Large-Scale Optimization

Interior-point methods
Primal-dual path-following methods

Jordi Castro
MSc in Statistics and Operations Research

Department of Statistics and Operations Research
School of Mathematics and Statistics
Universitat Politècnica de Catalunya
Barcelona

Contents

LP optimality conditions

- We develop main ideas for LP, later extend to QP and convex programming.
- LP primal and dual, m constraints, n variables

$$(P)\quad \begin{array}{lll} \min & c^T x & \\ \text{s.to} & Ax = b & [\lambda] \\ & x \geq 0 & [s] \end{array} \qquad (D)\quad \begin{array}{lll} \max & b^T \lambda & \\ \text{s.to} & A^T\lambda + s = c & \\ & s \geq 0 & \end{array}$$

- Using Lagrangian

$$L(x, \lambda, s) = c^T x + \lambda^T (b - Ax) - s^T x$$

we get KKT optimality conditions

$$\begin{array}{ll} A^T\lambda + s = c & \text{[dual feasibility]} \\ Ax = b & \text{[primal feasibility]} \\ x^T s = 0 \;\equiv\; x_i s_i = 0, i = 1, \ldots, n & \text{[complementarity]} \\ (x, s) \geq 0 & \end{array}$$

Reformulation of LP KKT conditions

- Original KKT conditions

$$\begin{array}{l} A^T\lambda + s = c \\ Ax = b \\ x_i^T s_i = 0 \quad i = 1, \ldots, n \\ (x, s) \geq 0 \end{array}$$

- Define diagonal matrices X, S and vector e:

$$X = \begin{bmatrix} x_1 & & \\ & \ddots & \\ & & x_n \end{bmatrix} \quad S = \begin{bmatrix} s_1 & & \\ & \ddots & \\ & & s_n \end{bmatrix} \quad e = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^n$$

- Usual reformulation of KKT conditions in IPMs:

$$\begin{array}{l} A^T\lambda + s = c \\ Ax = b \\ XSe = 0 \\ (x, s) \geq 0 \end{array}$$

Solution of KKT by Newton (I)

- KKT conditions

$$\begin{array}{l} A^T\lambda + s = c \\ Ax = b \\ XSe = 0 \\ (x, s) \geq 0 \end{array}$$

- Defining function $F : \mathbb{R}^{2n+m} \to \mathbb{R}^{2n+m}$, rewrite KKT:

$$F(x, \lambda, s) = \begin{bmatrix} A^T\lambda + s - c \\ Ax - b \\ XSe \end{bmatrix} \qquad \text{(KKT)} \quad \boxed{\begin{array}{l} F(x, \lambda, s) = 0 \\ (x, s) \geq 0 \end{array}}$$

- $F(x, \lambda, s) = 0$ is a nonlinear system of equations. The only nonlinear equations are $XSe = 0$, which is the closest to linearity.
- We can use Newton's method for $F(x, \lambda, s) = 0$, shortening if necessary the Newton direction due to $(x, s) \geq 0$: damped Newton method.

Solution of KKT by Newton (II)

- Newton direction $\Delta = (\Delta_x, \Delta_\lambda, \Delta_s)$ for $F(x, \lambda, s) = 0$ is

$$\nabla_{(x,\lambda,s)} F(x, \lambda, s)\Delta = -F(x, \lambda, s)$$

- Jacobian of $F(x, \lambda, s) = \begin{bmatrix} A^T\lambda + s - c \\ Ax - b \\ XSe \end{bmatrix}$ is:

$$\nabla_{(x,\lambda,s)} F(x, \lambda, s) = \begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix}$$

- Linear system for Δ is

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta_x \\ \Delta_\lambda \\ \Delta_s \end{bmatrix} = \begin{bmatrix} c - A^T\lambda - s \\ b - Ax \\ -XSe \end{bmatrix}$$

- If (x, λ, s) is feasible then two first vectors of RHS are 0, and feasibility will be preserved.
- Step length α used to guarantee $(x, s) > 0$. Whenever possible, use $\alpha = 1$, default value in Newton's method:

$$(x, \lambda, s) \leftarrow (x, \lambda, s) + \alpha(\Delta_x, \Delta_\lambda, \Delta_s)$$

Solution scheme of KKT based on Newton

Algorithm *KKT-DAMPED-NEWTON*
Initial point (x^0, λ^0, s^0), $k = 0$
while (x^k, λ^k, s^k) is not solution **do**

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta_x^k \\ \Delta_\lambda^k \\ \Delta_s^k \end{bmatrix} = \begin{bmatrix} c - A^T\lambda^k - s^k \\ b - Ax^k \\ -X^k S^k e \end{bmatrix}$$

Compute α such that $(x^{k+1}, s^{k+1}) > 0$
$(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + \alpha(\Delta_x^k, \Delta_\lambda^k, \Delta_s^k)$
$k := k + 1$
end_while
Return: $(x^*, \lambda^*, s^*) = (x^k, \lambda^k, s^k)$
End_algorithm

Inconvenience: Newton's step length $\alpha = 1$ can not be always used, and in general we get $\alpha \ll 1$ because we are very close to the boundary of $(x, s) \geq 0$. Procedure is slow and not useful in practice.

Matlab code of KKT-DAMPED-NEWTON (I)

```
 function xopt= ip_KKT_dampednewton(A,b,c,info)
% usage: ip_KKT_dampednewton(A,b,c,info)
%      A is full row rank matrix (m x n)
%      b is column vector of size m
%      c is column vector of size n
%      info is level of information:
%               if info>0 shows information at each iteration
%               if info=0 only shows solution

% check dimensions
if (nargin==3) info=0;
end
[m,n]= size(A);
[mc,nc]= size(c);
[mb,nb]= size(b);
if ((nc ~= 1) | (nb ~= 1))
        error('Error: c and b must be column vectors');
end
if ((mc ~= n) | (mb ~= m))
        error('Error: inconsistent dimensions for c, b and A');
end
% check rank of A
if (rank(A) ~= m)
        error('Error: matrix A is not full row rank');
end
```

Matlab code of KKT-DAMPED-NEWTON (II)

```
% parameters
optgap= 1.0e-6;
myeps= 1.0e-10;
rho= 0.9999;
% initializations
e= ones(n,1); x= ones(n,1)*10; y= ones(m,1); s= ones(n,1)*10;
dx= zeros(n,1); dy= zeros(m,1); ds= zeros(n,1);
X= diag(x); S= diag(s);
k= 0;
while ( norm([A'*y+s-c; A*x-b; X*S*e]) > optgap )
        M=[zeros(n,n) A' eye(n); A zeros(m,m) zeros(m,n);S zeros(n,m) X];
        dmov= M\[c-A'*y-s; b-A*x;-X*S*e];

        dx= dmov(1:n); dy= dmov(n+1:n+m); ds= dmov(n+m+1:n+m+n);

        alphx= min(-x(dx<myeps) ./ dx(dx<myeps));
        alphs= min(-s(ds<myeps) ./ ds(ds<myeps));
        alph= min([1 alphx alphs]);

        if (info > 0)
                fprintf('\n\t\t\t Iteration %d\n',k);
                fprimal= c'*x;
                fdual= b'*y;
                x,y,s,alph %dx,dy,dx,alph,fprimal,fdual
        end
        x= x+rho*alph*dx; y= y+rho*alph*dy; s= s+rho*alph*ds;
        X= diag(x); S= diag(s);
        k=k+1;
end
xopt=x;
```


Modification of Newton's direction

- Using standard Newton we get $\alpha \ll 1$, because we are very close to $(x, s) = 0$. Slow progress. We need another direction.
- Direction of primal-dual interior-point methods:
  - interior point because $(x, s) > 0$.
  - primal-dual because optimize both (x, λ, s)
  - direction avoids (x, s) gets close to 0
  - direction brings to interior of $(x, s) \geq 0$, allowing larger α.
- Above achieved using a perturbed KKT-τ system:

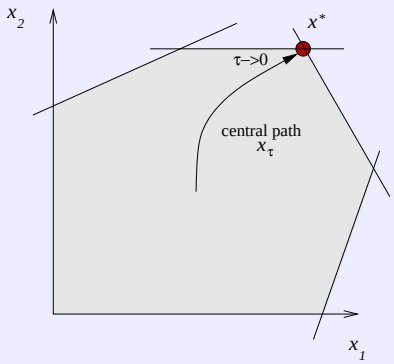
$$\begin{array}{l} A^T\lambda + s = c \\ Ax = b \\ XSe = \tau e, \tau \in \mathbb{R}^+ \\ (x,s) > 0 \end{array} \quad \equiv \quad F(x,\lambda,s) = \begin{bmatrix} 0 \\ 0 \\ \tau e \end{bmatrix}, \quad (x,s) > 0$$

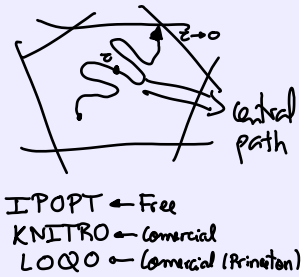
The primal-dual central path

- For each τ we have a solution $(x_\tau, \lambda_\tau, s_\tau)$ of strictly positive points: $x_\tau > 0$, $s_\tau > 0$. This solution exists under some conditions and it is unique.
- Primal-dual central path is the set of solutions to KKT-τ:

$$\mathcal{C} = \{(x_\tau, \lambda_\tau, s_\tau), \tau > 0\}$$

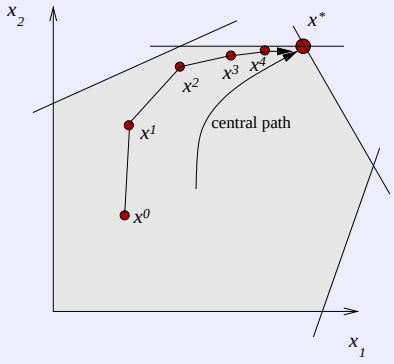
- Central path can be seen as a parametric curve inside the primal-dual positive orthant:





Central path leads us to the optimum

- When $\tau = 0$, we have a primal-dual solution (x, λ, s) to LP KKT optimality conditions.
- PD path-following IPMs approximately follow this path to the optimum:



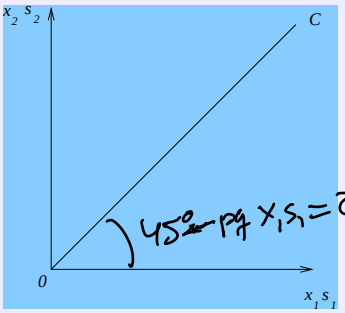
Example of central path

- Primal-dual problem:

$$(P)\quad \begin{array}{ll}\min & x_1 + x_2 \\ \text{s.to} & x_1 \geq 0, x_2 \geq 0\end{array} \qquad (D)\quad \begin{array}{ll}\max & 0 \\ \text{s.to} & s_1 = 1, s_2 = 1 \\ & s_1 \geq 0, s_2 \geq 0\end{array}$$

- KKT-τ conditions:

$$\begin{cases} s_1 & = 1 \\ s_2 & = 1 \\ x_1 s_1 & = \tau \\ x_2 s_2 & = \tau \\ (x_1, x_2, s_1, s_2) & > 0 \end{cases}$$



- Central path points are: $(x_{1_\tau}^*, x_{2_\tau}^*, s_{1_\tau}^*, s_{2_\tau}^*) = (\tau, \tau, 1, 1)$
- When $\tau \to 0$, $(x_{1_\tau}^*, x_{2_\tau}^*, s_{1_\tau}^*, s_{2_\tau}^*) \to (0, 0, 1, 1)$ we get the optimal solution.
- In general there is no closed form for central path.

Derivation of KKT-τ using logarithmic barrier

- Original and barrier problem: replace $x \geq 0$ by logarithmic barrier $\sum_{i=1}^n \ln x_i$ with parameter τ.

$$(P)\quad \begin{array}{ll}\min & c^T x \\ \text{s.to} & Ax = b \\ & x \geq 0\end{array} \qquad (P_\tau)\quad \begin{array}{ll}\min & c^T x - \tau \sum_{i=1}^n \ln x_i \\ \text{s.to} & Ax = b\end{array}$$

- Barrier avoids $x = 0$.
- Barrier properties:
  - For $\tau > 0$, $-\tau \sum_{i=1}^n \ln x_i < 0$ if $x_i \gg 0$
  - $-\tau \sum_{i=1}^n \ln x_i \to +\infty$ if some $x_i \to 0$
  - Minimizer of $-\sum_{i=1}^n \ln x_i$ is named the analytic center:

$$\min -\sum_{i=1}^n \ln x_i \Leftrightarrow \min e^{-\sum_{i=1}^n \ln x_i} \Leftrightarrow \min e^{-\ln(\Pi_{i=1}^n x_i)} \Leftrightarrow \min \frac{1}{\Pi_{i=1}^n x_i} \Leftrightarrow \max \Pi_{i=1}^n x_i$$

  Analytic center maximizes distance to hyperplanes $x_i = 0$
- [Fiacco, McCormick (1968)] show that when $\tau \to 0$ then solutions x_τ of (P_τ) tend to x^* of (P).

Optimality conditions of (P_τ)

- Lagrangian of

$$(P_\tau) \quad \begin{array}{ll} \min & c^T x - \tau \sum_{i=1}^n \ln x_i \\ \text{s.to} & Ax = b \end{array}$$

  is

$$L(x, \lambda) = c^T x - \tau \sum_{i=1}^n \ln x_i + \lambda^T (b - Ax)$$

- KKT conditions of (P_τ):

$$\begin{array}{rll} c - \tau X^{-1} e - A^T y & = 0 & \\ Ax & = b & \\ x & > 0 & \text{[because } \ln x_i \text{ not defined in 0]} \end{array}$$

- Defining $\tau X^{-1} e = s$, or equivalently $XSe = \tau e$, and including $s > 0$ since $x > 0$, rewrite KKT:

$$\begin{array}{rl} c - s - A^T \lambda & = 0 \\ Ax & = b \\ XSe & = \tau e \\ (x, s) & > 0 \end{array}$$

- We get perturbed KKT-τ conditions: a solution of (P_τ) is a point of central path.

Uniqueness of central path

- Objective function of (P_τ) is strictly convex for $x > 0$:

$$f(x) = c^T x - \tau \sum_{i=1}^n \ln x_i \qquad \nabla_x f(x) = c - \tau X^{-1} e \qquad \nabla^2_{xx} f(x) = \tau X^{-2} \succ 0$$

- If (P_τ) has solution, it is thus unique. Then if there is a central path point solution KKT-τ, this is unique too.
- But some problems don't have central path. Example:

$$(P) \quad \begin{array}{ll} \min & 0 \\ \text{s.to} & x \geq 0 \end{array} \qquad (D) \quad \begin{array}{ll} \max & 0 \\ \text{s.to} & s = 0 \\ & s \geq 0 \end{array}$$

  Barrier problem $\min 0 - \tau \ln x$, which has no minimum: $-\tau/x = 0 \Rightarrow x = +\infty$. Similarly, there is no solution to KKT-τ:

$$\begin{array}{l} s = 0 \\ xs = \tau \\ (x, s) > 0 \end{array}$$

- The reason is that the dual feasible region has empty interior...

Existence of central path

- Sufficient conditions for existence of central path is that primal and dual feasible sets have nonempty interior:

$$\mathcal{F}^0 \neq \emptyset \qquad \text{where} \quad \mathcal{F}^0 = \{(x, \lambda, s) : Ax = b, A^T\lambda + s = c, x > 0, s > 0\}$$

- **Theorem**
  If $\mathcal{F}^0 \neq \emptyset$ and $\tau > 0$ then there exists a solution $(x_\tau, \lambda_\tau, s_\tau)$ of central path.

  See, for instance, [Roos, Terlaky, Vial (2006), Vanderbei (1996), Wright (1997)]...

  ...or the proof in the blackboard...

Develope the algorithm

Alg. that follows the central path

Following approximately the central path

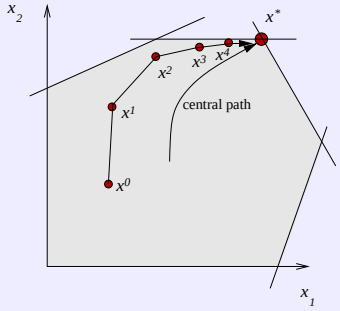
Reduce τ at each iteration

- KKT-τ system

$$\begin{aligned} A^T\lambda + s &= c \\ Ax &= b \\ XSe &= \tau e \\ (x, s) &> 0 \end{aligned}$$

  is solved iteratively, reducing τ (to 0) at each iteration

- For a particular τ, point $(x_\tau, \lambda_\tau, s_\tau)$ is not computed: that would mean exact solutions to a sequence of systems KKT-τ, which is computationally prohibitive.
- Instead, what is done is to compute only one damped Newton direction for τ, then τ is reduced, another damped Newton direction for the new τ is computed, and so on. Sequence of points are not in central path, they **follow** the central path:



Computing and reducing τ

- τ is reduced until 0, as follows:

$$\tau = \sigma\mu$$

  - $\sigma \in (0, 1)$ is a reduction parameter
  - μ means the perturbed complementarity of current point

- If (x, λ, s) is in central path, then perturbed complementarity is guaranteed: $x_i s_i = \mu, i = 1, \ldots, n$
- In general, iterates (x, λ, s) are not in central path, and we don't have $x_i s_i = \mu$ for all i. Therefore we consider as μ the average of $x_i s_i$:

$$\mu = \frac{\sum_{i=1}^{n} x_i s_i}{n} = \frac{x^T s}{n}$$

- If (x, λ, s) are primal and dual feasible then $n\mu$ is the duality gap:

$$n\mu = x^T s = x^T(c - A^T\lambda) = c^T x - (Ax)^T\lambda = c^T x - b^T\lambda$$

KKT-μ and σ

- System to be solved is KKT-μ:

$$\begin{aligned} A^T\lambda + s &= c \\ Ax &= b \\ XSe &= \sigma\mu e \\ (x, s) &> 0 \end{aligned}$$

- $\sigma \approx 0$ means $XSe = \sigma\mu e \approx 0$, forcing an aggressive direction that quickly leads to the optimum. Inconvenience: as in Newton's method ($\sigma = 0$), we will progress slowly.
- $\sigma \approx 1$ means $XSe = \sigma\mu e \approx \mu$, forcing a direction that leads to central path. Inconvenience: we don't move to optimality. Advantage: moving to well centered points, this allows significant progress in next iterations.
- σ updated according to the particular IPM used.

Generic primal-dual path-following IPM

Algorithm *primal-dual path-following IPM*
Initial point (x^0, λ^0, s^0), $k = 0$
while (x^k, λ^k, s^k) is not solution **do**
$\sigma_k \in (0,1)$, $\mu_k = (x^k)^T s^k / n$
$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta_x^k \\ \Delta_\lambda^k \\ \Delta_s^k \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -X^k S^k + \sigma_k \mu_k e \end{bmatrix}$$
Compute α such that $(x^{k+1}, s^{k+1}) > 0$
$(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + \alpha(\Delta_x^k, \Delta_\lambda^k, \Delta_s^k)$
$k := k + 1$
end_while
Return: $(x^*, \lambda^*, s^*) = (x^k, \lambda^k, s^k)$
End_algorithm

where

$r_c = A^T\lambda + s - c$ Dual infeasibilities
$r_b = Ax - b$ Primal infeasibilites

Short-step and long-step variants

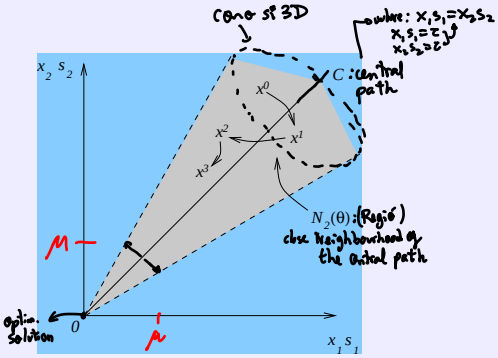
(Assume to simplify exposition in next slides that x^0 is feasible: $r_c = r_b = 0$)

- Allowing short or long movements when moving from (x^k, λ^k, s^k) to $(x^{k+1}, \lambda^{k+1}, s^{k+1})$ we obtain two variants. Both are polynomial time algorithms for PL:
  - Short-step method: best theoretical complexity, worse in practice.
  - Long-step method: worse theoretical complexity, but used and the most efficient in practice.

Short-step method

- Iterates (x^k, λ^k, s^k) in a narrow neighborhood of central path. This neighborhood is $N_2(\theta)$, where $\theta \in (0,1)$:

$$N_2(\theta) = \{(x, \lambda, s) \in \mathcal{F}^0 : ||XSe - \mu e|| \leq \theta\mu\}$$



$N_2(0)$ is central path.

- Narrow neighborhood of central path because deviations of $x_i s_i$ to μ must be small:

$$||XSe - \mu e|| \leq \theta\mu \Leftrightarrow \sqrt{\sum_{i=1}^{n}(x_i s_i - \mu)^2} \leq \theta\mu < \mu \Rightarrow \sum_{i=1}^{n}(x_i s_i/\mu - 1)^2 \leq \theta^2 < 1$$

Even when $\theta \approx 1$, condition forces points close to central path.

Generic short-step algorithm

Algorithm *Primal-dual path-following short-step algorithm*

Set $\theta \in (0,1)$, $\sigma = 1 - \delta/\sqrt{n}$, $\delta > 0$ $\quad (\theta = 0.4, \delta = 0.4)$

Initial point $(x^0, \lambda^0, s^0) \in N_2(\theta)$, $k = 0$

while (x^k, λ^k, s^k) is not solution **do**

$\sigma_k = \sigma \quad \mu_k = (x^k)^T s^k / n$

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta_x^k \\ \Delta_\lambda^k \\ \Delta_s^k \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -X^k S^k + \sigma_k \mu_k e \end{bmatrix}$$

$\alpha = 1$

$(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + \alpha(\Delta_x^k, \Delta_\lambda^k, \Delta_s^k)$

$k := k + 1$

end_while

Return: $(x^*, \lambda^*, s^*) = (x^k, \lambda^k, s^k)$

End_algorithm

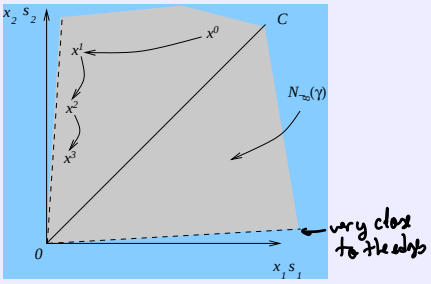
- It can be shown that iterates (x^k, λ^k, s^k) are in $N_2(\theta)$ if $\theta = 0.4$, $\delta = 0.4$
- Best known complexity for LP: number of iterations is

$$O(\sqrt{n}\,\log 1/\varepsilon)$$ n is number of variables, ε is optimality precission: $\mu_k \leq \varepsilon$

Long-step method

- Iterates (x^k, λ^k, s^k) in a wide neighborhood of central path. This neighborhood is $N_{-\infty}(\gamma)$, where $\gamma \in (0, 1)$:

$$\text{neighbourhood} \rightarrow N_{-\infty}(\gamma) = \{(x, \lambda, s) \in \mathcal{F}^0 : x_i s_i \geq \gamma\mu \;\; i = 1, \ldots, n\}$$



When $\gamma \approx 0$ then $N_{-\infty}(\gamma) \approx \mathcal{F}^0$

- σ_k selected from $[\sigma_{min}, \sigma_{max}]$ to avoid large or small reductions of complementarity μ. in practice $\sigma = 0.1$ works fine.
- α_k computed as maximum step length such that $(x^k, \lambda^k, s^k) + \alpha_k(\Delta_x^k, \Delta_\lambda^k, \Delta_s^k)$ is in $N_{-\infty}(\gamma)$. In practice, the maximum step length such that $(x, s) \geq 0$ is used, and later it is reduced by parameter $\rho \in [0.95, 0.99995]$.

Generic long-step algorithm

Algorithm *Primal-dual path-following long-step algorithm*
Set $\gamma \in (0, 1)$, $0 < \sigma_{min} < \sigma_{max} < 1$
Initial point $(x^0, \lambda^0, s^0) \in N_{-\infty}(\gamma)$, $k = 0$
while (x^k, λ^k, s^k) is not solution **do**
$\sigma_k \in [\sigma_{min}, \sigma_{max}]$ $\quad \mu_k = (x^k)^T s^k / n$

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta_x^k \\ \Delta_\lambda^k \\ \Delta_s^k \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -X^k S^k + \sigma_k \mu_k e \end{bmatrix}$$

compute α_k as maximum value such that new point is in $N_{-\infty}(\gamma)$
$(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + \alpha_k(\Delta_x^k, \Delta_\lambda^k, \Delta_s^k)$
$k := k + 1$
end_while
Return: $(x^*, \lambda^*, s^*) = (x^k, \lambda^k, s^k)$
End_algorithm

- Polynomial complexity: the number of iterations is

$$O(n \log 1/\varepsilon)$$

n is number of variables, ε is optimality precission: $\mu_k < \varepsilon$

Long-step method: infeasible variant

- In practice an infeasible point (x^0, λ^0, s^0) is considered. This point only needs to be interior: $x^0, s^0 > 0$.
- Infeasibilities in that point are vectors r_c, r_b. In first iterations, $r_c \neq 0$, $r_b \neq 0$. Usually, we first satisfy primal and dual feasibility ($r_c = 0$, $r_b = 0$), and in last iterations we reach complementarity: $\mu \approx 0$.
- Same algorithm than for the feasible long-step algorithm, only difference is that $r_c \neq 0$, $r_b \neq 0$ at first iterations.
- Polynomial complexity: the number of iterations is

$$O(n^2 \log 1/\varepsilon)$$

- It is the (generic) algorithm used in practice in most solvers: CPLEX, XPRESS, LIPSOL, HOPDM, BPMPD, COIN-CLP, etc...

Convergence of primal-dual path-following algorithms

- We concentrate on interior feasible starting point case: $(x, \lambda, s) \in \mathcal{F}^0$. Proof of infeasible method more elaborate (see for instance [Wright (1997), Chap. 6].
- We prove convergence and complexity of short-step variant, the one with the best complexity, although not used in practice.
- A proof for long-step variant can be found, for instance, in [Wright (1997), Chap. 5].

Convergence of short-step variant

We have to prove:

1. Sequence (x^k, λ^k, s^k) is in neighborhood $N_2(\theta) = \{(x, \lambda, s) \in \mathcal{F}^0 : ||XSe - \mu e|| \leq \theta\mu\}$
   - Not to be proved here. See, for instance, [Wright (1997), pp. 88–91]. All the iterations are in the neighbourhood (Proof) → $\sigma = 0.4 \rightarrow \sigma$, $\theta = 0.4 \rightarrow N_2(\theta)$
2. $\mu_k = (x^k)^T s^k / n$ decreases at each iteration, and in a number of iterations $O(\sqrt{n}\ \log 1/\varepsilon)$ we get $\mu_k \leq \varepsilon$.
   - This result proves global convergence and polynomial complexity.

iterations $\leq a_0 + a_1 n + a_2 n^2 + a_3 n^3$ you need to perform is bounded by polynomial

Examples de pol. time: $u \in \mathbb{R}^n$, $v \in \mathbb{R}^n$, $u + v = w$ → the complexity is $\theta(n)$

$A \cdot B = C$ → $o(n^3)$

General result of convergence

We prove that any IPM that satisfies (n is number of variables):

$$\mu_{k+1} \leq \left(1 - \frac{\delta}{n^\omega}\right) \mu_k$$

< 1 → at each iteration we reduce the complementarity

converges with polynomial complexity.

Theorem

Let $\varepsilon \in (0, 1)$ be the optimality precision required. Assume our algorithm satisfies

$$\mu_{k+1} \leq \left(1 - \frac{\delta}{n^\omega}\right) \mu_k \quad k = 0, 1, 2, \ldots$$

for some $\delta, \omega > 0$. Assume (x^0, λ^0, s^0) satisfies $\mu_0 \leq 1/\varepsilon^\kappa$ ← ill-conditioning for some $\kappa > 0$. Then

$$\exists K = O(n^\omega \log 1/\varepsilon)$$

such that

$$\mu_k \leq \varepsilon \quad \forall k \geq K.$$

→ optim. precision

$\mu_0 = \frac{(x^0)^T s^0}{n}$, $\mu_0 \leq \frac{1}{\varepsilon^\kappa}$

$\mu_0 \leq \frac{1}{\varepsilon^\kappa} = \frac{1}{(10^{-8})^2} = 10^{16}$ → it is guaranteed by any starting point (because $10^{16} \gg$) $\varepsilon = 10^{-8}$, $\kappa = 2$

Proof at blackboard...

Convergence and complexity of short-step variant

Introduce notation:

$$(x(\alpha), \lambda(\alpha), s(\alpha)) = (x, \lambda, s) + \alpha(\Delta_x, \Delta_\lambda, \Delta_s)$$
$$\mu(\alpha) = x^T(\alpha)s(\alpha)/n.$$

Lemma

Direction $(\Delta_x, \Delta_\lambda, \Delta_s)$ computed by short-step method satisfies:

$$a) \quad \Delta_x^T \Delta_s = 0$$
$$b) \quad \mu(\alpha) = (1 - \alpha(1 - \sigma))\mu$$

Proof at blackboard...

Theorem
Short-step method with $\alpha = 1$ and $\sigma = 1 - \delta/\sqrt{n}$ reaches a point $\mu_k \leq \varepsilon$ in a number of iterations

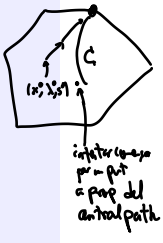
$$O(\sqrt{n} \log 1/\varepsilon)$$

Immediate from previous lemma and general convergence theorem.

Starting point

- Infeasible primal-dual path-following IPMs used in practice.
- A feasible point (x^0, λ^0, s^0) is not needed. The only requirement is that $(x^0, s^0) > 0$, and not too close to 0.
- In many cases, simply $(x^0, s^0) = 100$ can be a good choice. IPM quite robust.
- However, better if point is well centered (i.e., $x_i^0 s_i^0$ similar for all i), and it is not too infeasible (i.e., r_c and r_b not too large).
- For instance, look for a point that approximately satisfies

$$A^T\lambda + s = c$$
$$Ax = b$$
$$XSe = \mu e$$
$$(x, s) > 0$$



$x, s = 100$ (13 iter's)

by solving two least-squares problems related to primal and dual:

we move the feas. constraints to the O.F

$$\begin{cases} \min & \overbrace{c^T x}^{\text{minim a} -\infty \text{ gradient}} + \rho||x|| \\ \text{s.to} & Ax = b \end{cases} \qquad \begin{cases} \max & b^T\lambda + \rho||s|| \\ \text{s.to} & A^T\lambda + s = c, \end{cases}$$

difficult neg. in optim-probl's

and later making positive the resulting negative components.

Termination conditions

- Look for a point that solves

$$\begin{aligned} A^T\lambda + s &= c \\ Ax &= b \\ XSe &= \mu e \\ (x,s) &> 0 \end{aligned} \iff \begin{aligned} r_c \triangleq A^T\lambda + s - c &= 0 \\ r_b \triangleq Ax - b &= 0 \\ XSe &= \mu e \\ (x,s) &> 0 \end{aligned}$$

for $\mu \approx 0$.

- In practice we report (x^k, λ^k, s^k) as optimum if

$$\frac{||r_c||}{(1+||c||)} \leq \epsilon_{feas} \quad \wedge \quad \frac{||r_b||}{(1+||b||)} \leq \epsilon_{feas} \quad \wedge \quad \mu \leq \epsilon_{opt}$$

where ϵ_{feas} and ϵ_{opt} are required feasibility and optimality tolerances (e.g., $\epsilon_{feas} = \epsilon_{opt} = 10^{-8}$).

Different primal and dual step lengths

- Theory suggests same step length for primal and dual spaces:

$$(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + \alpha(\Delta_x^k, \Delta_\lambda^k, \Delta_s^k)$$

That way we can prove $\mu_{k+1} < \mu_k$:

$$\begin{aligned} n\mu_{k+1} = (x^{k+1})^T s^{k+1} &= (x^k + \alpha\Delta_x^k)^T(s^k + \alpha\Delta_s^k) \\ &= (x^k)^T s^k + \alpha((x^k)^T\Delta_s^k + (s^k)^T\Delta_x^k) + \alpha^2(\Delta_x^k)^T\Delta_s^k) = \dots \\ &= (x^k)^T s^k(1-\alpha(1-\sigma)) = n\mu_k(1-\alpha(1-\sigma)) \end{aligned}$$

- However, in practice better to advance as much as possible in each space, computing different α_p and α_d:

$$\begin{aligned} \alpha_p &= \min\{1, \rho \cdot \max\{\alpha \geq 0 : x^k + \alpha\Delta_x^k \geq 0\}\} \\ \alpha_d &= \min\{1, \rho \cdot \max\{\alpha \geq 0 : s^k + \alpha\Delta_s^k \geq 0\}\} \end{aligned}$$

$\rho < 1$ is reduction parameter close to 1 (p.e., $\rho = 0.99$)

- Next point is thus:

$$\begin{aligned} x^{k+1} &= x^k + \alpha_p\Delta_x^k \\ (\lambda^{k+1}, s^{k+1}) &= (\lambda^k, s^k) + \alpha_d(\Delta_\lambda^k, \Delta_s^k) \end{aligned}$$

- In convex problems we may need to be more conservative, using same step length for primal and dual spaces.

Practical primal-dual path-following algorithm

Implementar en Matlab

Algorithm *primal-dual path-following IPM*
Initial point (x^0, λ^0, s^0), $\sigma = 0.1$, $k = 0$
while (x^k, λ^k, s^k) is not solution **do**
$\mu_k = (x^k)^T s^k / n$

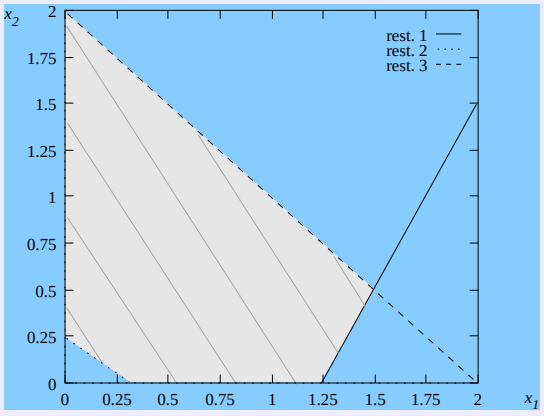
$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta_x^k \\ \Delta_\lambda^k \\ \Delta_s^k \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -X^k S^k + \sigma_k \mu_k e \end{bmatrix}$$

Compute α_p, α_d
$x^{k+1} = x^k + \alpha_p \Delta_x^k$
$(\lambda^{k+1}, s^{k+1}) = (\lambda^k, s^k) + \alpha_d (\Delta_\lambda^k, \Delta_s^k)$
$k := k + 1$
end_while
Return: $(x^*, \lambda^*, s^*) = (x^k, \lambda^k, s^k)$
End_algorithm

Example

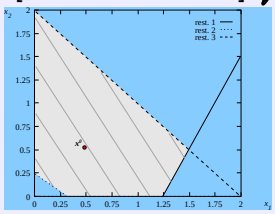
$$\begin{array}{ll} \min & -3x_1 - 2x_2 \\ \text{s.to} & 4x_1 - 2x_2 \le 5 \\ & 3x_1 + 4x_2 \ge 1 \\ & x_1 + x_2 \le 2 \\ & x_i \ge 0, i = 1, \ldots, 3 \end{array} \iff \begin{array}{ll} \min & -3x_1 - 2x_2 \\ \text{s.to} & 4x_1 - 2x_2 + x_3 = 5 \\ & 3x_1 + 4x_2 - x_4 = 1 \\ & x_1 + x_2 + x_5 = 2 \\ & x_i \ge 0, i = 1, \ldots, 5 \end{array}$$

5 variables in the primal space



Example: primal feasible starting point (I)

- Primal feasible starting point: $x^0 = [0.5 \;\; 0.5 \;\; 4 \;\; 2.5 \;\; 1]$, $\lambda = 0$, $s = 10e_5$.



- Compute r_b, r_c and μ_0:

$$r_b = Ax - b = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad r_c = A^T\lambda + s - c = \begin{bmatrix} 13 \\ 12 \\ 10 \\ 10 \\ 10 \end{bmatrix} \quad \mu_0 = (x^0)^T s^0/5 = 85/5 = 17$$

Solution not optimal. Iterate.

- Compute primal and dual directions $(\Delta_x^0, \Delta_\lambda^0, \Delta_s^0)$:

$$\Delta_x^0 = \begin{bmatrix} 0.0259 \\ 0.0604 \\ 0.0174 \\ 0.3194 \\ -0.0863 \end{bmatrix} \quad \Delta_\lambda^0 = \begin{bmatrix} -0.3815 \\ -0.5977 \\ -2.5633 \end{bmatrix} \quad \Delta_s^0 = \begin{bmatrix} -7.1175 \\ -7.8090 \\ -9.6185 \\ -10.5977 \\ -7.4367 \end{bmatrix}$$

Example: primal feasible starting point (II)

- Compute primal and dual step lengths, using $\rho = 0.99$:

$$\begin{aligned} \alpha_p^{\max} &= \max\{\alpha \geq 0 : x^0 + \alpha\Delta_x^0 \geq 0\} = 1/0.0863 = 11.584 \\ \alpha_p &= \min\{1, 0.99 \cdot 11.584\} = 1 \end{aligned}$$

$$\begin{aligned} \alpha_d^{\max} &= \max\{\alpha \geq 0 : s^0 + \alpha\Delta_s^0 \geq 0\} \\ &= \min\{10/7.1175, 10/7.8090, 10/9.6185, 10/10.5977, 10/7.4367\} = 0.9436 \\ \alpha_d &= \min\{1, 0.99 \cdot 0.9436\} = 0.9342 \end{aligned}$$

- Compute next point:

$$x^1 = x^0 + \alpha_p\Delta_x^0 = \begin{bmatrix} 0.5259 \\ 0.5604 \\ 4.0174 \\ 2.8194 \\ 0.9137 \end{bmatrix} \quad \lambda^1 = \lambda^0 + \alpha_d\Delta_\lambda^0 = \begin{bmatrix} -0.3564 \\ -0.5584 \\ -2.3945 \end{bmatrix} \quad s^1 = s^0 + \alpha_d\Delta_s^0 = \begin{bmatrix} 3.3510 \\ 2.7051 \\ 1.0148 \\ 0.1000 \\ 3.0529 \end{bmatrix}$$

Next point still primal feasible (feasibility not lost, once reached).

- Keep on iterating until optimum.

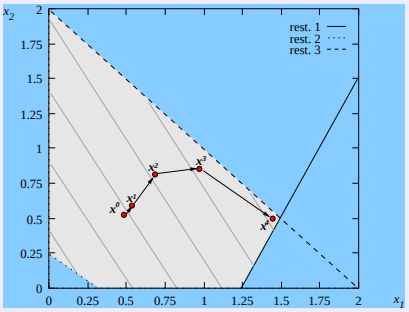
Example: primal feasible starting point (III)

Summary per iteration:

k	fprimal	fdual	rc	rb	mu
0	-2.500000	0.000000	5.375868	0.000000	17.00000
1	-2.698528	-7.129392	0.353922	0.000000	2.085254
2	-3.743938	-4.646623	0.076891	0.000000	0.477063
3	-5.114828	-5.440435	0.009454	0.000000	0.102437
4	-5.434246	-5.521104	0.000330	0.000000	0.018405
5	-5.497536	-5.506989	0.000000	0.000000	0.001891
6	-5.499620	-5.500565	0.000000	0.000000	0.000189
7	-5.499962	-5.500057	0.000000	0.000000	0.000019
8	-5.499996	-5.500006	0.000000	0.000000	0.000002
9	-5.500000	-5.500001	0.000000	0.000000	0.000000

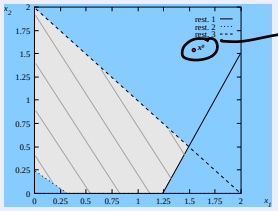
Primal variables per iteration:

k	xk_1	xk_2	xk_3	xk_4	xk_5
0	0.5000	0.5000	4.0000	2.5000	1.0000
1	0.5259	0.5604	4.0174	2.8194	0.9137
2	0.7240	0.7859	3.6757	4.3158	0.4900
3	1.1246	0.8705	2.2424	5.8558	0.0049
4	1.4429	0.5528	0.3339	5.5398	0.0043
5	1.4992	0.5000	0.0033	5.4976	0.0008
6	1.4998	0.5001	0.0011	5.4999	0.0001
7	1.5000	0.5000	0.0001	5.5000	0.0000
8	1.5000	0.5000	0.0000	5.5000	0.0000
9	1.5000	0.5000	0.0000	5.5000	0.0000



Example: primal and dual infeasible starting point (I)

- Infeasible starting point: $x^0 = [1.5 \;\; 1.5 \;\; 1 \;\; 1 \;\; 1]$, $y = 0$, $s = 10e_5$.



- Compute r_b, r_c and μ_0:

$$r_b = Ax - b = \begin{bmatrix} -1 \\ 8.5 \\ 2 \end{bmatrix} \quad r_c = A^T y + s - c = \begin{bmatrix} 13 \\ 12 \\ 10 \\ 10 \\ 10 \end{bmatrix} \quad \mu_0 = (x^0)^T s^0 / 5 = 60/5 = 12$$

Solution not optimal. Iterate.

- Compute primal and dual directions $(\Delta_x^0, \Delta_\lambda^0, \Delta_s^0)$:

$$\Delta_x^0 = \begin{bmatrix} -0.5799 \\ -1.5794 \\ 0.1606 \\ 0.4428 \\ 0.1593 \end{bmatrix} \quad \Delta_\lambda^0 = \begin{bmatrix} 0.4060 \\ -3.2275 \\ 0.3928 \end{bmatrix} \quad \Delta_s^0 = \begin{bmatrix} -5.3343 \\ 1.3294 \\ -10.4060 \\ -13.2275 \\ -10.3928 \end{bmatrix}$$

Example: primal and dual infeasible starting point (II)

- Compute primal and dual step lengths, using $\rho = 0.99$:

$$\begin{aligned} \alpha_p^{\max} &= \max\{\alpha \geq 0 : x^0 + \alpha\Delta_x^0 \geq 0\} \\ &= \min\{1.5/0.5799, 1.5/1.5794\} = 0.9497 \\ \alpha_p &= \min\{1, 0.99 \cdot 0.9497\} = 0.9402 \end{aligned}$$

$$\begin{aligned} \alpha_d^{\max} &= \max\{\alpha \geq 0 : s^0 + \alpha\Delta_s^0 \geq 0\} \\ &= \min\{10/5.3343, 10/10.4060, 10/13.2275, 10/10.3928\} = 0.7560 \\ \alpha_d &= \min\{1, 0.99 \cdot 0.7560\} = 0.7484 \end{aligned}$$

- Compute next point:

$$x^1 = x^0 + \alpha_p\Delta_x^0 = \begin{bmatrix} 0.9548 \\ 0.0150 \\ 1.1510 \\ 1.4163 \\ 1.1498 \end{bmatrix} \quad \lambda^1 = \lambda^0 + \alpha_d\Delta_\lambda^0 = \begin{bmatrix} 0.3039 \\ -2.4156 \\ 0.2940 \end{bmatrix} \quad s^1 = s^0 + \alpha_d\Delta_s^0 = \begin{bmatrix} 6.0076 \\ 10.9950 \\ 2.2117 \\ 0.1000 \\ 2.2217 \end{bmatrix}$$

- Keep on iterating until optimum.

Example: primal and dual infeasible starting point (III)

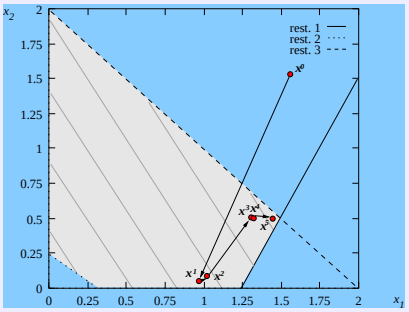
Summary per iteration:

k	fprimal	fdual	rc	rb	mu
0	-7.500000	0.000000	5.375868	1.356939	12.000000
1	-2.894414	-0.308251	1.352361	0.081117	2.228544
2	-3.182715	-3.305456	0.297756	0.000000	0.635964
3	-5.340267	-3.397607	0.244089	0.000000	0.425748
4	-5.312748	-5.616408	0.000000	0.000000	0.060732
5	-5.489901	-5.520267	0.000000	0.000000	0.006073
6	-5.498768	-5.501805	0.000000	0.000000	0.000607
7	-5.499879	-5.500182	0.000000	0.000000	0.000061
8	-5.499988	-5.500018	0.000000	0.000000	0.000006
9	-5.499999	-5.500002	0.000000	0.000000	0.000001

Dual feas. point

Primal variables per iteration:

k	xk_1	xk_2	xk_3	xk_4	xk_5
0	1.5000	1.5000	1.0000	1.0000	1.0000
1	0.9548	0.0150	1.1510	1.4163	1.1498
2	1.0347	0.0393	0.9398	2.2613	0.9260
3	1.3588	0.6320	0.8288	5.6042	0.0093
4	1.3703	0.6010	0.7210	5.5147	0.0288
5	1.4965	0.5003	0.0146	5.4904	0.0033
6	1.4993	0.5004	0.0034	5.4995	0.0003
7	1.4999	0.5000	0.0004	5.5000	0.0000
8	1.5000	0.5000	0.0000	5.5000	0.0000
9	1.5000	0.5000	0.0000	5.5000	0.0000



System 1: $\nabla F(x, \lambda, s)\Delta = -F(x)$

- System to be solved at each iteration is:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta_x \\ \Delta_\lambda \\ \Delta_s \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -r_{xs} \end{bmatrix}$$

where

$$\begin{aligned} r_c &= A^T\lambda + s - c \\ r_b &= Ax - b \\ r_{xs} &= XSe - \sigma\mu e \end{aligned}$$

Some practical variants (Mehrotra's direction, to be seen later) use alternative RHS, but what we will do is still valid.

- System of $2n + m$ variables and equations. Neither symmetric, nor positive definite: needs LU factorization.
- In practice it is not used, we perform Gaussian elimination of some variables.

System 2: augmented system

- Reordering system 1 we get:

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta_x \\ \Delta_\lambda \\ \Delta_s \end{bmatrix} = \begin{bmatrix} -r_b \\ -r_c \\ -r_{xs} \end{bmatrix}$$

- Eliminating Δ_s from last block of equations (add last row times X^{-1} to second row), we get:

$$\Delta_s = -X^{-1}(r_{xs} + S\Delta_x)$$

and

$$\begin{bmatrix} A & 0 \\ -\Theta^{-1} & A^T \end{bmatrix} \begin{bmatrix} \Delta_x \\ \Delta_\lambda \end{bmatrix} = \begin{bmatrix} -r_b \\ -r_c + X^{-1}r_{xs} \end{bmatrix}, \quad \Theta = XS^{-1}$$

- System of $n + m$ variables and constraints, symmetric, not positive definite (it is indefinite). Submatrix Θ is diagonal and positive definite.
- Solved using Bunch-Parlett factorization (1×1 and 2×2 pivots)

$$P \begin{bmatrix} A & 0 \\ -\Theta^{-1} & A^T \end{bmatrix} P^T = LDL^T$$

P is permutation matrix, and D is indefinite block-diagonal matrix with 1×1 and 2×2 blocks.

System 3: normal equations

- Adding last block of augmented system times $A\Theta$ to first block, we get normal equations (first solve for Δ_λ, next Δ_s i Δ_x):

$$\begin{aligned}(A\Theta A^T)\Delta_\lambda &= -r_b + A\Theta(-r_c + X^{-1}r_{xs}) &&= -r_b + A(-XS^{-1}r_c + S^{-1}r_{xs})\\ \Delta_s &= -r_c - A^T\Delta_\lambda && [\text{ from } A^T\Delta_\lambda + \Delta_s = -r_c]\\ \Delta_x &= -S^{-1}(r_{xs} + X\Delta_s) && [\text{ from } S\Delta_x + X\Delta_s = -r_{xs}]\end{aligned}$$

- Most expensive step is solution of $A\Theta A^T$. Symmetric, positive definite system (if A has full row rank) solved with sparse Cholesky factorization.
- In practice, normal equations more effective for most LP problems (Cholesky factorization very efficient).
- For some LP, QP and general Nonlinear Programming, we will see that augmented system preferred.

Reminder: Cholesky factorization

$M = A\Theta A^T$ symmetric positive definite: find factorization $M = LDL^T$, D diagonal and positive:

$$\begin{bmatrix} m_{11} & m_{21} & m_{31}\\ m_{21} & m_{22} & m_{32}\\ m_{31} & m_{32} & m_{33}\end{bmatrix} = \begin{bmatrix} 1 & & \\ l_{21} & 1 & \\ l_{31} & l_{32} & 1\end{bmatrix}\begin{bmatrix} d_1 & & \\ & d_2 & \\ & & d_3\end{bmatrix}\begin{bmatrix} 1 & l_{21} & l_{31}\\ & 1 & l_{32}\\ & & 1\end{bmatrix}$$

$$= \begin{bmatrix} d_1 & d_1 l_{21} & d_1 l_{31}\\ d_1 l_{21} & d_2 + d_1 l_{21}^2 & d_2 l_{32} + d_1 l_{21} l_{31}\\ d_1 l_{31} & d_2 l_{32} + d_1 l_{21} l_{31} & d_3 + d_1 l_{31}^2 + d_2 l_{32}^2\end{bmatrix} \Rightarrow \begin{cases} col.1: & d_1 = m_{11} \quad l_{21} = m_{21}/d_1 \quad l_{31} = m_{31}/d_1\\ col.2: & d_2 = m_{22} - d_1 l_{21}^2 \quad l_{32} = (m_{32} - d_1 l_{21} l_{31})/d_2\\ col.3: & d_3 = m_{33} - d_1 l_{31}^2 - d_2 l_{32}^2\end{cases}$$

Algorithm *Cholesky*
  for $j = 1$ to n
    $c_{jj} = m_{jj} - \sum_{s=1}^{j-1} d_s l_{js}^2$
    $d_j = c_{jj}$
    for $i = j+1$ to n
      $c_{ij} = m_{ij} - \sum_{s=1}^{j-1} d_s l_{is} l_{js}$
      $l_{ij} = c_{ij}/d_j$
    end_for
  end_for
End_algorithm

Properties of Cholesky factorization

- No need to use complete or partial pivoting (unlike Gaussian elimination): all diagonal pivots are > 0 and matrix symmetry is preserved. This does not happen in LU factorization of general matrices, e.g.:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

- Cholesky factorization is numerically stable, even without row/column pivoting. If $\tilde{z}$ is the numerical solution and z the exact solution then

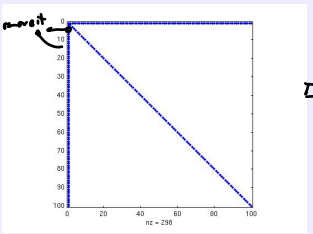
$$\frac{||\tilde{z} - z||}{||z||} \approx \kappa(M)u$$

where u is the computer precision (10^{-14} for double precision computations) and $\kappa(M) = ||M||_2||M^{-1}||_2$ is the condition number of matrix M.

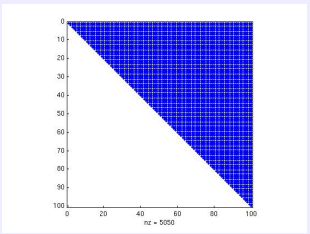
- We can compute factorization of $M = P(A\Theta A^T)P^T$, where P is a permutation matrix, instead of that of $M = A\Theta A^T$: the good properties of Cholesky do not change.

Difficulty 1 factoring $A\Theta A^T$: "fill-in"

- $A\Theta A^T$ sparse in general (stored in sparse structures). Factorization can degrade sparsity: fill-in (amount of new nonzero elements) can be large.

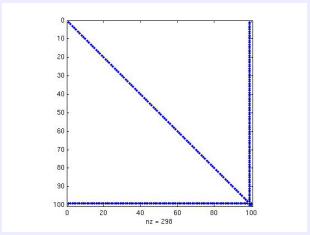


pattern of $A\Theta A^T$

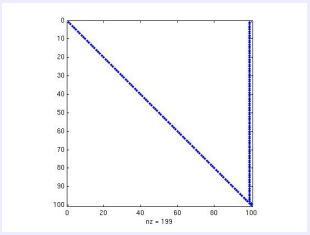


pattern of L : $LDL^T = A\Theta A^T$

- Solution: obtain a permutation matrix P of rows and columns. This matrix is valid for all iterations, since sparsity pattern of $A\Theta A^T$ is always the same, only Θ changes at each iteration. In example, first and last row/column need only to be interchanged:



pattern of $PA\Theta A^T P^T$



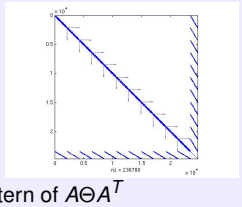
pattern of L : $LDL^T = PA\Theta A^T P^T$

How to compute P?

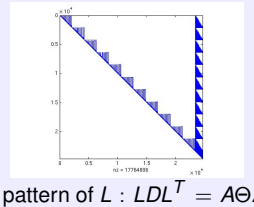
- Looking for the best P (that minimizing fill-in) is a difficult combinatorial problem. In practice three heuristics used:
- Minimum-degree ordering
  - Equivalent to Markovitz pivoting rule in standard LU factorization. Represents symmetric matrix as a graph of m nodes and $(nnz - m)/2$ arcs, then first chooses as pivots nodes of minimum degree. E.g., graph of matrix $A\Theta A^T$ of previous slide is a "star" with node 1 at center, connected to remaining $m-1$ with a single arc; node 1 has maximum degree, then left for final pivot.
  - Matlab implements this heuristic in function `symamd`.
  - Mostly used in IPMs.
- Minimum local fill-in.
  - Minimum-degree ordering overestimates the number of new nonzeroes, since it does not considers that some nonzeroes already existed. Minimum local fill-in takes this into account: tries to determine the number of new nonzeroes, looking ahead only at one elimination.
  - More computationally expensive that minimum-degree ordering.
- Nested dissection.
  - Represents symmetric matrix as a graph and tries to eliminate "separator" nodes such that, without them, we would get two independent subgraphs. The procedure is recursively applied for each subgraph.
  - Used in packages like METIS (METIS implements multilevel nested dissection).
  - Very efficient in some problems.

Example comparing MDO and ND: PDS15 LP problem

- Original matrix $A\Theta A^T$ of size 24634 and 236780 nonzeros; L has 17.764.806 nonzeros.

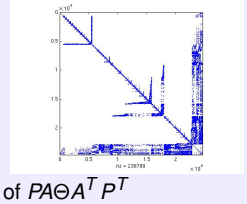


pattern of $A\Theta A^T$

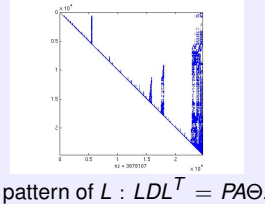


pattern of $L : LDL^T = A\Theta A^T$

- Permutation matrix with minimum-degree ordering; L has 3.878.107 nonzeros.

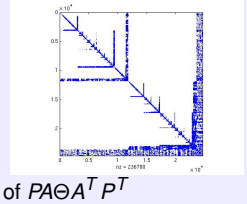


pattern of $PA\Theta A^T P^T$

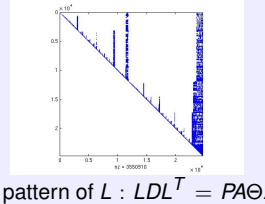


pattern of $L : LDL^T = PA\Theta A^T P^T$

- Permutation matrix with nested dissection (METIS); L has 3.550.910 nonzeros.



pattern of $PA\Theta A^T P^T$



pattern of $L : LDL^T = PA\Theta A^T P^T$

Computing P, symbolic and numerical factorization

(lab assignment)

In practice, IPM codes do next steps for efficient solution of $A\Theta A^T$:

- Before iterations start:
  - Compute column/row ordering P to reduce fill-in.
  - Symbolic factorization to detect new nonzeros and allocate space in memory for data structures.
- At each iteration:
  - Numerical factorization $LDL^T = PA\Theta A^T P^T$
  - Solution of $A\Theta A^T \Delta_\lambda = \bar{b}$:

$$\begin{aligned} A\Theta A^T \Delta_\lambda &= v \\ (PA\Theta A^T P^T)(P\Delta_\lambda) &= Pv \\ LDL^T u &= (Pv) \qquad [u = P\Delta_\lambda] \\ \Delta_\lambda &= P^T u \end{aligned}$$

- Some known packages for sparse Cholesky factorizations are: SprsblkIlt (Ng-Peyton), Cholmod (Tim Davis) (used in Matlab), WSSMP (A. Gupta), Pardiso (O. Schenk).

Difficulty 2 factoring $A\Theta A^T$: 0 pivots

- $A\Theta A^T$ may be positive semidefinite, not positive definite, and 0 pivots can be found if:
  - A is not full-row rank:

$$u \neq 0, u^T A A^T u = ||u^T A|| = 0 \Leftrightarrow \sum_{i=1}^{m} u_i A_{i.} = 0 \Leftrightarrow rank(A) < m$$

  - Solution is degenerate vertex (degenerate basic solution), i.e., $x = [x_B \;\; x_N = 0]$ and x_B less than m nonzero components:

$$A\Theta A^T = (A_B \Theta_B A_B^T) + (A_N \Theta_N A_N^T) \approx (A_B \Theta_B A_B^T) \text{ rank defficient since } \Theta_B = X_B S_B^{-1}$$

- In practice, when a zero pivot $d_i = 0$ is found in Cholesky factorization, it is replaced by large value $d_i = 10^{64}$; subdiagonal elements for $j > i$ are $l_{ij} = c_{ij}/d_j \approx 0$. Result is to make 0 component i of Δ_λ and to remove row/column i of $A\Theta A^T$ (redundant constraint of A):

Algorithm *Cholesky-semidefinite*
  for $j = 1$ to n
    $c_{jj} = m_{jj} - \sum_{s=1}^{j-1} d_s l_{js}^2$
    if $(c_{jj} > 0)$ then $d_j = c_{jj}$ else $d_j = 10^{64}$ end_if
    for $i = j + 1$ to n
      $c_{ij} = m_{ij} - \sum_{s=1}^{j-1} d_s l_{is} l_{js}$
    $l_{ij} = c_{ij}/d_j$
    end_for
  end_for
End_algorithm

- Matlab implements this option as `cholinc(M,'inf')`

Difficulty 3 factoring $A\Theta A^T$: dense columns

- If A has at least one dense columns then $A\Theta A^T$ is a dense matrix: computationally very expensive.
- The two strategies are:
  - Anyway, factorize $A\Theta A^T$ as the sum of sparse matrix plus low rank additional matrix: If $A = [A_S \; A_D]$ (A_S contains sparse columns, A_D dense ones) then

$$A\Theta A^T = A_S\Theta_S A_S^T + A_D\Theta_D A_D^T.$$

    $A_S\Theta_S A_S^T$ is sparse, $A_D\Theta_D A_D^T$ is a low-rank matrix.
    Exploit this using Sherman-Morrisson-Woodbury formula.
  - Use augmented system, insensible to dense column.
    In addition, augmented system useful for non-LP problems (QP, nonlinear convex problem...).

(aputs)

opots dividir per 2 el nº d'iteracions

Mehrotra predictor-corrector direction

The two key ingredients are [Mehrotra (1992)]:

Heurístic (no és segur que convergeixi, però dona bons resultats)

- Adaptive adjustment of centrality parameter σ at each iteration.
- Second-order Taylor approximation of KKT-μ (extensió Newton's method a grau 2)

$$F(x,\lambda,s) = \begin{bmatrix} A^T\lambda + s - c \\ Ax - b \\ XSe - \sigma\mu e \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$
$$(x,s) > 0$$

Remind first-order Taylor approximation was:

$$F(x+\Delta_x, \lambda+\Delta_\lambda, s+\Delta_s) \approx F(x,\lambda,s) + \nabla F(x,\lambda,s)(\Delta_x,\Delta_\lambda,\Delta_s)^T$$

giving rise to linear system for first-order direction:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta_x \\ \Delta_\lambda \\ \Delta_s \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe + \sigma\mu e \end{bmatrix}$$

- There are other methods based on multiple centrality corrections [Gondzio (1996)].

Second-order Taylor approximation

Given $F : \mathbb{R}^{2n+m} \to \mathbb{R}^{2n+m}$

$$F(x, \lambda, s) = \begin{bmatrix} A^T\lambda + s - c \\ Ax - b \\ XSe - \sigma\mu e \end{bmatrix}$$

consider each $F_i(w), i = 1, \ldots, 2n + m$ where $w = (x, \lambda, s)$, and its second-order Taylor approximation:

$$F_i(w + \Delta_w) \approx F_i(w) + \nabla F_i(w)\Delta_w + \frac{1}{2}\Delta_w^T \nabla^2 F_i(w)\Delta_w$$

The only $F_i(w)$ with non-empty Hessian are $F_i(w) = x_i s_i - \sigma\mu$: entry 1 in components $\nabla^2_{x_i,s_i}$ and $\nabla^2_{s_i,x_i}$, and 0 otherwise. The resulting system for the second-order direction is:

solve it approx ($\Delta x \Delta s$ → nonlinear)

$$\begin{cases} A^T\Delta_\lambda + \Delta_s = -r_c \\ A\Delta_x = -r_b \\ S\Delta_x + X\Delta_s + \Delta_X\Delta_S e = -XSe + \sigma\mu e \end{cases}$$

How to solve this nonlinear system? Mehrotra suggested a heuristic for approximate solution.

Solution in three steps

The three directions:

- Predictor direction: 1 linear system of equations.
- Centering direction: 1 linear system of equations.
- Corrector direction: 1 linear system of equations.

predictor (affine direction) → 1 system of; cent + correct → 1 system of eq's

In practice the centering and corrector directions are computed together in a single system of equations: we only need two systems of equations.
In addition these two systems have same coefficient matrix, only RHS changes: we only need to perform one factorization.

Predictor direction

Remind nonlinear system for second-order direction is

$$\begin{aligned} A^T\Delta_\lambda + \Delta_s &= -r_c \\ A\Delta_x &= -r_b \\ S\Delta_x + X\Delta_s + \Delta_X\Delta_S e &= -XSe + \sigma\mu e \end{aligned}$$

Remove nonlinear term $\Delta_X\Delta_S$ and consider $\sigma = 0$: optimality or affine-scaling direction. We solve the resulting linear system:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta_x^{aff} \\ \Delta_\lambda^{aff} \\ \Delta_s^{aff} \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe \end{bmatrix}$$

Next compute

$$\begin{aligned} \alpha_p^{aff} &= \max\{\alpha \in [0,1] : x + \alpha\Delta_x^{aff} \geq 0\} \\ \alpha_d^{aff} &= \max\{\alpha \in [0,1] : s + \alpha\Delta_s^{aff} \geq 0\} \end{aligned}$$

and compute complementarity of resulting point:

$$\mu^{aff} = (x + \alpha_p^{aff}\Delta_x^{aff})^T(s + \alpha_d^{aff}\Delta_s^{aff})/n$$

If $\mu^{aff} \ll \mu$ affine direction is fine: progress made in that direction. Then $0 < \sigma \ll 1$: no need for excessive centering. If $\mu^{aff} \approx \mu$ affine direction is bad: no progress in that direction. Then $1 > \sigma \gg 0$: need to center point. Proposal:

$$\sigma = \left(\mu^{aff}/\mu\right)^3$$

Centering direction

Once σ computed in predictor step, we compute centering direction $(\Delta_x^{cen}, \Delta_\lambda^{cen}, \Delta_s^{cen})$ solving:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta_x^{cen} \\ \Delta_\lambda^{cen} \\ \Delta_s^{cen} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sigma\mu e \end{bmatrix}$$

Observe direction $(\Delta_x^{aff} + \Delta_x^{cen}, \Delta_\lambda^{aff} + \Delta_\lambda^{cen}, \Delta_s^{aff} + \Delta_s^{cen})$ is solution of

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta_x^{aff+cen} \\ \Delta_\lambda^{aff+cen} \\ \Delta_s^{aff+cen} \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe + \sigma\mu e \end{bmatrix}$$

In practice $(\Delta_x^{cen}, \Delta_\lambda^{cen}, \Delta_s^{cen})$ not computed independently, but together with next corrector direction.

Corrector direction (I)

Second order approximation of equations $XSe = 0$ ($x_i s_i = 0,\ i = 1, \ldots, n$) is:

$$XSe + S\Delta_x + X\Delta_s + \Delta_X \Delta_S e = 0 \quad \Leftrightarrow \quad x_i s_i + s_i \Delta_{x_i} + x_i \Delta_{s_i} + \Delta_{x_i} \Delta_{s_i} = 0,\ i = 1, \ldots, n$$

How to approximately compute directions that satisfy above conditions?
Justification: moving along affine direction with step length 1 we get:

$$(x_i + \Delta_{x_i}^{aff})(s_i + \Delta_{s_i}^{aff}) = x_i s_i + x_i \Delta_{s_i}^{aff} + s_i \Delta_{x_i}^{aff} + \Delta_{x_i}^{aff} \Delta_{s_i}^{aff} = \Delta_{x_i}^{aff} \Delta_{s_i}^{aff}$$

Above equality obtained because one of equations of predictor step was $s_i \Delta_{x_i}^{aff} + s_i \Delta_{s_i}^{aff} = -x_i s_i$. Therefore, using affine direction we don't have the desired $(x_i + \Delta_{x_i}^{aff})(s_i + \Delta_{s_i}^{aff}) = 0$, but $(x_i + \Delta_{x_i}^{aff})(s_i + \Delta_{s_i}^{aff}) = \Delta_{x_i}^{aff} \Delta_{s_i}^{aff}$. Corrector step tries to compensate for this deviation solving:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta_x^{co} \\ \Delta_\lambda^{co} \\ \Delta_s^{co} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -\Delta_X^{aff} \Delta_S^{aff} e \end{bmatrix}$$

Corrector direction(II)

- Effect of corrector step is observed comparing error with predictor step and with predictor-corrector step in last equation:

$$(x_i + \Delta_{x_i}^{aff})(s_i + \Delta_{s_i}^{aff}) = x_i s_i + x_i \Delta_{s_i}^{aff} + s_i \Delta_{x_i}^{aff} + \Delta_{x_i}^{aff} \Delta_{s_i}^{aff} = \Delta_{x_i}^{aff} \Delta_{s_i}^{aff}$$

$$\begin{aligned} &(x_i + \Delta_{x_i}^{aff} + \Delta_{x_i}^{co})(s_i + \Delta_{s_i}^{aff} + \Delta_{s_i}^{co}) \\ &= x_i s_i + x_i \Delta_{s_i}^{aff} + s_i \Delta_{x_i}^{aff} + \Delta_{x_i}^{aff} \Delta_{s_i}^{aff} + s_i \Delta_{x_i}^{co} + \Delta_{s_i}^{aff} \Delta_{x_i}^{co} + \Delta_{s_i}^{co} \Delta_{x_i}^{co} + x_i \Delta_{s_i}^{co} + \Delta_{x_i}^{aff} \Delta_{s_i}^{co} \\ &= \Delta_{s_i}^{aff} \Delta_{x_i}^{co} + \Delta_{s_i}^{co} \Delta_{x_i}^{co} + \Delta_{x_i}^{aff} \Delta_{s_i}^{co} \end{aligned}$$

- Since RHS for predictor step is $-XSe$ then $||(\Delta_X^{aff}, \Delta_S^{aff}|| = O(\mu)$.
- Since RHS for corrector step is $-\Delta_X^{aff} \Delta_S^{aff} e$ then $||(\Delta_X^{co}, \Delta_S^{co}|| = O(\mu)^2$. Then when $\mu \to 0$, error smaller using predictor-corrector direction.

Predictor-corrector direction: summary

In practice we do:

- Compute predictor step:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta_x^{aff} \\ \Delta_\lambda^{aff} \\ \Delta_s^{aff} \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe \end{bmatrix}$$

- Compute σ and μ
- Compute together centering and corrector step:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta_x^{cc} \\ \Delta_\lambda^{cc} \\ \Delta_s^{cc} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sigma\mu e - \Delta_X^{aff}\Delta_S^{aff} e \end{bmatrix}$$

- Use direction:

$$(\Delta_x, \Delta_\lambda, \Delta_s) = (\Delta_x^{aff} + \Delta_x^{cc}, \Delta_\lambda^{aff} + \Delta_\lambda^{cc}, \Delta_s^{aff} + \Delta_s^{cc})$$

- In practice number of iterations is approximately halved. Two systems, but share matrix: only one factorization, then computational cost for iteration is not doubled.

Primal-dual path-following with Mehrotra's direction

Algorithm *Mehrotra predictor-corrector*

Starting point (x^0, λ^0, s^0), $k = 0$

while (x^k, λ^k, s^k) is not solution **do**

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta_x^{aff} \\ \Delta_\lambda^{aff} \\ \Delta_s^{aff} \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -X^k S^k e \end{bmatrix}$$

Compute α_p^{aff}, α_d^{aff}

$\mu^{aff} = (x^k + \alpha_p^{aff}\Delta_x^{aff})^T(s^k + \alpha_d^{aff}\Delta_s^{aff})/n$

$\mu = (x^k)^T s^k / n$

$\sigma = \left(\mu^{aff}/\mu\right)^3$

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta_x^{cc} \\ \Delta_\lambda^{cc} \\ \Delta_s^{cc} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sigma\mu e - \Delta_X^{aff}\Delta_S^{aff} e \end{bmatrix}$$

$(\Delta_x, \Delta_\lambda, \Delta_s) = (\Delta_x^{aff} + \Delta_x^{cc}, \Delta_\lambda^{aff} + \Delta_\lambda^{cc}, \Delta_s^{aff} + \Delta_s^{cc})$

Compute α_p, α_d

$x^{k+1} = x^k + \alpha_p \Delta_x$

$(\lambda^{k+1}, s^{k+1}) = (\lambda^k, s^k) + \alpha_d(\Delta_\lambda, \Delta_s)$

$k := k + 1$

end_while

Return: $(x^*, \lambda^*, s^*) = (x^k, \lambda^k, s^k)$

End_algorithm

First-order vs Mehrotra's second-order direction

Some Netlib results with home-made C implementation and Ng-Peyton Cholesky solver:

			Second-order		First-order	
Instance	rows	columns	Iter	CPU	Iter	CPU
80bau3b	2263	9799	33	41.4	71	61.3
bnl2	2325	3489	39	85.9	61	124.7
d2q06c	2172	5167	34	167.2	58	263.3
degen3	1504	1818	24	136.9	38	165.6
fit2d	26	10500	24	28.2	48	44.7
ganges	1310	1681	26	11.7	41	15.2
maros	847	1443	49	18.7	65	22.2

CPU times on old SunSparc 10/41 (year 1998)

IPMs vs simplex: number of iterations

- The complexity of IPMs was
  - Number of iterations of short-step method:
    $$O(\sqrt{n}\ \log 1/\varepsilon)$$
  - Number of iterations of long-step method:
    $$O(n\ \log 1/\varepsilon)$$
  - Number of iterations of infeasible long-step method (used in practice):
    $$O(n^2\ \log 1/\varepsilon)$$
- It could be argued this is *worst*, than simplex, which in practice requires (if problem not too degenerate) a linear number of iterations.
- But, also in practice, IPMs rarely need more than 100 iterations, and for most problems 50 or less are enough.
- However, according to [Bixby (2002)] dual simplex outperforms primal simplex and IPMs in significant instances...

IPM vs simplex: practical guidelines

- For very large and degenerate LP problems: too many simplex iterations, use IPMs.
- If A contains many dense columns, or $A\Theta A^T$ is too dense: IPM iterations expensive, use simplex.
- Simplex is sensitive to degeneracy, but insensitive to infeasibility or unboundness. The opposite holds for IPMs.
- If a basic feasible solution is needed for reoptimization (in integer programming, for instance): either use simplex, or IPM+crossover. In some integer problems, it may be worth using IPM+crossover at first branch-and-bound node, and dual simplex for the rest of nodes.
- For convex quadratic problems, and general convex problems: use IPMs.

IPM vs simplex: a data confidentiality problem

- Results for LP statistical data protection problem with CPLEX 10 [Castro (2006)]:

Instance	rows	columns	Iter Simplex	CPU simplex	Iter IP	CPU IP
bts4	73140	36310	20393	16.5	43	39.7
hier13	4040	3313	2697	3.2	20	6.9
hier16	7128	5484	6673	19.9	17	28.4
nine12	20798	11362	35864	382.1	21	47.4
nine5d	21466	17295	15070	126.7	25	43.0
ninenew	13092	7340	9228	27.1	19	24.0
two5in6	11362	9629	5973	13.6	20	16.9

Results on Pentium 1.8Ghz

Quadratic problems

- QP primal and dual, m constraints, n variables

$$\begin{array}{lll} & \min & c^T x + 1/2 x^T Q x \\ (P) & \text{s.to} & Ax = b \quad [\lambda] \\ & & x \geq 0 \quad [s] \end{array} \qquad \begin{array}{lll} & \max & b^T \lambda \\ (D) & \text{s.to} & A^T \lambda - Qx + s = c \\ & & s \geq 0 \end{array}$$

- KKT-μ conditions:

$$\begin{array}{ll} A^T\lambda - Qx + s = c & \text{[dual feasibility]} \\ Ax = b & \text{[primal feasibility]} \\ XSe = \sigma\mu e & \text{[complementarity]} \\ (x, s) > 0 & \end{array}$$

- System for directions: Newton's direction

in linear case $=0$

$$\begin{bmatrix} -Q & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta_x \\ \Delta_\lambda \\ \Delta_s \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe + \sigma\mu e \end{bmatrix}$$

Augmented system and normal equations for QPs

$\Theta_{LP} = (X^{-1}S)^{-1} = XS^{-1}$ (easy)

- Augmented system:

$$\Delta_s = -X^{-1}(r_{xs} + S\Delta_x)$$

and

$$\begin{bmatrix} A & 0 \\ -\Theta^{-1} & A^T \end{bmatrix} \begin{bmatrix} \Delta_x \\ \Delta_\lambda \end{bmatrix} = \begin{bmatrix} -r_b \\ -r_c + X^{-1} r_{xs} \end{bmatrix}, \quad \Theta = (Q + X^{-1}S)^{-1}$$

difficult

diag.

can be computed very fast if Q is diag, is easy

- Normal equations:

$$\begin{array}{rll} (A\Theta A^T)\Delta_\lambda & = -r_b + A\Theta(-r_c + X^{-1} r_{xs}) & = -r_b + A(-XS^{-1} r_c + S^{-1} r_{xs}) \\ \Delta_s & = -r_c - A^T \Delta_\lambda & [\text{ from } A^T\Delta_\lambda + \Delta_s = -r_c] \\ \Delta_x & = -S^{-1}(r_{xs} + X\Delta_s) & [\text{ from } S\Delta_x + X\Delta_s = -r_{xs}] \end{array}$$

- Similar to linear case but Q may increase fill-in of $(A\Theta A^T)$: $(A\Theta A^T)$ almost dense in practice!
- Solution is to use augmented system for QPs.
- One exception: for separable QPs (Q is diagonal, e.g. $\min_x \sum_{i=1}^n a_i x_i^2$) $\Theta = (Q + X^{-1}S)^{-1}$ is still a diagonal matrix. We can proceed as in LP: normal equations.

Problems with convex objective function

- Primal and dual when $f(x) : \mathbb{R}^n \to \mathbb{R}$ convex, m constraints, n variables

$$(P) \quad \begin{array}{lll} \min & f(x) & \\ \text{s.to} & Ax = b & [\lambda] \\ & x \geq 0 & [s] \end{array} \qquad (D) \quad \begin{array}{lll} \max & L(x,\lambda,s) = f(x) + \lambda^T(b - Ax) - s^T x & \\ \text{s.to} & \nabla_x L(x,\lambda,s) = A^T\lambda - \nabla f(x) + s = 0 & \\ & s \geq 0 & \end{array}$$

- KKT-μ conditions:

$$\begin{array}{ll} A^T\lambda - \nabla f(x) + s = 0 & \text{[dual feasibility]} \\ Ax = b & \text{[primal feasibility]} \\ XSe = \sigma\mu e & \text{[complementarity]} \\ (x,s) > 0 & \end{array}$$

- System for directions:

$$\begin{bmatrix} -\nabla^2 f(x) & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta_x \\ \Delta_\lambda \\ \Delta_s \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe + \sigma\mu e \end{bmatrix}$$

Same equations that in quadratic case, but Hessian changes at each iteration.

General convex problems

- Primal and dual $f(x) : \mathbb{R}^n \to \mathbb{R}$ convex, $g(x) : \mathbb{R}^n \to \mathbb{R}^m$ convex, m constraints, n variables

$$(P) \quad \begin{array}{lll} \min & f(x) & \\ \text{s.to} & g(x) \leq 0 & [\lambda] \end{array} \qquad (D) \quad \begin{array}{lll} \max & L(x,\lambda) = f(x) + \lambda g(x) & \\ \text{s.to} & \nabla f(x) + \lambda \nabla g(x) = 0 & \\ & \lambda \geq 0 & \end{array}$$

- KKT-μ conditions:

$$\begin{array}{ll} \nabla f(x) + \lambda^T \nabla g(x) = 0 & \text{[dual feasibility]} \\ g(x) + s = 0 & \text{[primal feasibility]} \\ \Lambda Se = \sigma\mu e & \text{[complementarity]} \\ (\lambda, s) > 0 & \end{array}$$

- System for directions:

$$\begin{bmatrix} \nabla^2 f(x) + \lambda^T \nabla^2 g(x) & \nabla g(x)^T & 0 \\ \nabla g(x) & 0 & I \\ 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta_x \\ \Delta_\lambda \\ \Delta_s \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -\Lambda Se + \sigma\mu e \end{bmatrix}$$

Augmented system should be used as for QP.

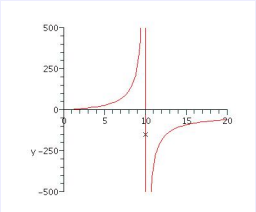
Example of application: the routing data problem

- Let $G = (V, E)$ be a directed/undirected graph with n edges and $m+1$ nodes representing a telecommunication network with k commodities or requirements of demand $b^i, i = 1, \ldots, k$. N is node-arc incidence matrix for G.
- Each link $j \in E$ has a mutual capacity b^0_j.
- Assume graph undirected, flows are $x^{k+},\ x^{k-} \geq 0$.
- The nonoriented convex data routing problem is:

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & Nx^{i+} - Nx^{i-} = b^i, \quad i = 1, \ldots, k, \\ & \sum_{i=1}^{k} (x^{i+} + x^{i-}) = x^0, \\ & x^0 \leq b^0, \\ & x^{i+},\ x^{i-} \geq 0, \quad i = 1, \ldots, k. \end{array}$$

- Objective is Kleinrock average delay function.reliability term:

$$f(x) \triangleq \sum_{j=1}^{n} \frac{x^0_j}{b^0_j - x^0_j}$$



Results for some small instances: IPMs vs active sets

- Home-made IPM implemented in Matlab.
- Results:

Instance	rows	columns	CPU IPM	CPU MINOS	CPU SNOPT
3	6040	28140	5.43	255	246.86
4	7468	36260	11.44	4123.71	2180.5
5	2591	17088	2.74	37.91	87.44

Results on PC with AMD Athlon 64 bits Dual Core 4400+, and 2 Gb RAM

- Instance 3 is a random network
- Instance 4 is nonoriented version of problem ndo148
- Instance 5 is a real network

Output for instance 3

```
-----------------------------------------------------------------------------------------
constraints= 6040        variables= 28140
subnonz(A*A)= 41400      subnonz(R*R) = 149856          fill-in= 72.4%
density(A*A)=  0.2%      density(R*R)=  0.8%
-----------------------------------------------------------------------------------------
Iter    mu       Primal Obj.  P. Infeas  Dual Obj.    D. Infeas   Gap      CGit
-----------------------------------------------------------------------------------------
0    9.26e+02   1.400000e+02 9.81e+01  -5.212921e+07 8.39e-01  3.70e+05  0
1    2.41e+02   9.087536e+02 2.51e+01  -7.667748e+06 8.90e-01  8.43e+03  0
2    7.34e+01   2.259299e+03 8.49e+00  -1.757566e+06 5.58e-01  7.79e+02  0
3    2.49e+01   4.012596e+03 2.41e+00  -7.122343e+05 3.65e-01  1.78e+02  0
4    1.08e+01   4.829588e+03 4.87e-01  -4.736653e+05 2.82e-01  9.91e+01  0
5    4.57e+00   4.406178e+03 5.95e-15  -2.528650e+05 7.71e-02  5.84e+01  0
6    1.59e+00   3.172205e+03 1.02e-14  -8.625947e+04 1.70e-01  2.82e+01  0
7    4.18e-01   2.159481e+03 1.44e-14  -2.135711e+04 4.76e-01  1.09e+01  0
8    1.07e-01   1.431467e+03 1.60e-14  -4.580762e+03 6.72e-01  4.20e+00  0
9    3.33e-02   9.260660e+02 3.17e-14  -9.495467e+02 7.37e-01  2.02e+00  0
10   9.70e-03   5.823557e+02 3.69e-14   3.644487e+01 7.76e-01  9.36e-01  0
11   3.55e-03   3.497692e+02 3.21e-14   1.497065e+02 7.37e-01  5.70e-01  0
12   1.03e-03   1.953901e+02 5.77e-14   1.374976e+02 6.63e-01  2.95e-01  0
13   4.64e-04   1.229650e+02 2.60e-14   9.685095e+01 5.41e-01  2.11e-01  0
14   3.26e-04   1.005926e+02 6.97e-14   8.222927e+01 4.56e-01  1.81e-01  0
15   2.46e-04   9.036824e+01 1.18e-13   7.650949e+01 3.63e-01  1.52e-01  0
16   1.35e-04   7.924612e+01 3.31e-13   7.162942e+01 2.50e-01  9.49e-02  0
17   6.96e-05   7.587734e+01 4.86e-13   7.196089e+01 9.69e-02  5.09e-02  0
18   2.51e-05   7.386615e+01 5.41e-13   7.245612e+01 1.76e-02  1.88e-02  0
19   9.70e-06   7.310720e+01 2.03e-13   7.256146e+01 5.39e-03  7.36e-03  0
20   3.47e-06   7.279880e+01 9.23e-14   7.260371e+01 1.75e-03  2.64e-03  0
21   3.48e-07   7.264109e+01 1.84e-13   7.262149e+01 8.70e-06  2.66e-04  0
CPU total  5.43000
```


Non-convex programming: just some IPM codes...

- The standard IPM approach needs to be updated with: line searches, trust-region models, filter methods...
- Some successful codes for non-convex IPMS are:
  - IPOPT [Wächter, Biegler (2006)]
  - KNITRO [Byrd, Hribar, Nocedal (1999)]
  - LOQO [Vanderbei, Shanno (1999)]

References I

- R. Byrd, M. E. Hribar, J. Nocedal. An interior point method for large scale nonlinear programming, *SIAM J. Optimization*, 9 877–900, 1999.
- R. E. Bixby. Solving real-world linear programs: a decade and more of progress, *Operations Research*, 50 3–15, 2002.
- J. Castro. Minimum-distance controlled perturbation methods for large-scale tabular data protection, *European Journal of Operational Research*, 17 39–52, 2006.
- A.V. Fiacco, G.P. McCormick. *Nonlinear Programming. Sequential Unconstrained Minimization Techniques, 2nd Ed.* SIAM, 1990.
- J. Gondzio. Multiple centrality corrections in a primal dual method for linear programming, *Computational Optimization and Applications* 6 137-156, 1996.
- S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization* 2 575–601, 1996.
- C. Roos, T. Terlaky, J.-P. Vial. *Interior Point methods for linear optimization, 2nd Ed.* Springer, 2006.
- R.J. Vanderbei. *Linear Programming: Foundations and Extensions*, Kluwer, 1996.

References II

- R.J. Vanderbei, D.F. Shanno. An interior-point algorithm for nonconvex nonlinear programming, *Computational Optimization and Applications*, 13 231–252, 1999.
- A. Wächter, L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming, *Mathematical Programming* 106 25–57, 2006.
- S.J. Wright. *Primal-dual interior-point methods* SIAM, 1997.