when $\|\text{grad} f(R)\| < 10^{-4}$.

**Figure 1:** Results for Discretized 1D Kohn-Sham Equation. $\alpha = 1$. 
Today’s Talk

- New developments of ADMM-based interior point (ABIP) Method
- Optimal Diagonal Preconditioner and HDSDP
- A Dimension Reduced Trust-Region Method
- A Homogeneous Second-Order Descent Method
A Descent Direction Using the Homogenized Quadratic Model I

- Big Question: How to drop Assumption (c) in DRSOM analyses?

Recall the classical trust-region method minimizes the quadratic model

\[
\begin{align*}
\min_{d \in \mathbb{R}^n} m_k(d) & := g_k^T d + \frac{1}{2} d^T H_k d \\
\text{s.t.} \|d\| & \leq \Delta_k.
\end{align*}
\]

- \(-g_k\) is the first-order steepest descent direction but ignores Hessian; the direction of \(H_k\)-negative curvature \(v\) meets Assumption (c) and also enables \(O(\epsilon^{1.5})\) decrease if

\[
R(H_k, v) = v^T H_k v / \|v\|^2 < -\sqrt{\epsilon},
\]

but such direction does not exist if it becomes nearly convex...

- Could we construct a direction integrating both?

**Answer:** Use the homogenized quadratic model!
A Descent Direction Using the Homogenized Quadratic Model II

• Using the homogenization trick by lifting with extra scalar $t$:

$$\psi_k(\xi_0, t; \delta) := \frac{1}{2} \begin{bmatrix} \xi_0 \\ t \end{bmatrix}^T \begin{bmatrix} H_k & g_k \\ g_k^T & -\delta \end{bmatrix} \begin{bmatrix} \xi_0 \\ t \end{bmatrix} = \frac{t^2}{2} \begin{bmatrix} \xi_0/t \\ 1 \end{bmatrix}^T \begin{bmatrix} H_k & g_k \\ g_k^T & -\delta \end{bmatrix} \begin{bmatrix} \xi_0/t \\ 1 \end{bmatrix}$$

• The homogeneous model is equivalent to $m_k$ up to scaling:

$$\psi_k(\xi_0, t; \delta) = t^2 \cdot (m_k(\xi_0/t) - \delta)$$

• Find a good direction $\xi = \xi_0/t$ (if $t = 0$ then set $t=1$) by the leftmost eigenvector:

$$\min_{||(\xi_0; t)|| \leq 1} \psi_k(\xi_0, t; \delta)$$

• Accessible at the cost of $O(\varepsilon^{-1/4})$ via the randomized Lanczos method.
This is the Classical Homogenization Trick in QCQP via SDP

- For inhomogeneous QP (and QCQP):

\[ \begin{align*}
\min & \quad x^T Q_0 x - 2b_0^T x \\
\text{s.t.} & \quad x^T Q_{i} x - 2b_{i}^T x + c_{i} \leq 0, \quad i = 1, \ldots, m
\end{align*} \quad \leftrightarrow \quad \begin{align*}
\min & \quad x^T Q_0 x - 2b_0^T x t \\
\text{s.t.} & \quad x^T Q_{i} x - 2b_{i}^T x t + c_{i} t^2 \leq 0, \quad i = 1, \ldots, m \\
& \quad t^2 = 1
\end{align*} \]

- Used with SDP relaxation:

\[ \begin{align*}
\min & \quad M_0 \cdot X \\
\text{s.t.} & \quad M_{i} \cdot X \leq 0, \quad i = 1, \ldots, m \\
& \quad X_{00} = 1, X \geq 0
\end{align*} \quad \leftrightarrow \quad M_i = \begin{bmatrix} c_i & b_i^T \\ b_i & Q_i \end{bmatrix}, \quad X = \begin{bmatrix} 1 & x^T \\ x^T & X_0 \end{bmatrix} \]

- Homogenized QCQP and SDP relaxation enables strong performance and theoretical analysis, and it guarantees a rank-one solution if \( m=1 \).

* Rojas and Sorensen 2001
The Descent Direction Using the Homogenized Quadratic Model

- Define the following parametrized \( (\delta) \) homogenized quadratic model at \( x_k \):

\[
\psi_k(\xi_0, t; \delta) := \frac{1}{2} \begin{bmatrix} \xi_0 \\ t \end{bmatrix}^T \begin{bmatrix} H_k & g_k \\ g_k^T & -\delta \end{bmatrix} \begin{bmatrix} \xi_0 \\ t \end{bmatrix} = \frac{t^2}{2} \begin{bmatrix} \xi_0/t \\ 1 \end{bmatrix}^T \begin{bmatrix} H_k & g_k \\ g_k^T & -\delta \end{bmatrix} \begin{bmatrix} \xi_0/t \\ 1 \end{bmatrix}
\]

- The “un-homogenized vector” \( \xi = \xi_0/t \) can be found by the leftmost eigenvalue computation and scaling (if \( t = 0 \) then set \( t=1 \)) ;

- Lemma 1 (strict negative curvature) : if \( g_k \neq 0, H_k \neq 0 \), let \( \lambda_1 \) be the leftmost eigenvalue of \( \begin{bmatrix} H_k & g_k \\ g_k^T & -\delta \end{bmatrix} \), then \( \lambda_1 \leq -\delta \).

- The motivates us to use \( \xi \) as a second-order descent direction resulting a single-looped (easy-to-implement) method
Theoretical Guarantees of HSODM

• Consider use the second-order homogenized direction, and the length of each step $\|\eta \xi\|$ is fixed: $\|\eta \xi\| \leq \Delta_k = \frac{2\sqrt{\varepsilon}}{M}$ where $f(x)$ has $L$-Lipschitz gradient and $M$-Lipschitz Hessian.

• Theorem 1 (Global convergence rate) : if $f(x)$ satisfies the Lipchitz Assumption and $\delta = \sqrt{\varepsilon}$, the iterate moves along homogeneous vector $\xi$: $x_{k+1} = x_k + \eta_k \xi$, then, if we choose $\eta_k = \Delta_k / \|\xi\|$, and terminate at $\|\xi\| < \Delta_k$, then algorithm has $O(\varepsilon^{-3/2})$ iteration complexity. Furthermore, $x_{k+1}$ satisfies approximate first-order and second-order conditions.
Global Convergence Rate: Outline of Analysis

- A concise analysis using fixed radius $\Delta$

| Let $x_{k+1} = x_k + \eta \xi$, $R(H_k, \xi) = \xi^T H_k \xi / \|\xi\|^2$, $\xi = \xi_0 / t$ |
| (sufficient decrease in large step) If $\|\xi\| \geq \Delta$, we choose $\eta = \Delta / \|\xi\|$ |
| $f(x_{k+1}) - f(x_k) \leq -\frac{\delta \Delta^2}{2} + \frac{M}{6} \Delta^3$, regardless of $t = 0$ or not |
| $\delta$ must be some greater than $O(\sqrt{\epsilon})$ to have $O(\epsilon^{\frac{3}{2}})$ decrease |
| (small step means convergence) Otherwise $\|\xi\| < \Delta$, then we choose step-size $\eta = 1$ and |
| $\|g_{k+1}\| \leq 4(L + \delta)^2 \Delta^3 + \frac{M}{2} \Delta^2 + (2L\delta + 2\delta^2) \Delta$ |
| $\delta$ must be some less than $O(\sqrt{\epsilon})$ and converge |

* The eigenvector does not change, and we do not have to solve $\xi$ again.
Theoretical Guarantees of HSODM (cont.)

• **Theorem 2** (Local convergence rate): If the iterate $x_k$ of HSODM converges to a strict local optimum $x^*$ such that $H(x^*) > 0$, and then $\eta_k = 1$ if $k$ is sufficiently large. If we do not terminate HSODM and set $\delta = 0$, then HSODM has a local superlinear (quadratic) speed of convergence, namely: $\|x_{k+1} - x^*\| = O(\|x_k - x^*\|^2)$

• The local convergence property of HSODM is very similar to classical trust-region method when the iterate becomes unconstrained Newton steps
Preliminary results: HSODM and DRSOM + HSODM

\[ \frac{1}{2} ||Ax - b||^2 + ||x||^p, p = 0.5, A \in R^{100 \times 300}, \text{nnz} = 0.5 \]

An example of L2-Lp

- **GD** and **LBFGS** both use a Line-search (Hager-Zhang)
- **DRSOM** uses 2-D subspace
- **HSODM** and **DRSOM + HSODM** are much better!
- **DRSOM** can also benefit from the homogenized system
The Effect of Warm-Starting the Eigenvector

Convex QP: $Q \in S^{200\times 200}$

- $\epsilon = 10^{-8}$ Lanczos iterations

An example of warm starting
- $\text{HSODM(warm)}$ uses the last eigenvector to warm start the Lanczos method
- $\text{HSODM(warm)}$ always needs less subproblem iter than $\text{HSODM(cold)}$
Ongoing Research and Future Directions on DRSOM

• Are there other alternatives to remove Assumption c) in DRSOM analyses?

• **Low-rank approximation** of the homogenized matrix
  \[
  \begin{bmatrix}
  H_k & g_k \\
  g_k^T & 0
  \end{bmatrix} (+\mu \cdot I, \text{ that is, adding sufficiently large scalar } \mu \text{ so that it is positive definite if necessary}) \text{ to make the leftmost eigenvector computing easier (Randomized rank reduction of a symmetric matrix to } \log(n), \text{ So et al. 08) and “Hot-Start” eigenvector computing by Power Methods (linear convergence of Liu et al. 2017)?}
  \]

• **Indefinite and Randomized** Hessian rank-one updating via BFGS/SR1

• **Dimension Reduced** Non-Smooth/Semi-Smooth Newton

**Takeaway:** Second-Order Information matters and better to integrate FOM and SOM!

• **THANK YOU**