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

November 27, 2007

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.
- 1999-2007: LO is employed extensively.
- 2007: You have learned about the simplex method.
- 2007: 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 5 released July 2007.
- 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:

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

http://www.mosek.com

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

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

Notes:

- ρ is a positive (barrier) parameter.
- $\lim\limits_{x \rightarrow 0} \rho \ln(x) = -\infty$.
- 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

$$\begin{array}{lccc} \min & x_1 & & \\ s.t. & 2x_1 & = & 3, \\ & x_1 \geq 0 & & \end{array} \quad (1)$$

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

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

Let

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

and hence

$$Xs = \rho e.$$

Equivalent optimality conditions

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

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:

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

($\gamma \geq 0$ 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 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:
$$\begin{array}{rcl} Ad_x & = & -(Ax^k - b), \\ A^T d_y + d_s & = & -(A^T y^k + s^k - c), \\ S^k d_x + X^k d_s & = & -X^k s^k + \gamma \mu^k e, \end{array}$$
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:

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

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:

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

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

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

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 case.
- Slightly more expensive per iteration than the primal-dual algorithm.
- Can be generalized.

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

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

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:

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

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

http://www.mosek.com

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,

$$d_s = X^{-1}(\hat{r}_{xs} - S d_x).$$

Hence,

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

Leading to

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

i.e.

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

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. $\quad\quad l_{jj} := \delta$
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. $\quad\quad l_{(k+1:m)k} := l_{(k+1:m)k} - l_{kj} l_{(k+1:m)j}$

- 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.

Software

What is available

Name	Features					
	PD	Homo.	BI	Den. col.	Pre. slv.	NLO
BPMPD	D	-	N	Y	Y	QO
Coin	D	-	?	?	Y	?
CPLEX	D	O	Y	Y	Y	QO/CQO
FortMP	D	-	Y	?	Y	QO
Lindo	-	D	Y	Y	Y	QO/CQO
MATLAB	D	-	N	Y	Y	N
MOSEK	-	D	Y	Y	Y	QO/CQO/CO
XPRESS-MP	D	-	Y	Y	Y	QO
HOPDM	D	-	N	Y	Y	QO/?
LIPSOL	D	-	N	Y	N	-
LOQO	D	-	N	Y	(N)	CO
PCx	D	-	N	Y	Y	-
SeDuMi	-	D	N	Y	N	SDO

Notation: (D=default, O=optional, Y=yes, N=no).

Observations

- Most codes use the primal-dual method.
- All codes use Mehrotra's predictor-corrector.
- Only commercial software can compute an optimal basis.
- Matlab TB is based on LIPSOL.
- Presolve facility is common.

Computational results

- Taken from Mittlemans benchmark results.
- See `http://plato.asu.edu/bench.html`.
- Problem size: Up to a million constraints and variable.

The results

s	problem	CPLEX-B	CPLEX-D/P	MOSEK-B	MOSEK-D/P	LOQO	LIPSOL
2	cont1	1186	2622/1640	3163	3658/3729	-	-
2	cont11	2471	72727/22679	1746	/	-	-
2	cont4	4334	3668/787	6262	/9566	-	-
2	cont1_l	f		1803			
2	cont11_l	3246		2011			
1	dano3mip	17	42/25	24	80/76	85	14
4	dbic1	127	142/25	110	1120/714	156	104
3	dfl001	15	23/43	12	39/99	152	13
2	fome12	191	168/429	53	>5100/1215	485	88
2	fome13	97	775/1263	96	778/2738	963	85
5	gen4	35	3/77	22	9/119	33	233
7	ken-18	12	14/74	17	31/115	274	25
5	l30	f	32/80	3	96/258	3	1691
4	lp22	7	47/77	8	94/315	62	9
4	mod2	11	99/214	15	154/614	61	25
2	neos	128	32/166	140	1866/250	754	321
2	neos1	35	930/18	27	39/8	153	37
2	neos2	25	532/31	24	75/13	92	29
2	neos3	229	4046/6350	395	7008/59	1505	502
4	nsct2	77	2/2	53	6/5	840	131
4	nug15	85	3704/1379	108	6955/>7800	1855	118
2	nug20	1406		1792		26402	3092
2	nug08-3rd	1579	3314/	1345	>16000/>17000		2004
2	pds-40	159	45/180	130	94/1131	15908	252
2	pds-100	827	193/1762	907	946/>25000		1673
3	qap12	11	240/119	16	368/386	201	17
3	qap15	83	3557/1315	140	7917/4807	1978	145
2	rail4284	295	6827/6923	306	8064/744	4123	534
4	rlfprim	5	1/7	4	27/3	64	7
8	self	82	146/106	3604	652/>7700	85	3194

2	sgpf5y6	13	4/2	24	8/7	fail	23
2	spal_004	m		5242		m	
4	stat96v1	456	259/682	188	fail/>20000		
4	stat96v2		44446/fail		/		
4	stat96v4	10	507/587	16	548/fail		
6	storm-125	22	18/29	52	77/110	35	62
2	storm_1000	383	768/1608	754	6583/7538	1269	802
1	stp3d	188	1295/12834	180	3365/>16000	2347	218
2	watson_2	58	225/385	52	237/891	313	70
4	world	12	111/271	20	146/912	44	28

"m": memory exceeded

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://www.mosek.com/fileadmin/homepages/e.d.andersen/papers/linopt.pdf.
- Student projects at MOSEK: http://mosek.com/studentprojects.
- Work at MOSEK: http://mosek.com/career.