Primal-Dual Interior Point Methods

John E. Mitchell
http://www.rpi.edu/~mitchj

RPI

November, 2010 & 2018
Outline

1. Introduction
2. Log barrier function
3. Primal-dual scaling and $\mu$-complementary slackness
4. The algorithm
5. An example
6. Computational effort
7. Further reading
Standard form

Our standard linear programming problem is

\[
\begin{align*}
\min_x & \quad c^T x \\
\text{subject to} & \quad Ax = b \quad (P) \\
& \quad x \geq 0
\end{align*}
\]

Here, \( c \) and \( x \) are \( n \) vectors, \( b \) is an \( m \) vector, and \( A \) is an \( m \times n \) matrix.

The dual problem can be written

\[
\begin{align*}
\max_{y,s} & \quad b^T y \\
\text{subject to} & \quad A^T y + s = c \quad (D) \\
& \quad s \geq 0
\end{align*}
\]

where \( y \) is an \( m \)-vector and \( s \) is the \( n \)-vector of dual slacks.
Simplex versus interior
Simplex versus interior
Simplex versus interior
Simplex versus interior
Simplex versus interior
Simplex versus interior
Simplex versus interior
Simplex versus interior
Simplex versus interior
Thus, we may try to use an algorithm which cuts across the middle of the feasible region. Such a method is called an interior point method.

There are many different interior point algorithms; we will just consider one: a primal dual method that is close to those implemented in packages such as CPLEX.

*Notation:* We will let $e = [1, \ldots, 1]^T$. 
Outline

1. Introduction
2. Log barrier function
3. Primal-dual scaling and $\mu$-complementary slackness
4. The algorithm
5. An example
6. Computational effort
7. Further reading
Short steps

We may only be able to take a very small step before we violate the nonnegativity constraint.
Short steps

We may only be able to take a very small step before we violate the nonnegativity constraint.

current iterate

updated iterate: $x_4 < 0$
Short steps

We may only be able to take a very small step before we violate the nonnegativity constraint.

updated iterate: $x_4 < 0$
Log barrier function

In order to make the iterates more centered, we add a penalty term to the objective of \((P)\), giving

\[
f_\mu(x) := c^T x - \mu \sum_{i=1}^{n} \ln(x_i)
\]

where \(\mu\) is a positive constant. Note that if \(x_i\) gets close to zero then the function \(f_\mu(x)\) gets large. This gives a nonlinear program

\[
\begin{align*}
\min_{x} & \quad c^T x - \mu \sum_{i=1}^{n} \ln(x_i) \\
\text{subject to} & \quad Ax = b \\
& \quad x \geq 0
\end{align*}
\]

\((P(\mu))\)

Given \(\mu > 0\), it can be shown that there is a unique optimal solution to the problem \((P(\mu))\).
Central path

The central path: the solutions to \((P(\mu))\) for different \(\mu > 0\).
Iterates following central path

Most primal dual algorithms take Newton steps towards points on $C$ with $\mu > 0$, rather than working to solve for a particular $\mu$. 

![Diagram showing iterative process with green iterates aiming for blue target points and optimal face highlighted.](image-url)
Iterates following central path

Most primal dual algorithms take Newton steps towards points on $C$ with $\mu > 0$, rather than working to solve for a particular $\mu$. 

**green** iterates aiming for **blue** target points

**optimal face**
Iterates following central path

Most primal dual algorithms take Newton steps towards points on $C$ with $\mu > 0$, rather than working to solve for a particular $\mu$. 

optimal face

green iterates aiming for
blue target points
Iterates following central path

Most primal dual algorithms take Newton steps towards points on $C$ with $\mu > 0$, rather than working to solve for a particular $\mu$. 

optimal face

green iterates aiming for blue target points
Iterates following central path

Most primal dual algorithms take Newton steps towards points on $C$ with $\mu > 0$, rather than working to solve for a particular $\mu$.
Iterates following central path

Most primal dual algorithms take Newton steps towards points on $C$ with $\mu > 0$, rather than working to solve for a particular $\mu$. 

![Diagram showing iterates aiming for optimal face](image)
Iterates following central path
Most primal dual algorithms take Newton steps towards points on $C$ with $\mu > 0$, rather than working to solve for a particular $\mu$.

optimal face

green iterates aiming for blue target points
Problem (LP1)

$$\begin{align*}
\text{min} & \quad -x_1 - x_2 + x_3 + x_4 \\
\text{subject to} & \quad x_1 + x_3 = 1 \\
& \quad x_2 + x_4 = 2 \\
& \quad x_i \geq 0 \quad i = 1, \ldots, 4
\end{align*}$$

(LP1)

Optimal solution: $x = (1, 2, 0, 0)$. 
Log barrier function example

For \( x \in \mathbb{R}^4 \) with \( c^T x = -x_1 - x_2 + x_3 + x_4 \), the barrier function is

\[
f_{\mu}(x) = -x_1 - x_2 + x_3 + x_4 - \mu \sum_{i=1}^{4} \ln(x_i).
\]

Consider two points:

\( x^k = (0.8, 0.1, 0.2, 1.9) \), \( \bar{x} = (0.95, 1.95, 0.05, 0.05) \),

<table>
<thead>
<tr>
<th>( \mu )</th>
<th>( f_{\mu}(x^k) )</th>
<th>( f_{\mu}(\bar{x}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>11.33</td>
<td>13.32</td>
</tr>
<tr>
<td>0.1</td>
<td>1.54</td>
<td>-2.26</td>
</tr>
</tbody>
</table>

Thus, for larger choices of \( \mu \), the value \( f_{\mu}(x^k) \) of the more central point is smaller than that of \( f_{\mu}(\bar{x}) \), but for smaller choices of \( \mu \) the results are reversed.
Successive minimization

Recall our nonlinear optimization problem:

\[
\begin{align*}
\min_x & \quad f_\mu(x) := c^T x - \mu \sum_{i=1}^n \ln(x_i) \\
\text{subject to} & \quad Ax = b \\
& \quad x \geq 0
\end{align*}
\]

\( (P(\mu)) \)

The motivation for this problem is that it should be easier to get an approximate optimal solution to \((P(\mu))\) for a large value of \(\mu\).

This solution can help in finding approximate optimal solutions for smaller values of \(\mu\). It provides a warm start.

The limiting solution as \(\mu \to 0\) is an optimal solution to \((P)\).

The gradient of \(f_\mu(x)\) is \(c - \mu X^{-1} e\). This eventually leads to a direction that combines what is called the affine direction with a centering direction, as we will see later.

**Notation:** We will let \(e = [1, \ldots, 1]^T\).
Predictor-Corrector

Current commercial implementations combine affine steps with centering and duality in predictor-corrector methods.

In addition to excellent computational performance, variants of these methods converge in a number of iterations that is polynomial in the size of the problem.
Outline

1. Introduction
2. Log barrier function
3. Primal-dual scaling and $\mu$-complementary slackness
4. The algorithm
5. An example
6. Computational effort
7. Further reading
Optimality conditions

The conditions for a point \( \hat{x} \) to be optimal for \((P(\mu))\) are closely related to the standard LP optimality conditions. In fact, we have the following theorem:

**Theorem**

Let \( \mu > 0 \). A point \( x > 0 \) is optimal for \((P(\mu))\) if and only if there exist vectors \( y \) and \( s \) satisfying

\[
\begin{align*}
Ax &= b \\
A^T y + s &= c \\
x_i s_i &= \mu \quad \text{for } i = 1, \ldots, n
\end{align*}
\]

(primal feasibility) \hspace{1cm} (dual feasibility) \hspace{1cm} (\( \mu \) – complementary slackness)
Exploiting dual iterates

This theorem makes it clear that there is no particular reason to favor $(P)$ over $(D)$.

Typically, algorithms use iterates $(\bar{x}, \bar{y}, \bar{s})$ of both primal and dual variables.

We will assume the iterates satisfy primal and dual feasibility, and work towards achieving $\mu$-complementary slackness (and complementary slackness in the limit as $\mu \to 0$).

Further, we assume $\bar{x} > 0$ and $\bar{s} > 0$.

(Recall: the iterates of the simplex algorithm satisfy primal feasibility and complementary slackness, and work towards achieving dual feasibility.)
Newton’s method for nonlinear equations

Our optimality conditions:

\[ A\bar{x} = b \quad \text{(primal feasibility)} \]
\[ A^T\bar{y} + \bar{s} = c \quad \text{(dual feasibility)} \]
\[ x_is_i = \mu \quad \text{for } i = 1, \ldots, n \quad \text{(} \mu \text{ - complementary slackness)} \]

This is a system of \(2n + m\) nonlinear equations in \(2n + m\) unknowns.

Newton’s method:

- Have current solutions \((\bar{x}, \bar{y}, \bar{s})\) which satisfy \(A\bar{x} = b, A^T\bar{y} + \bar{s} = c\), but violate the nonlinear constraints \(\bar{x}_i\bar{s}_i = \mu \forall i\).
- Find directions \(d^x, d^y, d^s\).
  
  So \(x \leftarrow \bar{x} + d^x, y \leftarrow \bar{y} + d^y, s \leftarrow \bar{s} + d^s\).
- The directions satisfy a linearization of the system of nonlinear equations.
Newton's method for nonlinear equations

Our optimality conditions:

\[
\begin{align*}
Ax &= b \\
A^T y + s &= c \\
x_i s_i &= \mu \quad \text{for } i = 1, \ldots, n
\end{align*}
\]

(primal feasibility)  
(dual feasibility)  
(\(\mu\) - complementary slackness)

This is a system of \(2n + m\) nonlinear equations in \(2n + m\) unknowns.

Newton's method:

- Have current solutions \((\bar{x}, \bar{y}, \bar{s})\) which satisfy \(A\bar{x} = b, A^T\bar{y} + \bar{s} = c\), but violate the nonlinear constraints \(\bar{x}_i \bar{s}_i = \mu \\forall i\).
- Find directions \(d^x, d^y, d^s\).
  So \(x \leftarrow \bar{x} + d^x, y \leftarrow \bar{y} + d^y, s \leftarrow \bar{s} + d^s\).
- The directions satisfy a linearization of the system of nonlinear equations.
Newton’s method for nonlinear equations

Our optimality conditions:

\[ \begin{align*}
A\bar{x} &= b & \text{(primal feasibility)} \\
A^T\bar{y} + \bar{s} &= c & \text{(dual feasibility)} \\
x_is_i &= \mu & \text{for } i = 1, \ldots, n & \text{(\(\mu\) – complementary slackness)}
\end{align*} \]

This is a system of \(2n + m\) nonlinear equations in \(2n + m\) unknowns.

Newton’s method:

- Have current solutions \((\bar{x}, \bar{y}, \bar{s})\) which satisfy \(A\bar{x} = b\), \(A^T\bar{y} + \bar{s} = c\), but violate the nonlinear constraints \(\bar{x}_i\bar{s}_i = \mu \ \forall i\).
- Find directions \(d^x, d^y, d^s\).
  - So \(x \leftarrow \bar{x} + d^x\), \(y \leftarrow \bar{y} + d^y\), \(s \leftarrow \bar{s} + d^s\).
- The directions satisfy a linearization of the system of nonlinear equations.
Linearization

After update, ideally want

\[ A(\bar{x} + d^x) = b \]
\[ A^T(\bar{y} + d^y) + (\bar{s} + d^s) = c \]
\[ (\bar{x}_i + d^x_i)(\bar{s}_i + d^s_i) = \mu \quad \text{for } i = 1, \ldots, n \]

We ignore the quadratic terms \( d^x_i d^s_i \).

We note that \( A\bar{x} = b, A^T\bar{y} + \bar{s} = c \), so the directions must satisfy

\[ Ad^x = 0 \quad \text{(so } d^x \text{ in nullspace of } A) \]
\[ A^T d^y + d^s = 0 \]
\[ \bar{s}_i d^x_i + \bar{x}_i d^s_i = \mu - \bar{x}_i \bar{s}_i \quad \text{for } i = 1, \ldots, n \]

This is a system of \( 2n + m \) linear equations in the \( 2n + m \) variables \( d^x, d^y, d^s \).
Scaling matrix

We can express the solution \((d^x, d^y, d^s)\) to the linearized system of equations in closed form.

To make the notation a little nicer, we introduce a scaling matrix and a projection matrix.

Because both primal and dual iterates are available, a scaling matrix can be used that combines the primal variables \(\bar{x}\) and the dual slacks \(\bar{s}\). This matrix is

\[
\bar{D} := \begin{bmatrix}
\sqrt{\frac{\bar{x}_1}{\bar{s}_1}} & 0 & \cdots & 0 \\
0 & \sqrt{\frac{\bar{x}_2}{\bar{s}_2}} & \cdots & \vdots \\
\vdots & \ddots & \ddots & 0 \\
0 & \cdots & 0 & \sqrt{\frac{\bar{x}_n}{\bar{s}_n}}
\end{bmatrix}.
\]

We want \(x_is_i = \mu\) for each \(i\) at optimality to \((P(\mu))\). Then \(x_i/s_i = x_i^2/\mu\).
Project

It is convenient to express $d^x$ using a projection matrix. This takes a vector and projects it onto the nullspace of $\bar{A}$, which is a scaling of $A$.

We define $\bar{A} = A\bar{D}$. The projection $v$ of a vector $g$ onto the nullspace of $\bar{A}$ can be written

$$v = P_{\bar{A}}g$$

which is calculated using the formula

$$P_{\bar{A}} = (I - (\bar{A})^T (\bar{A})(\bar{A})^T)^{-1} \bar{A},$$

and $I$ denotes the identity matrix.

(Aside: We assume that the rows of $\bar{A}$ are linearly independent. Under this assumption, the projection matrix is well-defined. Note that we need this assumption to hold in order to be able to obtain a basic feasible solution.)
Project $g$

Project $g$ onto the nullspace of $\bar{A}$ to find the direction $v$. 

current iterate

$\bar{A}z = b$
Project $g$

Project $g$ onto the nullspace of $\tilde{A}$ to find the direction $\nu$. 

$\tilde{A}z = b$

current iterate

projection of $g$
Updating directions

Solving the system of $2n + m$ linear equations in the $2n + m$ variables $d^x, d^y, d^s$ gives:

$$
d^x := -\bar{D}P_{\bar{A}\bar{D}}\bar{D}(s - \mu X^{-1}e)$$
$$
d^y := (A(\bar{D})^2A^T)^{-1}A(\bar{D})^2(s - \mu X^{-1}e)$$
$$
d^s := -A^Td^y$$

The direction $d^x$ is a combination of an affine direction $\bar{D}P_{\bar{A}\bar{D}}\bar{D}s$ and a centering direction $\bar{D}P_{\bar{A}\bar{D}}\bar{D}X^{-1}e$.

The affine direction emphasizes improving the objective function.

The centering direction pushes the iterates back towards the middle of the polyhedron in order to allow longer steps on future iterations.
Updating directions

We have \( d^x = -\bar{D}P_{\bar{A}\bar{D}}\bar{D}(s - \mu X^{-1}e) \).

The multiplication by \( \bar{D}P_{\bar{A}\bar{D}}\bar{D} \) serves to

1. \( \bar{D}P_{\bar{A}\bar{D}}\bar{D} \): rescale the problem,
2. \( \bar{D}P_{\bar{A}\bar{D}}\bar{D} \): project the direction in the rescaled space,
3. \( \bar{D}P_{\bar{A}\bar{D}}\bar{D} \): and scale back to the original space.

The dual direction \( d^y \) can be calculated on the way to calculating \( d^x \).
It can be shown that \( d^x = -(\bar{D})^2(s - \mu X^{-1}e) - (\bar{D})^2d^s \).
Updating directions

We have \( d^x = -\bar{D} P_{\bar{A}D} \bar{D} (s - \mu X^{-1} e) \).

The multiplication by \( \bar{D} P_{\bar{A}D} \bar{D} \) serves to

1. \( \bar{D} P_{\bar{A}D} \bar{D} \): rescale the problem,
2. \( \bar{D} P_{\bar{A}D} \bar{D} \): project the direction in the rescaled space,
3. \( \bar{D} P_{\bar{A}D} \bar{D} \): and scale back to the original space.

The dual direction \( d^y \) can be calculated on the way to calculating \( d^x \).
It can be shown that \( d^x = -(\bar{D})^2 (s - \mu X^{-1} e) - (\bar{D})^2 d^s \).
Updating directions

We have $d^x = -\bar{D}P_{\bar{A}\bar{D}}\bar{D}(s - \mu X^{-1}e)$. 

The multiplication by $\bar{D}P_{\bar{A}\bar{D}}\bar{D}$ serves to

1. $\bar{D}P_{\bar{A}\bar{D}}\bar{D}$: rescale the problem,
2. $\bar{D}P_{\bar{A}\bar{D}}\bar{D}$: project the direction in the rescaled space,
3. $\bar{D}P_{\bar{A}\bar{D}}\bar{D}$: and scale back to the original space.

The dual direction $d^y$ can be calculated on the way to calculating $d^x$. It can be shown that $d^x = -(\bar{D})^2(s - \mu X^{-1}e) - (\bar{D})^2 d^s$. 
Complementary slackness

The term \( s - \mu X^{-1}e \) is the gradient of the log barrier objective function
\[ s^T x - \mu \sum_{i=1}^{n} \ln(x_i) . \]

Note that if \( x \) is primal feasible and \((y, s)\) is dual feasible then
\[ s^T x = (c - A^T y)^T x = c^T x - b^T y . \]

Therefore, for a given \((y, s)\), it is equivalent to minimize \( c^T x \) or \( s^T x \).
We can say that \( \nabla f_\mu(x) = s - \mu X^{-1}e \).
Outline

1 Introduction
2 Log barrier function
3 Primal-dual scaling and $\mu$-complementary slackness
4 The algorithm
5 An example
6 Computational effort
7 Further reading
Algorithm

We have an iteration counter $k$.

1. Initialize with $x^0 > 0$ feasible in $(P)$ and $(y^0, s^0)$ feasible in $(D)$ with $s^0 > 0$. Let $k = 0$.
2. Let $\mu^k = ((x^k)^T s^k)/n^2$. (Other scalings are possible.)
3. Calculate $\bar{D}$ and $A^k = A\bar{D}$.
4. Calculate $d^y$, $d^s$, $d^x$.
5. Calculate primal and dual steplengths $\alpha_P$ and $\alpha_D$, ensuring $x^k + \alpha_P d^x > 0$ and $s^k + \alpha_D d^s > 0$.
6. Update $x^{k+1} = x^k + \alpha_P d^x$, $y^{k+1} = y^k + \alpha_D d^y$, $s^{k+1} = s^k + \alpha_D d^s$.
7. Let $k \rightarrow k + 1$. Calculate $(x^k)^T s^k$. If small enough, **STOP**. Else return to Step 2.
The algorithm

Algorithm

We have an iteration counter $k$. 

1. Initialize with $x^0 > 0$ feasible in $(P)$ and $(y^0, s^0)$ feasible in $(D)$ with $s^0 > 0$. Let $k = 0$.

2. Let $\mu^k = (\langle x^k \rangle^T s^k)/n^2$. (Other scalings are possible.)

3. Calculate $\bar{D}$ and $A^k = A\bar{D}$.

4. Calculate $d^y$, $d^s$, $d^x$.

5. Calculate primal and dual steplengths $\alpha_P$ and $\alpha_D$, ensuring $x^k + \alpha_P d^x > 0$ and $s^k + \alpha_D d^s > 0$.

6. Update $x^{k+1} = x^k + \alpha_P d^x$, $y^{k+1} = y^k + \alpha_D d^y$, $s^{k+1} = s^k + \alpha_D d^s$.

7. Let $k \to k + 1$. Calculate $(x^k)^T s^k$. If small enough, STOP. Else return to Step 2.
The algorithm

Algorithm

We have an iteration counter $k$.

1. Initialize with $x^0 > 0$ feasible in $(P)$ and $(y^0, s^0)$ feasible in $(D)$ with $s^0 > 0$. Let $k = 0$.
2. Let $\mu^k = ((x^k)^T s^k)/n^2$. (Other scalings are possible.)
3. Calculate $\bar{D}$ and $A^k = A\bar{D}$.
4. Calculate $d^y, d^s, d^x$.
5. Calculate primal and dual steplengths $\alpha_P$ and $\alpha_D$, ensuring $x^k + \alpha_P d^x > 0$ and $s^k + \alpha_D d^s > 0$.
6. Update $x^{k+1} = x^k + \alpha_P d^x$, $y^{k+1} = y^k + \alpha_D d^y$, $s^{k+1} = s^k + \alpha_D d^s$.
7. Let $k \rightarrow k + 1$. Calculate $(x^k)^T s^k$. If small enough, STOP. Else return to Step 2.
Algorithm

We have an iteration counter $k$.

1. Initialize with $x^0 > 0$ feasible in $(P)$ and $(y^0, s^0)$ feasible in $(D)$ with $s^0 > 0$. Let $k = 0$.
2. Let $\mu^k = ((x^k)^T s^k)/n^2$. (Other scalings are possible.)
3. Calculate $\bar{D}$ and $A^k = A\bar{D}$.
4. Calculate $d^y$, $d^s$, $d^x$.
5. Calculate primal and dual steplengths $\alpha_P$ and $\alpha_D$, ensuring $x^k + \alpha_P d^x > 0$ and $s^k + \alpha_D d^s > 0$.
6. Update $x^{k+1} = x^k + \alpha_P d^x$, $y^{k+1} = y^k + \alpha_D d^y$, $s^{k+1} = s^k + \alpha_D d^s$.
7. Let $k \rightarrow k + 1$. Calculate $(x^k)^T s^k$. If small enough, STOP. Else return to Step 2.
The algorithm

Algorithm

We have an iteration counter $k$.

1. Initialize with $x^0 > 0$ feasible in $(P)$ and $(y^0, s^0)$ feasible in $(D)$ with $s^0 > 0$. Let $k = 0$.
2. Let $\mu^k = \left(\langle x^k \rangle s^k \right)/n^2$. (Other scalings are possible.)
3. Calculate $\bar{D}$ and $A_k = A\bar{D}$.
4. Calculate $d_y, d_s, d_x$.
5. Calculate primal and dual steplengths $\alpha_P$ and $\alpha_D$, ensuring $x^k + \alpha_P d^x > 0$ and $s^k + \alpha_D d^s > 0$.
6. Update $x^{k+1} = x^k + \alpha_P d^x$, $y^{k+1} = y^k + \alpha_D d^y$, $s^{k+1} = s^k + \alpha_D d^s$.
7. Let $k \rightarrow k + 1$. Calculate $\langle x^k \rangle s^k$. If small enough, STOP. Else return to Step 2.
Algorithm

We have an iteration counter $k$.

1. Initialize with $x^0 > 0$ feasible in $(P)$ and $(y^0, s^0)$ feasible in $(D)$ with $s^0 > 0$. Let $k = 0$.
2. Let $\mu^k = ((x^k)^T s^k)/n^2$. (Other scalings are possible.)
3. Calculate $\bar{D}$ and $A^k = A\bar{D}$.
4. Calculate $d^y, d^s, d^x$.
5. Calculate primal and dual steplengths $\alpha_P$ and $\alpha_D$, ensuring $x^k + \alpha_P d^x > 0$ and $s^k + \alpha_D d^s > 0$.
6. Update $x^{k+1} = x^k + \alpha_P d^x, y^{k+1} = y^k + \alpha_D d^y, s^{k+1} = s^k + \alpha_D d^s$.
7. Let $k \rightarrow k + 1$. Calculate $(x^k)^T s^k$. If small enough, STOP. Else return to Step 2.
The algorithm

Algorithm

We have an iteration counter $k$.

1. Initialize with $x^0 > 0$ feasible in $(P)$ and $(y^0, s^0)$ feasible in $(D)$ with $s^0 > 0$. Let $k = 0$.
2. Let $\mu^k = ((x^k)^T s^k)/n^2$. (Other scalings are possible.)
3. Calculate $\bar{D}$ and $A^k = A\bar{D}$.
4. Calculate $d^y$, $d^s$, $d^x$.
5. Calculate primal and dual steplengths $\alpha_P$ and $\alpha_D$, ensuring $x^k + \alpha_P d^x > 0$ and $s^k + \alpha_D d^s > 0$.
6. Update $x^{k+1} = x^k + \alpha_P d^x$, $y^{k+1} = y^k + \alpha_D d^y$, $s^{k+1} = s^k + \alpha_D d^s$.
7. Let $k \rightarrow k + 1$. Calculate $(x^k)^T s^k$. If small enough, STOP. Else return to Step 2.
Outline

1. Introduction
2. Log barrier function
3. Primal-dual scaling and $\mu$-complementary slackness
4. The algorithm
5. An example
6. Computational effort
7. Further reading

Mitchell (RPI) Primal-Dual Methods November, 2010 & 2018 28 / 41
An example

The problem

\[\begin{align*}
\text{min} & \quad -x_1 - x_2 + x_3 + x_4 \\
\text{subject to} & \quad x_1 + x_3 = 1 \\
& \quad x_2 + x_4 = 2 \\
& \quad x_i \geq 0 \quad i = 1, \ldots, 4
\end{align*}\]  

\((LP1)\)

has dual

\[\begin{align*}
\text{max} & \quad y_1 + 2y_2 \\
\text{subject to} & \quad y_1 + s_1 = -1 \\
& \quad y_2 + s_2 = -1 \\
& \quad y_1 + s_3 = 1 \\
& \quad y_2 + s_4 = 1 \\
& \quad s_i \geq 0 \quad i = 1, \ldots, 4.
\end{align*}\]  

\((LD1)\)
Initial iterate

An initial feasible point for \((LP1)\) is \(x^0 = (0.6, 1.5, 0.4, 0.5)\) and an initial dual feasible solution is \(y^0 = (-5, -2)\) giving \(s^0 = (4, 1, 6, 3)\).

Note that we have \(x^0_i s^0_i = 2.4\) for \(i = 1, 3\) and \(x^0_i s^0_i = 1.5\) for \(i = 2, 4\). Thus, this point is not centered.

The current duality gap is 7.8, so we pick \(\mu = 0.5\) (approximately \(7.8/(4^2)\)). We get

\[
D^0 = \begin{bmatrix} \sqrt{0.15} & 0 & 0 & 0 \\ 0 & \sqrt{1.5} & 0 & 0 \\ 0 & 0 & \sqrt{1/15} & 0 \\ 0 & 0 & 0 & \sqrt{1/6} \end{bmatrix}, \quad s - \mu X^{-1} e = \begin{bmatrix} 3.17 \\ 0.67 \\ 4.75 \\ 2.00 \end{bmatrix},
\]

and

\[
A(D^0)^2 A^T = \begin{bmatrix} 0.217 & 0 \\ 0 & 1.667 \end{bmatrix}.
\]
Calculate directions

Further,

\[ A(D_0)^2(s - \mu X^{-1}e) = \begin{bmatrix} 0.792 \\ 1.333 \end{bmatrix} \]

so

\[ d^y = (A(D_0)^2 A^T)^{-1} A(D_0)^2(s - \mu X^{-1}e) = \begin{bmatrix} 3.654 \\ 0.800 \end{bmatrix} \]

and

\[ d^s = -A^T d^y = \begin{bmatrix} -3.654 \\ -0.800 \\ -3.654 \\ -0.800 \end{bmatrix} \]
Updating the iterates

Now,

\[ d^x = -(D^0)^2(s - \mu X^{-1} e) - (D^0)^2 d^s \]

\[
\begin{bmatrix}
0.475 \\
1.000 \\
0.316 \\
0.333
\end{bmatrix}
- 
\begin{bmatrix}
-0.548 \\
-1.200 \\
-0.243 \\
-0.133
\end{bmatrix}
= 
\begin{bmatrix}
0.073 \\
0.200 \\
-0.073 \\
-0.200
\end{bmatrix}.
\]

Taking \( \alpha_P = 2 \) and \( \alpha_D = 1 \) gives

\[ x^1 = 
\begin{bmatrix}
0.746 \\
1.900 \\
0.254 \\
0.100
\end{bmatrix},
\]
\[ y^1 = 
\begin{bmatrix}
-1.346 \\
-1.200
\end{bmatrix},
\]
\[ s^1 = 
\begin{bmatrix}
0.346 \\
0.200 \\
2.346 \\
2.200
\end{bmatrix}.
\]
Updated complementaritities

The complementarity values are now

\[ x_1^1 s_1^1 = 0.26, \quad x_2^1 s_2^1 = 0.38, \quad x_3^1 s_3^1 = 0.60, \quad x_4^1 s_4^1 = 0.22 \]

and the new duality gap is 1.45.

The individual complementarities are reasonably close to the target value of \( \mu = 0.5 \).

The duality gap has been noticeably reduced.
The updated iterate for the example

\[ x^k = (0.6, 1.5, 0.4, 0.5) \]
The updated iterate for the example

\[ x^k = (0.6, 1.5, 0.4, 0.5) \]

\[ x^{k+1} = (0.746, 1.900, 0.254, 0.100) \]
Complementarities

Want all $x_i s_i$ to be the same to be on the central path.

$C = \begin{pmatrix} 2.4, 1.5, 2.4, 1.5 \end{pmatrix}$
Complementarities

Want all $x_i s_i$ to be the same to be on the central path.

$C = (2.4, 1.5, 2.4, 1.5)$

target: $\mu = 0.5$
Complementarities

Want all $x_i s_i$ to be the same to be on the central path.

\[ x^k s^k = (2.4, 1.5, 2.4, 1.5) \]

Optimal solution:

- $\mu = 0.5$
- $x^{k+1} s^{k+1} = (0.26, 0.38, 0.60, 0.22)$
Outline

1. Introduction
2. Log barrier function
3. Primal-dual scaling and \( \mu \)-complementary slackness
4. The algorithm
5. An example
6. Computational effort
7. Further reading
Computational effort

Each iteration requires a lot of work.

The major effort at each iteration is in calculating the projections.

It is not necessary calculate the inverse \((A(\bar{D})^2A^T)^{-1}\) explicitly. Gaussian elimination can be used instead.

It seems that an interior point method will solve almost any problem in at most 40 iterations. The number of iterations required grows very slowly as the size of the problem increases.
Comparison with simplex

By contrast, the simplex algorithm seems to need approximately $1.5m$ iterations to solve a problem with $m$ constraints.

The work required at each iteration of an interior point method is far larger than the amount of work needed to compute a simplex pivot, so there is a trade-off:

\textit{an interior point algorithm needs far fewer iterations, but it takes considerably more time per iteration, when compared to the simplex algorithm.}

Commercial implementations of interior point methods use algorithms that are similar to the one presented, but they have some more modifications to make them faster.
Outline

1. Introduction
2. Log barrier function
3. Primal-dual scaling and $\mu$-complementary slackness
4. The algorithm
5. An example
6. Computational effort
7. Further reading
Further reading

Textbooks include [BV04, LY08, RTV05, Van08, Wri96].

Current computational results indicate that interior point methods outperform the simplex algorithm on large problems.
Textbooks


Textbooks


Textbooks


Textbooks


Textbooks


