Interior Point Methods

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

RPI

November, 2010 & 2018
Introduction

1 Introduction

2 The primal affine scaling algorithm
   - Rescaling so that we can take long steps
   - The relationship between \((P)\) and \((P^k)\)
   - Projections
   - Choosing the steplength
   - One iteration of the algorithm
   - Monotonicity of the algorithm
   - Getting a dual solution
   - Stopping the algorithm

3 Doing more centering

4 Historical notes

5 Bibliography
Introduction

Our standard linear programming problem is

$$\min c^T x$$
subject to $Ax = b$ \hspace{1cm} (P)
$$x \geq 0$$

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

The simplex algorithm moves from basic feasible solution to basic feasible solution. The basic feasible solutions are extreme points of the feasible region for (P). Furthermore, the simplex algorithm moves from one extreme point along an edge of the feasible region to another extreme point. If the feasible region is very big with many extreme points, the simplex algorithm may take a long time before it finds the optimal extreme point.
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: the primal affine scaling algorithm.

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

1. Introduction

2. The primal affine scaling algorithm
   - Rescaling so that we can take long steps
   - The relationship between \((P)\) and \((P^k)\)
   - Projections
   - Choosing the steplength
   - One iteration of the algorithm
   - Monotonicity of the algorithm
   - Getting a dual solution
   - Stopping the algorithm

3. Doing more centering

4. Historical notes

5. Bibliography
Affine scaling

The boundaries of the feasible region are given by the nonnegativity inequalities $x \geq 0$. Therefore, a point is in the interior of the feasible region if it satisfies

$$Ax = b \text{ and } x > 0.$$ 

Assume we know a strictly positive point $x^k$ which is feasible for the problem $(P)$.

The best direction to move may appear to be the direction $-c$, because this will result in the largest decrease in the objective function value for the smallest change in $x$.

There are two problems with this direction; infeasibility or short steps (see next slides).
Infeasibility

Moving in the direction $-c$ may result in a point which no longer satisfies $Ax = b$. We will return to this issue later when we discuss projections.
Infeasibility

Moving in the direction $-c$ may result in a point which no longer satisfies $Ax = b$. We will return to this issue later when we discuss projections.
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.

updated iterate: $x_4 < 0$
Short steps

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

\[ x_4 = 0 \]

updated iterate: \( x_4 < 0 \)
Outline

1. Introduction

2. The primal affine scaling algorithm
   - Rescaling so that we can take long steps
   - The relationship between \((P)\) and \((P^k)\)
   - Projections
   - Choosing the steplength
   - One iteration of the algorithm
   - Monotonicity of the algorithm
   - Getting a dual solution
   - Stopping the algorithm

3. Doing more centering

4. Historical notes

5. Bibliography
Rescaling

Consider the problem

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

so here $c = [-1, -1, 1, 1]^T$.

This problem has optimal solution $x^* = (1, 2, 0, 0)$ with value -3.

The point $x^k = (0.8, 0.1, 0.2, 1.9)$ is feasible in this problem, with objective function value 1.2.
Update $x$

We want to look at points of the form

$$x = x^k - \beta c$$

for some steplength $\beta > 0$, because the objective function value of such a point is

$$c^T x = c^T x^k - \beta c^T c < c^T x^k$$

because $c^T c$ is the dot product between the vector $c$ and itself, that is, it is just the square of the length of the vector $c$. 
Update $x$

Now,

$$x = x^k - \beta c$$
$$= (0.8, 0.1, 0.2, 1.9) - \beta(-1, -1, 1, 1)$$
$$= (0.8 + \beta, 0.1 + \beta, 0.2 - \beta, 1.9 - \beta)$$

This point still satisfies the equality constraints for any choice of $\beta$ (for this particular problem).

However, if we want $x \geq 0$, the largest possible value of $\beta$ is 0.2 and then we get $x = (1, 0.3, 0, 1.7)$ with value 0.4, so we have not moved very far towards the optimal point.
Rescale

To try to get around this difficulty, we rescale the problem so that we can take a step of length at least one in any direction.

Thus, we introduce a diagonal matrix

\[
D^k = \begin{bmatrix}
x_1^k & 0 & \cdots & 0 \\
0 & x_2^k & \ddots & \vdots \\
\vdots & \ddots & \ddots & 0 \\
0 & \cdots & 0 & x_n^k
\end{bmatrix}.
\]

We then scale the constraint matrix and the objective function, getting

\[
A^k := AD^k \text{ and } c^k := D^k c.
\]
Rescale

For the problem \((LP1)\), with \(x^k\) as given above, we get

\[
D^k = \begin{bmatrix}
0.8 & 0 & 0 & 0 \\
0 & 0.1 & 0 & 0 \\
0 & 0 & 0.2 & 0 \\
0 & 0 & 0 & 1.9 \\
\end{bmatrix}
\]

and

\[
A^k = AD^k = \begin{bmatrix}
0.8 & 0 & 0.2 & 0 \\
0 & 0.1 & 0 & 1.9 \\
\end{bmatrix}, \quad c^k = D^k c = \begin{bmatrix}
-0.8 \\
-0.1 \\
0.2 \\
1.9 \\
\end{bmatrix}
\]

since

\[
A = \begin{bmatrix}
1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
\end{bmatrix}, \quad c = \begin{bmatrix}
-1 \\
-1 \\
1 \\
1 \\
\end{bmatrix}.
\]
Rescaled problem

We then get a rescaled problem

\[
\begin{align*}
\min & \quad c^k T z \\
\text{subject to} & \quad A^k z = b \\
& \quad z \geq 0
\end{align*}
\]

(P^k)

The rescaled version of \((LP1)\) is

\[
\begin{align*}
\min & \quad -0.8z_1 - 0.1z_2 + 0.2z_3 + 1.9z_4 \\
\text{subject to} & \quad 0.8z_1 + 0.2z_3 = 1 \\
& \quad 0.1z_2 + 1.9z_4 = 2 \\
& \quad z_i \geq 0 \quad i = 1, \ldots, 4
\end{align*}
\]

\((LP1^k)\)

Notice that \(z = (1, 1, 1, 1)\) is feasible in \((LP1^k)\), with objective function value 1.2. Because all of the components of \(z\) are at least one, we can take a reasonable step in any direction.
An inequality version of the problem

Our original 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)

is equivalent to the problem

\[
\begin{align*}
\text{min} & \quad 2x_3 + 2x_4 - 3 \\
\text{subject to} & \quad 0 \leq x_3 \leq 1 \\
& \quad 0 \leq x_4 \leq 2
\end{align*}
\]

since \( x_1 = 1 - x_3, \ x_2 = 2 - x_4 \).
Rescaling the inequality version

Rescaling using \( x^k = (0.8, 0.1, 0.2, 1.9) \) the inequality problem

\[
\begin{align*}
\min & \quad 2x_3 + 2x_4 - 3 \\
\text{subject to} & \quad 0 \leq x_3 \leq 1 \\
& \quad 0 \leq x_4 \leq 2
\end{align*}
\]

gives

\[
\begin{align*}
\min & \quad 0.4z_3 + 3.8z_4 - 3 \\
\text{subject to} & \quad 0 \leq 0.2z_3 \leq 1 \\
& \quad 0 \leq 1.9z_4 \leq 2
\end{align*}
\]

with point \( z^k = (1, 1, 1, 1) \).

Notice that \( z_1 = \frac{1}{0.8} (1 - 0.2z_3) \) and \( z_2 = \frac{1}{0.1} (2 - 1.9z_4) \).
Rescaling in \((x_3, x_4)\)-space

Original:

\[
\begin{align*}
\text{min} & \quad 2x_3 + 2x_4 - 3 \\
\text{s.t.} & \quad 0 \leq x_3 \leq 1 \\
& \quad 0 \leq x_4 \leq 2
\end{align*}
\]

Rescaled:

\[
\begin{align*}
\text{min} & \quad 0.4z_3 + 3.8z_4 - 3 \\
\text{s.t.} & \quad 0 \leq 0.2z_3 \leq 1 \\
& \quad 0 \leq 1.9z_4 \leq 2
\end{align*}
\]
Rescaling in \((x_3, x_4)\)-space

Original:

\[
\begin{align*}
\min \quad & 2x_3 + 2x_4 - 3 \\
\text{s.t.} \quad & 0 \leq x_3 \leq 1 \\
\quad & 0 \leq x_4 \leq 2
\end{align*}
\]

Rescaled:

\[
\begin{align*}
\min \quad & 0.4z_3 + 3.8z_4 - 3 \\
\text{s.t.} \quad & 0 \leq 0.2z_3 \leq 1 \\
\quad & 0 \leq 1.9z_4 \leq 2
\end{align*}
\]
Rescaling in \((x_3, x_4)\)-space

Original:

\[
\begin{align*}
\text{min} & \quad 2x_3 + 2x_4 - 3 \\
\text{s.t.} & \quad 0 \leq x_3 \leq 1 \\
& \quad 0 \leq x_4 \leq 2
\end{align*}
\]

Rescaled:

\[
\begin{align*}
\text{min} & \quad 0.4z_3 + 3.8z_4 - 3 \\
\text{s.t.} & \quad 0 \leq 0.2z_3 \leq 1 \\
& \quad 0 \leq 1.9z_4 \leq 2
\end{align*}
\]
Rescaling in \((x_3, x_4)\)-space

Original:

\[
\begin{align*}
\min \quad & 2x_3 + 2x_4 - 3 \\
\text{s.t.} \quad & 0 \leq x_3 \leq 1 \\
& 0 \leq x_4 \leq 2
\end{align*}
\]

Rescaled:

\[
\begin{align*}
\min \quad & 0.4z_3 + 3.8z_4 - 3 \\
\text{s.t.} \quad & 0 \leq 0.2z_3 \leq 1 \\
& 0 \leq 1.9z_4 \leq 2
\end{align*}
\]
Rescaling in \((x_3, x_4)\)-space

Original:

\[
\begin{align*}
\text{min} & \quad 2x_3 + 2x_4 - 3 \\
\text{s.t.} & \quad 0 \leq x_3 \leq 1 \\
& \quad 0 \leq x_4 \leq 2 \\
\end{align*}
\]

Rescaled:

\[
\begin{align*}
\text{min} & \quad 0.4z_3 + 3.8z_4 - 3 \\
\text{s.t.} & \quad 0 \leq 0.2z_3 \leq 1 \\
& \quad 0 \leq 1.9z_4 \leq 2 \\
\end{align*}
\]
The primal affine scaling algorithm

The relationship between $(P)$ and $(P^k)$

Outline

1. Introduction
2. The primal affine scaling algorithm
   - Rescaling so that we can take long steps
   - The relationship between $(P)$ and $(P^k)$
   - Projections
   - Choosing the steplength
   - One iteration of the algorithm
   - Monotonicity of the algorithm
   - Getting a dual solution
   - Stopping the algorithm
3. Doing more centering
4. Historical notes
5. Bibliography
Properties

We list several properties of \((P_k)\). It may help you to understand these properties if you see how they apply to the problem \((LP_1)\).

- Since \(D^k\) is a diagonal matrix, it is equal to its transpose:
  \[(D^k)^T = D^k.\]

- The point \(z = (1, \ldots, 1)\) is feasible in \((P_k)\). It corresponds to the point \(x^k\) in \((P)\). Note that \(D^k e = x^k\) and \((D^k)^{-1} x^k = e.\)

- If \(\bar{x}\) is feasible in \((P)\) then \(\bar{z} := (D^k)^{-1} \bar{x}\) is feasible in \((P_k)\).
  Further, \(c^T \bar{x} = c^k T \bar{z}.\)
  Check: \(A^k \bar{z} = A D^k (D^k)^{-1} \bar{x} = A \bar{x} = b.\)
  Also: \(c^k T \bar{z} = (D^k c)^T (D^k)^{-1} \bar{x} = c^T D^k (D^k)^{-1} \bar{x} = c^T \bar{x}.\)

- If \(\tilde{z}\) is feasible in \((P_k)\) then \(\tilde{x} := D^k \tilde{z}\) is feasible in \((P)\). Further, \(c^T \tilde{x} = c^k T \tilde{z}.\)
  Check: \(A \tilde{x} = A D^k \tilde{z} = A^k \tilde{z} = b.\)
  Also: \(c^T \tilde{x} = c^T D^k \tilde{z} = (D^k c)^T \tilde{z} = c^k T \tilde{z}.\)
Properties

We list several properties of \((P^k)\). It may help you to understand these properties if you see how they apply to the problem \((LP1)\).

- Since \(D^k\) is a diagonal matrix, it is equal to its transpose:
  \[(D^k)^T = D^k.\]

- The point \(z = (1, \ldots, 1)\) is feasible in \((P^k)\). It corresponds to the point \(x^k\) in \((P)\). Note that \(D^k e = x^k\) and \((D^k)^{-1} x^k = e.\)

- If \(\bar{x}\) is feasible in \((P)\) then \(\bar{z} := (D^k)^{-1} \bar{x}\) is feasible in \((P^k)\). Further, \(c^T \bar{x} = c^k T \bar{z}.\)
  Check: \(A^k \bar{z} = AD^k (D^k)^{-1} \bar{x} = A\bar{x} = b.\)
  Also: \(c^k T \bar{z} = (D^k c)^T (D^k)^{-1} \bar{x} = c^T D^k (D^k)^{-1} \bar{x} = c^T \bar{x}.\)

- If \(\tilde{z}\) is feasible in \((P^k)\) then \(\tilde{x} := D^k \tilde{z}\) is feasible in \((P)\). Further, \(c^T \tilde{x} = c^k T \tilde{z}.\)
  Check: \(A\tilde{x} = AD^k \tilde{z} = A^k \tilde{z} = b.\)
  Also: \(c^T \tilde{x} = c^T D^k \tilde{z} = (D^k c)^T \tilde{z} = c^k T \tilde{z}.\)
Properties

We list several properties of $(P^k)$. It may help you to understand these properties if you see how they apply to the problem $(LP1)$.

- Since $D^k$ is a diagonal matrix, it is equal to its transpose:
  \[(D^k)^T = D^k.\]

- The point $z = (1, \ldots , 1)$ is feasible in $(P^k)$. It corresponds to the point $x^k$ in $(P)$. Note that $D^k e = x^k$ and $(D^k)^{-1} x^k = e$.

- If $\bar{x}$ is feasible in $(P)$ then $\tilde{z} := (D^k)^{-1} \bar{x}$ is feasible in $(P^k)$. Further, $c^T \bar{x} = c^k T \tilde{z}$.
  Check: $A^k \tilde{z} = AD^k(D^k)^{-1} \bar{x} = A \bar{x} = b$.
  Also: $c^k T \tilde{z} = (D^k c)^T(D^k)^{-1} \bar{x} = c^T D^k(D^k)^{-1} \bar{x} = c^T \bar{x}$.

- If $\tilde{z}$ is feasible in $(P^k)$ then $\tilde{x} := D^k \tilde{z}$ is feasible in $(P)$. Further, $c^T \tilde{x} = c^k T \tilde{z}$.
  Check: $A \tilde{x} = AD^k \tilde{z} = A^k \tilde{z} = b$.
  Also: $c^T \tilde{x} = c^T D^k \tilde{z} = (D^k c)^T \tilde{z} = c^k T \tilde{z}$. 

Mitchell (RPI)
Properties

We list several properties of \((P^k)\). It may help you to understand these properties if you see how they apply to the problem \((LP1)\).

- Since \(D^k\) is a diagonal matrix, it is equal to its transpose: 
  \[ (D^k)^T = D^k. \]
- The point \(z = (1, \ldots, 1)\) is feasible in \((P^k)\). It corresponds to the point \(x^k\) in \((P)\). Note that \(D^k e = x^k\) and \((D^k)^{-1} x^k = e\).
- If \(\tilde{x}\) is feasible in \((P)\) then \(\tilde{z} := (D^k)^{-1} \tilde{x}\) is feasible in \((P^k)\). Further, \(c^T \tilde{x} = c^k T \tilde{z}\).
  Check: \(A^k \tilde{z} = AD^k(D^k)^{-1} \tilde{x} = A \tilde{x} = b\).
  Also: \(c^k T \tilde{z} = (D^k c)^T (D^k)^{-1} \tilde{x} = c^T D^k (D^k)^{-1} \tilde{x} = c^T \tilde{x}\).
- If \(\tilde{z}\) is feasible in \((P^k)\) then \(\tilde{x} := D^k \tilde{z}\) is feasible in \((P)\). Further, \(c^T \tilde{x} = c^k T \tilde{z}\).
  Check: \(A \tilde{x} = AD^k \tilde{z} = A^k \tilde{z} = b\).
  Also: \(c^T \tilde{x} = c^T D^k \tilde{z} = (D^k c)^T \tilde{z} = c^k T \tilde{z}\).
The primal affine scaling algorithm

Rescaling so that we can take long steps
The relationship between \((P)\) and \((P^k)\)

Projections
Choosing the steplength
One iteration of the algorithm
Monotonicity of the algorithm
Getting a dual solution
Stopping the algorithm

Doing more centering

Historical notes

Bibliography
Keeping $Ax = b$

Conceptually, an iteration of the algorithm has the form:

1. Given a point $x^k > 0$ which is feasible for $(P)$, rescale to get new problem $(P^k)$. The point $z = e$ is feasible in the rescaled problem $(P^k)$.

2. Update $z$ to a new point: $z^{new} = e + \beta d$, where $d$ is a direction and $\beta$ is a step length. We pick $\beta$ to ensure that $z^{new} > 0$.

3. Scale back to get a feasible point in $(P)$: $x^{k+1} = D^k z^{new}$.

We would like to take $d = -c^k$, in order to decrease the objective function value as quickly as possible. Unfortunately, this may lead to a point which violates the constraint $A^k z = b$. 
Keeping $Ax = b$

Conceptually, an iteration of the algorithm has the form:

1. Given a point $x^k > 0$ which is feasible for $(P)$, rescale to get new problem $(P^k)$. The point $z = e$ is feasible in the rescaled problem $(P^k)$.

2. Update $z$ to a new point: $z^{new} = e + \beta d$, where $d$ is a direction and $\beta$ is a step length. We pick $\beta$ to ensure that $z^{new} > 0$.

3. Scale back to get a feasible point in $(P)$: $x^{k+1} = D^k z^{new}$.

We would like to take $d = -c^k$, in order to decrease the objective function value as quickly as possible. Unfortunately, this may lead to a point which violates the constraint $A^k z = b$. 
Keeping $Ax = b$

Conceptually, an iteration of the algorithm has the form:

1. **Given a point** $x^k > 0$ which is feasible for $(P)$, **rescale** to get new problem $(P^k)$. The point $z = e$ is feasible in the rescaled problem $(P^k)$.

2. **Update** $z$ to a new point: $z^{\text{new}} = e + \beta d$, where $d$ is a direction and $\beta$ is a step length. We pick $\beta$ to ensure that $z^{\text{new}} > 0$.

3. **Scale back** to get a feasible point in $(P)$: $x^{k+1} = D^k z^{\text{new}}$.

We would like to take $d = -c^k$, in order to decrease the objective function value as quickly as possible. Unfortunately, this may lead to a point which violates the constraint $A^k z = b$. 
For example, in the problem \((LP1^k)\), taking \(\beta = 0.5\) gives the point

\[
\begin{align*}
z &= e - 0.5c^k \\
&= (1, 1, 1, 1) - 0.5(-0.8, -0.1, 0.2, 1.9) \\
&= (1.4, 1.05, 0.9, 0.05)
\end{align*}
\]

which does not satisfy the constraints. We have

\[
A^k z = \begin{bmatrix} 0.8 & 0 & 0.2 & 0 \\ 0 & 0.1 & 0 & 1.9 \end{bmatrix} \begin{bmatrix} 1.4 \\ 1.05 \\ 0.9 \\ 0.05 \end{bmatrix} = \begin{bmatrix} 1.3 \\ 0.2 \end{bmatrix} \neq b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}
\]
Modify the direction

We need to move in a direction $d$ which satisfies $A^k d = 0$.

A direction $d$ which satisfies $A^k d = 0$ is said to be in the nullspace of the matrix $A^k$.

We then get

$$A^k (e + \beta d) = A^k e + \beta A^k d = A^k e = b,$$

because $e$ is feasible in $(P^k)$.
Project

Therefore, we project the direction $c^k$ onto the nullspace of $A^k$. Algebraically, this means that we take

$$d = -P_{A^k} c^k$$

(1)

where

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

(2)

and $I$ denotes the identity matrix.

(Aside: We assume that the rows of $A$ 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 $-c$

Project $-c^k$ onto the nullspace of $A^k$ to find the direction $d$. 

$Ax = b$
Project $-c$

Project $-c^k$ onto the nullspace of $A^k$ to find the direction $d$. 
Example

For the problem \((LP1)\), we have

\[
A^k(A^k)^T = \begin{bmatrix}
0.8 & 0 & 0.2 & 0 \\
0 & 0.1 & 0 & 1.9 \\
0 & 0.2 & 0 & 0 \\
0 & 0 & 1.9 & 0
\end{bmatrix}
\begin{bmatrix}
0.8 & 0 \\
0 & 0.1 \\
0 & 0 \\
0 & 1.9
\end{bmatrix}
\]

\[
= \begin{bmatrix}
0.68 & 0 \\
0 & 3.62
\end{bmatrix}
\]

so

\[
(A^k(A^k)^T)^{-1} = \begin{bmatrix}
1.4706 & 0 \\
0 & 0.2762
\end{bmatrix}
\]
Example

and

$$P_{A^k} = I - \begin{bmatrix} 0.8 & 0 \\ 0 & 0.1 \\ 0.2 & 0 \\ 0 & 1.9 \end{bmatrix} \begin{bmatrix} 1.4706 & 0 \\ 0 & 0.2762 \end{bmatrix} \begin{bmatrix} 0.8 & 0 & 0.2 & 0 \\ 0 & 0.1 & 0 & 1.9 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0.941 & 0 & 0.235 & 0 \\ 0 & 0.003 & 0 & 0.052 \\ 0.235 & 0 & 0.059 & 0 \\ 0 & 0.052 & 0 & 0.997 \end{bmatrix}$$

$$= \begin{bmatrix} 0.059 & 0 & -0.235 & 0 \\ 0 & 0.997 & 0 & -0.052 \\ -0.235 & 0 & 0.941 & 0 \\ 0 & -0.052 & 0 & 0.003 \end{bmatrix}.$$
Example

So, in the problem \((LP1^k)\), we use the direction

\[
d = -P_A^k c^k
\]

\[
= -
\begin{bmatrix}
0.059 & 0 & -0.235 & 0 \\
0 & 0.997 & 0 & -0.052 \\
-0.235 & 0 & 0.941 & 0 \\
0 & -0.052 & 0 & 0.003
\end{bmatrix}
\begin{bmatrix}
-0.8 \\
-1.0 \\
0.2 \\
1.9
\end{bmatrix}
\]

\[
= \begin{bmatrix}
0.0942 \\
0.1985 \\
-0.3762 \\
-0.0109
\end{bmatrix}.
\]
**Updated x**

This gives a new point of the form

\[ z^{\text{new}} = e + \beta d \]
\[ = (1, 1, 1, 1) + \beta(0.0941, 0.1994, -0.3765, -0.0160) \]

Taking \( \beta = 2 \) gives

\[ z^{\text{new}} = (1.1882, 1.3988, 0.237, 0.968) \]

and then we get a new iterate for \((P)\):

\[ x^{k+1} = D^k z^{\text{new}} \]
\[ = (0.95, 0.14, 0.05, 1.86) \]

with value 0.82.
Rescaling in \((x_3, x_4)\)-space

Original:

- \(x^k\)

Rescaled:

- \(d_3 = -0.3765\)
- \(d_4 = -0.160\)

Scales:

- \(z_3\):
  - 0
  - 5
  - \(z^k\)
  - \(\frac{2}{1.9}\)

- \(z_4\):
  - \(2\)
  - \(1.9\)
Rescaling in \((x_3, x_4)\)-space

Rescaled:

\[
\begin{align*}
d_3 &= -0.3765 \\
d_4 &= -0.160
\end{align*}
\]

Original:

\[
\begin{align*}
x_3 &= 2 \\
x_4 &= 1
\end{align*}
\]
Rescaling in \((x_3, x_4)\)-space

Rescaled:

\[ d_3 = -0.3765 \]
\[ d_4 = -0.160 \]

projection of \(-c\)
Rescaling in \((x_3, x_4)\)-space

Rescaled:

\[
d_3 = -0.3765 \\
d_4 = -0.160
\]

\[
z^k
\]

projection of \(-c\)

Original:

\[
x^k
\]

\[
x^{k+1}
\]
Outline

1. Introduction

2. The primal affine scaling algorithm
   - Rescaling so that we can take long steps
   - The relationship between \((P)\) and \((P^k)\)
   - Projections
   - Choosing the steplength
   - One iteration of the algorithm
   - Monotonicity of the algorithm
   - Getting a dual solution
   - Stopping the algorithm

3. Doing more centering

4. Historical notes

5. Bibliography
Steplength

We have $z^{new} = e + \beta d$, and we need to have $z^{new} > 0$. Thus, we need to select $\beta$ so that $1 + \beta d_i > 0$ for each component $i$. So, we need to pick

$$\beta < 1/(-d_i) \text{ if } d_i < 0,$$

so we take

$$\beta = 0.9/\max\{-d_i : d_i < 0\}.$$  \hspace{1cm} (3)

This will result in moving 0.9 of the way to the boundary of the feasible region of the rescaled problem ($P^k$).

Notice that one component of $z^{new}$ will be equal to 0.1.

(This discussion on choosing the steplength should remind you of the minimum ratio rule in the simplex algorithm.)
Choose steplength

Use a minimum ratio test to choose the steplength.

\[-P_{A_k}c^k\]
Choose steplength

Use a minimum ratio test to choose the steplength.

Current iterate

$z_4 = 0$
Choose steplength

Use a minimum ratio test to choose the steplength.
Outline

1. Introduction

2. The primal affine scaling algorithm
   - Rescaling so that we can take long steps
   - The relationship between \((P)\) and \((P^k)\)
   - Projections
   - Choosing the steplength
   - One iteration of the algorithm
     - Monotonicity of the algorithm
     - Getting a dual solution
     - Stopping the algorithm

3. Doing more centering

4. Historical notes

5. Bibliography
One iteration

Recall that when we looked at the revised simplex method, we found out that it was not necessary to calculate the whole of the simplex tableau.

In the same spirit of efficiency, it is not necessary to calculate the complete projection matrix $P_{Ak}$.

In practice, we would perform one iteration as follows:
Iteration

1. Given current iterate $x^k > 0$ feasible in $(P)$.
2. Calculate $D^k, A^k = AD^k, c^k = D^k c$.
3. Calculate $w^k = A^k c^k$.
4. Calculate $v^k = (A^k(A^k)^T)^{-1} w^k$. (Aside: In practice, we would not calculate the inverse explicitly, but we would solve the system of equations $(A^k(A^k)^T)v^k = w^k$ in order to find $v^k$.)
5. Calculate $g^k = (A^k)^T v^k$.
6. Calculate the direction $d^k$ in the problem $(P^k)$: $d^k = -c^k + g^k (= -P_{A^k} c^k)$.
7. Calculate step length $\beta$ using minimum ratio approach.
8. Update $z^{\text{new}} = e + \beta d^k$.
9. Update $x^{k+1} = D^k z^{\text{new}}$. 
Iteration

1. Given current iterate $x^k > 0$ feasible in $(P)$.
2. Calculate $D^k, A^k = AD^k, c^k = D^k c$.
3. Calculate $w^k = A^k c^k$.
4. Calculate $v^k = (A^k(A^k)^T)^{-1}w^k$. (Aside: In practice, we would not calculate the inverse explicitly, but we would solve the system of equations $(A^k(A^k)^T)v^k = w^k$ in order to find $v^k$.)
5. Calculate $g^k = (A^k)^Tv^k$.
6. Calculate the direction $d^k$ in the problem $(P^k)$:
   
   $$d^k = -c^k + g^k = -PAk c^k$$

7. Calculate step length $\beta$ using minimum ratio approach.
8. Update $z^{new} = e + \beta d^k$.
9. Update $x^{k+1} = D^k z^{new}$.
Iteration

1. Given current iterate \( x^k > 0 \) feasible in \((P)\).
2. Calculate \( D^k, A^k = AD^k, c^k = D^k c \).
3. Calculate \( w^k = A^k c^k \).
4. Calculate \( v^k = (A^k(A^k)^T)^{-1}w^k \). (Aside: In practice, we would not calculate the inverse explicitly, but we would solve the system of equations \((A^k(A^k)^T)v^k = w^k\) in order to find \(v^k\).)
5. Calculate \( g^k = (A^k)^Tv^k \).
6. Calculate the direction \( d^k \) in the problem \((P^k)\): \( d^k = -c^k + g^k (= -P_{A_k}c^k) \).
7. Calculate step length \( \beta \) using minimum ratio approach.
8. Update \( z^{new} = e + \beta d^k \).
9. Update \( x^{k+1} = D^k z^{new} \).
Iteration

1. Given current iterate $x^k > 0$ feasible in $(P)$.
2. Calculate $D^k, A^k = AD^k, c^k = D^k c$.
3. Calculate $w^k = A^k c^k$.
4. Calculate $v^k = (A^k (A^k)^T)^{-1} w^k$. (Aside: In practice, we would not calculate the inverse explicitly, but we would solve the system of equations $(A^k (A^k)^T) v^k = w^k$ in order to find $v^k$.)
5. Calculate $g^k = (A^k)^T v^k$.
6. Calculate the direction $d^k$ in the problem $(P^k)$:
   
   $d^k = -c^k + g^k (= - P_{A^k} c^k)$.

7. Calculate step length $\beta$ using minimum ratio approach.
8. Update $z^{new} = e + \beta d^k$.
9. Update $x^{k+1} = D^k z^{new}$.
The primal affine scaling algorithm

One iteration of the algorithm

Iteration

1. Given current iterate $x^k > 0$ feasible in $(P)$.
2. Calculate $D^k$, $A^k = AD^k$, $c^k = D^k c$.
3. Calculate $w^k = A^k c^k$.
4. Calculate $v^k = (A^k(A^k)^T)^{-1}w^k$. (Aside: In practice, we would not calculate the inverse explicitly, but we would solve the system of equations $(A^k(A^k)^T)v^k = w^k$ in order to find $v^k$.)
5. Calculate $g^k = (A^k)^Tv^k$.
6. Calculate the direction $d^k$ in the problem $(P^k)$:
   $d^k = -c^k + g^k (= -P_{A^k}c^k)$.
7. Calculate step length $\beta$ using minimum ratio approach.
8. Update $z^{new} = e + \beta d^k$.
9. Update $x^{k+1} = D^k z^{new}$.
Iteration

1. Given current iterate $x^k > 0$ feasible in $(P)$.
2. Calculate $D^k$, $A^k = AD^k$, $c^k = D^k c$.
3. Calculate $w^k = A^k c^k$.
4. Calculate $v^k = (A^k (A^k)^T)^{-1} w^k$. (Aside: In practice, we would not calculate the inverse explicitly, but we would solve the system of equations $(A^k (A^k)^T)v^k = w^k$ in order to find $v^k$.)
5. Calculate $g^k = (A^k)^T v^k$.
6. Calculate the direction $d^k$ in the problem $(P^k)$:
   \[ d^k = -c^k + g^k (= -P_{A^k} c^k) . \]
7. Calculate step length $\beta$ using minimum ratio approach.
8. Update $z^{new} = e + \beta d^k$.
9. Update $x^{k+1} = D^k z^{new}$.
Iteration

1. Given current iterate $x^k > 0$ feasible in $(P)$.
2. Calculate $D^k, A^k = AD^k, c^k = D^k c$.
3. Calculate $w^k = A^k c^k$.
4. Calculate $v^k = (A^k (A^k)^T)^{-1} w^k$. (Aside: In practice, we would not calculate the inverse explicitly, but we would solve the system of equations $(A^k (A^k)^T) v^k = w^k$ in order to find $v^k$.)
5. Calculate $g^k = (A^k)^T v^k$.
6. Calculate the direction $d^k$ in the problem $(P^k)$:
   $d^k = -c^k + g^k (= -P_{A^k} c^k)$.
7. Calculate step length $\beta$ using minimum ratio approach.
8. Update $z^{new} = e + \beta d^k$.
9. Update $x^{k+1} = D^k z^{new}$.
Iteration

1. Given current iterate $x^k > 0$ feasible in $(P)$.
2. Calculate $D^k$, $A^k = AD^k$, $c^k = D^k c$.
3. Calculate $w^k = A^k c^k$.
4. Calculate $v^k = (A^k (A^k)^T)^{-1} w^k$. (Aside: In practice, we would not calculate the inverse explicitly, but we would solve the system of equations $(A^k (A^k)^T) v^k = w^k$ in order to find $v^k$.)
5. Calculate $g^k = (A^k)^T v^k$.
6. Calculate the direction $d^k$ in the problem $(P^k)$:
   
   $d^k = -c^k + g^k (= -P_{A^k} c^k)$.
7. Calculate step length $\beta$ using minimum ratio approach.
8. Update $z^{new} = e + \beta d^k$.
9. Update $x^{k+1} = D^k z^{new}$.
The primal affine scaling algorithm

Iteration

1. Given current iterate $x^k > 0$ feasible in $(P)$.
2. Calculate $D^k, A^k = AD^k, c^k = D^k c$.
3. Calculate $w^k = A^k c^k$.
4. Calculate $v^k = (A^k (A^k)^T)^{-1} w^k$. (Aside: In practice, we would not calculate the inverse explicitly, but we would solve the system of equations $(A^k (A^k)^T) v^k = w^k$ in order to find $v^k$.)
5. Calculate $g^k = (A^k)^T v^k$.
6. Calculate the direction $d^k$ in the problem $(P^k)$:
   
   \[ d^k = -c^k + g^k (= -P_{A^k} c^k) \]
7. Calculate step length $\beta$ using minimum ratio approach.
8. Update $z^{new} = e + \beta d^k$.
9. Update $x^{k+1} = D^k z^{new}$.
Outline

1. Introduction

2. The primal affine scaling algorithm
   - Rescaling so that we can take long steps
   - The relationship between \((P)\) and \((P^k)\)
   - Projections
   - Choosing the steplength
   - One iteration of the algorithm
   - **Monotonicity of the algorithm**
   - Getting a dual solution
   - Stopping the algorithm

3. Doing more centering

4. Historical notes

5. Bibliography
Monotonicity

The algorithm does decrease the objective function at each iteration. This is because a projection matrix $M$ is *idempotent*, that is, it is symmetric and $MM = M$.

We then get that the objective function value of $z^{new}$ is

$$c^k T z^{new} = c^k T (e - \beta P_{A^k} c^k)$$
$$= c^k T e - \beta c^k T P_{A^k} c^k$$
$$= c^k T e - \beta c^k T P_{A^k} P_{A^k} c^k$$
$$= c^k T e - \beta (P_{A^k} c^k)^T (P_{A^k} c^k)$$

and this must be smaller than $c^k T e$ since it subtracts off the square of the length of the vector $P_{A^k} c^k$. 
Outline

1. Introduction
2. The primal affine scaling algorithm
   - Rescaling so that we can take long steps
   - The relationship between \((P)\) and \((P^k)\)
   - Projections
   - Choosing the steplength
   - One iteration of the algorithm
   - Monotonicity of the algorithm
   - Getting a dual solution
   - Stopping the algorithm
3. Doing more centering
4. Historical notes
5. Bibliography
Duality

It can be shown that

- $d^k = 0$ if and only if $x^k$ is a vertex.
- $c - A^T v^k \geq 0$ at the optimal vertex. The optimal vertex is the only one where we have $c - A^T v^k \geq 0$.

A possible dual iterate is

$$v^k = (A^k (A^k)^T)^{-1} A^k c^k. \tag{4}$$

Notice that the dual slacks are then $c - A^T v^k$, and that $d^k = -D^k (c - A^T v^k)$. Thus, the elements of $d^k$ give the product between the primal variable $d_i^k$ and the corresponding dual slack, so they give a measure of the complementary slackness. At optimality, we will have dual feasibility with this choice. It can be shown that this choice is actually dual optimal if $x^k$ is optimal for the primal problem.
Duality

It can be shown that

- \( d^k = 0 \) if and only if \( x^k \) is a vertex.
- \( c - A^T v^k \geq 0 \) at the optimal vertex. The optimal vertex is the only one where we have \( c - A^T v^k \geq 0 \).

A possible dual iterate is

\[
  v^k = (A^k (A^k)^T)^{-1} A^k c^k.
\]

(4)

Notice that the dual slacks are then \( c - A^T v^k \), and that \( d^k = -D^k (c - A^T v^k) \). Thus, the elements of \( d^k \) give the product between the primal variable \( d^k_i \) and the corresponding dual slack, so they give a measure of the complementary slackness. At optimality, we will have dual feasibility with this choice. It can be shown that this choice is actually dual optimal if \( x^k \) is optimal for the primal problem.
Outline

1. Introduction

2. The primal affine scaling algorithm
   - Rescaling so that we can take long steps
   - The relationship between (P) and (P^k)
   - Projections
   - Choosing the steplength
   - One iteration of the algorithm
   - Monotonicity of the algorithm
   - Getting a dual solution
   - Stopping the algorithm

3. Doing more centering

4. Historical notes

5. Bibliography
Termination

We stop the simplex algorithm when all the costs in the tableau are nonnegative. Unfortunately, we don’t have such a simple stopping rule for the primal affine scaling algorithm. However, there are some possibilities:

- Is $c - A^T v^k \geq -\epsilon e$ for some $\epsilon$ such as $10^{-6}$. (This corresponds to almost getting dual feasibility.)

- Is the change in the objective function from one iteration to the next smaller than some tolerance? That is, do we have $c^T x^k - c^T x^{k+1} \leq \epsilon$, or alternatively, $(c^T x^k - c^T x^{k+1})/c^T x^k \leq \epsilon$. This criterion indicates that we are not making much progress, so we should be very close to the optimal.

- Is $d^k$ a small vector? This indicates that we are close to getting complementary slackness. (See the previous subsection.)
Termination

We stop the simplex algorithm when all the costs in the tableau are nonnegative. Unfortunately, we don’t have such a simple stopping rule for the primal affine scaling algorithm. However, there are some possibilities:

- Is $c - A^T v^k \geq -\epsilon e$ for some $\epsilon$ such as $10^{-6}$. (This corresponds to almost getting dual feasibility.)

- Is the change in the objective function from one iteration to the next smaller than some tolerance? That is, do we have $c^T x^k - c^T x^{k+1} \leq \epsilon$, or alternatively, $(c^T x^k - c^T x^{k+1})/c^T x^k \leq \epsilon$. This criterion indicates that we are not making much progress, so we should be very close to the optimal.

- Is $d^k$ a small vector? This indicates that we are close to getting complementary slackness. (See the previous subsection.)
The primal affine scaling algorithm

Stopping the algorithm

### Termination

We stop the simplex algorithm when all the costs in the tableau are nonnegative. Unfortunately, we don’t have such a simple stopping rule for the primal affine scaling algorithm. However, there are some possibilities:

- Is \( c - A^T v^k \geq -\epsilon e \) for some \( \epsilon \) such as \( 10^{-6} \). (This corresponds to almost getting dual feasibility.)

- Is the change in the objective function from one iteration to the next smaller than some tolerance? That is, do we have \( c^T x^k - c^T x^{k+1} \leq \epsilon \), or alternatively, \( (c^T x^k - c^T x^{k+1})/c^T x^k \leq \epsilon \). This criterion indicates that we are not making much progress, so we should be very close to the optimal.

- Is \( d^k \) a small vector? This indicates that we are close to getting complementary slackness. (See the previous subsection.)
Outline

1. Introduction
2. The primal affine scaling algorithm
   - Rescaling so that we can take long steps
   - The relationship between $(P)$ and $(P^k)$
   - Projections
   - Choosing the steplength
   - One iteration of the algorithm
   - Monotonicity of the algorithm
   - Getting a dual solution
   - Stopping the algorithm
3. Doing more centering
4. Historical notes
5. Bibliography
Centering direction

The algorithm still sometimes makes very slow progress when it gets close to a corner of the feasible region. One possible remedy is to try explicitly to center the iterates. We do this by moving in the direction

$$d^{\text{center}} := P_A^k e$$

in the scaled problem ($P^k$). Notice that this direction is trying to increase all components simultaneously.

An algorithm will usually do some steps using the regular $d^k$ we found earlier and then mix in some centering steps.
Centering direction example

In the example we were looking at earlier, the centering direction is

\[ d_{\text{center}} = P_A^k e = \begin{bmatrix} 0.059 & 0 & -0.235 & 0 \\ 0 & 0.997 & 0 & -0.052 \\ -0.235 & 0 & 0.941 & 0 \\ 0 & -0.052 & 0 & 0.003 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.176 \\ 0.945 \\ 0.706 \\ -0.049 \end{bmatrix}. \]

Notice that this has the effect of increasing \( x_2 \) and decreasing \( x_1 \), which does seem to move us towards the middle of the feasible region.
Centering direction for the example

\[ \begin{align*}
  z_4 &\quad 0 \\
  \frac{2}{1.9} &\quad z^k \\
  z_3 &\quad 5
\end{align*} \]
Centering direction for the example

projection of $-c$
Centering direction for the example

centering direction:
\[ d_3 = 0.706 \]
\[ d_4 = -0.049 \]
Commercial implementations

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. The primal affine scaling algorithm
   - Rescaling so that we can take long steps
   - The relationship between $(P)$ and $(P^k)$
   - Projections
   - Choosing the steplength
   - One iteration of the algorithm
   - Monotonicity of the algorithm
   - Getting a dual solution
   - Stopping the algorithm
3. Doing more centering
4. Historical notes
5. Bibliography
Historical notes

Serious research on interior point methods started with the publication of Karmarkar’s paper [Kar84]. Karmarkar made some very bold claims for the performance of his algorithm, which were somewhat borne out by subsequent results.

The primal affine scaling method is a simplification of Karmarkar’s original algorithm that was proposed by several researchers in 1986, including Vanderbei, Meketon and Freedman [VMF86].

This method was in fact discovered by the Russian Dikin [Dik67] in 1967, although this discovery remained unknown in the west until much later. A good (mathematical) description of the algorithm can be found in [VL90].
Textbooks

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

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

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. 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.}
Outline

1. Introduction
2. The primal affine scaling algorithm
   - Rescaling so that we can take long steps
   - The relationship between \((P)\) and \((P^k)\)
   - Projections
   - Choosing the steplength
   - One iteration of the algorithm
   - Monotonicity of the algorithm
   - Getting a dual solution
   - Stopping the algorithm
3. Doing more centering
4. Historical notes
5. Bibliography
Early references

N. K. Karmarkar.
A new polynomial-time algorithm for linear programming. 
*Combinatorica, 4:373–395, 1984.*

A modification of Karmarkar’s linear programming algorithm. 

I. I. Dikin.
Iterative solution of problems of linear and quadratic programming. 

R. J. Vanderbei and J. C. Lagarias.
I. I. Dikin’s convergence result for the affine–scaling algorithm. 
Early references

N. K. Karmarkar.
A new polynomial-time algorithm for linear programming. 

A modification of Karmarkar’s linear programming algorithm. 

I. I. Dikin.
Iterative solution of problems of linear and quadratic programming. 

R. J. Vanderbei and J. C. Lagarias.
I. I. Dikin’s convergence result for the affine–scaling algorithm. 
Early references


Early references

N. K. Karmarkar.
A new polynomial-time algorithm for linear programming.

A modification of Karmarkar’s linear programming algorithm.

I. I. Dikin.
Iterative solution of problems of linear and quadratic programming.

R. J. Vanderbei and J. C. Lagarias.
I. I. Dikin’s convergence result for the affine–scaling algorithm.
Textbooks


Textbooks


Textbooks


Textbooks


Textbooks

S. P. Boyd and L. Vandenberghe. 
*Convex Optimization.*

D. G. Luenberger and Y. Ye.
*Linear and Nonlinear Programming.*

C. Roos, T. Terlaky, and J.-Ph. Vial.
*Interior Point Methods for Linear Optimization.*

R. J. Vanderbei.
*Linear Programming: Foundations and Extensions.*

S. J. Wright.
*Primal-dual interior point methods.*