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

History

- 1940s: Linear optimization i.e.

$$\begin{array}{lll} (P) & \text{minimize} & c^T x \\ & \text{subject to} & Ax = b, \\ & & x \geq 0, \end{array}$$

  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.
- Version 6 released July 2009.
- See www.mosek.com for further info.

Customers of MOSEK

- Financial institutions such as banks and investment funds.
- Companies/goverments managing forrest.
- 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

http://www.mosek.com

Outline

- 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.

Notation

Primal problem:

$$
\begin{array}{lll}
(P) & \text{minimize} & c^T x \\
& \text{subject to} & Ax = b, \\
& & x \geq 0,
\end{array}
$$

(m equalities, n variables).
Dual problem:

$$
\begin{array}{lll}
(D) & \text{maximize} & b^T y \\
& \text{subject to} & A^T y + s = c, \\
& & s \geq 0.
\end{array}
$$

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.

mosek
http://www.mosek.com

Step 1 - Introducing the barrier

$$
\begin{array}{lll}
(PB) & \text{minimize} & c^T x - \rho \sum_j \ln(x_j) \\
& \text{subject to} & Ax = b.
\end{array}
$$

Notes:

- ρ 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

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

$$\begin{array}{llll} \min & x_1 & & \\ s.t. & 2x_1 & = & 3, \\ & x_1 \geq 0 & & \end{array} \tag{1}$$

then

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

Step 2 - the Lagrange optimality conditions

Optimality conditions:

$$\begin{array}{rcccl} \nabla_x L(x,y) & = & c - \rho X^{-1} e - A^T y & = & 0, \\ \nabla_y L(x,y) & = & Ax - b & = & 0. \end{array}$$

where

$$X := \text{diag}(x) := \begin{bmatrix} x_1 & 0 & 0 & 0 \\ 0 & x_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & x_n \end{bmatrix}, \quad e := \begin{bmatrix} 1 \\ 1 \\ \vdots \\ \vdots \\ 1 \end{bmatrix}.$$

Let

$$s = \rho X^{-1} e$$

and hence

$$Xs = \rho e.$$

Equivalent optimality conditions

$$(O) \qquad \begin{array}{rcll} Ax & = & b, & x > 0, \\ A^T y + s & = & c, & s > 0, \\ Xs & = & \rho e. & \end{array}$$

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:

$$\begin{array}{rcl} \nabla f(x^k) d_x & = & f(x^k), \\ x^{k+1} & = & x^k + \alpha d_x. \end{array}$$

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 $\max(\left\|Ax^k - b\right\|, \left\|A^T y^k + s^k - c\right\|, (x^k)^T s^k) \geq \varepsilon$
5. $\mu^k := ((x^k)^T s^k)/n$
6. Solve:
$$\begin{aligned} Ad_x &= -(Ax^k - b), \\ A^T d_y + d_s &= -(A^T y^k + s^k - c), \\ S^k d_x + X^k d_s &= -X^k s^k + \gamma \mu^k e, \end{aligned}$$
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

$$\begin{bmatrix} Ax^1 - b \\ A^T y^1 + s^1 - c \end{bmatrix} = (1-\alpha) \begin{bmatrix} Ax^0 - b \\ A^T y^0 + s^0 - c \end{bmatrix}$$

and

$$(x^1)^T s^1 = (1 - (1-\gamma)\alpha)(x^0)^T s^0 + \alpha^2 d_x^T d_s.$$

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

- Residuals are reduced.
- $(x^1)^T s^1 < (x^0)^T s^0$ for sufficiently α small.
- Difficulty: $d_x^T d_s$ 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:

$$
\begin{array}{rcll}
Ax - b\tau & = & 0, & x \geq 0, \\
A^T y + s - c\tau & = & 0, & s \geq 0, \qquad (HLF) \\
-c^T x + b^T y - \kappa & = & 0, & \tau, \kappa \geq 0.
\end{array}
$$

Facts:

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

$$
\begin{array}{lll}
x_j^* s_j^* = 0 & \text{and} & x_j^* + s_j^* > 0, \quad j = 1, \ldots, n, \\
\tau^* \kappa^* = 0 & \text{and} & \tau^* + \kappa^* > 0.
\end{array}
$$

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:

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

If $c^T x^* < 0$, then

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

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

Conclusion: Compute a SCS solution to (HLF) using an IPM.

The homogeneous algorithm

1. 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 $\theta, \gamma \in (0,1)$. k := 0.
2. while $(x^k)^T s^k + \tau^k \kappa^k > \varepsilon$
3. Solve

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

4. $\alpha :=$ stepsize$((x^k; \tau^k; s^k; \kappa^k), (d_x; d_\tau; d_s; d_\kappa), \theta)$.
5.

$$\begin{array}{rcl} (x^{k+1}; \tau^{k+1}) & := & (x^k; \tau^k) + \alpha(d_x; d_\tau), \\ (y^{k+1}; s^{k+1}; \kappa^{k+1}) & := & (y^k; s^k; \kappa^k) + \alpha(d_y; d_s; d_\kappa) \end{array}$$

6. $k := k+1$
7. 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.

Powell's example (Math. Prog., 93, no. 1)

Problem:

$$\max \quad 0y_1 - 1y_2 \quad \text{st.} \quad y_1^2 + y_2^2 \leq 1.$$

Discretized:

$$\begin{array}{ll} \max & 0y_1 - 1y_2 \\ \text{st.} & \cos(2j\pi/n)y_1 + \sin(2j\pi/n)y_2 \geq -1, \\ & \quad j = 1, \ldots, n. \end{array}$$

Results from a simple implementation

n	Iter.	y_1	y_2	$y_1^2 + y_2^2$
10	10	0.0000	-1.0515	1.1056
100	12	0.0000	-1.0000	1.0000
500	13	0.0000	-1.0000	1.0000
1000	14	0.0000	-1.0000	1.0000
5000	14	0.0000	-1.0000	1.0000
10000	16	0.0000	-1.0000	1.0000
50000	17	0.0000	-1.0000	1.0000
100000	18	0.0000	-1.0000	1.0000
250000	18	0.0000	-1.0000	1.0000
500000	19	0.0000	-1.0000	1.0000

Mehrotra's predictor-corrector method

Problem: Better choice of γ.
Define (d_x^a, d_y^a, d_s^a) by

$$\begin{array}{rcl} A d_x^a & = & -(Ax^k - b), \\ A^T d_y^a + d_s^a & = & -(A^T y^k + s^k - c), \\ S^k d_x^a + X^k d_s^a & = & -X^k s^k. \end{array}$$

Let

$$\hat{\alpha} \equiv \text{step-size}((x^k; s^k), (d_x^a; d_s^a), 1).$$

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:

$$d_{x_j} d_{s_j} = d_\tau^a d_\kappa^a.$$

"Final" direction:

$$\begin{array}{rcl} A d_x & = & -(A x^k - b), \\ A^T d_y + d_s & = & -(A^T y^k + s^k - c), \\ S^k d_x + X^k d_s & = & -X^k s^k + \hat{\gamma} \mu^k e - D_x^a d_s^a, \end{array}$$

where $D_x^a = \text{diag}(d_x^a)$.

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.

Linear algebra

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,

$$\begin{aligned} A^T d_y + X^{-1}(\hat{r}_{xs} - Sd_x) &= \hat{r}_d, \\ Ad_x &= \hat{r}_p. \end{aligned}$$

Leading to

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

i.e.

$$d_x = S^{-1}(XA^T d_y - \hat{r}_{xs}) - S^{-1}X\hat{r}_d$$

But

$$Ad_x = A(S^{-1}(XA^T d_y + \hat{r}_{xs}) - S^{-1}X\hat{r}_d)$$

and finally we reach at

$$AS^{-1}XA^T d_y = \hat{r}_p - AS^{-1}(\hat{r}_{xs} - X\hat{r}_d)$$

or

$$Md_y = \ldots$$

where

$$M := A(S)^{-1}XA^T = ADA^T = \sum_{j=1}^{n} \frac{x_j}{s_j} A_{:j} A_{:j}^T.$$

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

- 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.

The sparse Cholesky decomposition

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

$$M = \begin{bmatrix} x & x & x & x & x & x \\ x & x & & & & \\ x & & x & & & \\ x & & & x & & \\ x & & & & x & \\ x & & & & & x \end{bmatrix}$$

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:

$$\begin{array}{ll} \text{minimize} & e^T x \\ \text{subject to} & e^T x \geq 1, \quad x \geq 0. \end{array}$$

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

Details

- 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

CPLEX-B/D/P	http://www.cplex.com/ (ILOG-CPLEX-11.1)
MOSEK-B/D/P	http://www.mosek.com (MOSEK-5.0.0.93)
LOQO-6.07	http://www.princeton.edu/~rvdb/
LIPSOL	linprog in Matlab 7.6

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.

s	problem	CPLEX-B	CPLEX-D/P	MOSEK-B	MOSEK-D/P	LOQO	LIPSOL
2	cont1	1445	948/911	6427	1593/1069	89	766
2	cont11	913	32767/7580	871	36023/4052	183	1047
2	cont4	1754	826/499	256	6167/1766	933	304
2	cont1_l$	289		914			
2	cont11_l$	7599		940			
1	dano3mip	10	19/9	7	27/22	71	8
4	dbic1	32	47/7	32	303/160	95	30
3	dfl001	9	8/13	7	23/28	112	8
2	fome12	141	48/174	27	191/972	463	30
2	fome13	45	216/339	50	370/1784	786	55
5	gen4	20	1/39	3067	5/124	21	33
7	ken-18	5	5/28	8	12/38	52	13
5	l30	19	9/140	1	613/49	1	3
4	lp22	3	14/35	4	40/125	41	7
4	mod2	6	27/77	7	47/319	49	12
2	neos	67	13/70	77	1369/68	684	86
2	neos1	16	331/8	13	24/3	146	17
2	neos2	12	245/15	11	43/3	81	15
2	neos3	100	1406/	165	7635/21	1490	187
2	ns1687037	>75000	35938/16620	143	/		193
2	ns1688926	89	33/563	8	/		
4	nsct2	43	1/1	27	2/3	854	77

4	nug15	44	1541/704	52	6497/2566	1580	55
2	nug20	767	/39000	920	/60453	22856	1037
2	nug08-3rd	751	1392/	820	4402/	6112	754
2	pds-40	82	14/53	66	37/422	15682	77
2	pds-100	461	66/522	439	376/8627		618
3	qap12	5	69/45	8	227/236	176	11
3	qap15	43	1543/599	71	8342/3557	1739	83
2	rail4284	136	2820/2333	148	3691/291	2654	203
4	rlfprim	2	1/3	2	5/1	56	5
8	self	34	111/49	3072	198/240	34	4330
2	sgpf5y6	8	2/1	6	3/3	fail	14
2	spal_004$	2958		2345		m	
4	stat96v1	314	133/238	33	6205/460	156	90
4	stat96v4	4	146/266	7	194/295	12	16
6	storm-125	11	7/6	27	27/137	21	34
2	storm_1000	205	361/571	348	2926/14521	457	490
1	stp3d	97	315/4579	91	1772/16743	2460	103
2	watson_2	25	74/224	27	116/428	206	43
4	world	6	33/123	10	55/415	46	145

==

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

Links

- An unfinished book:
  - http://mosek.com/fileadmin/homepages/e.d.andersen/papers/linopt.pdf.