Primal-dual interior-point methods for linear optimization problems

Erling D. Andersen
MOSEK ApS
Email: e.d.andersen@mosek.com
WWW: http://www.mosek.com

September 15, 2009
Introduction
1940s: Linear optimization i.e.

\[
(P) \quad \text{minimize} \quad c^T x \\
\text{subject to} \quad Ax = b, \\
\quad x \geq 0,
\]

is invented by Dantzig (and Von Neuman).

1947: The simplex method was invented.

1984: Kamarkar presents a polynomial time interior-point method.

1984-1999:

- A large amount of work on interior-point methods is performed.
- Implementations of the simplex alg. is improved a lot.

1999-2009: LO is employed extensively.

2009: You have learned about the simplex method.

2009: You will learn about interior-point methods.
What is MOSEK

- A software package for solving large-scale optimization problems.
- Solves linear, conic, and nonlinear convex problems.
- Has mixed-integer capabilities.
- Stand-alone as well as embedded.
- Used to solve problems with up to millions of constraints and variables.
- Version 1 released in 1999.
- See www.mosek.com for further info.
Customers of MOSEK

- Financial institutions such as banks and investment funds.
- Companies/governments managing forest.
- Chip designers.
- Public transport companies.
  - Trapeze.
- ISVs such as Energy Exemplar.
- TV Commercial scheduling.
Overview

- The basics of an interior-point method.
- Implementation specific details.
- Existing IPM software.
- Some computational results.
  - Comparison with the simplex algorithm.
Primal-dual interior-point methods
Notation.

The primal-dual algorithm (the infeasible variant).

◆ A homogeneous model.
◆ Mehrotra’s predictor-corrector method.
◆ Further enhancements.
◆ Linear algebra issues (The Cholesky factorization).

Other issues.
Primal problem:

\[(P) \quad \text{minimize} \quad c^T x \]
\[\text{subject to} \quad Ax = b, \quad x \geq 0,\]

(m equalities, n variables).

Dual problem:

\[(D) \quad \text{maximize} \quad b^T y \]
\[\text{subject to} \quad A^T y + s = c, \quad s \geq 0.\]
The primal-dual algorithm

Derivation summary:
- **Step 1:** Remove the inequalities from (P) using a barrier term.
- **Step 2:** State the Lagrange optimality conditions.
- **Step 3:** Apply Newton’s method to the optimality conditions.
Step 1 - Introducing the barrier

(PB) minimize $c^T x - \rho \sum_j \ln(x_j)$
subject to $Ax = b.$

Notes:

- $\rho$ is a positive (barrier) parameter.
- $\lim_{x \to 0} \rho \ln(x) = -\infty.$
- What is the relation between (P) and (PB)?
  - Feasibility?
  - Optimality?
- Could $\ln(x)$ be replaced by another function?
The Lagrange function:

\[ L(x, y) := c^T x - \rho \sum_j \ln(x_j) - y^T (Ax - b) \]

where \( y \) is the Lagrange multipliers.

Given

\[
\begin{align*}
\min & \quad x_1 \\
\text{s.t.} & \quad 2x_1 = 3, \\
& \quad x_1 \geq 0
\end{align*}
\]

then

1. State the Lagrange function.
2. State the optimality conditions.
Optimality conditions:

\[
\nabla_x L(x, y) = c - \rho X^{-1}e - A^T y = 0,
\]

\[
\nabla_y L(x, y) = Ax - b = 0.
\]

where

\[
X := \text{diag}(x) := \begin{bmatrix}
x_1 & 0 & 0 & 0 \\
0 & x_2 & 0 & 0 \\
0 & 0 & \ddots & 0 \\
0 & 0 & \cdots & x_n \\
\end{bmatrix}, \quad e := \begin{bmatrix}
1 \\
1 \\
\vdots \\
\vdots \\
1 \\
\end{bmatrix}.
\]
Let
\[ s = \rho X^{-1}e \]
and hence
\[ Xs = \rho e. \]

Equivalent optimality conditions
\[
\begin{align*}
(O) & : Ax = b, \quad x > 0, \\
& : A^T y + s = c, \quad s > 0, \\
& : Xs = \rho e.
\end{align*}
\]

Observe this implies
\[ x_j s_j = \rho. \]

- What is the interpretation of the optimality conditions?
- How does the optimality conditions relate to the optimality conditions for \((P)\)?
Step 3 - solving the optimality conditions

How to solve the optimality conditions.

- They are nonlinear.
- Hence apply Newton’s method:

\[ \nabla f(x^k) d_x = f(x^k), \]
\[ x^{k+1} = x^k + \alpha d_x. \]

where \( \alpha \in ]0, 1] \) is step size. Solves \( f(x) = 0 \).

Define:

\[ F_\gamma(x, y, s) := \begin{bmatrix} Ax - b \\ A^T y + s - c \\ Xs - \gamma \mu e \end{bmatrix}, \quad \rho := \gamma \mu = \gamma x^T s/n. \]

\( (\gamma \geq 0 \) is a parameter to be chosen).
Given \((x^0, s^0) > 0\) then one step of Newton’s method applied to

\[ F_\gamma(x, y, s) = 0, \quad x, s \geq 0 \]

is given by

\[
\nabla F_\gamma(x^0, y^0, s^0) \begin{bmatrix} d_x \\ d_y \\ d_s \end{bmatrix} = -F_\gamma(x^0, y^0, s^0).
\]

and

\[
\begin{bmatrix} x^1 \\ y^1 \\ s^1 \end{bmatrix} := \begin{bmatrix} x^0 \\ y^0 \\ s^0 \end{bmatrix} + \alpha \begin{bmatrix} d_x \\ d_y \\ d_s \end{bmatrix}.
\]

\(\alpha \in (0, 1]\) is a step-size.
The complete primal-dual algorithm

1. Choose \((x^0, y^0, s^0)\) such that \(x^0, s^0 > 0\).
2. Choose \(\gamma, \theta \in (0, 1), \varepsilon > 0\)
3. \(k := 0\)
4. while max\(|Ax^k - b|, |A^T y^k + s^k - c|, (x^k)^T s^k\) \(\geq \varepsilon\)
5. \(\mu^k := ((x^k)^T s^k)/n\)
6. Solve:
   \[
   \begin{align*}
   Ad_x &= -(Ax^k - b), \\
   A^T d_y + d_s &= -(A^T y^k + s^k - c), \\
   S^k d_x + X^k d_s &= -X^k s^k + \gamma \mu^k e,
   \end{align*}
   \]
7. Compute:
   \[
   \alpha^k := \theta \max\{\bar{\alpha} : x^k + \bar{\alpha}d_x \geq 0, s^k + \bar{\alpha}d_s \geq 0, \theta \bar{\alpha} \leq 1\}
   \]
8. \((x^{k+1}; y^{k+1}; s^{k+1}) := (x^k; y^k; s^k) + \alpha^k (d_x; d_y; d_s)\)
9. \(k := k + 1\)
10. end while
\[
\begin{bmatrix}
Ax^1 - b \\
ATy^1 + s^1 - c
\end{bmatrix} = (1 - \alpha) \begin{bmatrix}
Ax^0 - b \\
ATy^0 + s^0 - c
\end{bmatrix}
\]

and
\[
(x^1)^T s^1 = (1 - (1 - \gamma)\alpha)(x^0)^T s^0 + \alpha^2 d^T_x d_s.
\]

Given $\alpha > 0$ and $\gamma \in [0, 1)$:

- Residuals are reduced.
- $(x^1)^T s^1 < (x^0)^T s^0$ for sufficiently $\alpha$ small.
- **Difficulty**: $d^T_x d_s$ is not under control.
- Always interior: $x, s > 0$. 
Observations

- Fairly simple algorithm.
- Insensitive to degeneration.
- Few but computational expensive iterations.
- What about infeasible or unbounded problems?
- Theoretical convergence analysis is messy.
The homogeneous model

A homogeneous and self-dual model:

\[
Ax - b\tau = 0, \quad x \geq 0,
\]
\[
A^T y + s - c\tau = 0, \quad s \geq 0,
\]
\[
-c^T x + b^T y - \kappa = 0, \quad \tau, \kappa \geq 0.\]

Facts:

- A homogeneous LP.
- Always has a solution (0).
- Always has a SCS solution i.e.

\[
x^*_j s^*_j = 0 \quad \text{and} \quad x^*_j + s^*_j > 0, \quad j = 1, \ldots, n,
\]
\[
\tau^* \kappa^* = 0 \quad \text{and} \quad \tau^* + \kappa^* > 0.
\]
Let \((x^*, \tau^*, y^*, s^*, \kappa^*)\) be any SCS then

- \(\tau^* > 0\) in the **feasible** case: \((x^*, y^*, s^*)/\tau^*\) is an optimal solution to \((P)\).

- \(\kappa^* > 0\) in the **infeasible** case:

\[
\begin{align*}
Ax^* &= 0, \\
A^T y^* + s^* &= 0, \\
-c^T x^* + b^T y^* &= \kappa^* > 0.
\end{align*}
\]

If \(c^T x^* < 0\), then

\[
\min c^T x \quad \text{s.t.} \quad Ax = 0, \quad x \geq 0
\]

is unbounded implying dual infeasibility. \((b^T y^* > 0\) implies primal infeasibility.\)

**Conclusion:** Compute a SCS solution to \((HLF)\) using an IPM.
Choose \((x^0, \tau^0, y^0, s^0, \kappa^0)\) such that \((x^0, \tau^0, s^0, \kappa^0) > 0, \varepsilon > 0, \) and \(\theta, \gamma \in (0, 1). \) 

2. \(\text{while} \ (x^k)^T s^k + \tau^k \kappa^k > \varepsilon \)

3. Solve

\[
\begin{align*}
Ad_x - bd_\tau &= (1 - \gamma)(b \tau^k - Ax^k), \\
A^T d_y + d_s - cd_\tau &= (1 - \gamma)(c \tau^k - A^T y^k - s^k), \\
-c^T d_x + b^T d_y - d_\kappa &= (1 - \gamma)(\kappa^k + c^T x^k - b^T y^k), \\
S^k d_x + X^k d_s &= -X^k s^k + \gamma \mu^k e, \\
\kappa^k d_\tau + \tau^k d_\kappa &= -\tau^k \kappa^k + \gamma \mu^k.
\end{align*}
\]

4. \(\alpha := \text{stepsizes}(x^k; \tau^k; s^k; \kappa^k), (d_x; d_\tau; d_s; d_\kappa), \theta). \)

5. 

\[
\begin{align*}
(x^{k+1}; \tau^{k+1}) &:= (x^k; \tau^k) + \alpha(d_x; d_\tau), \\
(y^{k+1}; s^{k+1}; \kappa^{k+1}) &:= (y^k; s^k; \kappa^k) + \alpha(d_y; d_s; d_\kappa)
\end{align*}
\]

6. \(k := k + 1\)

7. \(\text{end while}\)
Summary for homogeneous model

- Fairly easy to prove polynomial convergence $O(n^{3.5}L)$.
- Works in the primal and dual infeasible cases.
- Slightly more expensive per iteration than the primal-dual algorithm.
- Can be generalized to symmetric cone optimization.
Problem:
\[
\max \quad 0 y_1 - 1 y_2 \quad \text{st.} \quad y_1^2 + y_2^2 \leq 1.
\]

Discretized:
\[
\max \quad 0 y_1 - 1 y_2 \\
\text{st.} \quad \cos\left(\frac{2j\pi}{n}\right)y_1 + \sin\left(\frac{2j\pi}{n}\right)y_2 \geq -1, \\
\quad \quad \quad \quad \quad j = 1, \ldots, n.
\]
Results from a simple implementation

<table>
<thead>
<tr>
<th>n</th>
<th>Iter.</th>
<th>$y_1$</th>
<th>$y_2$</th>
<th>$y_1^2 + y_2^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>0.0000</td>
<td>-1.0515</td>
<td>1.1056</td>
</tr>
<tr>
<td>100</td>
<td>12</td>
<td>0.0000</td>
<td>-1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>500</td>
<td>13</td>
<td>0.0000</td>
<td>-1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>1000</td>
<td>14</td>
<td>0.0000</td>
<td>-1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>5000</td>
<td>14</td>
<td>0.0000</td>
<td>-1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>10000</td>
<td>16</td>
<td>0.0000</td>
<td>-1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>50000</td>
<td>17</td>
<td>0.0000</td>
<td>-1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>100000</td>
<td>18</td>
<td>0.0000</td>
<td>-1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>250000</td>
<td>18</td>
<td>0.0000</td>
<td>-1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>500000</td>
<td>19</td>
<td>0.0000</td>
<td>-1.0000</td>
<td>1.0000</td>
</tr>
</tbody>
</table>
Mehrotra's predictor-corrector method

Problem: Better choice of $\gamma$.

Define $(d^a_x, d^a_y, d^a_s)$ by

\[
\begin{align*}
Ad^a_x &= -(Ax^k - b), \\
A^T d^a_y + d^a_s &= -(A^T y^k + s^k - c), \\
S^k d^a_x + X^k d^a_s &= -X^k s^k.
\end{align*}
\]

Let

$$\hat{\alpha} \equiv \text{step-size}((x^k; s^k), (d^a_x; d^a_s), 1).$$

Reduction for $\gamma = 0$:

$$1 - \hat{\alpha}.$$

Heuristic choice:

$$\hat{\gamma} \equiv (1 - \hat{\alpha})^2 \min(0.1, 1 - \hat{\alpha}).$$
Want to solve

\[(x_j^k + d_{x_j})(s_j^k + d_{s_j}) = \gamma \mu^k\]

implies

\[x_j^k d_{s_j} + s_j^k d_{x_j} = -x_j^k s_j^k - d_{x_j} d_{s_j} + \gamma \mu^k.\]

Mehrotra’s high-order estimate:

\[d_{x_j} d_{s_j} = d^\tau d^\kappa.\]

“Final” direction:

\[
\begin{align*}
Ad_x &= -(Ax^k - b), \\
A^T d_y + d_s &= -(A^T y^k + s^k - c), \\
S^k d_x + X^k d_s &= -X^k s^k + \hat{\gamma} \mu^k e - D^a_x d^a_s,
\end{align*}
\]

where \(D^a_x = \text{diag}(d^a_x)\).
Summary for Mehrotra’s predictor-corrector method

- A high-order method.
- Reuses a matrix factorization of the Newton equations system.
- Increases the number of solves by 1.
- Reduces the number of iterations significantly (> 20%).
- Is a heuristic.
The Newton equations system:

\[
\begin{bmatrix}
    A & 0 & 0 \\
    0 & A^T & I \\
    S & 0 & X
\end{bmatrix}
\begin{bmatrix}
    d_x \\
    d_y \\
    d_s
\end{bmatrix} =
\begin{bmatrix}
    \hat{r}_p \\
    \hat{r}_d \\
    \hat{r}_{xs}
\end{bmatrix}.
\]

Therefore,

\[
d_s = X^{-1}(\hat{r}_{xs} - Sd_x).
\]

Hence,

\[
A^T d_y + X^{-1}(\hat{r}_{xs} - Sd_x) = \hat{r}_d,
\]

\[
Ad_x = \hat{r}_p.
\]

Leading to

\[
S^{-1}(X A^T d_y + \hat{r}_{xs}) - d_x = S^{-1} X \hat{r}_d
\]
i.e.

\[ d_x = S^{-1}(X A^T d_y - \hat{r}_{xs}) - S^{-1} X \hat{r}_d \]

But

\[ A d_x = A(S^{-1}(X A^T d_y + \hat{r}_{xs}) - S^{-1} X \hat{r}_d) \]

and finally we reach at

\[ A S^{-1} X A^T d_y = \hat{r}_p - A S^{-1}(\hat{r}_{xs} - X \hat{r}_d) \]

or

\[ M d_y = \ldots \]

where

\[ M := A(S)^{-1} X A^T = ADA^T = \sum_{j=1}^{n} \frac{x_j}{s_j} A_{:j} A_{:j}^T. \]
The Cholesky factorization

- \( M \) is symmetric and positive definite.
- A Cholesky decomposition \( (L) \) exists
  \[
  M = LL^T.
  \]

Notes:
- Works if \( M \) is positive definite.
- \( \frac{1}{6}m^3 + O(m^2) \) complexity.
- Cholesky = Gaussian elimination using diagonal pivots.
- Numerically stable without pivoting.
- Problem: \( M \) is only P.S.D. occasionally.
Modified algorithm:

1. for $j = 1, \ldots, m$
2. if $l_{jj} \leq \varepsilon$
3. $l_{jj} := \delta$
4. $l_{jj} := \sqrt{l_{jj}}$
5. $l_{(j+1:m)j} := l_{(j+1:m)j} / l_{jj}$
6. for $k = j + 1, \ldots, m$
7. $l_{(k+1:m)k} := l_{(k+1:m)k} - l_{kj} l_{(k+1:m)j}$

- **Choice:** $\varepsilon = 1.0e - 12$, $\delta = 1.0e30$.
- **Corresponds to removing dependent rows in** $A(XS^{-1})^{\frac{1}{2}}$.
- **Analyzed by** Y. Zhang and S. Wright.
Observations:

- $A$ is very sparse in practice.
- $M$ is usually very sparse.
- $L$ is usually very sparse.
- Only nonzeros in $L$ are stored.
- Sparsity pattern of $M$ and $L$ is constant over all iterations.
An example

\[
M = \begin{bmatrix}
x & x & x & x & x & x \\
x & x & x \\
x & x \\
x & x \\
x & x \\
x & x \\
\end{bmatrix}
\]

Notes:
- Pivot order is important for **fill-in** and **work**.
- \( M \) is represented by an undirected graph.

Ordering methods:
- (Multiple) minimum-degree (George and Liu; Liu).
- Minimum-local fill (better but is expensive).
New ordering methods

- Approximate minimum degree (Amestoy, Davis and Duff).
- Approximate minimum local fill (Mészáros; Rothberg; Rothberg and Eisenstat).
- Graph partitioning (Kumar et al.; Hendrickson and Rothberg; Gupta).

\[
M = \begin{bmatrix}
M_{11} & 0 & M_{31}^T \\
0 & M_{22} & M_{32}^T \\
M_{31} & M_{32} & M_{33}
\end{bmatrix}.
\]

(used recursively).
Summary for the normal equation system

- **Iteration 0:**
  - Find sparsity pattern of $AA^T$.
  - Choose a sparsity preserving ordering.
  - Find sparsity pattern of $L$.

- **At iteration k:**
  - Form $M = ADA^T$.
  - Factorize $M$.
  - Do solves.
Efficient implementation of Cholesky computation

- Exploit hardware cache.
- Do loop unrolling.
- Can be implemented efficiently for shared memory parallel.
- Dense columns in $A$ leads to inefficiency.

$$M = \sum_j \frac{x_j}{s_j} A_{:j} A_{:j}^T$$
Basis identification

- **Problem:** An optimal basic and nonbasic partition of the variables is required.
- **Reasons:**
  - Easy sensitivity analysis.
  - Integer programming.
  - Efficient warm-start.

An example:

\[
\begin{aligned}
\text{minimize} & \quad e^T x \\
\text{subject to} & \quad e^T x \geq 1, \quad x \geq 0.
\end{aligned}
\]

Basic sol.: \( x^* = (0, \ldots, 0, 1, 0, \ldots, 0) \).

IP sol.: \( x^* = (1/n, \ldots, 1/n) \).
Summary for basis identification

- Has a primal and dual phase (symmetric).
- Requires at most $n$ simplex type pivots.
- May need some simplex clean-up iterations.
- Implementation can exploit problem structure to gain computational efficiency.
- Combined approach leads to a highly reliably optimization package.
Computational results
Taken from Mittlemans benchmark results (23 aug 2008).
See http://plato.la.asu.edu/bench.html.
Problem size: Up to a million constraints and variable.
The results

CPU times are user times in secs including input and crossover to a feasible basis for all codes except LOQO and LIPSOL. "$\$" without crossover. LOQO has no presolver; sigfig=6 was used for it.

<table>
<thead>
<tr>
<th>problem</th>
<th>CPLEX-B</th>
<th>CPLEX-D/P</th>
<th>MOSEK-B</th>
<th>MOSEK-D/P</th>
<th>LOQO</th>
<th>LIPSOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>cont1</td>
<td>1445</td>
<td>948/911</td>
<td>6427</td>
<td>1593/1069</td>
<td>89</td>
<td>766</td>
</tr>
<tr>
<td>cont11</td>
<td>913</td>
<td>32767/7580</td>
<td>871</td>
<td>36023/4052</td>
<td>183</td>
<td>1047</td>
</tr>
<tr>
<td>cont4</td>
<td>1754</td>
<td>826/499</td>
<td>256</td>
<td>6167/1766</td>
<td>933</td>
<td>304</td>
</tr>
<tr>
<td>cont1_L</td>
<td>$ 289</td>
<td></td>
<td>914</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cont11_L</td>
<td>$ 7599</td>
<td></td>
<td>940</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dano3mip</td>
<td>10</td>
<td>19/9</td>
<td>7</td>
<td>27/22</td>
<td>71</td>
<td>8</td>
</tr>
<tr>
<td>dbic1</td>
<td>32</td>
<td>47/7</td>
<td>32</td>
<td>303/160</td>
<td>95</td>
<td>30</td>
</tr>
<tr>
<td>df1001</td>
<td>9</td>
<td>8/13</td>
<td>7</td>
<td>23/28</td>
<td>112</td>
<td>8</td>
</tr>
<tr>
<td>fome12</td>
<td>141</td>
<td>48/174</td>
<td>27</td>
<td>191/972</td>
<td>463</td>
<td>30</td>
</tr>
<tr>
<td>fome13</td>
<td>45</td>
<td>216/339</td>
<td>50</td>
<td>370/1784</td>
<td>786</td>
<td>55</td>
</tr>
<tr>
<td>gen4</td>
<td>20</td>
<td>1/39</td>
<td>3067</td>
<td>5/124</td>
<td>21</td>
<td>33</td>
</tr>
<tr>
<td>ken-18</td>
<td>5</td>
<td>5/28</td>
<td>8</td>
<td>12/38</td>
<td>52</td>
<td>13</td>
</tr>
<tr>
<td>l130</td>
<td>19</td>
<td>9/140</td>
<td>1</td>
<td>613/49</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>lp22</td>
<td>3</td>
<td>14/35</td>
<td>4</td>
<td>40/125</td>
<td>41</td>
<td>7</td>
</tr>
<tr>
<td>mod2</td>
<td>6</td>
<td>27/77</td>
<td>7</td>
<td>47/319</td>
<td>49</td>
<td>12</td>
</tr>
<tr>
<td>neos</td>
<td>67</td>
<td>13/70</td>
<td>77</td>
<td>1369/68</td>
<td>684</td>
<td>86</td>
</tr>
<tr>
<td>neos1</td>
<td>16</td>
<td>331/8</td>
<td>13</td>
<td>24/3</td>
<td>146</td>
<td>17</td>
</tr>
<tr>
<td>neos2</td>
<td>12</td>
<td>245/15</td>
<td>11</td>
<td>43/3</td>
<td>81</td>
<td>15</td>
</tr>
<tr>
<td>neos3</td>
<td>100</td>
<td>1406/</td>
<td>165</td>
<td>7635/21</td>
<td>1490</td>
<td>187</td>
</tr>
<tr>
<td>ns1687037</td>
<td>&gt;75000</td>
<td>35938/16620</td>
<td>143</td>
<td>/</td>
<td>193</td>
<td></td>
</tr>
<tr>
<td>ns1688926</td>
<td>89</td>
<td>33/563</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nsct2</td>
<td>43</td>
<td>1/1</td>
<td>27</td>
<td>2/3</td>
<td>854</td>
<td>77</td>
</tr>
</tbody>
</table>
Introduction
Primal-dual interior-point methods
Computational results
Details
The results
Conclusions
Further information

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Number</th>
<th>Iterations</th>
<th>Solver Time</th>
<th>Wall Time</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>nug15</td>
<td>4</td>
<td>44</td>
<td>1541/704</td>
<td>52</td>
<td>6497/2566</td>
<td>1580</td>
</tr>
<tr>
<td>nug20</td>
<td>2</td>
<td>767</td>
<td>/39000</td>
<td>920</td>
<td>/60453</td>
<td>22856</td>
</tr>
<tr>
<td>nug08-3rd</td>
<td>2</td>
<td>751</td>
<td>1392/1</td>
<td>820</td>
<td>4402/1</td>
<td>6112</td>
</tr>
<tr>
<td>pds-40</td>
<td>2</td>
<td>82</td>
<td>14/53</td>
<td>66</td>
<td>37/422</td>
<td>15682</td>
</tr>
<tr>
<td>pds-100</td>
<td>2</td>
<td>461</td>
<td>66/522</td>
<td>439</td>
<td>376/8627</td>
<td>618</td>
</tr>
<tr>
<td>qap12</td>
<td>3</td>
<td>5</td>
<td>69/45</td>
<td>8</td>
<td>227/236</td>
<td>176</td>
</tr>
<tr>
<td>qap15</td>
<td>3</td>
<td>43</td>
<td>1543/599</td>
<td>71</td>
<td>8342/3557</td>
<td>1739</td>
</tr>
<tr>
<td>rail4284</td>
<td>2</td>
<td>136</td>
<td>2820/2333</td>
<td>148</td>
<td>3691/291</td>
<td>2654</td>
</tr>
<tr>
<td>rlfprim</td>
<td>4</td>
<td>2</td>
<td>1/3</td>
<td>2</td>
<td>5/1</td>
<td>56</td>
</tr>
<tr>
<td>self</td>
<td>8</td>
<td>34</td>
<td>111/49</td>
<td>3072</td>
<td>198/240</td>
<td>34</td>
</tr>
<tr>
<td>spal_004$</td>
<td>2</td>
<td>8</td>
<td>2/1</td>
<td>6</td>
<td>3/3</td>
<td>fail</td>
</tr>
<tr>
<td>stat96v1</td>
<td>4</td>
<td>314</td>
<td>133/238</td>
<td>33</td>
<td>6205/460</td>
<td>156</td>
</tr>
<tr>
<td>stat96v4</td>
<td>4</td>
<td>4</td>
<td>146/266</td>
<td>7</td>
<td>194/295</td>
<td>12</td>
</tr>
<tr>
<td>storm-125</td>
<td>6</td>
<td>11</td>
<td>7/6</td>
<td>27</td>
<td>27/137</td>
<td>21</td>
</tr>
<tr>
<td>storm_1000</td>
<td>2</td>
<td>205</td>
<td>361/571</td>
<td>348</td>
<td>2926/14521</td>
<td>457</td>
</tr>
<tr>
<td>stp3d</td>
<td>1</td>
<td>97</td>
<td>315/4579</td>
<td>91</td>
<td>1772/16743</td>
<td>2460</td>
</tr>
<tr>
<td>watson_2</td>
<td>2</td>
<td>25</td>
<td>74/224</td>
<td>27</td>
<td>116/428</td>
<td>206</td>
</tr>
<tr>
<td>world</td>
<td>4</td>
<td>6</td>
<td>33/123</td>
<td>10</td>
<td>55/415</td>
<td>46</td>
</tr>
</tbody>
</table>

==================================================================
Conclusions
Summary

- Interior-point methods are stable and fast.
- Implementations are mature.
- Even public domain codes are quite good.
- For cold start and large models interior-point methods tend dominate the simplex methods.
Observations about interior-point methods

- Highly reliable using default options.
  - Insensitive to degeneration.
  - Insensitive to the problem size.

- Few but expensive iterations.

- No generally efficient warm-start is known (at least to my knowledge).

- Formulation:
  - Search direction is a function of $c$, $A$, and $b$.
  - Avoid large numbers.
Further information
An unfinished book: