The Ongoing Impact of Interior-Point Methods

Stephen Wright
University of Wisconsin-Madison

SIAM Conference on Optimization
Toronto
May 20, 2002

(Slightly expanded version, prepared May 27, 2002)
themes

Since 1984 (Karmarkar), interior-point techniques and ideas have pervaded all aspects of optimization: algorithms, theory, software, applications, discrete and continuous, linear and nonlinear, convex and nonconvex.

Research has moved beyond the “frenetic” stage into a phase of consolidation and maturity. Important developments continue to occur.

This talk gives some background, and presents a selection of highlights from the past 10 years.

Disclaimer: Opinions are personal!
interior-point methods have been used in:

- linear programming
- quadratic programming
  - support vector machines
  - optimal control, model predictive control
- monotone linear complementarity
- column generation algorithms for large LP
- analytic center methods
- network optimization / multicommodity flow
• semidefinite programming:
  ▶ combinatorial optimization: approximations
  ▶ control
  ▶ cluster analysis and statistics
  ▶ structural optimization

• second-order cone programming

• branch-and-bound methods for integer and combinatorial optimization

• convex nonlinear programming, monotone nonlinear complementarity

• general nonlinear programming

• stochastic optimization
background on interior-point methods: outline

- the problem classes:
  - linear programming (LP)
  - nonlinear programming (NLP)
  - semidefinite programming (SDP)
  - second-order cone programming (2OCP)
  - conic programming (CP)
- primal methods: general framework
- primal-dual methods for LP and SDP
linear programming (LP)

\[
\begin{align*}
\text{min } & \quad c^T x \quad \text{subject to } \quad Ax = b, \quad x \geq 0, \\
\text{where } & \quad x \in \mathbb{R}^n, \quad A \in \mathbb{R}^{m \times n}, \quad \text{etc. Dual:} \\
\text{max } & \quad b^T y \quad \text{subject to } \quad A^T y + s = c, \quad s \geq 0, \\
\text{where } & \quad y \in \mathbb{R}^m, \quad s \in \mathbb{R}^n. \quad \text{KKT (optimality conditions) can be expressed as} \\
& \quad A x - b = 0, \\
& \quad A^T y + s - c = 0, \\
& \quad X S e = 0, \\
& \quad x \geq 0, \quad s \geq 0, \\
\text{where } & \quad X = \text{diag}(x_1, x_2, \ldots, x_n), \quad S = \text{diag}(s_1, s_2, \ldots, s_n), \quad e = (1, 1, \ldots, 1)^T. \\
\text{The equation } & \quad X S e = 0 \text{ expresses \textit{complementarity}, i.e.} \\
& \quad x_i s_i = 0 \text{ for all } i = 1, 2, \ldots, n.
\end{align*}
\]
nonlinear programming (NLP)

\[
\min f(x) \text{ subject to } c(x) = 0, \ h(x) \geq 0,
\]
where \( f : \mathbb{R}^n \to \mathbb{R} \), \( c : \mathbb{R}^n \to \mathbb{R}^m \), \( h : \mathbb{R}^n \to \mathbb{R}^p \) are smooth functions.

KKT conditions: If \( x \) is a stationary point (no first-order feasible descent directions) and a constraint qualification is satisfied, there are multipliers \( \lambda \in \mathbb{R}^m \) and \( \rho \in \mathbb{R}^p \) and a slack vector \( s \in \mathbb{R}^p \) such that

\[
\nabla f(x) - \nabla c(x) \lambda - \nabla h(x) \rho = 0, \\
c(x) = 0, \\
h(x) - s = 0, \\
\ s_i \rho_i = 0, \ i = 1, 2, \ldots, p, \\
s \geq 0, \quad \rho \geq 0.
\]
max \sum_{i=1}^{m} c_i^T x_i + d_i t_i \\
subject to \\
\sum_{i=1}^{m} A_i x_i + a_i t_i = b, \\
\|x_i\|_2 \leq t_i, i = 1, 2, \ldots, m, \\
where x_i \in R^{n_i} and t_i \in R, i = 1, 2, \ldots, m.

Dual: \\
\min b^T y \text{ subject to } \|A_i^T y - c_i\|_2 \leq a_i^T y - d_i, i = 1, 2, \ldots, m.
semidefinite programming (SDP)

\[
\min \langle C, X \rangle \text{ subject to } \langle A_i, X \rangle = b_i, \ i = 1, 2, \ldots, m, \ X = X^T \geq 0,
\]

- \( X \in \mathbb{R}^{n \times n}, A_i \in \mathbb{R}^{n \times n} \) symmetric, \( C \in \mathbb{R}^{n \times n} \) symmetric, \( b_i \in \mathbb{R} \);
- Inner product: \( \langle F, G \rangle = \text{trace } F^T G = \sum_{j,k=1}^{n} F_{jk}G_{jk} \);
- \( X \geq 0 \) indicates that \( X \) is semidefinite.

Dual is:

\[
\max \ b^T y \text{ subject to } \sum_{i=1}^{m} y_i A_i + S = C, \ S = S^T \geq 0.
\]
SDP: optimality conditions

\[ \langle A_i, X \rangle = b_i, \quad i = 1, 2, \ldots, m, \]
\[ \sum_{i=1}^{m} y_i A_i + S = C, \]
\[ \langle X, S \rangle = 0, \]
\[ X \geq 0, \quad S \geq 0. \]
general form: conic programming (CP)

\[
\min \langle c, x \rangle \text{ subject to } Ax = b, \ x \in K,
\]

where \( Ax = b \) is an affine constraint, \( K \) is a cone.

- **LP**: \( K = \{ x \in R^n \mid x \geq 0 \} \);
- **SDP**: \( K = \{ X \mid X = X^T, \ X \geq 0 \} \);
- **2OCP**: \( K_i = \{ (x_i, t_i) \mid \|x_i\|_2 \leq t_i \}, \ K = K_1 \times K_2 \times \cdots \).

These are all convex, pointed cones, and special in another sense too...

Can even write NLP as a CP (but \( K \) not convex).
We can even write NLP in the form (1). Given

$$\min f(x) \text{ subject to } h(x) \geq 0,$$

define

$$K = \{(x,t) \mid t \geq 0, \ h(x/t) \geq 0\}.$$ 

Then constraint $h(x) \geq 0$ can be expressed as

$$t = 1, \ (x, t) \in K.$$ 

(Of course, this $K$ is not convex, in general.)
primal interior-point methods

In the CP setting, define a barrier function \( F : \text{int}(K) \to R \):

\[ F(x) \to \infty \quad \text{as} \quad x \to \text{bdry} K \]

Seek successive minima of

\[ \langle c, x \rangle + \mu F(x) \quad \text{subject to} \quad Ax = b, \]

for a sequence of \( \mu \) values, decreasing to zero.

Under reasonable conditions, the sequence of minimizers \( x(\mu) \) approaches the solution \( x^* \) of the CP as \( \mu \downarrow 0 \).

Find the minimizers using some variant of Newton’s method (or SQP, if constraints \( Ax = b \) are present).
self-concordant barrier functions

(Nesterov, Nemirovskii 1993): If the barrier function $F$ is “self-concordant”, we can devise primal IP algorithms with nice properties.

- $F$ is “not too nonquadratic”:
  \[ |F'''(x)| \leq 2 \left( F''(x) \right)^{3/2}, \quad \text{all } x \in \text{int} (K); \]

- “Newton decrement” is bounded:
  \[ \langle F'(x), [F''(x)]^{-1} F'(x) \rangle \leq \vartheta, \quad \text{for some } \vartheta \in \mathbb{R}, \text{ all } x \in \text{int} (K). \]

Moreover, it’s possible (in principle) to construct self-concordant barriers for all convex, pointed cones with nonempty interiors.
self-concordant barriers: examples

- $K = \{ x \in \mathbb{R}^n \mid x \geq 0 \}$ (LP):
  \[ F(x) = - \sum_{i=1}^{n} \log x_i \quad (\vartheta = n). \]

- $K = \{ X \mid X = X^T, X \geq 0 \}$ (SDP):
  \[ F(X) = - \log \det(X) \quad (\vartheta = n). \]

- $K = \{(x, t) \in \mathbb{R}^{n+1} \mid \|x\|_2 \leq t\}$ (2OCP):
  \[ F(x, t) = - \log \left( t^2 - \|x\|_2^2 \right) \quad (\vartheta = 2). \]
primal interior-point method: general form

- start at point $x^0 \in \text{int} \ (K)$, $\mu = \mu_0$ (possibly also $Ax^0 = b$);
- apply Newton-like method (with “enhanced” starting point? with line search?) to find approximate solution of $\langle c, x \rangle + \mu F(x)$ s.t. $Ax = b$;
- decrease $\mu$ and repeat.

Nesterov & Nemirovskii show that to reduce $\mu$ by a factor of $\epsilon$, require

- $O(\sqrt{\vartheta} \log(1/\epsilon))$ iterations for “short-step” methods (modest decreases in $\mu$ at each iteration, single full Newton step);
- $O(\vartheta \log(1/\epsilon))$ iterations for “long-step” methods (larger decreases in $\mu$ at each iteration, multiple Newton steps with line search).
Recall the KKT conditions:

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

The first three conditions form a nonlinear system of \( 2n + m \) equations in \( 2n + m \) unknowns.

Primal-dual methods apply a modified Newton’s method to this system, choosing modifications and step lengths to ensure that \( x > 0 \) and \( s > 0 \) are satisfied at all iterates.
central path

Many primal-dual methods use the concept of the central path $C$: set of points $(x(\mu), y(\mu), s(\mu))$ satisfying

\[
\begin{align*}
Ax - b &= 0, \\
A^T y + s - c &= 0, \\
XSe &= \mu e, \quad \text{for } \mu > 0 \\
x > 0, \quad s > 0.
\end{align*}
\]

That is, feasible $(x, y, s)$ with $x_i s_i = \mu$, all $i = 1, 2, \ldots, n$.

Connection to primal methods: $x(\mu) \in C$ is the same $x(\mu)$ that minimizes the primal barrier problem, when

\[F(x) = -\sum \log x_i.\]
primal-dual search directions

Search directions obtained by modifying the complementarity equation as follows:

\[ XSe = \sigma \mu e, \]

where \( \mu = \frac{x^T s}{n} \) and \( \sigma \in [0, 1] \). Calculate Newton step:

\[
\begin{bmatrix}
A & 0 & 0 \\
0 & A^T & I \\
S & 0 & X
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta y \\
\Delta s
\end{bmatrix}
=
-
\begin{bmatrix}
Ax - b \\
A^T y + s - c \\
XSe - \sigma \mu e
\end{bmatrix}.
\]
primal-dual algorithms for LP

path-following:

• short-step: modest decreases in $\mu$ (i.e. $\sigma$ slightly less than 1), one full Newton step at each iteration;

• long-step: large decrease in $\mu$ (i.e. $\sigma$ closer to 0), use a line search to stay inside a (generous) nbd of $C$;

• feasible / infeasible: depends on whether conditions $Ax = b$, $ATy + s = c$ are/are not satisfied at each iteration.

potential-reduction:

• generate steps in the same way, but use a primal-dual potential function to monitor progress and choose step lengths.
convergence of primal-dual algorithms for LP

- **complexity**
  - need $O(n^\tau \log(1/\epsilon))$ iterations, where $\tau = .5, 1, 2$ depending on the algorithm

- **rate**
  - superlinear variants of long-step algorithms make aggressive choices of $\sigma$ (closer to 0) on later iterations
  - besides improving convergence rate, superlinear algorithms have better robustness

- **practical algorithm**
  - Mehrotra (1992) (includes many enhancements)
SDP central path

Recall optimality conditions:

\[ \sum_{i=1}^{m} y_i A_i + S = C, \]  
\[ \langle A_i, X \rangle = b_i, \quad i = 1, 2, \ldots, m, \]  
\[ \langle X, S \rangle = 0, \]  
\[ X \geq 0, \quad S \geq 0. \]  

Define central path \( C \) by replacing (2c) by

\[ XS = \mu I, \quad \text{for } \mu > 0. \]
primal-dual methods for SDP

Can’t apply Newton-like method to (2b), (2a), (3), because this system is not square!

- variables: \((X, y, S) \in S^n \times R^m \times S^n\);
- equations: \(S^n \times R^m \times R^{n \times n}\), because \(XS\) is not symmetric.

Restate complementarity condition in a symmetric form, to recover a “square” system.

Example: “AHO” method uses \(XS + SX = 2\mu I\).

Search directions calculated similarly to LP primal-dual methods (but the linear algebra is more complicated).
selected highlights

1. semidefinite programming algorithms
2. primal-dual algorithms and symmetric cones
3. geometry of the central path in LP
4. nonlinear programming algorithms
5. software tools for LP, NLP, SDP, 2OCP, QP
6. stable linear algebra
7. classification of self-scaled barriers
1. primal-dual semidefinite programming algorithms

- use different symmetrizations of the complementarity condition
- **MZ** family: For nonsingular $P$, define

  $$PXSP^{-1} + P^{-T}SXPT = 2\mu I$$

  - **AHO**: $P = I$
  - **H..K..M**: $P = S^{1/2}$; dual **H..K..M**: $P = X^{-1/2}$;
  - **NT**: $P = W^{-1/2}$, where $W = X^{1/2}(X^{1/2}SX^{1/2})^{-1/2}X^{1/2}$

- many others—(Todd, 1999) lists twenty!
convergence of SDP algorithms

primal-dual, path-following.

- similar to the analogous LP algorithms: choice of $\sigma$, definition of neighborhood, and Mehrotra heuristics for practical codes

- short-step and long-step variants; convergence in $O(n^\tau \log(1/\epsilon))$ iterations, where $\tau = .5, 1, 1.5$

- infeasible variants require only $X > 0$ and $S > 0$ at starting point

- superlinearly convergent variants

SDP least-squares approaches

Alternatively, don’t bother symmetrizing. Treat the central path conditions:

\[
\sum_{i=1}^{m} y_i A_i + S = C, \tag{4a}
\]

\[
\langle A_i, X \rangle = b_i, \quad i = 1, 2, \ldots, m, \tag{4b}
\]

\[
XS = \mu I \tag{4c}
\]

as an overdetermined system (zero residual at the solution). Apply a form of Gauss-Newton. (Kruk et al., 1997)

(Wolkowicz, 2001): Find particular solution \( \bar{X} \) of (4b), and a linear parametrization \( \mathcal{N} : R^{n(n+1)/2-m} \rightarrow \text{null}(\mathcal{A}) \) of the null space of (4b); express \( X = \bar{X} + \mathcal{N}(x) \).
Substitute for $X$ and $S$ in (4c) to obtain a reduced, overdetermined system:

$$F_\mu(x, y) = (\bar{X} + \mathcal{N}(x)) \left( C - \sum_{i=1}^{m} y_i A_i \right) - \mu I = 0.$$ 

(Require $\bar{X} + \mathcal{N}(x) \geq 0$, $C - \sum_{i=1}^{m} y_i A_i \geq 0$ at most iterates and at solution.)

- apply Gauss-Newton, with preconditioning;
- long-step strategy for decreasing $\mu$;
- eventually “cross over” and set $\mu = 0$;
- results obtained for max-cut problem: (4b) is $\text{diag}(X) = (1, 1, \ldots, 1)^T$. 

28
dual barrier algorithms for SDP

\[ \max b^T y \quad \text{subject to} \quad \sum_{i=1}^{m} y_i A_i + S = C, \quad S = S^T \geq 0, \]

yields dual barrier subproblem:

\[ \max f(y, S; \mu) \stackrel{\text{def}}{=} b^T y + \mu \log \det(S) \quad \text{subject to} \]
\[ \sum_{i=1}^{m} y_i A_i + S = C, \quad S = S^T > 0. \quad (5) \]

or alternatively

\[ \max g(y; \mu) \stackrel{\text{def}}{=} b^T y + \mu \log \det \left( C - \sum_{i=1}^{m} y_i A_i \right). \quad (6) \]

Assume that constraints include “\( \text{diag}(X) = \text{given} \)” or “\( \text{trace}(X) = \text{given} \)”.  

29
• (Burer et al., 1998, 2001) find a reparametrization that maintains $S' = S^T \geq 0$
  ▶ substitute for $S'$ to obtain barrier function with only nonnegativity constraints;
  ▶ use steepest descent to solve it.

• (Kojima et al., 2001) use both forms (5), (6)
  ▶ Newton or BFGS to solve the barrier subproblem;
  ▶ preconditioned CG to calculate predictor for each new subproblem.

• gradients can be calculated efficiently, sparsity exploited.
2. self-scaled cones

(a.k.a. homogeneous self-dual cones, symmetric cones)

$K$ in the conic program

$$\min \langle c, x \rangle, \ \text{subject to} \ Ax = b, \ x \in K,$$

is self-scaled if it is convex, self-dual ($K = K^*$), pointed, has nonempty interior, and admits a self-concordant barrier function $F$ satisfying

- $F(\tau x) = F(x) - \vartheta \log \tau$ for all $x \in \text{int } K$, $\tau > 0$;
- $F''(v)x \in \text{int } K$ for all $x, v \in \text{int } K$;
- $F_*(F''(v)x) = F(x) - 2F(v) - \vartheta$.

(Nesterov & Todd, 1997, 1998)
primal-dual algorithms for self-scaled cones

For any $x, s \in \text{int } K$, there is a unique scaling point $w$ such that $F''(w)x = s$. (Critical in development of the theory and algorithms.)

Primal-dual algorithms for CP on self-scaled cones use search directions satisfying

$$
\begin{bmatrix}
A & 0 & 0 \\
0 & A^* & I \\
F''(w) & 0 & I
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta y \\
\Delta s
\end{bmatrix}
= -
\begin{bmatrix}
0 \\
0 \\
F'(x) + \frac{1}{\sigma \mu} s
\end{bmatrix},
$$

from feasible $(x, y, s)$, for $\mu = \langle x, s \rangle$.

This is the NT direction (already encountered in context of SDP). It’s good in practice.
self-scaled cones: notes

- self-scaled cones provide an vehicle for extending primal-dual LP algorithms to a larger class of conic programming problems.

- they include the cones from LP, 2OCP, SDP (and their Cartesian products)—but not much else.

- they had been well studied already by analysts, differential geometers.

- contributions also from Güler (1996) and Faybusovich (1997) (explored connections with Jordan algebras).

3. geometry of the central path in LP

(Vavasis & Ye, 1994)

Recall the system defining central path points $(x(\mu), y(\mu), s(\mu))$:

\[
Ax - b = 0, \\
ATy + s - c = 0, \\
XSe = \mu e, \quad (x_is_i = \mu, \ i = 1, 2, \ldots, n) \\
x > 0, \quad s > 0,
\]

The geometry of $C$ is relevant to path-following methods. Sharp turns, highly nonlinear behavior make $C$ hard to follow.
Vavasis-Ye observations

- $\mathcal{C}$ consists of “nearly straight” segments (easy to traverse) with possibly sharp turns in between.

- turns are associated with crossover events: values of $\mu$ below which $s_i$ definitely larger than $s_j$ for some index pair $(i, j)$;

- define a “layered step” algorithm to follow the path efficiently
  - traverse nearly straight segments in one step;
  - use short steps to round the turns.
simple example of a twisted path

Given $\epsilon > 0$:

$$\min x_0 \text{ subject to } x_0 \geq \epsilon^i x_i, \quad 0 \leq x_i \leq 1, \quad i = 1, 2, \ldots, n,$$

Solution $(x_0^*, x_1^*, \ldots, x_n^*) = (0, 0, \ldots, 0)$. 


results from LP solvers

\( n = 20, \epsilon = 0.1 \). Turn off presolve, scaling, crossover to simplex.

- **MOSEK:**

  \[
  \begin{array}{c|cccccccc}
  \text{Iter.} & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \\
  \text{log } \mu & -1.2 & -2.2 & -3.2 & -4.2 & -5.1 & -6.2 & -7.3 & -8.4 \\
  \end{array}
  \]

  \( x^* \approx (10^{-11}, 10^{-10}, 10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, .0024, .39, .49, .5, .5, \ldots) \)

- **PCx:**

  \[
  \begin{array}{c|cccccccc}
  \text{Iter.} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
  \text{log } \mu & -2.6 & -3.7 & -4.8 & -5.8 & -6.8 & -7.8 & -8.8 & -10.1 \\
  \end{array}
  \]

  \( x^* \approx (10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-3}, .40, .49, .5, .5, \ldots) \)

Each iteration resolves a single component.
results from LP solvers

$n = 20, \epsilon = 0.5$. Turn off presolve, scaling, crossover to simplex.

- **MOSEK:**
  
  \[
  \begin{array}{c|ccccccc}
  \text{Iter.} & 7 & 8 & 9 & 10 & 11 & 12 \\
  \hline
  \log \mu & -0.7 & -1.5 & -2.3 & -3.0 & -4.0 & -6.3 \\
  \end{array}
  \]

  \[x^* \approx (2 \times 10^{-10}, 3 \times 10^{-10}, 1 \times 10^{-9}, \ldots, 8 \times 10^{-5}, 1 \times 10^{-4})\]

- **PCx:**
  
  \[
  \begin{array}{c|cccccc}
  \text{Iter.} & 4 & 5 & 6 & 7 & 8 & 9 \\
  \hline
  \log \mu & -4.7 & -5.4 & -6.2 & -6.9 & -7.7 & -8.6 \\
  \end{array}
  \]

  \[x^* \approx (4 \times 10^{-12}, 7 \times 10^{-11}, 1 \times 10^{-10}, \ldots, 3 \times 10^{-5}, 2 \times 10^{-5})\]

Each iteration resolves multiple components.
consequences of the geometry

- In theory, superlinear convergence doesn’t happen until the final almost-straight leg (basic and nonbasic indices identified);
  - Might happen only when $\mu$ is very small
- Practical algorithms can step past many corners of $C$ at once, provided they are not too sharp.
  - Can we get “semi-local” convergence results that yield fast convergence beyond the final leg?
Consider the problem with only inequality constraints, slacks added:

\[
\min f(x) \text{ subject to } h(x) - s = 0, \quad s \geq 0,
\]

where \( f : R^n \to R, h : R^n \to R^p \) are smooth functions.

Barrier form:

\[
\min f(x) - \mu \sum_{i=1}^{p} \log s_i \text{ subject to } h(x) - s = 0.
\]

KKT conditions: There is \( \rho \in R^p \) such that

\[
\nabla f(x) - \nabla h(x) \rho = 0,
\]

\[
h(x) - s = 0,
\]

\[
s_i \rho_i = 0, \quad i = 1, 2, \ldots, p,
\]

\[
s \geq 0, \quad \rho \geq 0.
\]
primal-dual approach

As in LP, can obtain steps by applying Newton’s method to perturbations of the equality conditions in the KKT system:

\[
\begin{bmatrix}
W(x, \rho) & -\nabla h(x) & 0 \\
\nabla h(x)^T & 0 & -I \\
0 & S & R
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta \rho \\
\Delta s
\end{bmatrix}
=
\begin{bmatrix}
\nabla f(x) - \nabla h(x)\rho \\
h(x) - s \\
SRe - \sigma\mu e
\end{bmatrix}.
\]

For **convex** problems, can define “infeasible path-following” algorithms based on these steps, that maintain \((s, \rho) > 0\). Nice properties (global convergence, fast asymptotic convergence). (Ralph, Wright, others).

For **nonconvex** problems, need much more complex algorithmic strategies to ensure global convergence. (Wächter-Biegler example.)
primal-dual modifications

• line search for merit function: primal barrier, $\ell_1$, augmented Lagrangian
  ▶ doesn’t ameliorate Wächter-Biegler

• don’t insist on satisfaction of linearized equality constraint
  $$\nabla h(x)^T \Delta x - \Delta s = -h(x) + s$$
  ▶ relax and penalize instead (Tits et al.)
  ▶ require only a fraction of the best possible improvement in feasibility
    (Byrd et al., KNITRO)

• use line search filter approach (Wächter)
  ▶ use two merit functions: $\| h(x) - s \|$ and primal barrier;
  ▶ includes feasibility restoration steps, second-order correction.
combining primal and primal-dual approaches

KNITRO (Byrd, Gilbert, Hribar, Liu, Nocedal, Waltz): Find *approximate* solution of a trust-region SQP-like subproblem for the primal barrier:

$$\min \nabla f(x)^T \Delta s + \frac{1}{2} \Delta x^T W(x, \rho) \Delta x - \mu e^T S^{-1} \Delta s + \frac{1}{2} \Delta s^T (S^{-1} R) \Delta s$$

$$h(x) + \nabla h(x)^T \Delta x - s - \Delta s = r_h$$

$$\Delta s \geq -\tau s,$$

$$\| (\Delta x, S^{-1} \Delta s) \|_2 \leq \Delta_k.$$  

$r_h$ close to the smallest vector that makes this problem feasible (obtained by solving for a “normal step” that minimizes

$$\| r_h \|_2 = \| h(x) + \nabla h(x)^T \Delta x - s - \Delta s \|_2$$

subject to trust-region constraints.
5. software

A large range of interior-point software tools is now available.

Mittlemann’s optimization software benchmarks give battery-test performance data, including comparisons with non-interior-point competitors.
All based on Mehrotra’s predictor-corrector method, Gondzio correctors. Most use sparse Cholesky factorization of the $A(XS^{-1})A^T$ formulation.

Main difference between codes is in the sparse linear algebra.

Preprocessing, crossover to simplex, linear algebra modifications also improve efficiency and robustness.

In general, highly competitive with simplex codes on larger problems.

**Cplex Barrier, Xpress-MP Barrier, MOSEK, PCx, BPMPD, HOPDM.**
software: quadratic programming

**Convex:** Primal-dual codes have similar algorithmic heuristics to LP, but require a symmetric indefinite solver (rather than a Cholesky).

**CPLEX Barrier, XPRESS-MP Barrier, MOSEK.**

Object-oriented code: **OOQP.** C++; default version uses MA27, MA47 factorizers; can incorporate specialized linear algebra modules.

**Nonconvex:** Can be solved using NLP codes (**LOQO, KNITRO**).

Specialized code: **Galahad/QPB** (two-phase, primal trust-region algorithm with primal-dual scaling, preconditioned CG/Lanczos linear algebra).
software: semidefinite programming

- primal-dual codes (typically implement H..K..M, NT search directions)
  - SDPT3: Matlab/C, exploits sparsity;
  - SeDuMi: Matlab/C, product-form storage of $X$ and $S'$ promotes numerical stability;
  - CDSP: C, modified Cholesky;
  - SDPA: C++, exploits sparsity;
  - PENNON, numerous others...

- dual codes (avoid storage of potentially dense $X$)
  - DSDP
  - BMZ, BMPR
semidefinite programming software issues

- input format: nice ones available
- homogeneous and self-dual reformulations;
- starting points
- exploiting structure (sparse, block diagonal) is critical to efficiency
- linear algebra software used:
  - Meschach
  - Ng-Peyton / modified Cholesky
  - Matlab sparse matrix routines
  - LAPACK
software: second-order cone programming

- **MOSEK**: C, presolver, self-dual formulation, NT search direction. Also supports “rotated” second-order cones.

- **SDPT3**: see above.

- **SeDuMi**: see above.
Interior-point codes are proving to be highly competitive with codes based on other algorithms.

- **IPOPT** ([www.coin-or.org](http://www.coin-or.org)): line search, filter, preconditioned CG.
- **KNITRO** ([www.ziena.com](http://www.ziena.com)): trust-region, primal barrier/SQP with primal-dual scaling, preconditioned CG.
- **LOQO** ([orfe.princeton.edu/~loqo/](http://orfe.princeton.edu/~loqo/)): primal-dual, direct factorization of a regularized KKT matrix.
6. linear algebra

- robust, efficient linear algebra essential to practical effectiveness
- structure and size of linear systems to be solved typically the same at all iterations
  - easier to exploit structure, sparsity than in other approaches
  - easier to write an efficient code
- ill conditioning may cause concern, particularly near the solution
stable linear algebra for LP

Formulate search direction linear system as either “augmented system:”

\[
\begin{bmatrix}
-X^{-1}S & AT \\
A & 0
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta y
\end{bmatrix}
= 
\begin{bmatrix}
r_b \\
r_c
\end{bmatrix}
\]

or “normal equations:”

\[
A(XS^{-1})A^T \Delta y = \bar{r},
\]

where \(X, S\) are positive diagonal (but \(X^{-1}S\) increasingly ill conditioned near the solution).

Even when \(A\) has full rank, \(A(XS^{-1})A^T\) may become ill conditioned on later iterations—may lose numerical positive definiteness.

How to deal with the small or negative pivots that arise during Cholesky factorization?
• simple fixes often work:
  ▶ replace the offending diagonal with $\infty$;
  ▶ replace the diagonal, and its row/column, and the corresponding right-hand side element, by 0.

• can show that these measures (and roundoff) may induce a large error in $(\Delta x, \Delta y, \Delta s)$, but still yield a good step for the algorithm.
  ▶ errors are in a subspace that doesn’t matter much (S. Wright, 1996)

• more sophisticated fix: product-form Cholesky (Goldfarb & Scheinberg, 2000).
stable linear algebra for NLP

Compute Newton-like steps for

\[ \nabla f(x) - \nabla h(x) \rho = 0, \]
\[ h(x) - s = 0, \]
\[ s_i \rho_i = 0, \quad i = 1, 2, \ldots, p \]

from points \((x, s, \rho)\) with \((s, \rho) > 0\).

Usually use an “augmented system” or a “condensed” form of the step equations, which approaches a singular limit.

Direct factorization approaches applied to these formulations yield reasonable steps for the primal-dual, until \(\mu \approx \sqrt{u}\).

(M. Wright, 1997; S. Wright, 1998)
7. classification of self-scaled barriers

(Schmieta, Hauser, Güler, Lim, 1999-2001)

- self-scaled cones were classified (by their identity with symmetric cones), but self-scaled barriers were not

- for each of $R^n_+$, $S^n$, second-order cone, the self-scaled barrier function $F$ with smallest possible $\vartheta$ is well known. Is it unique?

  YES! (up to an additive constant)
proof via Jordan algebras

self-scaled barrier for $K$

$\downarrow$

Euclidean Jordan algebra $\mathcal{J}$ for which $K$ is the cone of squares

$\downarrow$

the self-scaled barrier for $\mathcal{J}$: $-\log \det x + \text{constant}$.

- technique for deriving $\mathcal{J}$ from $K$ and a self-scaled barrier for $K$;
- technique for generating the self-scaled barrier from $\mathcal{J}$:

  $-\log \det x + \text{constant}$. 
topics for further investigation

- NLP algorithms
  - more computation-guided development of algorithms and theory
  - practical algorithms appear to tie together many ideas, new and old (SQP, trust-region, filter, linear algebra of different types, ...)
- linear algebra / software implementations
  - SDP issues;
  - NLP issues: iterative vs. direct linear algebra
- structured applications in emerging areas
  - bioinformatics?
interior-point research has changed optimization

- LP software much better than in 1984
- opened a new line of research in practical NLP algorithms
- cross-fertilization between optimization disciplines
- semidefinite programming!