ISyE/Math/CS/Stat 525
Linear Optimization

3. The simplex method

Prof. Carla Michini
University of Wisconsin-Madison

Spring 2020

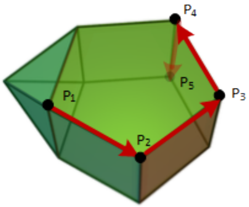
Outline

- From Theorem 2.7: If a linear programming problem in standard form has an optimal solution, then there exists a basic feasible solution that is optimal.

Outline

- From Theorem 2.7: If a linear programming problem in standard form has an optimal solution, then there exists a basic feasible solution that is optimal.

- The simplex method is based on this fact.
- It searches for an optimal solution by moving from one basic feasible solution to another, along the edges of the feasible set, always in a cost reducing direction.
- Eventually, a basic feasible solution is reached at which none of the available edges leads to a cost reduction.
- Such a basic feasible solution is optimal and the algorithm terminates.



Outline

Sec. 3.1 We present necessary and sufficient condition for a feasible solution to be optimal.

Sec. 3.2 We develop the simplex method.

Sec. 1.6 We review the conventions used in counting the number of operations performed by algorithms.

Sec. 3.3 We discuss a few different implementations, including the simplex tableau and the revised simplex method.

Sec. 3.4 We discuss how the simplex method avoids cycling in order to reach an optimal solution.

Sec. 3.5 We understand how it finds an initial basic feasible solution.

Sec. 3.6 The roots of the name "simplex method".

Sec. 3.7 We discuss its running time.

Outline

- Throughout this chapter, we consider the standard form problem

$$\begin{aligned} \text{minimize} \quad & c'x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0. \end{aligned}$$

- We let P be the corresponding feasible set.
- We assume that the dimensions of the matrix A are $m \times n$ and that its rows are linearly independent.
- We continue using our previous notation:
  - A_i is the ith column of the matrix A,
  - a_i' is the ith row of the matrix A.

3.1 Optimality conditions

Optimality conditions

Many optimization algorithms are structured as follows:

- Given a feasible solution, we search its neighborhood to find a nearby feasible solution with lower cost.
- If no nearby feasible solution leads to a cost improvement, the algorithm terminates and we have a locally optimal solution.

Optimality conditions

Many optimization algorithms are structured as follows:

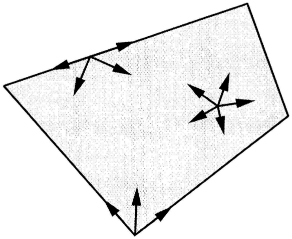
- Given a feasible solution, we search its neighborhood to find a nearby feasible solution with lower cost.
- If no nearby feasible solution leads to a cost improvement, the algorithm terminates and we have a locally optimal solution.

- For general optimization problems, a locally optimal solution need not be (globally) optimal.
- Fortunately, in linear programming, local optimality implies global optimality. You will show this in an exercise.
- In this section, we concentrate on the problem of searching for a direction of cost decrease in a neighborhood of a given basic feasible solution, and on the associated optimality conditions.

Optimality conditions

- Suppose that we are at a point $x \in P$ and that we contemplate moving away from x, in the direction of a vector $d \in \mathbb{R}^n$.
- Clearly, we should only consider those choices of d that do not immediately take us outside the feasible set.
- This leads to the following definition.

Definition 3.1
Let x be a point in a polyhedron P. A vector $d \in \mathbb{R}^n$ is said to be a feasible direction at x, if there exists a positive scalar θ for which $x + \theta d \in P$.



Optimality conditions

- Let x be a basic feasible solution to the standard form problem.
- Let $B(1), \ldots, B(m)$ be the indices of the basic variables, and let
$$B = [A_{B(1)} \cdots A_{B(m)}]$$
be the corresponding basis matrix.
- In particular, we have $x_i =$ for every nonbasic variable, while the vector $x_B = (x_{B(1)}, \ldots, x_{B(m)})$ of basic variables is given by
$$x_B =$$

Optimality conditions

- Let x be a basic feasible solution to the standard form problem.
- Let $B(1), \ldots, B(m)$ be the indices of the basic variables, and let
$$B = [A_{B(1)} \cdots A_{B(m)}]$$
be the corresponding basis matrix.
- In particular, we have $x_i = 0$ for every nonbasic variable, while the vector $x_B = (x_{B(1)}, \ldots, x_{B(m)})$ of basic variables is given by
$$x_B = B^{-1}b.$$

Optimality conditions

- We consider the possibility of moving away from x, to a new vector
$$x + \theta d.$$
- We do so by selecting a nonbasic variable x_j (which is initially at zero level), and increasing it to a positive value θ, while keeping the remaining nonbasic variables at zero.
- Algebraically, $d_j = 1$, and $d_i = 0$ for every nonbasic index i other than j.
- At the same time, the vector x_B of basic variables changes to
$$x_B + \theta d_B,$$
where $d_B = (d_{B(1)}, d_{B(2)}, \ldots, d_{B(m)})$ is the vector with those components of d that correspond to the basic variables.

Optimality conditions

- Given that we are only interested in feasible solutions, we require

$$A(x + \theta d) = b \quad \Leftrightarrow \quad \cancel{Ax} + \theta A d = \cancel{b} \quad \Leftrightarrow \quad \theta A d = 0,$$

  where we have used $Ax = b$, since x is feasible.
- Thus, for the equality constraints to be satisfied for $\theta > 0$, we need

$$Ad = 0.$$

Optimality conditions

- Given that we are only interested in feasible solutions, we require

$$A(x+\theta d)=b \quad \Leftrightarrow \quad \cancel{Ax}+\theta Ad=\cancel{b} \quad \Leftrightarrow \quad \theta Ad=0,$$

  where we have used $Ax=b$, since x is feasible.
- Thus, for the equality constraints to be satisfied for $\theta>0$, we need

$$Ad=0.$$

- Recall now that $d_j=1$, and that $d_i=0$ for all other nonbasic indices i.
- Then,

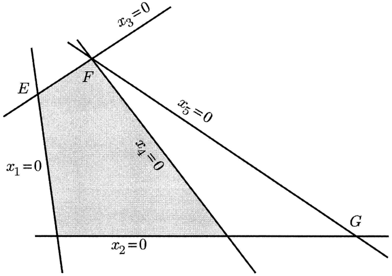
$$0=Ad=\sum_{i=1}^{n}A_i d_i=\sum_{i=1}^{m}A_{B(i)}d_{B(i)}+A_j=Bd_B+A_j.$$

- Since the basis matrix B is invertible, we obtain

$$d_B=-B^{-1}A_j.$$

Optimality conditions

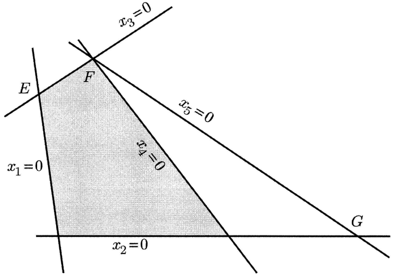
- The direction vector d that we have just constructed is called the jth basic direction.
- We have so far guaranteed that the equality constraints are respected by the vectors $x + \theta d$, for $\theta > 0$.



$n = 5, \ m = 3$

Optimality conditions

- How about the nonnegativity constraints?
- We recall that the variable x_j is increased, and all other nonbasic variables stay at zero level.
- Thus, we need only worry about the basic variables. We distinguish two cases:



$n = 5,\ m = 3$

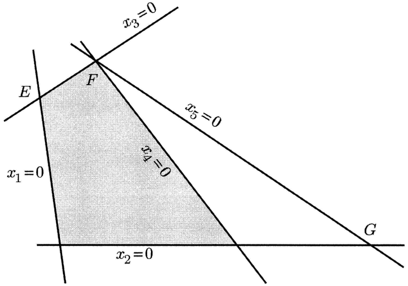
Optimality conditions

Case (a): x is a nondegenerate basic feasible solution.

- Then, $x_B > 0$, from which it follows that, when θ is sufficiently small,

$$x_B + \theta d_B \geq 0,$$

  and feasibility is maintained.
- In particular, d is a feasible direction.



$n = 5,\ m = 3$

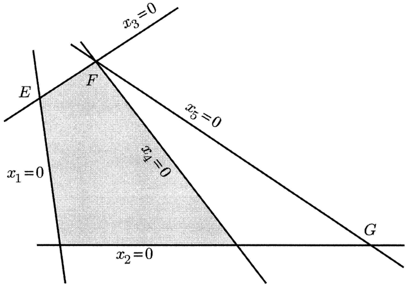
Optimality conditions

Case (b): x is a degenerate basic feasible solution.

- Then, d is not always a feasible direction.
- Indeed, it is possible that, for a basic variable,

$$x_{B(i)} = 0$$
$$d_{B(i)} < 0.$$

- In that case, if we follow the jth basic direction, the nonnegativity constraint for $x_{B(i)}$ is immediately violated, and we are led to infeasible solutions.



$n = 5,\ m = 3$

Optimality conditions

We now study the effects on the cost function if we move along a basic direction.

- If d is the jth basic direction, the cost change is given by

$$c'(x + \theta d) - c'x = \theta c' d.$$

- The rate of cost change along the direction d is given by

$$c' d = \sum_{i=1}^{n} c_i d_i = \sum_{i=1}^{m} c_{B(i)} d_{B(i)} + c_j = c'_B d_B + c_j = c_j - c'_B B^{-1} A_j.$$

  where $c_B = (c_{B(1)}, \ldots, c_{B(m)})$, and where we have used $d_B = -B^{-1}A_j$.
- For an intuitive interpretation:
  - c_j is the cost per unit increase in the variable x_j,
  - $-c'_B B^{-1} A_j$ is the cost of the compensating change in the basic variables necessitated by the constraint $Ax = b$.

Optimality conditions

- The latter quantity is important enough to warrant a definition.

Definition 3.2
Let x be a basic solution, let B be an associated basis matrix, and let c_B be the vector of costs of the basic variables. For each j, we define the <u>reduced cost</u> $\bar{c}_j$ of the variable x_j according to the formula

$$\bar{c}_j = c_j - c_B' B^{-1} A_j.$$

- Note that the definition holds also if j is a basic index!

Example 3.1

Consider the linear programming problem

$$\begin{aligned} \text{minimize} \quad & c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 \\ \text{subject to} \quad & x_1 + x_2 + x_3 + x_4 = 2 \\ & 2x_1 + 3x_3 + 4x_4 = 2 \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

- We have $A_1 = (1, 2)$ and $A_2 = (1, 0)$.
- Since they are linearly independent, we can choose x_1 and x_2 as our basic variables.
- The corresponding basis matrix is

$$B = \begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix}.$$

Example 3.1

Consider the linear programming problem

$$\begin{aligned} \text{minimize} \quad & c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 \\ \text{subject to} \quad & x_1 + x_2 + x_3 + x_4 = 2 \\ & 2x_1 + 3x_3 + 4x_4 = 2 \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

- We set $x_3 = x_4 = 0$, and solve for x_1, x_2, to obtain $x_1 = 1$ and $x_2 = 1$.
- We have thus obtained the nondegenerate basic feasible solution

$$(1, 1, 0, 0).$$

Example 3.1

Consider the linear programming problem

$$\begin{aligned} \text{minimize} \quad & c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 \\ \text{subject to} \quad & x_1 + x_2 + x_3 + x_4 = 2 \\ & 2x_1 + 3x_3 + 4x_4 = 2 \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

- The 3rd basic direction d is constructed as follows:
  - We have $d_3 = 1$ and $d_4 = 0$.
  - The vector $d_B = (d_1, d_2)$ is obtained using $d_B = -B^{-1}A_j$:

$$\begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = -B^{-1}A_3 = -\begin{bmatrix} 0 & 1/2 \\ 1 & -1/2 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} -3/2 \\ 1/2 \end{bmatrix}.$$

  - The 3rd basic direction is then the vector $d = (-\frac{3}{2}, \frac{1}{2}, 1, 0)$.

Example 3.1

Consider the linear programming problem

$$\begin{aligned} \text{minimize} \quad & c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 \\ \text{subject to} \quad & x_1 + x_2 + x_3 + x_4 = 2 \\ & 2x_1 + 3x_3 + 4x_4 = 2 \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

- The rate of cost change along this basic direction is

$$c'd = -\frac{3}{2}c_1 + \frac{1}{2}c_2 + c_3.$$

- This is the ~~same as~~ the reduced cost of the variable x_3.

这是按照x_3增加1计算得到的cost变化量，按照
定义，就是reduced cost of the variable x3

Optimality conditions

We now calculate the reduced cost

$$\bar{c}_j = c_j - c_B' B^{-1} A_j.$$

for the case of a basic variable ($j = B(i)$ for some $i \in \{1, \ldots, m\}$).

- Since $B = [A_{B(1)} \cdots A_{B(m)}]$, we have

$$B^{-1}[A_{B(1)} \cdots A_{B(m)}] = I,$$

  where I is the $m \times m$ identity matrix.
- In particular, $B^{-1}A_{B(i)}$ is the ith column of the identity matrix, which is the ith unit vector e_i.
- Therefore, for every basic variable $x_{B(i)}$, we have

$$\bar{c}_{B(i)} = c_{B(i)} - c_B' B^{-1} A_{B(i)} = c_{B(i)} - c_B' e_i = c_{B(i)} - c_{B(i)} = 0.$$

- Thus the reduced cost of every basic variable is zero.

Optimality conditions

- Our next result provides us with optimality conditions.
- Given our interpretation of the reduced costs as rates of cost change along certain directions, this result is intuitive.

Theorem 3.1
Consider a basic feasible solution x associated with a basis matrix B, and let $\bar{c}$ be the corresponding vector of reduced costs.

(a) If $\bar{c} \geq 0$, then x is optimal.

(b) If x is optimal and nondegenerate, then $\bar{c} \geq 0$.

Note that the contrapositive of (b) is:

(b') If $\bar{c}_j < 0$ for some j, then x is degenerate or not optimal.

Let's prove Theorem 3.1!

Optimality conditions

- Note that Theorem 3.1 allows the possibility that x is a (degenerate) optimal basic feasible solution, but that $\bar{c}_j < 0$ for some nonbasic index j.
- According to Theorem 3.1, in order to decide whether a nondegenerate basic feasible solution is optimal, we need only to check whether all reduced costs are nonnegative, which is the same as examining the $n - m$ basic directions.
- If x is a degenerate basic feasible solution, an equally simple computational test for determining whether x is optimal is not available.
- Fortunately, the simplex method manages to get around this difficulty in an effective manner.

Optimality conditions

- In order to use Theorem 3.1 and assert that a certain basic solution is optimal, we need to satisfy two conditions:
  (a) Feasibility,
  (b) Nonnegativity of the reduced costs.
- This leads us to the following definition.

Definition 3.3
A basis matrix B is said to be <u>optimal</u> if:
(a) $B^{-1}b \geq 0$, and
(b) $\bar{c}' = c' - c_B' B^{-1} A \geq 0'$.

Optimality conditions

- If an optimal basis is found, the corresponding basic solution is feasible, satisfies the optimality conditions, and is therefore optimal.
- On the other hand, it can happen that we have found a basis that is not optimal, and the corresponding basic solution is optimal.
- In this case at least one reduced cost is negative.
- Thus the basic solution is degenerate.

Definition 3.3
A basis matrix B is said to be <u>optimal</u> if:

(a) $B^{-1}b \geq 0$, and
(b) $\bar{c}' = c' - c_B' B^{-1} A \geq 0'$.

3.2 Development of the simplex method

Development of the simplex method

We now continue with the development of the simplex method.

- Our main task is to work out the details of how to move to a better basic feasible solution, whenever a profitable basic direction is discovered.

- Let us assume that every basic feasible solution is nondegenerate.
- This assumption will remain in effect until it is explicitly relaxed later in this section.

Development of the simplex method

- Suppose that we are at a basic feasible solution x and that we have computed the reduced costs $\bar{c}_j$ of the nonbasic variables.
- If all of them are nonnegative, Theorem 3.1 shows that we have an optimal solution, and we stop.
- Otherwise, the reduced cost $\bar{c}_j$ of a nonbasic variable x_j is negative, and the jth basic direction d is a feasible direction of cost decrease.
- While moving along this direction d, the nonbasic variable x_j becomes positive and all other nonbasic variables remain at zero.
- We describe this situation by saying that A_j (or x_j) enters the basis.

Development of the simplex method

- Once we start moving away from x along the direction d, we are tracing points of the form

$$x + \theta d, \qquad \text{where } \theta \geq 0.$$

- Since costs decrease along the direction d, it is desirable to move as far as possible.
- This takes us to the point $x + \theta^* d$, where

$$\theta^* = \max\{\theta \geq 0 \mid x + \theta d \in P\}.$$

- The resulting cost change is

$$\theta^* c' d = \theta^* \bar{c}_j.$$

Development of the simplex method

We now derive a formula for θ^*.

- Given that $Ad = 0$, we have

$$A(x + \theta d) = Ax = b \qquad \forall \theta \in \mathbb{R},$$

  and the equality constraints will never be violated.
- Thus, $x + \theta d$ can become infeasible only if one of its components becomes negative.

Development of the simplex method

We distinguish two cases:

(a) If $d \geq 0$, then $x + \theta d \geq 0$ for all $\theta \geq 0$, the vector $x + \theta d$ never becomes infeasible, and we let $\theta^* = \infty$.

(b) If $d_i < 0$ for some i, the constraint $x_i + \theta d_i \geq 0$ becomes

$$\theta \leq -\frac{x_i}{d_i}.$$

- This constraint on θ must be satisfied for every i with $d_i < 0$.
- Thus, the largest possible value of θ is

$$\theta^* = \min_{i \mid d_i < 0} \left(-\frac{x_i}{d_i} \right).$$

Development of the simplex method

$$\theta^* = \min_{i \mid d_i < 0} \left(-\frac{x_i}{d_i} \right).$$

- Recall that if x_i is a nonbasic variable, then either x_i is the entering variable and $d_i = 1$, or else $d_i = 0$.
- In either case, d_i is nonnegative.
- Thus, we only need to consider the basic variables and we have the equivalent formula

$$\theta^* = \min_{i=1,\dots,m \mid d_{B(i)} < 0} \left(-\frac{x_{B(i)}}{d_{B(i)}} \right).$$

- Note that $\theta^* > 0$, because $x_{B(i)} > 0$ for all i, as a consequence of nondegeneracy.

Example 3.2

This is a continuation of Example 3.1.

$$\begin{aligned} \text{minimize} \quad & c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 \\ \text{subject to} \quad & x_1 + x_2 + x_3 + x_4 = 2 \\ & 2x_1 + 3x_3 + 4x_4 = 2 \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

- We again consider the basic feasible solution

$$x = (1, 1, 0, 0).$$

- The reduced cost $\bar{c}_3$ of the nonbasic variable x_3 was

$$\bar{c}_3 = -\frac{3}{2}c_1 + \frac{1}{2}c_2 + c_3.$$

- Suppose that $c = (2, 0, 0, 0)$, in which case, we have

$$\bar{c}_3 = -3.$$

Example 3.2

- Since $\bar{c}_3$ is negative, we form the 3rd basic direction, which is

$$d = \left(-\frac{3}{2}, \frac{1}{2}, 1, 0\right).$$

- We consider vectors of the form

$$x + \theta d, \qquad \text{with } \theta \geq 0.$$

- As θ increases, the only component of x that decreases is the first one (because $d_1 < 0$).
- The largest possible value of θ is given by

$$\theta^* = -\frac{x_1}{d_1} = \frac{2}{3}.$$

- This takes us to the point

$$y = x + \frac{2}{3}d = \left(0, \frac{4}{3}, \frac{2}{3}, 0\right).$$

Example 3.2

$$\begin{aligned} \text{minimize} \quad & c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 \\ \text{subject to} \quad & x_1 + x_2 + x_3 + x_4 = 2 \\ & 2x_1 + 3x_3 + 4x_4 = 2 \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

- Consider our new vector y

$$y = \left(0, \frac{4}{3}, \frac{2}{3}, 0\right).$$

- Note that the columns A_2 and A_3 corresponding to the nonzero variables at the new vector y are $(1, 0)$ and $(1, 3)$, respectively, and are linearly independent.
- Therefore, they form a basis and the vector y is the corresponding basic feasible solution.
- In particular, A_3 (or x_3) has entered the basis and A_1 (or x_1) has exited the basis.

Development of the simplex method

- Once θ^* is chosen, and assuming it is finite, we move to the new feasible solution
$$y = x + \theta^* d.$$
- Since $x_j = 0$ and $d_j = 1$, we have $y_j = \theta^* > 0$.
- Let ℓ be a minimizing index in the choice of θ^*, that is,
$$-\frac{x_{B(\ell)}}{d_{B(\ell)}} = \min_{i=1,\dots,m \,|\, d_{B(i)}<0} \left(-\frac{x_{B(i)}}{d_{B(i)}} \right) = \theta^*.$$
- Hence
$$y_{B(\ell)} = x_{B(\ell)} + \theta^* d_{B(\ell)} = x_{B(\ell)} - \frac{x_{B(\ell)}}{\cancel{d_{B(\ell)}}} \cancel{d_{B(\ell)}} = 0.$$

Development of the simplex method

- The basic variable $x_{B(\ell)}$ has become zero, and the nonbasic variable x_j has become positive.
- This suggests that x_j should replace $x_{B(\ell)}$ in the basis.
- Accordingly, we take the old basis matrix B and replace $A_{B(\ell)}$ with A_j:

$$\bar{B} = \left[A_{B(1)} \cdots A_{B(\ell-1)} \; A_j \; A_{B(\ell+1)} \cdots A_{B(m)}\right].$$

- Equivalently, we are replacing the set $\{B(1), \ldots, B(m)\}$ of basic indices by a new set $\{\bar{B}(1), \ldots, \bar{B}(m)\}$ of indices given by

$$\bar{B}(i) = \begin{cases} B(i), & i \neq \ell, \\ j, & i = \ell. \end{cases}$$

Development of the simplex method

Theorem 3.2

(a) The columns $A_{\bar{B}(i)}$, $i = 1, \ldots, m$, are linearly independent and, therefore, $\bar{B}$ is a basis matrix.

(b) The vector $y = x + \theta^* d$ is a basic feasible solution associated with the basis matrix $\bar{B}$.

Let's prove it!

Development of the simplex method

- Since $\theta^* > 0$, the new basic feasible solution $x + \theta^* d$ is distinct from x.
- Since d is a direction of cost decrease, the cost of this new basic feasible solution is strictly smaller than the cost of x.
- We have therefore accomplished our objective of moving to a new basic feasible solution with lower cost.

Development of the simplex method

- We can now summarize a typical iteration of the simplex method, also known as a pivot.
- It is convenient to define a vector $u = (u_1, \ldots, u_m)$ by letting

$$u = -d_B = B^{-1}A_j,$$

  where A_j is the column that enters the basis.
- In particular,

$$u_i = -d_{B(i)}, \qquad \text{for } i = 1, \ldots, m.$$

Development of the simplex method

An iteration of the simplex method

1. We start with a basis consisting of the basic columns $A_{B(1)}, \ldots, A_{B(m)}$, and an associated basic feasible solution x.
2. Compute the reduced costs $\bar{c}_j = c_j - c_B' B^{-1} A_j$ for all nonbasic indices j.
   - If they are all nonnegative, the current basic feasible solution is optimal, and the algorithm terminates.
   - Else, choose some j for which $\bar{c}_j < 0$.
3. Compute $u = B^{-1} A_j$. If no component of u is positive, we have $\theta^* = \infty$, the optimal cost is $-\infty$, and the algorithm terminates.

Development of the simplex method

An iteration of the simplex method

4. If some component of u is positive, let
$$\theta^* = \min_{i=1,\ldots,m \,|\, u_i > 0} \frac{x_{B(i)}}{u_i}.$$

5. Let ℓ be such that
$$\theta^* = \frac{x_{B(\ell)}}{u_\ell}.$$

   Form a new basis by replacing $A_{B(\ell)}$ with A_j. If y is the new basic feasible solution, the values of the new basic variables are $y_j = \theta^*$ and $y_{B(i)} = x_{B(i)} - \theta^* u_i$, $i \neq \ell$.

Development of the simplex method

- The simplex method is initialized with an arbitrary basic feasible solution, which, for feasible standard form problems, is guaranteed to exist (cf. Corollary 2.2).

Development of the simplex method

- The following theorem states that, in the nondegenerate case, the simplex method works correctly and terminates after a finite number of iterations.

Theorem 3.3
Assume that the feasible set is nonempty and that every basic feasible solution is nondegenerate. Then, the simplex method terminates after a finite number of iterations. At termination, there are the following two possibilities:

(a) We have an optimal basis B and an associated basic feasible solution which is optimal.

(b) We have found a vector d satisfying $Ad = 0$, $d \geq 0$, and $c'd < 0$, and the optimal cost is $-\infty$.

Let's prove it!

Development of the simplex method

- Theorem 3.3 provides an independent proof of some of the results of Chapter 2 for nondegenerate standard form problems.
- In particular, it shows that for standard form problems that are nondegenerate and feasible:
  - either the optimal cost is $-\infty$, or
  - there exists a basic feasible solution which is optimal.

  (cf. Theorem 2.8 in Section 2.6.)

The simplex method for degenerate problems

The simplex method for degenerate problems

- We have been working so far under the assumption that all basic feasible solutions are nondegenerate.
- Suppose now that the same algorithm is used in the presence of degeneracy.
- Then, the following new possibilities may be encountered in the course of the algorithm.

The simplex method for degenerate problems

(a) If the current basic feasible solution x is degenerate, θ^* can be equal to zero, in which case, the new basic feasible solution y is the same as x.

- This happens if some basic variable $x_{B(\ell)}$ is equal to zero and the corresponding component $d_{B(\ell)}$ of the direction vector d is negative.

The simplex method for degenerate problems

(a) If the current basic feasible solution x is degenerate, θ^* can be equal to zero, in which case, the new basic feasible solution y is the same as x.

- This happens if some basic variable $x_{B(\ell)}$ is equal to zero and the corresponding component $d_{B(\ell)}$ of the direction vector d is negative.
- Nevertheless, we can still define a new basis $\bar{B}$, by replacing $A_{B(\ell)}$ with A_j, and Theorem 3.2 is still valid.

Theorem 3.2

(a) The columns $A_{B(i)}$, $i \neq \ell$, and A_j are linearly independent and, therefore, $\bar{B}$ is a basis matrix.

(b) The vector $y = x + \cancel{\theta^*} d$ is a basic feasible solution associated with the basis matrix $\bar{B}$.

The simplex method for degenerate problems

(b) Even if θ^* is positive, it may happen that more than one of the original basic variables becomes zero at the new point $x + \theta^* d$.

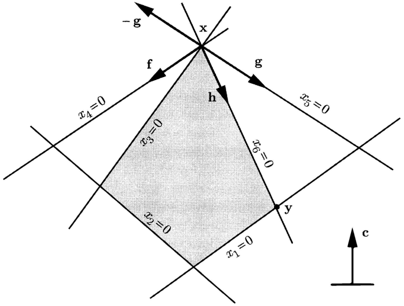
- Since only one of them exits the basis, the others remain in the basis at zero level, and the new basic feasible solution is degenerate.

The simplex method for degenerate problems

- Basis changes while staying at the same basic feasible solution are not in vain.
- A sequence of such basis changes may lead to the eventual discovery of a cost reducing feasible direction.

The simplex method for degenerate problems

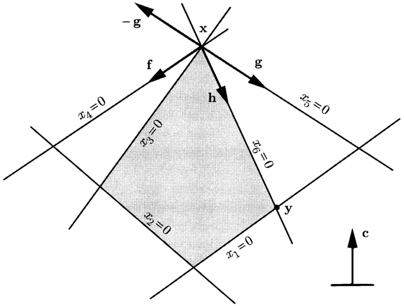
Example:



- The basic feasible solution x is degenerate.
- If x_4 and x_5 are the nonbasic variables, then
  - the 4th basic direction is g,
  - the 5th basic direction is f.
- For either of these two basic directions, we have $\theta^* = 0$.

The simplex method for degenerate problems

Example:



- However, if we perform a change of basis, with x_4 entering the basis and x_6 exiting, the new nonbasic variables are x_5 and x_6.
- Then
  - the 5th basic direction is h,
  - the 6th basic direction is $-g$.
- In particular, we can now follow direction h to reach a new basic feasible solution y with lower cost.

The simplex method for degenerate problems

- A sequence of basis changes might lead back to the initial basis, in which case the algorithm may loop indefinitely.
- This undesirable phenomenon is called cycling.

- It is sometimes maintained that cycling is an exceptionally rare phenomenon.
- However, for many highly structured linear programming problems, most basic feasible solutions are degenerate, and cycling is a real possibility.

- We will see that cycling can be avoided by judiciously choosing the variables that will enter or exit the basis.
- We now discuss the freedom available in this respect.

Pivot Selection

Pivot Selection

- The simplex algorithm, as we described it, has certain degrees of freedom:
  - In Step 2, we are free to choose any j whose reduced cost $\bar{c}_j$ is negative.
  - In Step 5, there may be several indices ℓ that attain the minimum in the definition of θ^*, and we are free to choose any one of them.
- Rules for making such choices are called pivoting rules.

Pivot Selection

Regarding the choice of the entering column, the following rules are some natural candidates:

(a) Choose a column A_j, with $\bar{c}_j < 0$, whose reduced cost is the most negative.

- Since the reduced cost is the rate of change of the cost function, this rule chooses a direction along which the cost decreases at the fastest rate.

Pivot Selection

Regarding the choice of the entering column, the following rules are some natural candidates:

(a) Choose a column A_j, with $\bar{c}_j < 0$, whose reduced cost is the most negative.

- Since the reduced cost is the rate of change of the cost function, this rule chooses a direction along which the cost decreases at the fastest rate.
- However, the actual cost decrease depends on how far we move along the chosen direction.
- This suggests the next rule.

Pivot Selection

(b) Choose a column with $\bar{c}_j < 0$ for which the corresponding cost decrease $\theta^* |\bar{c}_j|$ is largest.

- This rule offers the possibility of reaching optimality after a smaller number of iterations.
- On the other hand, the computational burden at each iteration is larger, because we need to compute θ^* for each column with $\bar{c}_j < 0$.
- The available empirical evidence suggests that the overall running time does not improve.

Pivot Selection

- For large problems, even rule (a) can be computationally expensive, because it requires the computation of the reduced cost of every variable.

- In practice, simpler rules are sometimes used, such as the smallest subscript rule, that chooses the smallest j for which $\bar{c}_j$ is negative.
- Under this rule, once a negative reduced cost is discovered, there is no reason to compute the remaining reduced costs.

Pivot Selection

- Regarding the choice of the exiting column, the simplest option is again the smallest subscript rule: out of all variables eligible to exit the basis, choose one with the smallest subscript.
- It turns out that by following the smallest subscript rule for both the entering and the exiting column, cycling can be avoided (cf. Section 3.4).

3.3 Implementations of the simplex method

Implementations of the simplex method

- In this section, we discuss some ways of carrying out the mechanics of the simplex method.
- But first, we review the conventions used in describing the computational requirements (operation count) of algorithms.

1.6 Algorithms and operation counts

Algorithms and operation counts

- An algorithm is a finite set of instructions of the type used in common programming languages.
- We are interested in comparing algorithms without having to examine the details of a particular implementation.
- As a first approximation, this can be accomplished by counting the number of arithmetic operations required by an algorithm.
- This approach is often adequate even though it ignores the fact that adding or multiplying large integers or high-precision floating point numbers is more demanding than adding or multiplying single-digit integers.

Example 1.9

(a) Inner product. Given vectors $a, b \in \mathbb{R}^n$, compute $a'b$.

$$a'b = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n.$$

- The natural algorithm requires n multiplications and $n - 1$ additions.
- Total number of arithmetic operations: $2n - 1$.

Example 1.9

(a) Inner product. Given vectors $a, b \in \mathbb{R}^n$, compute $a'b$.

$$a'b = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n.$$

- The natural algorithm requires n multiplications and $n-1$ additions.
- Total number of arithmetic operations: $2n-1$.

(b) Matrix multiplication. Given matrices A, B of dimensions $n \times n$, compute $C = AB$.

$$c_{ij} = a_i' B_j, \qquad \forall i, j = 1, \ldots, n.$$

- There are n^2 entries of AB to be evaluated.
- To obtain each one, the natural algorithm forms the inner product of a row of A and a column of B.
- Total number of arithmetic operations: $n^2(2n-1)$.

Example 1.9

We estimate the rate of growth of the number of arithmetic operations:

(a) Inner product.

- Total number of arithmetic operations: $2n - 1$.
- It increases linearly with n.

Example 1.9

We estimate the rate of growth of the number of arithmetic operations:

(a) Inner product.

- Total number of arithmetic operations: $2n - 1$.
- It increases linearly with n.

(b) Matrix multiplication.

- Total number of arithmetic operations: $n^2(2n - 1)$.
- It increases cubically with n.

Algorithms and operation counts

Definition 1.2

Let f and g be functions that map positive numbers to positive numbers.

(a) We write $f(n) = O(g(n))$ if there exist positive numbers n_0 and c such that

$$f(n) \le cg(n) \quad \text{for all } n \ge n_0.$$

(b) We write $f(n) = \Omega(g(n))$ if there exist positive numbers n_0 and c such that

$$f(n) \ge cg(n) \quad \text{for all } n \ge n_0.$$

Example: $3n^3 + n^2 + 10 = O(n^3)$, $n \log n = O(n^2)$, $n \log n = \Omega(n)$.

Algorithms and operation counts

O and Ω can also be used with two or more variables.

Definition 1.2+

Let f and g be functions that map pairs of positive numbers to positive numbers.

(a) We write $f(m,n) = O(g(m,n))$ if there exist positive numbers n_0 and c such that

$$f(m,n) \le cg(m,n) \quad \text{for all } (m,n) \text{ with } m \ge n_0 \text{ or } n \ge n_0.$$

(b) We write $f(m,n) = \Omega(g(m,n))$ if there exist positive numbers n_0 and c such that

$$f(m,n) \ge cg(m,n) \quad \text{for all } (m,n) \text{ with } m \ge n_0 \text{ or } n \ge n_0.$$

Algorithms and operation counts

- The number of operations performed by an algorithm is called running time.
- Instead of trying to estimate the running time for each possible input, it is customary to estimate the running time for the worst possible input data in a given family.
- For example, if we have an algorithm for linear programming, we might be interested in estimating its worst-case running time over all problems with a given number of variables and constraints.
- In practice, the "average" running time of an algorithm might be more relevant than the "worst case" running time. However, the average running time is much more difficult to estimate.

Example 1.10

System of linear equations. Given a matrix A of dimension $n \times n$, and a vector $b \in \mathbb{R}^n$, either compute a solution or decide that no solution exists for the system of linear equations

$$Ax = b.$$

- The classical method that eliminates one variable at a time (Gaussian elimination) is known to require $O(n^3)$ arithmetic operations. (Exercise!)
- Practical methods for matrix inversion also require $O(n^3)$ arithmetic operations.

Polynomial time algorithms

Is the $O(n^3)$ running time of Gaussian elimination good or bad?

- Each time that technological advances lead to computer hardware that is faster by a factor of 2^3 (roughly every 3 years by Moore's Law), we can solve problems of twice the size than earlier possible:
$$(2n)^3 = 2^3 \cdot n^3.$$
- A similar argument applies to algorithms whose running time is $O(n^c)$ for some positive constant c:
Roughly every c years we can solve problems of twice the size than earlier possible.
- Such algorithms are said to run in polynomial time.

Exponential time algorithms

- Algorithms also exist whose running time is $\Omega(2^n)$, where n is a parameter representing problem size; these are said to take at least exponential time.
- For such algorithms, each time that computer hardware becomes faster by a factor of 2 (roughly every year by Moore's Law), we can increase the value of n that we can handle only by 1:
$$2^{n+1} = 2 \cdot 2^n.$$
- A similar argument applies to algorithms whose running time is $\Omega(2^{cn})$ for some positive constant c:
Roughly every 2^c years we can increase the value of n that we can handle only by 1.
- It is then reasonable to expect that no matter how much technology improves, problems with truly large values of n will always be difficult to handle.

Polynomial vs Exponential time algorithms

Example 1.11

Suppose that we have a choice of two algorithms:

- The running time of the first is $10^n/100$ (exponential).
- The running time of the second is $10n^3$ (polynomial).

Polynomial vs Exponential time algorithms

Example 1.11
Suppose that we have a choice of two algorithms:

- The running time of the first is $10^n/100$ (exponential).
- The running time of the second is $10n^3$ (polynomial).

- For very small n, e.g., for $n = 3$, the exponential time algorithm is preferable:

$$10^3/100 = 10 \quad < \quad 10 \cdot 3^3 = 270.$$

Polynomial vs Exponential time algorithms

Example 1.11

Suppose that we have a choice of two algorithms:

- The running time of the first is $10^n/100$ (exponential).
- The running time of the second is $10n^3$ (polynomial).

- Suppose that we have access to a workstation that can execute 10^7 arithmetic operations per second and that we are willing to let it run for 1000 seconds ($\sim$17 minutes).
- Let us figure out what size problems can each algorithm handle within this time frame:
  - The equation $10^n/100 = 10^7 \times 1000$ yields $n = 12$.
  - The equation $10n^3 = 10^7 \times 1000$ yields $n = 1000$.
- Thus the polynomial time algorithm allows us to solve much larger problems.

Algorithms and operation counts

As a first cut, it is useful to juxtapose polynomial and exponential time algorithms:

- Polynomial time algorithms are viewed as fast and efficient.
- Exponential time algorithms are viewed as slow.

3.3 Implementations of the simplex method

Implementations of the simplex method

Let's now get back to the simplex method.

- It should be clear from the statement of the algorithm that the vectors $B^{-1}A_j$ play a key role.
- If these vectors are available, then we can easily compute:
  - The reduced costs
    $$\bar{c}_j = c_j - c_B' B^{-1} A_j.$$
  - The direction of motion
    $$-u = -B^{-1} A_j.$$
  - The stepsize
    $$\theta^* = \min_{i=1,\dots,m \,|\, u_i > 0} \frac{x_{B(i)}}{u_i}.$$
- The main difference between alternative implementations lies in:
  - The way that the vectors $B^{-1}A_j$ are computed,
  - The amount of related information that is carried from one iteration to the next.

Implementations of the simplex method

When comparing different implementations, it is important to keep the following facts in mind (see Section 1.6).

- Let B be a given $m \times m$ matrix, and let $b, p \in \mathbb{R}^m$ be given vectors.
- Computing the inverse of B or solving a linear system of the form $Bx = b$ takes $O(m^3)$ arithmetic operations.
- Computing a matrix-vector product Bb takes $O(m^2)$ operations.
- Computing an inner product $p'b$ takes $O(m)$ arithmetic operations.

Naive implementation

Naive implementation

We start by describing the most straightforward implementation.

- At the beginning of a typical iteration, we have the indices $B(1), \ldots, B(m)$ of the current basic variables.
- We form the basis matrix B and solve the linear system $p'B = c_B'$ to compute

$$p' = c_B' B^{-1}.$$

This vector $p \in \mathbb{R}^m$ is called the vector of simplex multipliers associated with the basis B.

Naive implementation

We start by describing the most straightforward implementation.

- At the beginning of a typical iteration, we have the indices $B(1), \ldots, B(m)$ of the current basic variables.
- We form the basis matrix B and solve the linear system $p'B = c'_B$ to compute

$$p' = c'_B B^{-1}. \qquad [O(m^3) \text{ operations}]$$

  This vector $p \in \mathbb{R}^m$ is called the vector of simplex multipliers associated with the basis B.

Naive implementation

We start by describing the most straightforward implementation.

- At the beginning of a typical iteration, we have the indices $B(1), \ldots, B(m)$ of the current basic variables.
- We form the basis matrix B and solve the linear system $p'B = c_B'$ to compute

$$p' = c_B' B^{-1}. \qquad [O(m^3) \text{ operations}]$$

  This vector $p \in \mathbb{R}^m$ is called the vector of simplex multipliers associated with the basis B.
- The reduced cost $\bar{c}_j = c_j - c_B' B^{-1} A_j$ of any variable x_j is then obtained according to the formula

$$\bar{c}_j = c_j - p' A_j.$$

- Regardless of the pivoting rule employed, we may have to compute all of the reduced costs.

Naive implementation

We start by describing the most straightforward implementation.

- At the beginning of a typical iteration, we have the indices $B(1), \ldots, B(m)$ of the current basic variables.
- We form the basis matrix B and solve the linear system $p'B = c'_B$ to compute

$$p' = c'_B B^{-1}. \qquad [O(m^3) \text{ operations}]$$

  This vector $p \in \mathbb{R}^m$ is called the vector of simplex multipliers associated with the basis B.
- The reduced cost $\bar{c}_j = c_j - c'_B B^{-1} A_j$ of any variable x_j is then obtained according to the formula

$$\bar{c}_j = c_j - p' A_j. \qquad [n \cdot O(m) = O(mn) \text{ operations}]$$

- Regardless of the pivoting rule employed, we may have to compute all of the reduced costs.

Naive implementation

- Once a column A_j is selected to enter the basis, we solve the linear system $Bu = A_j$ in order to determine the vector

$$u = B^{-1}A_j.$$

Naive implementation

- Once a column A_j is selected to enter the basis, we solve the linear system $Bu = A_j$ in order to determine the vector

$$u = B^{-1}A_j. \qquad [O(m^3) \text{ operations}]$$

Naive implementation

- Once a column A_j is selected to enter the basis, we solve the linear system $Bu = A_j$ in order to determine the vector

$$u = B^{-1}A_j. \qquad [O(m^3) \text{ operations}]$$

- At this point, we can form the direction along which we will be moving away from the current basic feasible solution.
- We finally determine

$$\theta^* = \min_{i=1,\dots,m \,|\, u_i > 0} \frac{x_{B(i)}}{u_i}$$

and the variable that will exit the basis, and construct the new basic feasible solution.

Naive implementation

- Once a column A_j is selected to enter the basis, we solve the linear system $Bu = A_j$ in order to determine the vector

$$u = B^{-1}A_j. \qquad [O(m^3) \text{ operations}]$$

- At this point, we can form the direction along which we will be moving away from the current basic feasible solution.
- We finally determine

$$\theta^* = \min_{i=1,\ldots,m \,|\, u_i > 0} \frac{x_{B(i)}}{u_i} \qquad [O(m) \text{ operations}]$$

and the variable that will exit the basis, and construct the new basic feasible solution.

Naive implementation: running time

- Thus, the total computational effort per iteration is

$$O(m^3 + mn + m^3 + m) = O(m^3 + mn).$$

- We will see shortly that alternative implementations require only

$$O(m^2 + mn)$$

  arithmetic operations.
- Therefore, the naive implementation is rather inefficient.

Revised simplex method

Revised simplex method

- Much of the computational burden in the naive implementation is due to the need for solving the two linear systems of equations

$$p'B = c'_B \qquad \text{and} \qquad Bu = A_j.$$

- In an alternative implementation, the matrix B^{-1} is made available at the beginning of each iteration, and the vectors

$$p' = c'_B B^{-1} \qquad \text{and} \qquad u = B^{-1}A_j$$

  are computed by a matrix-vector multiplication.
- For this approach to be practical, we need an efficient method for updating the matrix B^{-1} each time that we effect a change of basis.

Revised simplex method

- Let
$$B = [A_{B(1)} \cdots A_{B(m)}]$$
be the basis matrix at the beginning of an iteration and let
$$\bar{B} = [A_{B(1)} \cdots A_{B(\ell-1)}\ A_j\ A_{B(\ell+1)} \cdots A_{B(m)}]$$
be the basis matrix at the beginning of the next iteration.
- These two basis matrices have the same columns except that the ℓth column $A_{B(\ell)}$ has been replaced by A_j.
- It is then reasonable to expect that B^{-1} contains information that can be exploited in the computation of $\bar{B}^{-1}$.

Revised simplex method

Definition 3.4
Given a matrix, the operation of adding a constant multiple of one row to the same or to another row is called an elementary row operation.

- Performing an elementary row operation on a matrix C is equivalent to forming the matrix QC, where Q is a suitably constructed square matrix.

Example 3.3

Let

$$Q = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix},$$

and note that

$$QC = \begin{bmatrix} 1 + 2 \cdot 5 & 2 + 2 \cdot 6 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} 11 & 14 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}.$$

- Multiplication from the left by the matrix Q has the effect of multiplying the third row of C by two and adding it to the first row.

Revised simplex method

Let's generalize Example 3.3.

- Multiplying the jth row by β and adding it to the ith row (for $i \neq j$) is the same as left-multiplying by the matrix

$$Q = I + D_{ij},$$

  where D_{ij} is a matrix with all entries equal to zero, except for the (i, j)th entry which is equal to β.
- The determinant of such a matrix Q is equal to 1 and, therefore, Q is invertible. Really?

Revised simplex method

- Suppose now that we apply a sequence of K elementary row operations and that the kth such operation corresponds to left-multiplication by a certain invertible matrix Q_k.
- Then, the sequence of these elementary row operations is the same as left-multiplication by the invertible matrix

$$Q_K Q_{K-1} \cdots Q_2 Q_1.$$

- We conclude that performing a sequence of elementary row operations on a given matrix is equivalent to left-multiplying that matrix by a certain invertible matrix.
- We will now see how we can use elementary row operations to compute of $\bar{B}^{-1}$ exploiting B^{-1}.

Revised simplex method

- Since $B^{-1}B = I$, we see that $B^{-1}A_{B(i)} = e_i$.
- Using this observation, we have

$$B^{-1}\bar{B} = [e_1 \cdots e_{\ell-1} \; u \; e_{\ell+1} \cdots e_m]$$

$$= \begin{bmatrix} 1 & & u_1 & & \\ & \ddots & \vdots & & \\ & & u_\ell & & \\ & & \vdots & \ddots & \\ & & u_m & & 1 \end{bmatrix},$$

where $u = B^{-1}A_j$. (!!)

- We can change the above matrix to the identity matrix by applying the following sequence of elementary row operations:

(a) For each $i \neq \ell$, we add the ℓth row times $-u_i/u_\ell$ to the ith row. (Recall that $u_\ell > 0$. Why?) This replaces u_i by zero.

(b) We divide the ℓth row by u_ℓ. Why is this an elementary row operation? This replaces u_ℓ by one.

Revised simplex method

- This sequence of elementary row operations replaces the ℓth column u by the ℓth unit vector e_ℓ.
- Furthermore, it is equivalent to left-multiplying $B^{-1}\bar{B}$ by a certain invertible matrix Q.
- Since the result is the identity, we have

$$QB^{-1}\bar{B} = I \qquad \Rightarrow \qquad QB^{-1} = \bar{B}^{-1}.$$

- The last equation shows that if we apply the same sequence of row operations to the matrix B^{-1}, we obtain $\bar{B}^{-1}$.
- We conclude that all it takes to generate $\bar{B}^{-1}$, is to start with B^{-1} and apply the sequence of elementary row operations described above.
- Total number of arithmetic operations:

Revised simplex method

- This sequence of elementary row operations replaces the ℓth column u by the ℓth unit vector e_ℓ.
- Furthermore, it is equivalent to left-multiplying $B^{-1}\bar{B}$ by a certain invertible matrix Q.
- Since the result is the identity, we have

$$QB^{-1}\bar{B} = I \qquad \Rightarrow \qquad QB^{-1} = \bar{B}^{-1}.$$

- The last equation shows that if we apply the same sequence of row operations to the matrix B^{-1}, we obtain $\bar{B}^{-1}$.
- We conclude that all it takes to generate $\bar{B}^{-1}$, is to start with B^{-1} and apply the sequence of elementary row operations described above.
- Total number of arithmetic operations: $O(m^2)$.

Example 3.4

$$B^{-1} = \begin{bmatrix} 1 & 2 & 3 \\ -2 & 3 & 1 \\ 4 & -3 & -2 \end{bmatrix}, \qquad u = \begin{bmatrix} -4 \\ 2 \\ 2 \end{bmatrix}, \qquad \ell = 3.$$

Example 3.4

$$B^{-1} = \begin{bmatrix} 1 & 2 & 3 \\ -2 & 3 & 1 \\ 4 & -3 & -2 \end{bmatrix}, \qquad u = \begin{bmatrix} -4 \\ 2 \\ 2 \end{bmatrix}, \qquad \ell = 3.$$

- We have

$$B^{-1}\bar{B} = [e_1 \; e_2 \; u] = \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & 2 \\ 0 & 0 & 2 \end{bmatrix}.$$

- Thus, our objective is to transform the vector u to the unit vector $e_3 = (0, 0, 1)$.

Example 3.4

$$B^{-1} = \begin{bmatrix} 1 & 2 & 3 \\ -2 & 3 & 1 \\ 4 & -3 & -2 \end{bmatrix}, \qquad u = \begin{bmatrix} -4 \\ 2 \\ 2 \end{bmatrix}, \qquad \ell = 3.$$

- We multiply the third row by 2 and add it to the first row.
- We multiply the third row by -1 and add it to the second.
- We divide the third row by 2.

Applying the same row operations to B^{-1} we obtain

$$\bar{B}^{-1} = \begin{bmatrix} 9 & -4 & -1 \\ -6 & 6 & 3 \\ 2 & -1.5 & -1 \end{bmatrix}.$$

- When the matrix B^{-1} is updated in the manner we have described, we obtain an implementation of the simplex method known as the revised simplex method.

Revised simplex method

An iteration of the revised simplex method

1. We start with a basis consisting of the basic columns $A_{B(1)}, \ldots, A_{B(m)}$, an associated basic feasible solution x, and the inverse B^{-1} of the basis matrix.
2. Compute the row vector $p' = c'_B B^{-1}$ and then compute the reduced costs $\bar{c}_j = c_j - p' A_j$.
   - If they are all nonnegative, the current basic feasible solution is optimal, and the algorithm terminates.
   - Else, choose some j for which $\bar{c}_j < 0$.
3. Compute $u = B^{-1} A_j$. If no component of u is positive, the optimal cost is $-\infty$, and the algorithm terminates.

Revised simplex method

An iteration of the revised simplex method

4. If some component of u is positive, let

$$\theta^* = \min_{i=1,\ldots,m \,|\, u_i > 0} \frac{x_{B(i)}}{u_i}.$$

5. Let ℓ be such that $\theta^* = x_{B(\ell)}/u_\ell$. Form a new basis by replacing $A_{B(\ell)}$ with A_j. If y is the new basic feasible solution, the values of the new basic variables are $y_j = \theta^*$ and $y_{B(i)} = x_{B(i)} - \theta^* u_i$, $i \neq \ell$.
6. Form the $m \times (m+1)$ matrix $[B^{-1} \mid u]$. Add to each one of its rows a multiple of the ℓth row to make the last column equal to the unit vector e_ℓ. The first m columns of the result is the matrix $\bar{B}^{-1}$.

Revised simplex method: implementation

- At the beginning of a typical iteration, we have the indices $B(1), \ldots, B(m)$ of the current basic variables, and the inverse B^{-1} of the basis matrix.
- We compute

$$p' = c_B' B^{-1}.$$

Revised simplex method: implementation

- At the beginning of a typical iteration, we have the indices $B(1), \ldots, B(m)$ of the current basic variables, and the inverse B^{-1} of the basis matrix.
- We compute

$$p' = c_B' B^{-1}. \qquad [O(m^2) \text{ operations}]$$

Revised simplex method: implementation

- At the beginning of a typical iteration, we have the indices $B(1), \ldots, B(m)$ of the current basic variables, and the inverse B^{-1} of the basis matrix.
- We compute

$$p' = c_B' B^{-1}. \qquad [O(m^2) \text{ operations}]$$

- The reduced cost $\bar{c}_j = c_j - c_B' B^{-1} A_j$ of any variable x_j is then obtained according to the formula

$$\bar{c}_j = c_j - p' A_j.$$

- Regardless of the pivoting rule employed, we may have to compute all of the reduced costs.

Revised simplex method: implementation

- At the beginning of a typical iteration, we have the indices $B(1), \ldots, B(m)$ of the current basic variables, and the inverse B^{-1} of the basis matrix.
- We compute

$$p' = c_B' B^{-1}. \qquad [O(m^2) \text{ operations}]$$

- The reduced cost $\bar{c}_j = c_j - c_B' B^{-1} A_j$ of any variable x_j is then obtained according to the formula

$$\bar{c}_j = c_j - p' A_j. \qquad [O(mn) \text{ operations}]$$

- Regardless of the pivoting rule employed, we may have to compute all of the reduced costs.

Revised simplex method: implementation

- Once a column A_j is selected to enter the basis, we compute the vector

$$u = B^{-1}A_j.$$

Revised simplex method: implementation

- Once a column A_j is selected to enter the basis, we compute the vector

$$u = B^{-1}A_j. \qquad [O(m^2) \text{ operations}]$$

Revised simplex method: implementation

- Once a column A_j is selected to enter the basis, we compute the vector

$$u = B^{-1} A_j. \qquad [O(m^2) \text{ operations}]$$

- We determine

$$\theta^* = \min_{i=1,\dots,m \,|\, u_i > 0} \frac{x_{B(i)}}{u_i}$$

  and the variable that will exit the basis, and construct the new basic feasible solution, and the new basis matrix $\bar{B}$.

Revised simplex method: implementation

- Once a column A_j is selected to enter the basis, we compute the vector

$$u = B^{-1}A_j. \qquad [O(m^2) \text{ operations}]$$

- We determine

$$\theta^* = \min_{i=1,\dots,m \,|\, u_i>0} \frac{x_{B(i)}}{u_i} \qquad [O(m) \text{ operations}]$$

  and the variable that will exit the basis, and construct the new basic feasible solution, and the new basis matrix $\bar{B}$.

Revised simplex method: implementation

- Once a column A_j is selected to enter the basis, we compute the vector

$$u = B^{-1}A_j. \qquad [O(m^2) \text{ operations}]$$

- We determine

$$\theta^* = \min_{i=1,\dots,m \,|\, u_i > 0} \frac{x_{B(i)}}{u_i} \qquad [O(m) \text{ operations}]$$

  and the variable that will exit the basis, and construct the new basic feasible solution, and the new basis matrix $\bar{B}$.

- We construct the inverse $\bar{B}^{-1}$ of $\bar{B}$.

Revised simplex method: implementation

- Once a column A_j is selected to enter the basis, we compute the vector

$$u = B^{-1} A_j. \qquad [O(m^2) \text{ operations}]$$

- We determine

$$\theta^* = \min_{i=1,\ldots,m \,|\, u_i > 0} \frac{x_{B(i)}}{u_i} \qquad [O(m) \text{ operations}]$$

  and the variable that will exit the basis, and construct the new basic feasible solution, and the new basis matrix $\bar{B}$.
- We construct the inverse $\bar{B}^{-1}$ of $\bar{B}$. $[O(m^2)$ operations$]$

Revised simplex method: running time

- Thus, the total number of operations per iteration is

$$O(m^2 + mn + m^2 + m + m^2) = O(m^2 + mn) = O(mn).$$

- Therefore, the revised simplex method is more efficient than the naive implementation, which required

$$O(m^3 + mn)$$

arithmetic operations.

The full tableau implementation

The full tableau implementation

We now describe the implementation of the simplex method in terms of the so-called full tableau.

- Here, instead of maintaining and updating the matrix B^{-1}, we maintain and update the $m \times (n+1)$ matrix

$$B^{-1}[b \mid A]$$

  with columns $B^{-1}b, B^{-1}A_1, \ldots, B^{-1}A_n$.
- This matrix is called the simplex tableau.

The full tableau implementation

- The column $B^{-1}b$ is called the zeroth column and contains the values of the basic variables.
- The column $B^{-1}A_i$ is called the *i*th column of the tableau.
- The column $u = B^{-1}A_j$ corresponding to the variable that enters the basis is called the pivot column.

$$
\begin{array}{|c|ccccc|}
\hline
x_{B(1)} & (B^{-1}A_1)_1 & \dots & u_1 & \dots & (B^{-1}A_n)_1 \\
\vdots & \vdots & & \vdots & & \vdots \\
x_{B(\ell)} & (B^{-1}A_1)_\ell & \dots & u_\ell & \dots & (B^{-1}A_n)_\ell \\
\vdots & \vdots & & \vdots & & \vdots \\
x_{B(m)} & (B^{-1}A_1)_m & \dots & u_m & \dots & (B^{-1}A_n)_m \\
\hline
\end{array}
$$

The full tableau implementation

- If the ℓth basic variable exits the basis, the ℓth row of the tableau is called the pivot row.
- The element belonging to both the pivot row and the pivot column is called the pivot element.
- Note that the pivot element is u_ℓ and is always positive (unless $u \leq 0$, in which case the algorithm has met the termination condition in Step 3).

$x_{B(1)}$	$(B^{-1}A_1)_1$	$\ldots$	u_1	$\ldots$	$(B^{-1}A_n)_1$
$\vdots$	$\vdots$		$\vdots$		$\vdots$
$x_{B(\ell)}$	$(B^{-1}A_1)_\ell$	$\ldots$	u_ℓ	$\ldots$	$(B^{-1}A_n)_\ell$
$\vdots$	$\vdots$		$\vdots$		$\vdots$
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\ldots$	u_m	$\ldots$	$(B^{-1}A_n)_m$

The full tableau implementation

The information contained in the rows of the tableau

$$B^{-1}[b \mid A]$$

admits the following interpretation.

- The equality constraints are initially given to us in the form

$$b = Ax.$$

- Given the current basis matrix B, these equality constraints can also be expressed in the equivalent form

$$B^{-1}b = B^{-1}Ax.$$

- The tableau provides us with the coefficients of these equality constraints.

The full tableau implementation

- At the end of each iteration, we need to update the tableau $B^{-1}[b \mid A]$ and compute
$$\bar{B}^{-1}[b \mid A].$$
- This can be accomplished by left-multiplying the simplex tableau with a matrix Q satisfying $QB^{-1} = \bar{B}^{-1}$.
- As explained earlier, this is the same as performing those elementary row operations that turn B^{-1} to $\bar{B}^{-1}$.
- That is, we add to each row a multiple of the pivot row to set all entries of the pivot column to zero, with the exception of the pivot element which is set to one.

The full tableau implementation

Regarding the determination of the exiting column $A_{B(\ell)}$ and the stepsize θ^*, Steps 4 and 5 of the simplex method amount to:

- $\frac{x_{B(i)}}{u_i}$ is the ratio of the ith entry in the zeroth column of the tableau to the ith entry in the pivot column of the tableau.
- We only consider those i for which u_i is positive.
- The smallest ratio is equal to θ^* and determines ℓ.

$$\begin{array}{|c|ccccc|}
\hline
x_{B(1)} & (B^{-1}A_1)_1 & \dots & u_1 & \dots & (B^{-1}A_n)_1 \\
\vdots & \vdots & & \vdots & & \vdots \\
x_{B(\ell)} & (B^{-1}A_1)_\ell & \dots & u_\ell & \dots & (B^{-1}A_n)_\ell \\
\vdots & \vdots & & \vdots & & \vdots \\
x_{B(m)} & (B^{-1}A_1)_m & \dots & u_m & \dots & (B^{-1}A_n)_m \\
\hline
\end{array}$$

The zeroth row

It is customary to augment the simplex tableau by including a top row, to be referred to as the zeroth row.

$x_{B(1)}$	$(B^{-1}A_1)_1$	$\ldots$	u_1	$\ldots$	$(B^{-1}A_n)_1$
$\vdots$	$\vdots$		$\vdots$		$\vdots$
$x_{B(\ell)}$	$(B^{-1}A_1)_\ell$	$\ldots$	u_ℓ	$\ldots$	$(B^{-1}A_n)_\ell$
$\vdots$	$\vdots$		$\vdots$		$\vdots$
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\ldots$	u_m	$\ldots$	$(B^{-1}A_n)_m$

The zeroth row

It is customary to augment the simplex tableau by including a top row, to be referred to as the zeroth row.

- The entry at the top left corner contains the negative of the current cost:
$$-c'_B x_B = -c'_B B^{-1} b.$$
- The reason for the minus sign is that it allows for a simple update rule.

$-\ c'_B x_B$					
$x_{B(1)}$	$(B^{-1}A_1)_1$	$\dots$	u_1	$\dots$	$(B^{-1}A_n)_1$
$\vdots$	$\vdots$		$\vdots$		$\vdots$
$x_{B(\ell)}$	$(B^{-1}A_1)_\ell$	$\dots$	u_ℓ	$\dots$	$(B^{-1}A_n)_\ell$
$\vdots$	$\vdots$		$\vdots$		$\vdots$
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\dots$	u_m	$\dots$	$(B^{-1}A_n)_m$

The zeroth row

It is customary to augment the simplex tableau by including a top row, to be referred to as the zeroth row.

- The rest of the zeroth row is the row vector of reduced costs, that is, the vector

$$\bar{c}' = c' - c_B' B^{-1} A.$$

$-c_B' x_B$	$\bar{c}_1$	$\ldots$	$\bar{c}_j$	$\ldots$	$\bar{c}_n$
$x_{B(1)}$	$(B^{-1}A_1)_1$	$\ldots$	u_1	$\ldots$	$(B^{-1}A_n)_1$
$\vdots$	$\vdots$		$\vdots$		$\vdots$
$x_{B(\ell)}$	$(B^{-1}A_1)_\ell$	$\ldots$	u_ℓ	$\ldots$	$(B^{-1}A_n)_\ell$
$\vdots$	$\vdots$		$\vdots$		$\vdots$
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\ldots$	u_m	$\ldots$	$(B^{-1}A_n)_m$

The zeroth row

- The rule for updating the zeroth row turns out to be identical to the rule used for the other rows of the tableau:

  Add a multiple of the pivot row to the zeroth row to set the reduced cost of the entering variable to zero.

- We will now verify that this update rule produces the correct results for the zeroth row.

The zeroth row

- At the beginning of a typical iteration, the zeroth row is of the form

$$[-c_B' B^{-1} b \mid c' - c_B' B^{-1} A] = [0 \mid c'] - \underbrace{c_B' B^{-1}}_{\text{a row vector}} [b \mid A].$$

- Hence, it is equal to $[0 \mid c']$ plus a linear combination of the rows of $[b \mid A]$.
- Let column j be the pivot column, and row ℓ be the pivot row.
- Note that the pivot row is of the form

$$h'[b \mid A],$$

  where the vector h' is the ℓth row of B^{-1}.
- Hence, after a multiple of the pivot row is added to the zeroth row, that row is again equal to $[0 \mid c']$ plus a (different) linear combination of the rows of $[b \mid A]$, and is of the form

$$[0 \mid c'] - p'[b \mid A],$$

  for some vector p.

The zeroth row

Beginning of iteration: $[0 \mid c'] - c_B' B^{-1}[b \mid A]$
End of iteration: $[0 \mid c'] - p'[b \mid A]$

- We now calculate the vector p using our update rule.
- We should obtain
$$p' = c_{\bar{B}}' \bar{B}^{-1}.$$

The zeroth row

a) Consider the column $\bar{B}(\ell)$ of the tableau.

$-c_B' x_B$	$\bar{c}_1$	$\dots$	$\bar{c}_j$	$\dots$	$\bar{c}_n$
$x_{B(1)}$	$(B^{-1}A_1)_1$	$\dots$	u_1	$\dots$	$(B^{-1}A_n)_1$
$\vdots$	$\vdots$		$\vdots$		$\vdots$
$x_{B(\ell)}$	$(B^{-1}A_1)_\ell$	$\dots$	u_ℓ	$\dots$	$(B^{-1}A_n)_\ell$
$\vdots$	$\vdots$		$\vdots$		$\vdots$
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\dots$	u_m	$\dots$	$(B^{-1}A_n)_m$

- Recall that $\bar{B}(\ell) = j$, thus this is the pivot column.
- Our update rule is such that the pivot column entry of the zeroth row becomes zero.
- We obtain

$$c_{\bar{B}(\ell)} - p' A_{\bar{B}(\ell)} = 0.$$

The zeroth row

b) Consider the column $\bar{B}(i)$ of the tableau, for $i \neq \ell$.

$-c_B' x_B$	$\bar{c}_1$	$\ldots$	$\bar{c}_j$	$\ldots$	$\bar{c}_{\bar{B}(i)}$	$\ldots$	$\bar{c}_n$
$x_{B(1)}$	$(B^{-1}A_1)_1$	$\ldots$	u_1	$\ldots$	$(B^{-1}A_{\bar{B}(i)})_1$	$\ldots$	$(B^{-1}A_n)_1$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
$x_{B(\ell)}$	$(B^{-1}A_1)_\ell$	$\ldots$	u_ℓ	$\ldots$	$(B^{-1}A_{\bar{B}(i)})_\ell$	$\ldots$	$(B^{-1}A_n)_\ell$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\ldots$	u_m	$\ldots$	$(B^{-1}A_{\bar{B}(i)})_m$	$\ldots$	$(B^{-1}A_n)_m$

- This is a column corresponding to a basic variable that stays in the basis. Thus $\bar{B}(i) = B(i)$.
- The zeroth row entry of that column is zero, before the change of basis, since it is the reduced cost of a basic variable.

The zeroth row

b) Consider the column $\bar{B}(i)$ of the tableau, for $i \neq \ell$.

$-c_B' x_B$	$\bar{c}_1$	$\dots$	$\bar{c}_j$	$\dots$	$\bar{c}_{B(i)}$	$\dots$	$\bar{c}_n$
$x_{B(1)}$	$(B^{-1}A_1)_1$	$\dots$	u_1	$\dots$	$(B^{-1}A_{B(i)})_1$	$\dots$	$(B^{-1}A_n)_1$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
$x_{B(\ell)}$	$(B^{-1}A_1)_\ell$	$\dots$	u_ℓ	$\dots$	$(B^{-1}A_{B(i)})_\ell$	$\dots$	$(B^{-1}A_n)_\ell$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\dots$	u_m	$\dots$	$(B^{-1}A_{B(i)})_m$	$\dots$	$(B^{-1}A_n)_m$

- This is a column corresponding to a basic variable that stays in the basis. Thus $\bar{B}(i) = B(i)$.
- The zeroth row entry of that column is zero, before the change of basis, since it is the reduced cost of a basic variable.

The zeroth row

b) Consider the column $\bar{B}(i)$ of the tableau, for $i \neq \ell$.

$-c_B' x_B$	$\bar{c}_1$	$\dots$	$\bar{c}_j$	$\dots$	0	$\dots$	$\bar{c}_n$
$x_{B(1)}$	$(B^{-1}A_1)_1$	$\dots$	u_1	$\dots$	$(B^{-1}A_{B(i)})_1$	$\dots$	$(B^{-1}A_n)_1$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
$x_{B(\ell)}$	$(B^{-1}A_1)_\ell$	$\dots$	u_ℓ	$\dots$	$(B^{-1}A_{B(i)})_\ell$	$\dots$	$(B^{-1}A_n)_\ell$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\dots$	u_m	$\dots$	$(B^{-1}A_{B(i)})_m$	$\dots$	$(B^{-1}A_n)_m$

- This is a column corresponding to a basic variable that stays in the basis. Thus $\bar{B}(i) = B(i)$.
- The zeroth row entry of that column is zero, before the change of basis, since it is the reduced cost of a basic variable.

The zeroth row

b) Consider the column $\bar{B}(i)$ of the tableau, for $i \neq \ell$.

$-c'_B x_B$	$\bar{c}_1$	$\dots$	$\bar{c}_j$	$\dots$	0	$\dots$	$\bar{c}_n$
$x_{B(1)}$	$(B^{-1}A_1)_1$	$\dots$	u_1	$\dots$	$(B^{-1}A_{B(i)})_1$	$\dots$	$(B^{-1}A_n)_1$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
$x_{B(\ell)}$	$(B^{-1}A_1)_\ell$	$\dots$	u_ℓ	$\dots$	$(B^{-1}A_{B(i)})_\ell$	$\dots$	$(B^{-1}A_n)_\ell$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\dots$	u_m	$\dots$	$(B^{-1}A_{B(i)})_m$	$\dots$	$(B^{-1}A_n)_m$

- Before the operation, the $B(i)$th column is $B^{-1}A_{B(i)}$, thus it is the ith unit vector.
- Since $i \neq \ell$, the entry in the pivot row for that column is equal to zero.

The zeroth row

b) Consider the column $\bar{B}(i)$ of the tableau, for $i \neq \ell$.

$-c_B' x_B$	$\bar{c}_1$	$\dots$	$\bar{c}_j$	$\dots$	0	$\dots$	$\bar{c}_n$
$x_{B(1)}$	$(B^{-1}A_1)_1$	$\dots$	u_1	$\dots$	0	$\dots$	$(B^{-1}A_n)_1$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
$x_{B(\ell)}$	$(B^{-1}A_1)_\ell$	$\dots$	u_ℓ	$\dots$	0	$\dots$	$(B^{-1}A_n)_\ell$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
$x_{B(i)}$	$(B^{-1}A_1)_i$	$\dots$	u_i	$\dots$	1	$\dots$	$(B^{-1}A_n)_i$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\dots$	u_m	$\dots$	0	$\dots$	$(B^{-1}A_n)_m$

- Before the operation, the $B(i)$th column is $B^{-1}A_{B(i)}$, thus it is the ith unit vector.
- Since $i \neq \ell$, the entry in the pivot row for that column is equal to zero.

The zeroth row

b) Consider the column $\bar{B}(i)$ of the tableau, for $i \neq \ell$.

$-c_B' x_B$	$\bar{c}_1$	$\ldots$	$\bar{c}_j$	$\ldots$	0	$\ldots$	$\bar{c}_n$
$x_{B(1)}$	$(B^{-1}A_1)_1$	$\ldots$	u_1	$\ldots$	0	$\ldots$	$(B^{-1}A_n)_1$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
$x_{B(\ell)}$	$(B^{-1}A_1)_\ell$	$\ldots$	u_ℓ	$\ldots$	0	$\ldots$	$(B^{-1}A_n)_\ell$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
$x_{B(i)}$	$(B^{-1}A_1)_i$	$\ldots$	u_i	$\ldots$	1	$\ldots$	$(B^{-1}A_n)_i$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\ldots$	u_m	$\ldots$	0	$\ldots$	$(B^{-1}A_n)_m$

- Hence, adding a multiple of the pivot row to the zeroth row of the tableau does not affect the zeroth row entry of that column, which is left at zero.
- Thus for $i \neq \ell$ we have

$$c_{\bar{B}(i)} - p' A_{\bar{B}(i)} = 0.$$

The zeroth row

- a) and b) imply that the vector p satisfies

$$c_{\bar{B}(i)} - p' A_{\bar{B}(i)} = 0 \qquad i = 1, \ldots, m.$$

- In matrix form we have

$$c'_{\bar{B}} - p' \bar{B} = 0 \qquad \iff \qquad p' = c'_{\bar{B}} \bar{B}^{-1}.$$

- Hence, with our update rule, the updated zeroth row of the tableau is equal to

$$[0 \mid c'] - p'[b \mid A] = [0 \mid c'] - c'_{\bar{B}} \bar{B}^{-1} [b \mid A],$$

as desired.

The full tableau implementation

We can now summarize the mechanics of the full tableau implementation.

An iteration of the full tableau implementation

1. A typical iteration starts with the tableau associated with a basis matrix B and the corresponding basic feasible solution x.
2. Examine the reduced costs in the zeroth row of the tableau.
   - If they are all nonnegative, the current basic feasible solution is optimal, and the algorithm terminates.
   - Else, choose some j for which $\bar{c}_j < 0$.
3. Consider the vector $u = B^{-1}A_j$, which is the jth column (the pivot column) of the tableau. If no component of u is positive, the optimal cost is $-\infty$, and the algorithm terminates.

The full tableau implementation

An iteration of the full tableau implementation

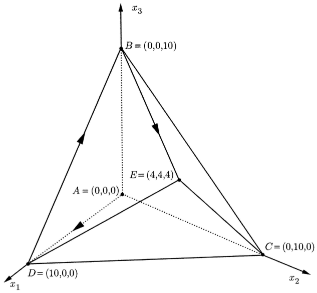
4. For each i for which u_i is positive, compute the ratio

$$\frac{x_{B(i)}}{u_i}.$$

   Let ℓ be the index of a row that corresponds to the smallest ratio. The column $A_{B(\ell)}$ exits the basis and the column A_j enters the basis.

5. Add to each row of the tableau a constant multiple of the ℓth row (the pivot row) so that u_ℓ (the pivot element) becomes one and all other entries of the pivot column become zero.

Example 3.5



$$\begin{array}{rl} \text{minimize} & -10x_1 - 12x_2 - 12x_3 \\ \text{subject to} & x_1 + 2x_2 + 2x_3 \le 20 \\ & 2x_1 + x_2 + 2x_3 \le 20 \\ & 2x_1 + 2x_2 + x_3 \le 20 \\ & x_1, x_2, x_3 \ge 0. \end{array}$$

Example 3.5

After introducing slack variables, we obtain the standard form problem

$$\begin{aligned} \text{minimize} \quad & -10x_1 - 12x_2 - 12x_3 \\ \text{subject to} \quad & x_1 + 2x_2 + 2x_3 + x_4 = 20 \\ & 2x_1 + x_2 + 2x_3 + x_5 = 20 \\ & 2x_1 + 2x_2 + x_3 + x_6 = 20 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0. \end{aligned}$$

$$\begin{aligned} \text{minimize} \quad & -10x_1 - 12x_2 - 12x_3 \\ \text{subject to} \quad & x_1 + 2x_2 + 2x_3 \leq 20 \\ & 2x_1 + x_2 + 2x_3 \leq 20 \\ & 2x_1 + 2x_2 + x_3 \leq 20 \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

Example 3.5

After introducing slack variables, we obtain the standard form problem

$$\begin{aligned} \text{minimize} \quad & -10x_1 - 12x_2 - 12x_3 \\ \text{subject to} \quad & x_1 + 2x_2 + 2x_3 + x_4 = 20 \\ & 2x_1 + x_2 + 2x_3 + x_5 = 20 \\ & 2x_1 + 2x_2 + x_3 + x_6 = 20 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0. \end{aligned}$$

- Note that $x = (0, 0, 0, 20, 20, 20)$ is a basic feasible solution and can be used to start the algorithm.
- Let accordingly, $B(1) = 4$, $B(2) = 5$, and $B(3) = 6$.
- The corresponding basis matrix is the identity matrix I.
- To obtain the zeroth row of the initial tableau, we note that $c_B = 0$ and, therefore, $c_B' x_B = 0$ and $\bar{c}' = c' - c_B' B^{-1} A = c'$.

Example 3.5: Initial tableau

$$\begin{array}{r|r|rrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
\hline
 & 0 & -10 & -12 & -12 & 0 & 0 & 0 \\
\hline
x_4 = & 20 & 1 & 2 & 2 & 1 & 0 & 0 \\
x_5 = & 20 & 2 & 1 & 2 & 0 & 1 & 0 \\
x_6 = & 20 & 2 & 2 & 1 & 0 & 0 & 1 \\
\hline
\end{array}$$

Example 3.5: Initial tableau

$$\begin{array}{r|r|rrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ \cline{2-8}
 & 0 & -10 & -12 & -12 & 0 & 0 & 0 \\ \cline{2-8}
x_4 = & 20 & 1 & 2 & 2 & 1 & 0 & 0 \\
x_5 = & 20 & 2 & 1 & 2 & 0 & 1 & 0 \\
x_6 = & 20 & 2 & 2 & 1 & 0 & 0 & 1 \\ \cline{2-8}
\end{array}$$

We note a few conventions in the format of the above tableau:

- The label x_i on top of the ith column indicates the variable associated with that column.
- The labels "$x_i =$" to the left of the tableau tell us which are the basic variables and in what order:
  - $x_{B(1)} = x_4 = 20$,
  - $x_{B(2)} = x_5 = 20$,
  - $x_{B(3)} = x_6 = 20$.

Example 3.5: Initial tableau

$$
\begin{array}{r|r|rrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
\hline
 & 0 & -10 & -12 & -12 & 0 & 0 & 0 \\
\hline
x_4 = & 20 & 1 & 2 & 2 & 1 & 0 & 0 \\
x_5 = & 20 & 2 & 1 & 2 & 0 & 1 & 0 \\
x_6 = & 20 & 2 & 2 & 1 & 0 & 0 & 1 \\
\hline
\end{array}
$$

We note a few conventions in the format of the above tableau:

- These labels are not necessary.
- We know that the column in the tableau associated with the first basic variable must be the first unit vector.
- Once we observe that the column associated with the variable x_4 is the first unit vector, it follows that x_4 is the first basic variable.

Example 3.5: Initial tableau

$$\begin{array}{r|r|rrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
\hline
 & 0 & -10 & -12 & -12 & 0 & 0 & 0 \\
\hline
x_4 = & 20 & 1 & 2 & 2 & 1 & 0 & 0 \\
x_5 = & 20 & 2 & 1 & 2 & 0 & 1 & 0 \\
x_6 = & 20 & 2 & 2 & 1 & 0 & 0 & 1 \\
\hline
\end{array}$$

We continue with our example.

- The reduced cost of x_1 is negative and we let that variable enter the basis.
- The pivot column is $u = (1, 2, 2)$.
- We form the ratios $x_{B(i)}/u_i$, $i = 1, 2, 3$:
  - $x_{B(1)}/u_1 = 20/1 = 20$,
  - $x_{B(2)}/u_2 = 20/2 = 10$,
  - $x_{B(3)}/u_3 = 20/2 = 10$.
- The smallest ratio corresponds to $i = 2$ and $i = 3$.
- We break this tie by choosing $\ell = 2$.

Example 3.5: Initial tableau

$$\begin{array}{r|r|rrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
\hline
 & 0 & -10 & -12 & -12 & 0 & 0 & 0 \\
\hline
x_4 = & 20 & 1 & 2 & 2 & 1 & 0 & 0 \\
x_5 = & 20 & 2 & 1 & 2 & 0 & 1 & 0 \\
x_6 = & 20 & 2 & 2 & 1 & 0 & 0 & 1 \\
\hline
\end{array}$$

- The second basic variable $x_{B(2)}$, which is x_5, exits the basis. This determines the pivot row and the pivot element.
- The new basis is given by $\bar{B}(1) = 4$, $\bar{B}(2) = 1$, and $\bar{B}(3) = 6$.

Example 3.5: Initial tableau

		x_1	x_2	x_3	x_4	x_5	x_6
	0	-10	-12	-12	0	0	0
$x_4 =$	20	1	2	2	1	0	0
$x_5 =$	20	2	1	2	0	1	0
$x_6 =$	20	2	2	1	0	0	1

- We multiply the pivot row by 5 and add it to the zeroth row.
- We multiply the pivot row by $1/2$ and subtract it from the first row.
- We subtract the pivot row from the third row.
- Finally, we divide the pivot row by 2.
- This leads us to the new tableau:

Example 3.5: First pivot

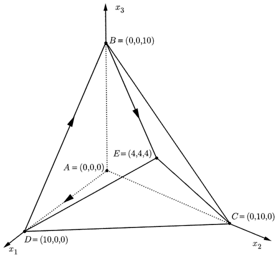
$$\begin{array}{r|r|rrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
\hline
 & 100 & 0 & -7 & -2 & 0 & 5 & 0 \\
\hline
x_4 = & 10 & 0 & 1.5 & 1 & 1 & -0.5 & 0 \\
x_1 = & 10 & 1 & 0.5 & 1 & 0 & 0.5 & 0 \\
x_6 = & 0 & 0 & 1 & -1 & 0 & -1 & 1 \\
\hline
\end{array}$$

- The cost has been reduced to -100.
- The corresponding basic feasible solution is $x = (10, 0, 0, 10, 0, 0)$.
- Note that this is a degenerate basic feasible solution, because the basic variable x_6 is equal to zero.

Example 3.5: First pivot

$$
\begin{array}{r|r|rrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
\hline
 & 100 & 0 & -7 & -2 & 0 & 5 & 0 \\
\hline
x_4 = & 10 & 0 & 1.5 & 1 & 1 & -0.5 & 0 \\
x_1 = & 10 & 1 & 0.5 & 1 & 0 & 0.5 & 0 \\
x_6 = & 0 & 0 & 1 & -1 & 0 & -1 & 1 \\
\hline
\end{array}
$$

- In terms of the original variables x_1, x_2, x_3, we have moved to the degenerate solution $D = (10, 0, 0)$.



Example 3.5: First pivot

$$\begin{array}{r|r|rrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
\hline
 & 100 & 0 & -7 & -2 & 0 & 5 & 0 \\
\hline
x_4 = & 10 & 0 & 1.5 & 1 & 1 & -0.5 & 0 \\
x_1 = & 10 & 1 & 0.5 & 1 & 0 & 0.5 & 0 \\
x_6 = & 0 & 0 & 1 & -1 & 0 & -1 & 1 \\
\hline
\end{array}$$

- We have mentioned earlier that the rows of the tableau (other than the zeroth row) amount to a representation of the equality constraints $B^{-1}Ax = B^{-1}b$, which are equivalent to the original constraints $Ax = b$.
- In our current example, the tableau indicates that the equality constraints can be written in the equivalent form:

$$\begin{array}{rcccccc}
10 = & & 1.5x_2 & +x_3 & +x_4 & -0.5x_5 & \\
10 = & x_1 & +0.5x_2 & +x_3 & & +0.5x_5 & \\
0 = & & x_2 & -x_3 & & -x_5 & +x_6.
\end{array}$$

Example 3.5: First pivot

$$\begin{array}{r|r|rrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ \hline
 & 100 & 0 & -7 & -2 & 0 & 5 & 0 \\ \hline
x_4 = & 10 & 0 & 1.5 & 1 & 1 & -0.5 & 0 \\
x_1 = & 10 & 1 & 0.5 & 1 & 0 & 0.5 & 0 \\
x_6 = & 0 & 0 & 1 & -1 & 0 & -1 & 1 \\ \hline
\end{array}$$

- We now return to the simplex method.
- With the current tableau, the variables x_2 and x_3 have negative reduced costs.
- We choose x_3 to be the one that enters the basis.
- The pivot column is $u = (1, 1, -1)$.
- Since $u_3 < 0$, we only form the ratios $x_{B(i)}/u_i$, for $i = 1, 2$:
  - $x_{B(1)}/u_1 = 10/1 = 10$,
  - $x_{B(2)}/u_2 = 10/1 = 10$.
- There is again a tie, which we break by letting $\ell = 1$.

Example 3.5: First pivot

$$\begin{array}{r|r|rrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
\hline
 & 100 & 0 & -7 & -2 & 0 & 5 & 0 \\
x_4 = & 10 & 0 & 1.5 & 1 & 1 & -0.5 & 0 \\
x_1 = & 10 & 1 & 0.5 & 1 & 0 & 0.5 & 0 \\
x_6 = & 0 & 0 & 1 & -1 & 0 & -1 & 1 \\
\hline
\end{array}$$

- The first basic variable, x_4, exits the basis.
  This determines the pivot row and the pivot element.
- We multiply the pivot row by 2 and add it to the zeroth row.
- We subtract the pivot row from the second row.
- Finally, we add the pivot row to the third row.
- We obtain the following new tableau:

Example 3.5: Second pivot

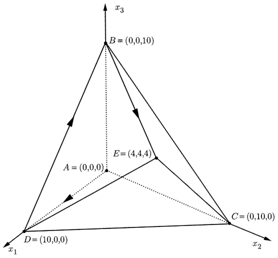
$$
\begin{array}{r|r|rrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
\cline{2-8}
 & 120 & 0 & -4 & 0 & 2 & 4 & 0 \\
\cline{2-8}
x_3 = & 10 & 0 & 1.5 & 1 & 1 & -0.5 & 0 \\
x_1 = & 0 & 1 & -1 & 0 & -1 & 1 & 0 \\
x_6 = & 10 & 0 & 2.5 & 0 & 1 & -1.5 & 1 \\
\cline{2-8}
\end{array}
$$

- The cost has been reduced to -120.
- The corresponding basic feasible solution is $x = (0, 0, 10, 0, 0, 10)$.

Example 3.5: Second pivot

$$\begin{array}{r|r|rrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
\hline
 & 120 & 0 & -4 & 0 & 2 & 4 & 0 \\
\hline
x_3 = & 10 & 0 & 1.5 & 1 & 1 & -0.5 & 0 \\
x_1 = & 0 & 1 & -1 & 0 & -1 & 1 & 0 \\
x_6 = & 10 & 0 & 2.5 & 0 & 1 & -1.5 & 1 \\
\hline
\end{array}$$

- In terms of the original variables x_1, x_2, x_3, we have moved to point $B = (0, 0, 10)$.



Example 3.5: Second pivot

$$\begin{array}{r|r|rrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
\hline
 & 120 & 0 & -4 & 0 & 2 & 4 & 0 \\
\hline
x_3 = & 10 & 0 & 1.5 & 1 & 1 & -0.5 & 0 \\
x_1 = & 0 & 1 & -1 & 0 & -1 & 1 & 0 \\
x_6 = & 10 & 0 & 2.5 & 0 & 1 & -1.5 & 1 \\
\hline
\end{array}$$

- At this point, x_2 is the only variable with negative reduced cost.
- We bring x_2 into the basis.
- The pivot column is $u = (1.5, -1, 2.5)$.
- Since $u_2 < 0$, we only form the ratios $x_{B(i)}/u_i$, for $i = 1, 3$:
  - $x_{B(1)}/u_1 = 10/1.5 = 6.\bar{6}$,
  - $x_{B(3)}/u_3 = 10/2.5 = 4$.
- We obtain $\ell = 3$, and the third basic variable, x_6 exits the basis.

Example 3.5: Second pivot

$$\begin{array}{r|r|rrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ \hline
 & 120 & 0 & -4 & 0 & 2 & 4 & 0 \\ \hline
x_3 = & 10 & 0 & 1.5 & 1 & 1 & -0.5 & 0 \\
x_1 = & 0 & 1 & -1 & 0 & -1 & 1 & 0 \\
x_6 = & 10 & 0 & 2.5 & 0 & 1 & -1.5 & 1 \\ \hline
\end{array}$$

- This determines the pivot row and the pivot element.
- We multiply the pivot row by $4/2.5$ and add it to the zeroth row.
- We multiply the pivot row by $1.5/2.5$ and subtract it to the first row.
- We multiply the pivot row by $1/2.5$ and add it to the second row.
- Finally, we divide the pivot row by 2.5.
- We obtain the following new tableau:

Example 3.5: Third pivot

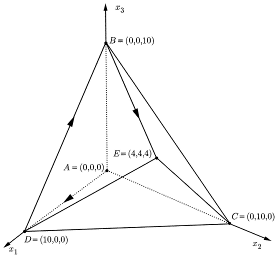
$$\begin{array}{r|r|rrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ \hline
 & 136 & 0 & 0 & 0 & 3.6 & 1.6 & 1.6 \\ \hline
x_3 = & 4 & 0 & 0 & 1 & 0.4 & 0.4 & -0.6 \\
x_1 = & 4 & 1 & 0 & 0 & -0.6 & 0.4 & 0.4 \\
x_2 = & 4 & 0 & 1 & 0 & 0.4 & -0.6 & 0.4 \\ \hline
\end{array}$$

- The cost has been reduced to -136.
- The corresponding basic feasible solution is $x = (4, 4, 4, 0, 0, 0)$.

Example 3.5: Third pivot

		x_1	x_2	x_3	x_4	x_5	x_6
	136	0	0	0	3.6	1.6	1.6
$x_3 =$	4	0	0	1	0.4	0.4	-0.6
$x_1 =$	4	1	0	0	-0.6	0.4	0.4
$x_2 =$	4	0	1	0	0.4	-0.6	0.4

- In terms of the original variables x_1, x_2, x_3, we have moved to point $E = (4, 4, 4)$.

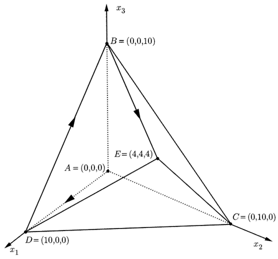


Example 3.5: Third pivot

$$
\begin{array}{r|r|rrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
\cline{2-8}
 & 136 & 0 & 0 & 0 & 3.6 & 1.6 & 1.6 \\
\cline{2-8}
x_3 = & 4 & 0 & 0 & 1 & 0.4 & 0.4 & -0.6 \\
x_1 = & 4 & 1 & 0 & 0 & -0.6 & 0.4 & 0.4 \\
x_2 = & 4 & 0 & 1 & 0 & 0.4 & -0.6 & 0.4 \\
\cline{2-8}
\end{array}
$$

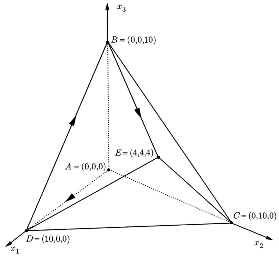
- The optimality of this solution is confirmed by observing that all reduced costs are nonnegative.

Example 3.5



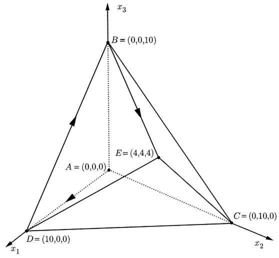
- In this example, the simplex method took three changes of basis to reach the optimal solution, and it traced the path $A - D - B - E$.
- With different pivoting rules, a different path would have been traced.

Example 3.5



- Question: Could the simplex method have solved the problem by tracing the path $A-D-E$, which involves only two edges, with only two iterations?

Example 3.5



- Question: Could the simplex method have solved the problem by tracing the path $A - D - E$, which involves only two edges, with only two iterations? The answer is no.
- The initial and final bases differ in three columns, and therefore at least three basis changes are required.
- In particular, if the method were to trace the path $A - D - E$, there would be a degenerate change of basis at point D (with no edge being traversed), which would again bring the total to three.

The full tableau implementation: running time

What is the total computational effort per iteration?

- The full tableau method requires a constant (and small) number of arithmetic operations for updating each entry of the tableau.
- Thus, the amount of computation per iteration is proportional to the size of the tableau, which is

$$O(mn).$$

- Therefore, the full tableau method is as efficient as the revised simplex method.

Comparison of the full tableau and the revised simplex methods

Comparison of the full tableau and the revised simplex

- Consider a linear programming problem in standard form

$$\begin{aligned} \text{minimize} \quad & c'x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0. \end{aligned}$$

- Let us pretend that the problem is changed to

$$\begin{aligned} \text{minimize} \quad & c'x + 0'y \\ \text{subject to} \quad & Ax + Iy = b \\ & x, y \geq 0 \end{aligned}$$

Comparison of the full tableau and the revised simplex

$$\begin{aligned} \text{minimize} \quad & c'x + 0'y \\ \text{subject to} \quad & Ax + Iy = b \\ & x, y \geq 0. \end{aligned}$$

- We implement the simplex method on this new problem, except that we never allow any of the components of the vector y to become basic.
- Then, we always have $y = 0$, and the simplex method performs basis changes as if the vector y were entirely absent.

Comparison of the full tableau and the revised simplex

- The equality constraints of our new standard form problem are $Ax + Iy = b$, thus the new constraint matrix is

$$[A \mid I].$$

- The vector of reduced costs in the augmented problem is

$$[c' \mid 0'] - c_B' B^{-1}[A \mid I] = [\bar{c}' \mid -c_B' B^{-1}].$$

- Thus, the simplex tableau for the augmented problem is

$-c_B' B^{-1} b$	$\bar{c}'$	$-c_B' B^{-1}$
$B^{-1} b$	$B^{-1} A$	B^{-1}

- In particular, by following the mechanics of the full tableau method on the above tableau, the inverse basis matrix B^{-1} is made available at each iteration.

Comparison of the full tableau and the revised simplex

$-c_B' B^{-1} b$	$\bar{c}'$	$-c_B' B^{-1}$
$B^{-1} b$	$B^{-1} A$	B^{-1}

- Consider now the revised simplex method, with a pivoting rule that evaluates one reduced cost at a time, until a negative reduced cost is found.
- It is essentially the full tableau method applied to the above augmented problem, except that the part of the tableau containing $\bar{c}'$ and $B^{-1} A$ is never formed explicitly.
- Instead, reduced costs are evaluated one at a time, and once the entering variable x_j is chosen, the pivot column $B^{-1} A_j$ is computed on the fly.

Comparison of the full tableau and the revised simplex

$-c_B' B^{-1} b$	$\bar{c}'$	$-c_B' B^{-1}$
$B^{-1} b$	$B^{-1} A$	B^{-1}

- Thus, the revised simplex method is just a variant of the full tableau method, with more efficient bookkeeping.
- If the revised simplex method also updates the zeroth row entries that lie on top of B^{-1} (by the usual elementary operations), the simplex multipliers $p' = c_B' B^{-1}$ become available, thus eliminating the need for computing $p' = c_B' B^{-1}$ at each iteration.

Comparison of the full tableau and the revised simplex

$-c_B' B^{-1} b$	$\bar{c}'$	$-c_B' B^{-1}$
$B^{-1} b$	$B^{-1} A$	B^{-1}

We now discuss the relative merits of the two methods.

- The full tableau method updates all the tableau at each iteration, and so the computational requirements per iteration are $O(mn)$.

Comparison of the full tableau and the revised simplex

$-c_B' B^{-1} b$	$\bar{c}'$	$-c_B' B^{-1}$
$B^{-1} b$	$B^{-1} A$	B^{-1}

- The revised simplex method updates B^{-1} and $p' = c_B' B^{-1}$, and since only $O(m^2)$ entries are updated, the computational requirements per iteration are $O(m^2)$.
- In addition, the reduced cost of each variable x_j can be computed by forming the inner product $p' A_j$, which requires $O(m)$ operations.
- In the worst case, the reduced cost of every variable is computed, for a total of $O(mn)$ computations per iteration.
- Since $m \leq n$, The worst-case computational effort per iteration is
$$O(mn + m^2) = O(mn).$$

Comparison of the full tableau and the revised simplex

$-c_B' B^{-1} b$	$\bar{c}'$	$-c_B' B^{-1}$
$B^{-1} b$	$B^{-1} A$	B^{-1}

- On the other hand, a typical iteration of the revised simplex method might require a lot less work.
- In the best case, if the first reduced cost computed is negative, and the corresponding variable is chosen to enter the basis, the total computational effort is only

$$O(m^2).$$

- The conclusion is that the revised simplex method cannot be slower than the full tableau method, and could be much faster during most iterations.

Comparison of the full tableau and the revised simplex

$-c_B' B^{-1} b$	$\bar{c}'$	$-c_B' B^{-1}$
$B^{-1} b$	$B^{-1} A$	B^{-1}

- Another important element in favor of the revised simplex method is that memory requirements are reduced from $O(mn)$ to $O(m^2)$. Why?
- As n is often much larger than m, this effect can be quite significant.

Comparison of the full tableau and the revised simplex

$-c_B' B^{-1} b$	$\bar{c}'$	$-c_B' B^{-1}$
$B^{-1} b$	$B^{-1} A$	B^{-1}

- It could be counterargued that the memory requirements of the revised simplex method are also $O(mn)$ because of the need to store the matrix A.
- However, in most large scale problems that arise in applications, the matrix A is very sparse (has many zero entries) and can be stored compactly.
- The sparsity of A does not usually help in the storage of the full simplex tableau because even if A and B are sparse, $B^{-1}A$ is not sparse, in general.

Comparison of the full tableau and the revised simplex

We summarize this discussion in the following table:

	Full tableau	Revised simplex
Worst-case time	$O(mn)$	$O(mn)$
Worst-case memory	$O(mn)$	$O(mn)$
Best-case time	$O(mn)$	$O(m^2)$
Best-case memory	$O(mn)$	$O(m^2)$

3.4 Anticycling: lexicography and Bland's rule

Example 3.6

- We now see an example that shows that the simplex method can cycle.
- We consider a problem described in terms of the following initial tableau.

$$\begin{array}{r|c|rrrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ \hline
 & 3 & -3/4 & 20 & -1/2 & 6 & 0 & 0 & 0 \\ \hline
x_5 = & 0 & 1/4 & -8 & -1 & 9 & 1 & 0 & 0 \\
x_6 = & 0 & 1/2 & -12 & -1/2 & 3 & 0 & 1 & 0 \\
x_7 = & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \hline
\end{array}$$

We use the following pivoting rules:

(a) We select a nonbasic variable with the most negative reduced cost $\bar{c}_j$ to be the one that enters the basis.

(b) Out of all basic variables that are eligible to exit the basis, we select the one with the smallest subscript.

We then obtain the following sequence of tableaux:

Example 3.6

Initial tableau:

$$\begin{array}{r|c|ccccccc|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\
\hline
 & 3 & -3/4 & 20 & -1/2 & 6 & 0 & 0 & 0 \\
\hline
x_5 = & 0 & 1/4 & -8 & -1 & 9 & 1 & 0 & 0 \\
x_6 = & 0 & 1/2 & -12 & -1/2 & 3 & 0 & 1 & 0 \\
x_7 = & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
\hline
\end{array}$$

Example 3.6

Initial tableau:

$$
\begin{array}{r|r|rrrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\
\hline
 & 3 & -3/4 & 20 & -1/2 & 6 & 0 & 0 & 0 \\
x_5 = & 0 & 1/4 & -8 & -1 & 9 & 1 & 0 & 0 \\
x_6 = & 0 & 1/2 & -12 & -1/2 & 3 & 0 & 1 & 0 \\
x_7 = & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
\hline
\end{array}
$$

First pivot:

$$
\begin{array}{r|r|rrrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\
\hline
 & 3 & 0 & -4 & -7/2 & 33 & 3 & 0 & 0 \\
\hline
x_1 = & 0 & 1 & -32 & -4 & 36 & 4 & 0 & 0 \\
x_6 = & 0 & 0 & 4 & 3/2 & -15 & -2 & 1 & 0 \\
x_7 = & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
\hline
\end{array}
$$

Example 3.6

First pivot:

$$\begin{array}{r|r|rrrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ \hline
 & 3 & 0 & -4 & -7/2 & 33 & 3 & 0 & 0 \\ \hline
x_1 = & 0 & 1 & -32 & -4 & 36 & 4 & 0 & 0 \\
x_6 = & 0 & 0 & 4 & 3/2 & -15 & -2 & 1 & 0 \\
x_7 = & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \hline
\end{array}$$

Example 3.6

Second pivot:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	3	0	0	-2	18	1	1	0
$x_1 =$	0	1	0	8	-84	-12	8	0
$x_2 =$	0	0	1	$3/8$	$-15/4$	$-1/2$	$1/4$	0
$x_7 =$	1	0	0	1	0	0	0	1

First pivot:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	3	0	-4	$-7/2$	33	3	0	0
$x_1 =$	0	1	-32	-4	36	4	0	0
$x_6 =$	0	0	4	$3/2$	-15	-2	1	0
$x_7 =$	1	0	0	1	0	0	0	1

Example 3.6

Second pivot:

$$\begin{array}{r|r|rrrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ \hline
 & 3 & 0 & 0 & -2 & 18 & 1 & 1 & 0 \\
x_1 = & 0 & 1 & 0 & 8 & -84 & -12 & 8 & 0 \\
x_2 = & 0 & 0 & 1 & 3/8 & -15/4 & -1/2 & 1/4 & 0 \\
x_7 = & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \hline
\end{array}$$

Example 3.6

Second pivot:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	3	0	0	-2	18	1	1	0
$x_1 =$	0	1	0	8	-84	-12	8	0
$x_2 =$	0	0	1	$3/8$	$-15/4$	$-1/2$	$1/4$	0
$x_7 =$	1	0	0	1	0	0	0	1

Third pivot:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	3	$1/4$	0	0	-3	-2	3	0
$x_3 =$	0	$1/8$	0	1	$-21/2$	$-3/2$	1	0
$x_2 =$	0	$-3/64$	1	0	$3/16$	$1/16$	$-1/8$	0
$x_7 =$	1	$-1/8$	0	0	$21/2$	$3/2$	-1	1

Example 3.6

Third pivot:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	3	$1/4$	0	0	-3	-2	3	0
$x_3 =$	0	$1/8$	0	1	$-21/2$	$-3/2$	1	0
$x_2 =$	0	$-3/64$	1	0	$3/16$	$1/16$	$-1/8$	0
$x_7 =$	1	$-1/8$	0	0	$21/2$	$3/2$	-1	1

Example 3.6

Fourth pivot:

$$\begin{array}{r|c|ccccccc|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\
\hline
 & 3 & -1/2 & 16 & 0 & 0 & -1 & 1 & 0 \\
\hline
x_3 = & 0 & -5/2 & 56 & 1 & 0 & 2 & -6 & 0 \\
x_4 = & 0 & -1/4 & 16/3 & 0 & 1 & 1/3 & -2/3 & 0 \\
x_7 = & 1 & 5/2 & -56 & 0 & 0 & -2 & 6 & 1 \\
\hline
\end{array}$$

Third pivot:

$$\begin{array}{r|c|ccccccc|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\
\hline
 & 3 & 1/4 & 0 & 0 & -3 & -2 & 3 & 0 \\
\hline
x_3 = & 0 & 1/8 & 0 & 1 & -21/2 & -3/2 & 1 & 0 \\
x_2 = & 0 & -3/64 & 1 & 0 & 3/16 & 1/16 & -1/8 & 0 \\
x_7 = & 1 & -1/8 & 0 & 0 & 21/2 & 3/2 & -1 & 1 \\
\hline
\end{array}$$

Example 3.6

Fourth pivot:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	3	$-1/2$	16	0	0	-1	1	0
$x_3 =$	0	$-5/2$	56	1	0	2	-6	0
$x_4 =$	0	$-1/4$	$16/3$	0	1	$1/3$	$-2/3$	0
$x_7 =$	1	$5/2$	-56	0	0	-2	6	1

Example 3.6

Fourth pivot:

$$\begin{array}{r|r|rrrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\
\hline
 & 3 & -1/2 & 16 & 0 & 0 & -1 & 1 & 0 \\
x_3 = & 0 & -5/2 & 56 & 1 & 0 & 2 & -6 & 0 \\
x_4 = & 0 & -1/4 & 16/3 & 0 & 1 & 1/3 & -2/3 & 0 \\
x_7 = & 1 & 5/2 & -56 & 0 & 0 & -2 & 6 & 1 \\
\hline
\end{array}$$

Fifth pivot:

$$\begin{array}{r|r|rrrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\
\hline
 & 3 & -7/4 & 44 & 1/2 & 0 & 0 & -2 & 0 \\
\hline
x_5 = & 0 & -5/4 & 28 & 1/2 & 0 & 1 & -3 & 0 \\
x_4 = & 0 & 1/6 & -4 & -1/6 & 1 & 0 & 1/3 & 0 \\
x_7 = & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
\hline
\end{array}$$

Example 3.6

Fifth pivot:

$$
\begin{array}{r|c|rrrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\
\hline
 & 3 & -7/4 & 44 & 1/2 & 0 & 0 & -\,2 & 0 \\
\hline
x_5 = & 0 & -5/4 & 28 & 1/2 & 0 & 1 & -\,3 & 0 \\
x_4 = & 0 & 1/6 & -4 & -1/6 & 1 & 0 & 1/3 & 0 \\
x_7 = & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
\hline
\end{array}
$$

Example 3.6

Sixth pivot:

$$\begin{array}{r|c|ccccccc|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\
\hline
 & 3 & -3/4 & 20 & -1/2 & 6 & 0 & 0 & 0 \\
\hline
x_5 = & 0 & 1/4 & -8 & -1 & 9 & 1 & 0 & 0 \\
x_6 = & 0 & 1/2 & -12 & -1/2 & 3 & 0 & 1 & 0 \\
x_7 = & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
\hline
\end{array}$$

Fifth pivot:

$$\begin{array}{r|c|ccccccc|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\
\hline
 & 3 & -7/4 & 44 & 1/2 & 0 & 0 & -2 & 0 \\
\hline
x_5 = & 0 & -5/4 & 28 & 1/2 & 0 & 1 & -3 & 0 \\
x_4 = & 0 & 1/6 & -4 & -1/6 & 1 & 0 & 1/3 & 0 \\
x_7 = & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
\hline
\end{array}$$

Example 3.6

Sixth pivot:

$$\begin{array}{r|c|ccccccc|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\
\hline
 & 3 & -3/4 & 20 & -1/2 & 6 & 0 & 0 & 0 \\
\hline
x_5 = & 0 & 1/4 & -8 & -1 & 9 & 1 & 0 & 0 \\
x_6 = & 0 & 1/2 & -12 & -1/2 & 3 & 0 & 1 & 0 \\
x_7 = & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
\hline
\end{array}$$

Example 3.6

Sixth pivot:

$$\begin{array}{r|r|rrrrrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\
\hline
 & 3 & -3/4 & 20 & -1/2 & 6 & 0 & 0 & 0 \\
\hline
x_5 = & 0 & 1/4 & -8 & -1 & 9 & 1 & 0 & 0 \\
x_6 = & 0 & 1/2 & -12 & -1/2 & 3 & 0 & 1 & 0 \\
x_7 = & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
\hline
\end{array}$$

- After six pivots, we have the same basis and the same tableau that we started with.
- At each basis change, we had $\theta^* = 0$.
- In particular, for each intermediate tableau, we had the same feasible solution and the same cost.
- The same sequence of pivots can be repeated over and over, and the simplex method never terminates.

Anticycling: lexicography and Bland's rule

- Next, we discuss anticycling rules under which the simplex method is guaranteed to terminate, thus extending Theorem 3.3 to degenerate problems.
- As an important corollary, we conclude that if the optimal cost is finite, then there exists an optimal basis, that is, a basis satisfying

$$B^{-1}b \geq 0$$

and

$$\bar{c}' = c' - c_B' B^{-1} A \geq 0'.$$

Lexicography

Lexicography

- We present here the lexicographic pivoting rule and see that it prevents the simplex method from cycling.
- We start with a definition.

Definition 3.5
A vector $u \in \mathbb{R}^n$ is said to be lexicographically larger (or smaller) than another vector $v \in \mathbb{R}^n$ if $u \neq v$ and the first component that is different in u and v is larger (or smaller, respectively) in u. Symbolically, we write

$$u >^L v \qquad \text{or} \qquad u <^L v.$$

Example:

$$(0,2,3,0) >^L (0,2,1,4),$$
$$(0,4,5,0) <^L (1,2,1,2).$$

Lexicography

- We present here the lexicographic pivoting rule and see that it prevents the simplex method from cycling.
- We start with a definition.

Definition 3.5
A vector $u \in \mathbb{R}^n$ is said to be lexicographically larger (or smaller) than another vector $v \in \mathbb{R}^n$ if $u \neq v$ and the first component that is different in u and v is larger (or smaller, respectively) in u. Symbolically, we write

$$u >^L v \qquad \text{or} \qquad u <^L v.$$

Remark: A vector $u \in \mathbb{R}^n$ is lexicographically positive if

$$u >^L 0,$$

i.e., if $u \neq 0$ and the first nonzero entry of u is positive.

Lexicography

Lexicographic pivoting rule

1. Choose an entering column A_j arbitrarily, as long as its reduced cost $\bar{c}_j$ is negative. Let $u = B^{-1}A_j$ be the jth column of the tableau.
2. For each i with $u_i > 0$, divide the ith row of the tableau (including the entry in the zeroth column) by u_i and choose the lexicographically smallest row. If row ℓ is lexicographically smallest, then the ℓth basic variable $x_{B(\ell)}$ exits the basis.

Example 3.7

- Consider the following tableau (the zeroth row is omitted), and suppose that the pivot column is the third one ($j = 3$).

$$\begin{array}{r|c|cccc|}
\hline
x_{B(1)} = & 1 & 0 & 5 & 3 & \cdots \\
x_{B(2)} = & 2 & 4 & 6 & -1 & \cdots \\
x_{B(3)} = & 3 & 0 & 7 & 9 & \cdots \\
\hline
\end{array}$$

- There is a tie in trying to determine the exiting variable:
  - $x_{B(1)}/u_1 = 1/3$,
  - $x_{B(3)}/u_3 = 3/9 = 1/3$.

Example 3.7

- Consider the following tableau (the zeroth row is omitted), and suppose that the pivot column is the third one ($j = 3$).

$$\begin{array}{r|c|ccc|}
\hline
x_{B(1)} = & 1 & 0 & 5 & 3 & \cdots \\
x_{B(2)} = & 2 & 4 & 6 & -1 & \cdots \\
x_{B(3)} = & 3 & 0 & 7 & 9 & \cdots \\
\hline
\end{array}$$

- We divide the first and third rows of the tableau by $u_1 = 3$ and $u_3 = 9$, respectively, to obtain:

$$\begin{array}{r|c|ccc|}
\hline
x_{B(1)} = & 1/3 & 0 & 5/3 & 1 & \cdots \\
x_{B(2)} = & * & * & * & * & \cdots \\
x_{B(3)} = & 1/3 & 0 & 7/9 & 1 & \cdots \\
\hline
\end{array}$$

Example 3.7

- Consider the following tableau (the zeroth row is omitted), and suppose that the pivot column is the third one ($j = 3$).

$x_{B(1)} =$	1	0	5	3	$\cdots$
$x_{B(2)} =$	2	4	6	-1	$\cdots$
$x_{B(3)} =$	3	0	7	9	$\cdots$

- We divide the first and third rows of the tableau by $u_1 = 3$ and $u_3 = 9$, respectively, to obtain:

$x_{B(1)} =$	$1/3$	0	$5/3$	1	$\cdots$
$x_{B(2)} =$	$*$	$*$	$*$	$*$	$\cdots$
$x_{B(3)} =$	$1/3$	0	$7/9$	1	$\cdots$

- The tie between the first and third rows is resolved by performing a lexicographic comparison.

Example 3.7

- Consider the following tableau (the zeroth row is omitted), and suppose that the pivot column is the third one ($j = 3$).

$$\begin{array}{r|c|cccc|}
\hline
x_{B(1)} = & 1 & 0 & 5 & 3 & \cdots \\
x_{B(2)} = & 2 & 4 & 6 & -1 & \cdots \\
x_{B(3)} = & 3 & 0 & 7 & 9 & \cdots \\
\hline
\end{array}$$

- We divide the first and third rows of the tableau by $u_1 = 3$ and $u_3 = 9$, respectively, to obtain:

$$\begin{array}{r|c|cccc|}
\hline
x_{B(1)} = & 1/3 & 0 & 5/3 & 1 & \cdots \\
x_{B(2)} = & * & * & * & * & \cdots \\
x_{B(3)} = & 1/3 & 0 & 7/9 & 1 & \cdots \\
\hline
\end{array}$$

- Since $7/9 < 5/3$, the third row is chosen to be the pivot row, and the variable $x_{B(3)}$ exits the basis.

Lexicography

The lexicographic pivoting rule always leads to a unique choice for the exiting variable.

- Indeed, if this were not the case, two of the rows in the tableau would have to be proportional.
- But if two rows of the matrix $B^{-1}A$ are proportional, the matrix $B^{-1}A$ has rank smaller than m.
- Therefore, A also has rank less than m, which contradicts our standing assumption that A has linearly independent rows.

Lexicography

Theorem 3.4
Suppose that the simplex algorithm starts with all the rows in the simplex tableau, except the zeroth row, lexicographically positive. If the lexicographic pivoting rule is followed, then:

(a) Every row of the tableau, except the zeroth row, remains lexicographically positive throughout the algorithm.

(b) The zeroth row strictly increases lexicographically at each iteration.

(c) The simplex method terminates after a finite number of iterations.

Let's prove it!

Lexicography

Before we start, some simple properties of "$>^L$".

$$
\begin{array}{lcl}
u >^L v & \Leftrightarrow & u - v >^L 0 \\
 & & \\
u >^L 0,\ \alpha > 0 & \Rightarrow & \alpha u >^L 0 \\
u >^L 0,\ v >^L 0 & \Rightarrow & u + v >^L 0 \\
u >^L 0 & \Rightarrow & u + v >^L v.
\end{array}
$$

Lexicography

Theorem 3.4
Suppose that the simplex algorithm starts with all the rows in the simplex tableau, except the zeroth row, lexicographically positive. If the lexicographic pivoting rule is followed, then:

(a) Every row of the tableau, except the zeroth row, remains lexicographically positive throughout the algorithm.

Proof (a). Suppose that x_j enters the basis and that the pivot row is the ℓ-th. We have $u_\ell > 0$ and

$$\frac{(\text{old } \ell\text{-th row})}{u_\ell} <^L \frac{(\text{old } i\text{-th row})}{u_i}, \text{ if } i \neq \ell \text{ and } u_i > 0. \qquad (*)$$

- $(\text{new } \ell\text{-th row}) \leftarrow \frac{(\text{old } \ell\text{-th row})}{u_\ell}$, still lexicographically positive since $u_\ell > 0$.

Lexicography

Theorem 3.4
Suppose that the simplex algorithm starts with all the rows in the simplex tableau, except the zeroth row, lexicographically positive. If the lexicographic pivoting rule is followed, then:

(a) Every row of the tableau, except the zeroth row, remains lexicographically positive throughout the algorithm.

Proof (a). Suppose that x_j enters the basis and that the pivot row is the ℓ-th. We have $u_\ell > 0$ and

$$\frac{(\text{old } \ell\text{-th row})}{u_\ell} <^L \frac{(\text{old } i\text{-th row})}{u_i}, \text{ if } i \neq \ell \text{ and } u_i > 0. \qquad (*)$$

- (new i-th row) $\leftarrow$ (old i-th row) $- \frac{u_i}{u_\ell}$(old ℓ-th row) for $i \neq \ell$.

Lexicography

Theorem 3.4
Suppose that the simplex algorithm starts with all the rows in the simplex tableau, except the zeroth row, lexicographically positive. If the lexicographic pivoting rule is followed, then:

(a) Every row of the tableau, except the zeroth row, remains lexicographically positive throughout the algorithm.

Proof (a). Suppose that x_j enters the basis and that the pivot row is the ℓ-th. We have $u_\ell > 0$ and

$$\frac{(\text{old } \ell\text{-th row})}{u_\ell} <^L \frac{(\text{old } i\text{-th row})}{u_i}, \text{ if } i \neq \ell \text{ and } u_i > 0. \qquad (*)$$

- (new i-th row) $\leftarrow$ (old i-th row) $- \frac{u_i}{u_\ell}$(old ℓ-th row) for $i \neq \ell$.
  - If $u_i \leq 0$, then $-\frac{u_i}{u_\ell} \geq 0 \Rightarrow$ (new i-th row) is lexicographically positive.
  - If $u_i > 0$, then $(*) \Rightarrow$ (new i-th row) is lexicographically positive.

Lexicography

Theorem 3.4
Suppose that the simplex algorithm starts with all the rows in the simplex tableau, except the zeroth row, lexicographically positive. If the lexicographic pivoting rule is followed, then:

(b) The zeroth row strictly increases lexicographically at each iteration.

Proof (b).

- At the beginning of an iteration, the pivot element is positive and the reduced cost in the pivot column is negative.
- To make this reduced cost 0, we add a positive multiple of the pivot row, which is lexicographically positive.
- Thus, the zeroth row increases lexicographically.

Lexicography

Theorem 3.4
Suppose that the simplex algorithm starts with all the rows in the simplex tableau, except the zeroth row, lexicographically positive. If the lexicographic pivoting rule is followed, then:

(c) The simplex method terminates after a finite number of iterations.

Proof (c).

- The zeroth row is completely determined by the current basis.
- From (b), the zeroth row strictly increases lexicographically at each iteration, thus no basis can be repeated twice.
- Since there is a finite number of bases, the simplex method must terminate in a finite number of iterations. □

Lexicography

- The lexicographic pivoting rule is straightforward to use if the simplex method is implemented in terms of the full tableau.
- It can also be used in conjunction with the revised simplex method, provided that the inverse basis matrix B^{-1} is formed explicitly.

Lexicography

- In order to apply the lexicographic pivoting rule, an initial tableau with lexicographically positive rows is required.

- Assume that an initial tableau is available (methods for obtaining an initial tableau are discussed in the next section).
- We can rename the variables so that the basic variables are the first m ones.
- This is equivalent to rearranging the tableau so that the first m columns of $B^{-1}A$ are the m unit vectors.
- The resulting tableau has lexicographically positive rows, as desired.

Bland's rule

Bland's rule

The smallest subscript pivoting rule, also known as Bland's rule, is as follows.

Smallest subscript pivoting rule

1. Find the smallest j for which the reduced cost $\bar{c}_j$ is negative and have the column A_j enter the basis.
2. Out of all variables x_i that are tied in the test for choosing an exiting variable, select the one with the smallest value of i.

Remark: Selecting the variable x_i with the smallest value of i is not the same as selecting the variable $x_{B(i)}$ with the smallest value of $B(i)$.

Bland's rule

- This pivoting rule is compatible with an implementation of the revised simplex method in which the reduced costs of the nonbasic variables are computed one at a time, in the natural order, until a negative one is discovered.
- Under this pivoting rule, it is known that cycling never occurs and the simplex method is guaranteed to terminate after a finite number of iterations.

Theorem (Termination with Blands rule)
If the simplex method uses Blands rule, it terminates after a finite number of iterations.

Let's prove it!

3.5 Finding an initial basic feasible solution

Finding an initial basic feasible solution

- In order to start the simplex method, we need to find an initial basic feasible solution.

- Sometimes this is straightforward, like in Example 3.5.
- More generally, suppose that we are dealing with a problem involving constraints of the form

$$\begin{aligned} Ax &\leq b \\ x &\geq 0, \end{aligned}$$

  where $b \geq 0$.
- We can then introduce nonnegative slack variables s and rewrite the constraints in the form

$$\begin{aligned} Ax + s &= b \\ x, s &\geq 0. \end{aligned}$$

Finding an initial basic feasible solution

- In order to start the simplex method, we need to find an initial basic feasible solution.

- Sometimes this is straightforward, like in Example 3.5.
- More generally, suppose that we are dealing with a problem involving constraints of the form

$$\begin{aligned} Ax &\leq b \\ x &\geq 0, \end{aligned}$$

  where $b \geq 0$.
- We can then introduce nonnegative slack variables s and rewrite the constraints in the form

$$\begin{aligned} Ax + s &= b \\ x, s &\geq 0. \end{aligned}$$

- The vector (x, s) defined by $x = 0$ and $s = b$ is a basic feasible solution and the corresponding basis matrix is the identity.

Finding an initial basic feasible solution

- In general, finding an initial basic feasible solution is not easy.
- It can be done by solving an auxiliary linear programming problem.
- Let's see how!

Finding an initial basic feasible solution

- Consider the problem

$$\begin{aligned} \text{minimize} \quad & c'x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0. \end{aligned}$$

- By possibly multiplying some of the equality constraints by -1, we can assume, without loss of generality, that $b \geq 0$.

Finding an initial basic feasible solution

- Consider the problem

$$\begin{aligned} \text{minimize} \quad & c'x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0. \end{aligned}$$

- By possibly multiplying some of the equality constraints by -1, we can assume, without loss of generality, that $b \geq 0$.
- We now introduce a vector $y \in \mathbb{R}^m$ of artificial variables and use the simplex method to solve the auxiliary problem

$$\begin{aligned} \text{minimize} \quad & y_1 + y_2 + \cdots + y_m \\ \text{subject to} \quad & Ax + y = b \\ & x \geq 0 \\ & y \geq 0. \end{aligned}$$

- Initialization is easy for the auxiliary problem: by letting $x = 0$ and $y = b$, we have a basic feasible solution and the corresponding basis matrix is the identity.

Finding an initial basic feasible solution

$$\begin{array}{rl} \text{minimize} & c'x \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array} \qquad \begin{array}{rl} \text{minimize} & y_1 + y_2 + \cdots + y_m \\ \text{subject to} & Ax + y = b \\ & x \geq 0 \\ & y \geq 0 \end{array}$$

- If x is a feasible solution to the original problem, this choice of x together with $y = 0$, yields a zero cost solution to the auxiliary problem.
- Therefore, if the optimal cost in the auxiliary problem is nonzero, we conclude that the original problem is infeasible.
- If we obtain a zero cost solution to the auxiliary problem, it must satisfy $y = 0$, and x is a feasible solution to the original problem.

Finding an initial basic feasible solution

- We have a method that either detects infeasibility or finds a feasible solution to the original problem.
- However, in order to initialize the simplex method for the original problem, we need
  - a basic feasible solution,
  - an associated basis matrix B, and
  - the corresponding tableau (depending on the implementation).

Finding an initial basic feasible solution

- All this is straightforward if the simplex method, applied to the auxiliary problem, terminates with a basis matrix B consisting exclusively of columns of A.

$\cancel{0}$	$\cancel{c' - c'_B B^{-1} A}$	$\cancel{c' - c'_B B^{-1}}$
$B^{-1}b$	$B^{-1}A$	$\cancel{B^{-1}}$

$$\downarrow$$

$-c'_B B^{-1} b$	$c' - c'_B B^{-1} A$
$B^{-1}b$	$B^{-1}A$

- We drop the columns that correspond to the artificial variables.
- We use B as the starting basis matrix.
- We recompute the zeroth row using the original cost vector c.
- We continue with the simplex method on the original problem.

Driving artificial variables out of the basis

Driving artificial variables out of the basis

The situation is more complex if:

- the original problem is feasible, and
- the simplex method applied to the auxiliary problem terminates with a feasible solution x^* to the original problem, where some of the artificial variables are in the final basis.

Since the final value of the artificial variables is zero, this implies that we have a degenerate basic feasible solution to the auxiliary problem.

- Our task is to obtain a different basis of x^* consisting only of columns of A.

Driving artificial variables out of the basis

- Let k be the number of columns of A that belong to the final basis ($k < m$) and, without loss of generality, assume that these are the columns $A_{B(1)}, \ldots, A_{B(k)}$.
- Note that the columns $A_{B(1)}, \ldots, A_{B(k)}$ must be linearly independent since they are part of a basis.

Driving artificial variables out of the basis

- Let k be the number of columns of A that belong to the final basis ($k < m$) and, without loss of generality, assume that these are the columns $A_{B(1)}, \dots, A_{B(k)}$.
- Note that the columns $A_{B(1)}, \dots, A_{B(k)}$ must be linearly independent since they are part of a basis.
- Under our standard assumption that the matrix A has full rank, the columns of A span $\mathbb{R}^m$, and we can choose $m - k$ additional columns $A_{B(k+1)}, \dots, A_{B(m)}$ of A, to obtain a set of m linearly independent columns, that is, a basis consisting exclusively of columns of A.

Driving artificial variables out of the basis

- Let k be the number of columns of A that belong to the final basis ($k < m$) and, without loss of generality, assume that these are the columns $A_{B(1)}, \ldots, A_{B(k)}$.
- Note that the columns $A_{B(1)}, \ldots, A_{B(k)}$ must be linearly independent since they are part of a basis.
- Under our standard assumption that the matrix A has full rank, the columns of A span $\mathbb{R}^m$, and we can choose $m - k$ additional columns $A_{B(k+1)}, \ldots, A_{B(m)}$ of A, to obtain a set of m linearly independent columns, that is, a basis consisting exclusively of columns of A.
- With this basis, all nonbasic components of x^* are at zero level, and it follows that x^* is the basic feasible solution associated with this new basis as well (cf. Theorem 2.4).
- At this point, the artificial variables and the corresponding columns of the tableau can be dropped.

Driving artificial variables out of the basis

- The procedure we have just described is called

  driving the artificial variables out of the basis,

  and depends crucially on the assumption that the matrix A has rank m.
- If A has rank less than m, constructing a basis for $\mathbb{R}^m$ using the columns of A is impossible and there exist redundant equality constraints that must be eliminated, as described by Theorem 2.5.
- All of the above can be carried out mechanically, in terms of the simplex tableau, in the following manner.

Driving artificial variables out of the basis

- Suppose that the ℓth basic variable is an artificial variable, which is in the basis at zero level.
- We examine the ℓth row of the tableau and search for some j such that the ℓth entry of $B^{-1}A_j$ is nonzero.

0	$\bar{c}_1$	$\dots$	$\bar{c}_n$	$\dots$	0	$\dots$
$x_{B(1)}$	$(B^{-1}A_1)_1$	$\dots$	$(B^{-1}A_n)_1$	$\dots$	0	$\dots$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	
$x_{B(\ell)}$	$(B^{-1}A_1)_\ell$	$\dots$	$(B^{-1}A_n)_\ell$	$\dots$	1	$\dots$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\dots$	$(B^{-1}A_n)_m$	$\dots$	0	$\dots$

- We either find this index j or not. We consider separately these two cases.

Driving artificial variables out of the basis

Case 1. We find some j such that the ℓth entry of $B^{-1}A_j$ is nonzero.

0	$\bar{c}_1$	$\dots$	$\bar{c}_j$	$\dots$	$\bar{c}_n$	$\dots$	0	$\dots$
$x_{B(1)}$	$(B^{-1}A_1)_1$	$\dots$	$(B^{-1}A_j)_1$	$\dots$	$(B^{-1}A_n)_1$	$\dots$	0	$\dots$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$	
$x_{B(\ell)}$	$(B^{-1}A_1)_\ell$	$\dots$	$(B^{-1}A_j)_\ell$	$\dots$	$(B^{-1}A_n)_\ell$	$\dots$	1	$\dots$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$	
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\dots$	$(B^{-1}A_j)_m$	$\dots$	$(B^{-1}A_n)_m$	$\dots$	0	$\dots$

Driving artificial variables out of the basis

Case 1. We find some j such that the ℓth entry of $B^{-1}A_j$ is nonzero.

0	$\bar{c}_1$	$\dots$	$\bar{c}_j$	$\dots$	$\bar{c}_n$	$\dots$	0	$\dots$
$x_{B(1)}$	$(B^{-1}A_1)_1$	$\dots$	$(B^{-1}A_j)_1$	$\dots$	$(B^{-1}A_n)_1$	$\dots$	0	$\dots$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$	
$x_{B(\ell)}$	$(B^{-1}A_1)_\ell$	$\dots$	$(B^{-1}A_j)_\ell$	$\dots$	$(B^{-1}A_n)_\ell$	$\dots$	1	$\dots$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$	
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\dots$	$(B^{-1}A_j)_m$	$\dots$	$(B^{-1}A_n)_m$	$\dots$	0	$\dots$

- We claim that A_j is linearly independent from the columns $A_{B(1)}, \dots, A_{B(k)}$.

Driving artificial variables out of the basis

Case 1. We find some j such that the ℓth entry of $B^{-1}A_j$ is nonzero.

0	$\bar{c}_1$	$\dots$	$\bar{c}_j$	$\dots$	$\bar{c}_n$	$\dots$	0	$\dots$
$x_{B(1)}$	$(B^{-1}A_1)_1$	$\dots$	$(B^{-1}A_j)_1$	$\dots$	$(B^{-1}A_n)_1$	$\dots$	0	$\dots$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$	
$x_{B(\ell)}$	$(B^{-1}A_1)_\ell$	$\dots$	$(B^{-1}A_j)_\ell$	$\dots$	$(B^{-1}A_n)_\ell$	$\dots$	1	$\dots$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$	
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\dots$	$(B^{-1}A_j)_m$	$\dots$	$(B^{-1}A_n)_m$	$\dots$	0	$\dots$

- To see this, note that $B^{-1}A_{B(i)} = e_i$, $i = 1, \dots, k$, and since $k < \ell$, the ℓth entry of these vectors is zero.
- Since the ℓth entry of $B^{-1}A_j$ is nonzero, this vector is not a linear combination of the vectors $B^{-1}A_{B(1)}, \dots, B^{-1}A_{B(k)}$.
- Equivalently, A_j is not a linear combination of the vectors $A_{B(1)}, \dots, A_{B(k)}$, which proves our claim.

Driving artificial variables out of the basis

Case 1. We find some j such that the ℓth entry of $B^{-1}A_j$ is nonzero.

0	$\bar{c}_1$	$\dots$	$\bar{c}_j$	$\dots$	$\bar{c}_n$	$\dots$	0	$\dots$
$x_{B(1)}$	$(B^{-1}A_1)_1$	$\dots$	$(B^{-1}A_j)_1$	$\dots$	$(B^{-1}A_n)_1$	$\dots$	0	$\dots$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$	
$x_{B(\ell)}$	$(B^{-1}A_1)_\ell$	$\dots$	$(B^{-1}A_j)_\ell$	$\dots$	$(B^{-1}A_n)_\ell$	$\dots$	1	$\dots$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$	
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\dots$	$(B^{-1}A_j)_m$	$\dots$	$(B^{-1}A_n)_m$	$\dots$	0	$\dots$

- We now bring A_j into the basis and have the ℓth basic variable exit the basis.

Driving artificial variables out of the basis

Case 1. We find some j such that the ℓth entry of $B^{-1}A_j$ is nonzero.

0	$\bar{c}_1$	$\dots$	$\bar{c}_j$	$\dots$	$\bar{c}_n$	$\dots$	0	$\dots$
$x_{B(1)}$	$(B^{-1}A_1)_1$	$\dots$	$(B^{-1}A_j)_1$	$\dots$	$(B^{-1}A_n)_1$	$\dots$	0	$\dots$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$	
$x_{B(\ell)}$	$(B^{-1}A_1)_\ell$	$\dots$	$(B^{-1}A_j)_\ell$	$\dots$	$(B^{-1}A_n)_\ell$	$\dots$	1	$\dots$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$	
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\dots$	$(B^{-1}A_j)_m$	$\dots$	$(B^{-1}A_n)_m$	$\dots$	0	$\dots$

- This is accomplished in the usual manner: perform those elementary row operations that replace $B^{-1}A_j$ by the ℓth unit vector.
- The only difference from the usual mechanics of the simplex method is that the pivot element could be negative.

Driving artificial variables out of the basis

Case 1. We find some j such that the ℓth entry of $B^{-1}A_j$ is nonzero.

0	$\bar{c}_1$	$\dots$	$\bar{c}_j$	$\dots$	$\bar{c}_n$	$\dots$ 0 $\dots$
$x_{B(1)}$	$(B^{-1}A_1)_1$	$\dots$	$(B^{-1}A_j)_1$	$\dots$	$(B^{-1}A_n)_1$	$\dots$ 0 $\dots$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
$x_{B(\ell)}$	$(B^{-1}A_1)_\ell$	$\dots$	$(B^{-1}A_j)_\ell$	$\dots$	$(B^{-1}A_n)_\ell$	$\dots$ 1 $\dots$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\dots$	$(B^{-1}A_j)_m$	$\dots$	$(B^{-1}A_n)_m$	$\dots$ 0 $\dots$

- Because $x_{B(\ell)} = 0$, adding a multiple of the ℓth row to the other rows does not change the values of the basic variables.
- This means that after the change of basis, we are still at the same basic feasible solution to the auxiliary problem, but we have reduced the number of basic artificial variables by one.

Driving artificial variables out of the basis

Case 2. We cannot find some j such that the ℓth entry of $B^{-1}A_j$ is nonzero.

0	$\bar{c}_1$	$\dots$	$\bar{c}_n$	$\dots$	0	$\dots$
$x_{B(1)}$	$(B^{-1}A_1)_1$	$\dots$	$(B^{-1}A_n)_1$	$\dots$	0	$\dots$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	
$x_{B(\ell)}$	0	$\dots$	0	$\dots$	1	$\dots$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\dots$	$(B^{-1}A_n)_m$	$\dots$	0	$\dots$

- In this case, the ℓth row of $B^{-1}A$ is zero.
- Note that the ℓth row of $B^{-1}A$ is equal to $g'A$, where g' is the ℓth row of B^{-1}.
- Hence, $g'A = 0'$ for some nonzero vector g, and the matrix A has linearly dependent rows.

Driving artificial variables out of the basis

Case 2. We cannot find some j such that the ℓth entry of $B^{-1}A_j$ is nonzero.

0	$\bar{c}_1$	$\dots$	$\bar{c}_n$	$\dots$	0	$\dots$
$x_{B(1)}$	$(B^{-1}A_1)_1$	$\dots$	$(B^{-1}A_n)_1$	$\dots$	0	$\dots$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	
$x_{B(\ell)}$	0	$\dots$	0	$\dots$	1	$\dots$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	
$x_{B(m)}$	$(B^{-1}A_1)_m$	$\dots$	$(B^{-1}A_n)_m$	$\dots$	0	$\dots$

- Since we are dealing with a feasible problem (why?), we must also have $g'b = 0$.
- Thus, the constraint $g'Ax = g'b$ is redundant and can be eliminated (cf. Theorem 2.5 in Section 2.3).
- Since this constraint is the information provided by the ℓth row of the tableau, we can eliminate that row and continue from there.

Example 3.8

Consider the linear programming problem

$$\begin{aligned} \text{minimize} \quad & x_1 + x_2 + x_3 \\ \text{subject to} \quad & x_1 + 2x_2 + 3x_3 = 3 \\ & x_1 - 2x_2 - 6x_3 = -2 \\ & 4x_2 + 9x_3 = 5 \\ & 3x_3 + x_4 = 1 \\ & x_1, \ldots, x_4 \geq 0. \end{aligned}$$

Example 3.8

Consider the linear programming problem

$$\begin{aligned} \text{minimize} \quad & x_1 + x_2 + x_3 \\ \text{subject to} \quad & x_1 + 2x_2 + 3x_3 = 3 \\ & x_1 - 2x_2 - 6x_3 = -2 \\ & 4x_2 + 9x_3 = 5 \\ & 3x_3 + x_4 = 1 \\ & x_1, \dots, x_4 \geq 0. \end{aligned}$$

In order to find a feasible solution, we form the auxiliary problem

$$\begin{aligned} \text{minimize} \quad & x_5 + x_6 + x_7 + x_8 \\ \text{subject to} \quad & x_1 + 2x_2 + 3x_3 + x_5 = 3 \\ & -x_1 + 2x_2 + 6x_3 + x_6 = 2 \\ & 4x_2 + 9x_3 + x_7 = 5 \\ & 3x_3 + x_4 + x_8 = 1 \\ & x_1, \dots, x_4, x_5, \dots, x_8 \geq 0. \end{aligned}$$

Example 3.8

$$\begin{aligned}\text{minimize}\quad & x_5 + x_6 + x_7 + x_8 \\ \text{subject to}\quad & x_1 + 2x_2 + 3x_3 + x_5 = 3 \\ & -x_1 + 2x_2 + 6x_3 + x_6 = 2 \\ & 4x_2 + 9x_3 + x_7 = 5 \\ & 3x_3 + x_4 + x_8 = 1 \\ & x_1, \ldots, x_4, x_5, \ldots, x_8 \geq 0.\end{aligned}$$

- A basic feasible solution to the auxiliary problem is obtained by letting

$$\begin{aligned} x_1, x_2, x_3, x_4 &= 0, \\ (x_5, x_6, x_7, x_8) &= b = (3, 2, 5, 1).\end{aligned}$$

- The corresponding basis matrix is the identity, and the cost of the solution is 11.

Example 3.8

$$\begin{aligned} \text{minimize} \quad & x_5 + x_6 + x_7 + x_8 \\ \text{subject to} \quad & x_1 + 2x_2 + 3x_3 + x_5 = 3 \\ & -x_1 + 2x_2 + 6x_3 + x_6 = 2 \\ & 4x_2 + 9x_3 + x_7 = 5 \\ & 3x_3 + x_4 + x_8 = 1 \\ & x_1, \ldots, x_4, x_5, \ldots, x_8 \geq 0. \end{aligned}$$

- Furthermore, we have $c' = [0' \mid 1']$, $c'_B = 1'$.
- The vector of reduced costs in the auxiliary problem is

$$c' - c'_B B^{-1}[A \mid I] = [0' \mid 1'] - 1' I [A \mid I]$$

Example 3.8

$$\begin{aligned}
\text{minimize} \quad & x_5 + x_6 + x_7 + x_8 \\
\text{subject to} \quad & x_1 + 2x_2 + 3x_3 + x_5 = 3 \\
& -x_1 + 2x_2 + 6x_3 + x_6 = 2 \\
& 4x_2 + 9x_3 + x_7 = 5 \\
& 3x_3 + x_4 + x_8 = 1 \\
& x_1, \ldots, x_4, x_5, \ldots, x_8 \geq 0.
\end{aligned}$$

- Furthermore, we have $c' = [0' \mid 1']$, $c_B' = 1'$.
- The vector of reduced costs in the auxiliary problem is

$$\begin{aligned}
c' - c_B' B^{-1}[A \mid I] &= [0' \mid 1'] - 1' I [A \mid I] \\
&= [-1'A \mid 1' - 1']
\end{aligned}$$

Example 3.8

$$\begin{aligned}
\text{minimize} \quad & x_5 + x_6 + x_7 + x_8 \\
\text{subject to} \quad & x_1 + 2x_2 + 3x_3 + x_5 = 3 \\
& -x_1 + 2x_2 + 6x_3 + x_6 = 2 \\
& 4x_2 + 9x_3 + x_7 = 5 \\
& 3x_3 + x_4 + x_8 = 1 \\
& x_1, \ldots, x_4, x_5, \ldots, x_8 \geq 0.
\end{aligned}$$

- Furthermore, we have $c' = [0' \mid 1']$, $c_B' = 1'$.
- The vector of reduced costs in the auxiliary problem is

$$\begin{aligned}
c' - c_B' B^{-1}[A \mid I] &= [0' \mid 1'] - 1' I [A \mid I] \\
&= [-1'A \mid 1' - 1'] = [-1'A \mid 0'].
\end{aligned}$$

Example 3.8

$$\begin{aligned} \text{minimize} \quad & x_5 + x_6 + x_7 + x_8 \\ \text{subject to} \quad & x_1 + 2x_2 + 3x_3 + x_5 = 3 \\ & -x_1 + 2x_2 + 6x_3 + x_6 = 2 \\ & 4x_2 + 9x_3 + x_7 = 5 \\ & 3x_3 + x_4 + x_8 = 1 \\ & x_1, \ldots, x_4, x_5, \ldots, x_8 \geq 0. \end{aligned}$$

- Furthermore, we have $c' = [0' \mid 1']$, $c_B' = 1'$.
- The vector of reduced costs in the auxiliary problem is

$$\begin{aligned} c' - c_B' B^{-1} [A \mid I] &= [0' \mid 1'] - 1' I [A \mid I] \\ &= [-1'A \mid 1' - 1'] = [-1'A \mid 0']. \end{aligned}$$

- We form the initial tableau:

Example 3.8

$$\begin{array}{r|r|rrrr|rrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
\hline
 & -11 & 0 & -8 & -21 & -1 & 0 & 0 & 0 & 0 \\
\hline
x_5 = & 3 & 1 & 2 & 3 & 0 & 1 & 0 & 0 & 0 \\
x_6 = & 2 & -1 & 2 & 6 & 0 & 0 & 1 & 0 & 0 \\
x_7 = & 5 & 0 & 4 & 9 & 0 & 0 & 0 & 1 & 0 \\
x_8 = & 1 & 0 & 0 & 3 & 1 & 0 & 0 & 0 & 1 \\
\hline
\end{array}$$

- We bring x_4 into the basis and have x_8 exit the basis.
- The basis matrix B is still the identity and only the zeroth row of the tableau changes.
- We obtain:

Example 3.8

$$\begin{array}{r|r|rrrr|rrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
\hline
 & -11 & 0 & -8 & -21 & -1 & 0 & 0 & 0 & 0 \\
\hline
x_5 = & 3 & 1 & 2 & 3 & 0 & 1 & 0 & 0 & 0 \\
x_6 = & 2 & -1 & 2 & 6 & 0 & 0 & 1 & 0 & 0 \\
x_7 = & 5 & 0 & 4 & 9 & 0 & 0 & 0 & 1 & 0 \\
x_8 = & 1 & 0 & 0 & 3 & 1 & 0 & 0 & 0 & 1 \\
\hline
\end{array}$$

- We bring x_4 into the basis and have x_8 exit the basis.
- The basis matrix B is still the identity and only the zeroth row of the tableau changes.
- We obtain:

Example 3.8

$$\begin{array}{r|r|rrrr|rrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
\hline
 & -10 & 0 & -8 & -18 & 0 & 0 & 0 & 0 & 1 \\
\hline
x_5 = & 3 & 1 & 2 & 3 & 0 & 1 & 0 & 0 & 0 \\
x_6 = & 2 & -1 & 2 & 6 & 0 & 0 & 1 & 0 & 0 \\
x_7 = & 5 & 0 & 4 & 9 & 0 & 0 & 0 & 1 & 0 \\
x_4 = & 1 & 0 & 0 & 3 & 1 & 0 & 0 & 0 & 1 \\
\hline
\end{array}$$

- We now bring x_3 into the basis and have x_4 exit the basis.
- The new tableau is:

Example 3.8

		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
	-10	0	-8	-18	0	0	0	0	1
$x_5 =$	3	1	2	3	0	1	0	0	0
$x_6 =$	2	-1	2	6	0	0	1	0	0
$x_7 =$	5	0	4	9	0	0	0	1	0
$x_4 =$	1	0	0	3	1	0	0	0	1

- We now bring x_3 into the basis and have x_4 exit the basis.
- The new tableau is:

Example 3.8

		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
	-4	0	-8	0	6	0	0	0	7
$x_5 =$	2	1	2	0	-1	1	0	0	-1
$x_6 =$	0	-1	2	0	-2	0	1	0	-2
$x_7 =$	2	0	4	0	-3	0	0	1	-3
$x_3 =$	$1/3$	0	0	1	$1/3$	0	0	0	$1/3$

- We now bring x_2 into the basis and x_6 exits.
- Note that this is a degenerate pivot with $\theta^* = 0$.
- The new tableau is:

Example 3.8

		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
	-4	0	-8	0	6	0	0	0	7
$x_5 =$	2	1	2	0	-1	1	0	0	-1
$x_6 =$	0	-1	2	0	-2	0	1	0	-2
$x_7 =$	2	0	4	0	-3	0	0	1	-3
$x_3 =$	$1/3$	0	0	1	$1/3$	0	0	0	$1/3$

- We now bring x_2 into the basis and x_6 exits.
- Note that this is a degenerate pivot with $\theta^* = 0$.
- The new tableau is:

Example 3.8

$$
\begin{array}{r|r|rrrr|rrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
\hline
 & -4 & -4 & 0 & 0 & -2 & 0 & 4 & 0 & -1 \\
\hline
x_5 = & 2 & 2 & 0 & 0 & 1 & 1 & -1 & 0 & 1 \\
x_2 = & 0 & -1/2 & 1 & 0 & -1 & 0 & 1/2 & 0 & -1 \\
x_7 = & 2 & 2 & 0 & 0 & 1 & 0 & -2 & 1 & 1 \\
x_3 = & 1/3 & 0 & 0 & 1 & 1/3 & 0 & 0 & 0 & 1/3 \\
\hline
\end{array}
$$

- We now have x_1 enter the basis and x_5 exit the basis.
- We obtain the following tableau:

Example 3.8

$$\begin{array}{r|r|rrrr|rrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
\hline
 & -4 & -4 & 0 & 0 & -2 & 0 & 4 & 0 & -1 \\
x_5 = & 2 & 2 & 0 & 0 & 1 & 1 & -1 & 0 & 1 \\
x_2 = & 0 & -1/2 & 1 & 0 & -1 & 0 & 1/2 & 0 & -1 \\
x_7 = & 2 & 2 & 0 & 0 & 1 & 0 & -2 & 1 & 1 \\
x_3 = & 1/3 & 0 & 0 & 1 & 1/3 & 0 & 0 & 0 & 1/3 \\
\hline
\end{array}$$

- We now have x_1 enter the basis and x_5 exit the basis.
- We obtain the following tableau:

Example 3.8

$$\begin{array}{r|r|rrrr|rrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ \hline
 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 0 & 1 \\ \hline
x_1 = & 1 & 1 & 0 & 0 & 1/2 & 1/2 & -1/2 & 0 & 1/2 \\
x_2 = & 1/2 & 0 & 1 & 0 & -3/4 & 1/4 & 1/4 & 0 & -3/4 \\
x_7 = & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 0 \\
x_3 = & 1/3 & 0 & 0 & 1 & 1/3 & 0 & 0 & 0 & 1/3 \\ \hline
\end{array}$$

- The cost in the auxiliary problem has dropped to zero: we have a feasible solution to the original problem.
- The artificial variable x_7 is still in the basis, at zero level.
- In order to obtain a basic feasible solution to the original problem and the corresponding basis, we need to drive x_7 out of the basis.

Example 3.8

$$\begin{array}{r|r|rrrr|rrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
\hline
 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 0 & 1 \\
\hline
x_1 = & 1 & 1 & 0 & 0 & 1/2 & 1/2 & -1/2 & 0 & 1/2 \\
x_2 = & 1/2 & 0 & 1 & 0 & -3/4 & 1/4 & 1/4 & 0 & -3/4 \\
x_7 = & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 0 \\
x_3 = & 1/3 & 0 & 0 & 1 & 1/3 & 0 & 0 & 0 & 1/3 \\
\hline
\end{array}$$

- x_7 is the third basic variable and the third row of $B^{-1}A$ is zero.
- This indicates that the matrix A has linearly dependent rows (Case 2).
- We remove the third row of the tableau, because it corresponds to a redundant constraint.
- The new tableau is:

Example 3.8

$$\begin{array}{r|r|rrrr|rrrr|}
 & & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
\hline
 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 0 & 1 \\
\hline
x_1 = & 1 & 1 & 0 & 0 & 1/2 & 1/2 & -1/2 & 0 & 1/2 \\
x_2 = & 1/2 & 0 & 1 & 0 & -3/4 & 1/4 & 1/4 & 0 & -3/4 \\
x_3 = & 1/3 & 0 & 0 & 1 & 1/3 & 0 & 0 & 0 & 1/3 \\
\hline
\end{array}$$

- There are no more artificial variables in the basis.
- Thus we can obtain an initial tableau for the original problem by removing all of the artificial variables.

Example 3.8

$$\begin{array}{r|r|rrrr|}
 & & x_1 & x_2 & x_3 & x_4 \\
\hline
 & 0 & 0 & 0 & 0 & 0 \\
\hline
x_1 = & 1 & 1 & 0 & 0 & 1/2 \\
x_2 = & 1/2 & 0 & 1 & 0 & -3/4 \\
x_3 = & 1/3 & 0 & 0 & 1 & 1/3 \\
\hline
\end{array}$$

- We compute the reduced costs of the original variables

$$\bar{c}' = c' - c_B' B^{-1} A.$$

- The original cost vector is $c = (1, 1, 1, 0)$, so $c_B = (1, 1, 1)$.
- The matrix $B^{-1}A$ is the tableau without the zeroth row and zeroth column.
- The vector of reduced costs is then $\bar{c} = (0, 0, 0, -1/12)$, and the cost of the solution $(1, 1/2, 1/3)$ is $11/6$.
- We fill in the zeroth row of the tableau and obtain:

Example 3.8

$$\begin{array}{r|r|rrrr|}
 & & x_1 & x_2 & x_3 & x_4 \\
\hline
 & -11/6 & 0 & 0 & 0 & -1/12 \\
\hline
x_1 = & 1 & 1 & 0 & 0 & 1/2 \\
x_2 = & 1/2 & 0 & 1 & 0 & -3/4 \\
x_3 = & 1/3 & 0 & 0 & 1 & 1/3 \\
\hline
\end{array}$$

- We can now start executing the simplex method on the original problem.
- Exercise: Do it!

Example 3.8

$$\begin{aligned}
\text{minimize} \quad & x_5 + x_6 + x_7 + x_8 \\
\text{subject to} \quad & x_1 + 2x_2 + 3x_3 + x_5 = 3 \\
& -x_1 + 2x_2 + 6x_3 + x_6 = 2 \\
& 4x_2 + 9x_3 + x_7 = 5 \\
& 3x_3 + x_4 + x_8 = 1 \\
& x_1, \ldots, x_4, x_5, \ldots, x_8 \geq 0.
\end{aligned}$$

- We observe that in this example, the artificial variable x_8 was unnecessary.
- Instead of starting with $x_8 = 1$, we could have started with $x_4 = 1$ thus eliminating the need for the first pivot.
- More generally, whenever there is a variable that appears in a single constraint and with a positive coefficient, we can always let that variable be in the initial basis and we do not have to associate an artificial variable with that constraint.

The two-phase simplex method

The two-phase simplex method

We can now summarize a complete algorithm for linear programming problems in standard form.

Phase I:

1. By multiplying some of the constraints by -1, change the problem so that $b \geq 0$.
2. Introduce artificial variables $y_1, \ldots, y_m$, if necessary, and apply the simplex method to the auxiliary problem with cost $\sum_{i=1}^{m} y_i$.
3. If the optimal cost in the auxiliary problem is positive, the original problem is infeasible and the algorithm terminates.

The two-phase simplex method

Phase I:

4. If the optimal cost in the auxiliary problem is zero, a feasible solution to the original problem has been found. If no artificial variable is in the final basis, the artificial variables and the corresponding columns are eliminated, and a feasible basis for the original problem is available.
5. If the ℓth basic variable is an artificial one, examine the ℓth entry of the columns $B^{-1}A_j$, $j = 1, \ldots, n$.
   - If all of these entries are zero, the ℓth row represents a redundant constraint and is eliminated.
   - Otherwise, if the ℓth entry of the jth column is nonzero, apply a change of basis (with this entry serving as the pivot element): the ℓth basic variable exits and x_j enters the basis.

   Repeat this operation until all artificial variables are driven out of the basis.

The two-phase simplex method

Phase II:

1. Let the final basis obtained from **Phase I** be the initial basis for **Phase II**.
2. Let the initial tableau for **Phase II** be obtained from the final tableau of **Phase I** by discarding the columns corresponding to the artificial variables and the zeroth row.
3. Compute the cost of the feasible solution and the reduced costs of all variables for this initial basis, using the cost coefficients of the original problem.
4. Apply the simplex method to the original problem.

The two-phase simplex method

- The two-phase simplex algorithm is a complete method, in the sense that it can handle all possible outcomes.
- As long as cycling is avoided (due to either nondegeneracy, an anticycling rule, or luck), one of the following possibilities will materialize:
  (a) If the problem is infeasible, this is detected at the end of **Phase I**.
  (b) If the problem is feasible but the rows of A are linearly dependent, this is detected and corrected at the end of **Phase I**, by eliminating redundant equality constraints.
  (c) If the optimal cost is equal to $-\infty$, this is detected while running **Phase II**.
  (d) Else, **Phase II** terminates with an optimal solution.

3.6 Column geometry and the simplex method

Column geometry and the simplex method

- We introduce an alternative way of visualizing the workings of the simplex method.
- We consider the problem

$$\begin{aligned} \text{minimize} \quad & c'x \\ \text{subject to} \quad & Ax = b \\ & \sum_{i=1}^{n} x_i = 1 \qquad \text{convexity constraint} \\ & x \geq 0. \end{aligned}$$

  where A is an $m \times n$ matrix.
- This is a special type of a linear programming problem.
- However, every linear programming problem with a bounded feasible set can be brought into this form. (Exercise 3.28.)

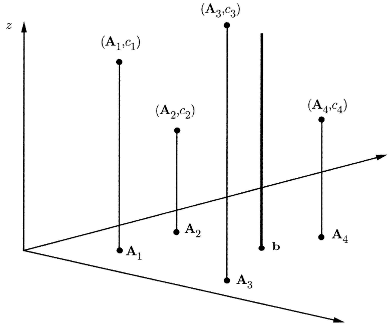
Column geometry and the simplex method

$$\begin{aligned} \text{minimize} \quad & c'x \\ \text{subject to} \quad & Ax = b \\ & \sum_{i=1}^{n} x_i = 1 \\ & x \geq 0. \end{aligned}$$

- We introduce an auxiliary variable z defined by $z = c'x$.
- Our problem can then be written in the form

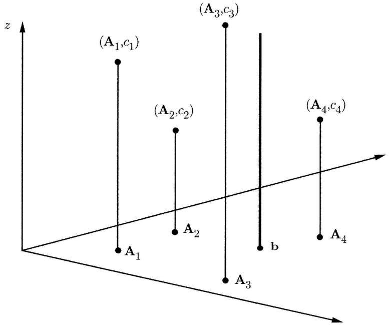
$$\begin{aligned} \text{minimize} \quad & z \\ \text{subject to} \quad & x_1 \begin{bmatrix} A_1 \\ c_1 \end{bmatrix} + x_2 \begin{bmatrix} A_2 \\ c_2 \end{bmatrix} + \cdots + x_n \begin{bmatrix} A_n \\ c_n \end{bmatrix} = \begin{bmatrix} b \\ z \end{bmatrix} \\ & \sum_{i=1}^{n} x_i = 1 \\ & x \geq 0. \end{aligned}$$

Column geometry and the simplex method



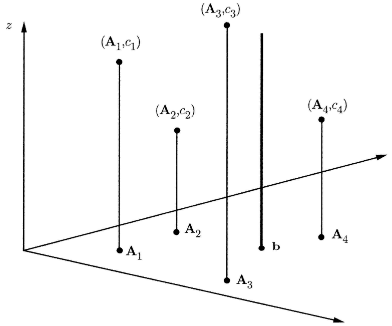
- We view the horizontal plane as an m-dimensional space containing the columns of A.
- We view the vertical axis as the one-dimensional space associated with the cost components c_i.
- Then, each point in the resulting three-dimensional space corresponds to a point (A_i, c_i).

Column geometry and the simplex method



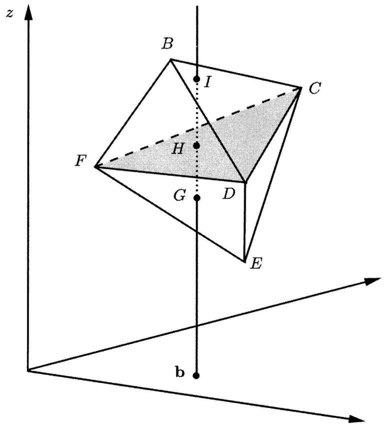
- Our objective is to construct a vector (b, z), which is a convex combination of the vectors (A_i, c_i), such that z is as small as possible.
- Note that the vectors of the form (b, z) lie on a vertical line, which we call the requirement line, and which intersects the horizontal plane at b.

Column geometry and the simplex method



- If the requirement line does not intersect the convex hull of the points (A_i, c_i), the problem is infeasible.
- If it does intersect it, the problem is feasible and an optimal solution corresponds to the lowest point in the intersection of the convex hull and the requirement line.

Column geometry and the simplex method



Example:

- The requirement line intersects the convex hull of the points (A_i, c_i).
- The point G corresponds to an optimal solution.
- The height of G is the optimal cost.

Column geometry and the simplex method

Definition 3.6

(a) A collection of vectors

$$y^1,\ y^2 \ldots,\ y^{k+1} \in \mathbb{R}^n$$

are said to be <u>affinely independent</u> if the vectors

$$y^1 - y^{k+1},\ y^2 - y^{k+1}, \ldots,\ y^k - y^{k+1}$$

are linearly independent. (Note that we must have $k \leq n$.)

(b) The convex hull of $k+1$ affinely independent vectors in $\mathbb{R}^n$ is called a k-dimensional <u>simplex</u>.

- Three points are either collinear or they are affinely independent and determine a two-dimensional simplex (a triangle).

Column geometry and the simplex method

Definition 3.6

(a) A collection of vectors

$$y^1,\ y^2 \dots,\ y^{k+1} \in \mathbb{R}^n$$

are said to be affinely independent if the vectors

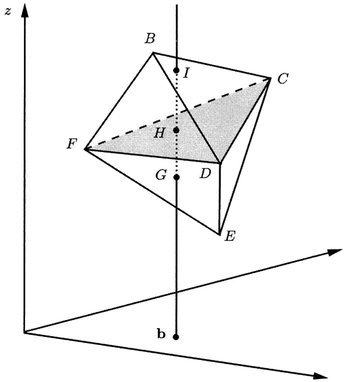
$$y^1 - y^{k+1},\ y^2 - y^{k+1}, \dots,\ y^k - y^{k+1}$$

are linearly independent. (Note that we must have $k \leq n$.)

(b) The convex hull of $k+1$ affinely independent vectors in $\mathbb{R}^n$ is called a k-dimensional simplex.

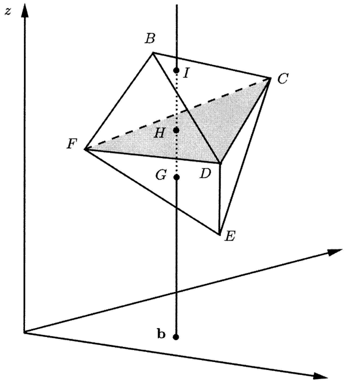
- Four points either lie on the same plane, or they are affinely independent and determine a three-dimensional simplex (a pyramid).

Column geometry and the simplex method



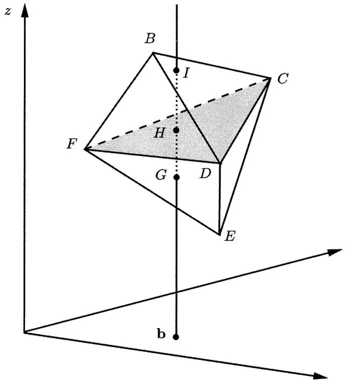
- We now give an interpretation of basic feasible solutions to our problem in this geometry.

Column geometry and the simplex method



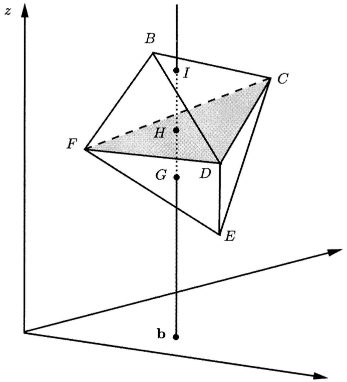
- In our original problem we have $m+1$ equality constraints.
- Thus, a basic feasible solution is associated with a collection of $m+1$ linearly independent columns $(A_i, 1)$ of our linear programming problem.

Column geometry and the simplex method



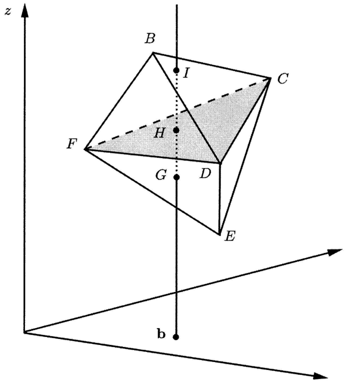
- These are in turn associated with $m + 1$ points (A_i, c_i), which we call basic points; the remaining points (A_i, c_i) are called the nonbasic points.
- Example: A possible choice of basic points is C, D, F.

Column geometry and the simplex method



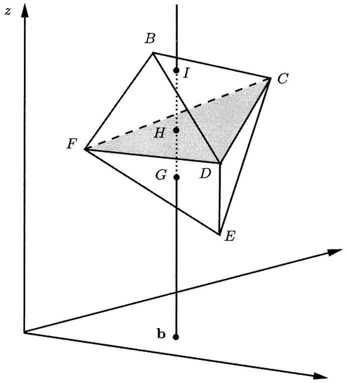
- The $m+1$ basic points are affinely independent (Exercise 3.29) and, therefore, their convex hull is an m-dimensional simplex, which we call the basic simplex.
- Example: The shaded triangle CDF is the basic simplex associated with the basic points C, D, F.

Column geometry and the simplex method



- Let the requirement line intersect the m-dimensional basic simplex at some point (b, z).
- The vector of weights x_i used in expressing (b, z) as a convex combination of the basic points, is the current basic feasible solution, and z represents its cost.

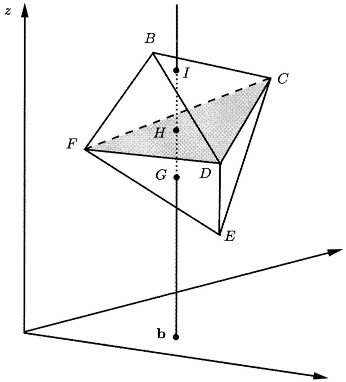
Column geometry and the simplex method



Example:

- The point H corresponds to the basic feasible solution associated with the basic points C, D, F.
- The height of H is the cost of this solution.

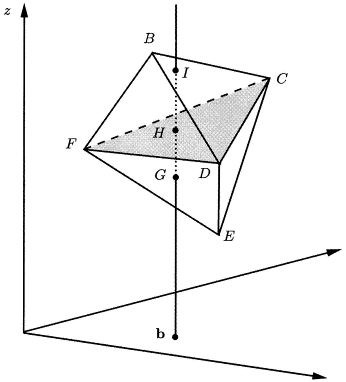
Column geometry and the simplex method



We now interpret a change of basis geometrically.
In a change of basis:

- A new point (A_j, c_j) becomes basic;
- One of the currently basic points is to become nonbasic.

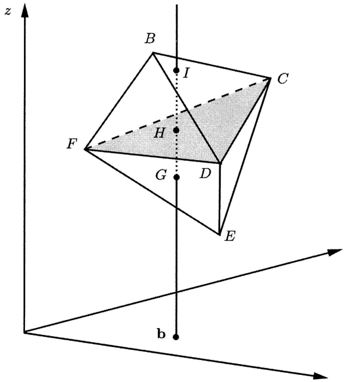
Column geometry and the simplex method



Example:

- Let C, D, F, be the current basic points,
- We could make point B basic, replacing F.
- The new basic simplex would be the convex hull of B, C, D.
- The new basic feasible solution would correspond to point I.

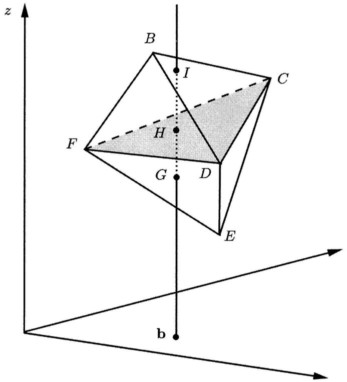
Column geometry and the simplex method



Example:

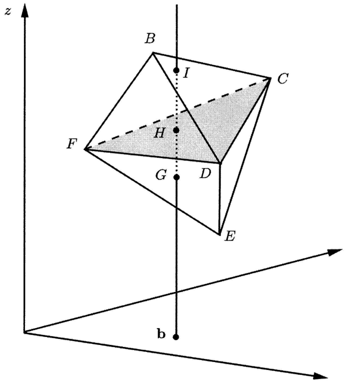
- Alternatively, we could make point E basic, replacing C.
- The new basic simplex would be the convex hull of D, E, F.
- The new basic feasible solution would correspond to point G.

Column geometry and the simplex method



- The plane that passes through the basic points is called the dual plane.
- After a change of basis, the cost decreases, if and only if the new basic point is below the dual plane.

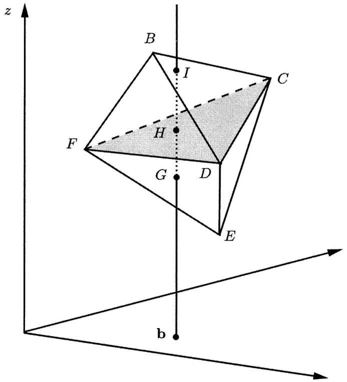
Column geometry and the simplex method



Example:

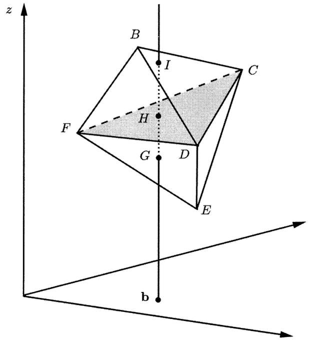
- Point E is below the dual plane and having it enter the basis is profitable.
- This is not the case for point B.

Column geometry and the simplex method



- In fact, the vertical distance from the dual plane to a point (A_j, C_j) is equal to the reduced cost of the associated variable x_j. (Exercise 3.30.)
- Requiring the new basic point to be below the dual plane is therefore equivalent to requiring the entering column to have negative reduced cost.

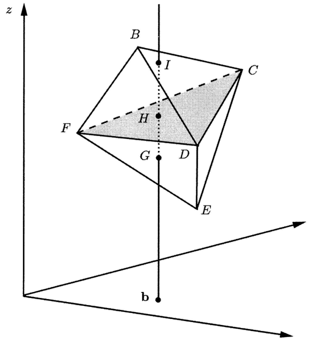
Column geometry and the simplex method



We discuss next the selection of the basic point that will exit the basis.

- Each possible choice of the exiting point leads to a different basic simplex.
- These m basic simplices, together with the original basic simplex (before the change of basis) form the boundary (the faces) of an $(m+1)$-dimensional simplex.

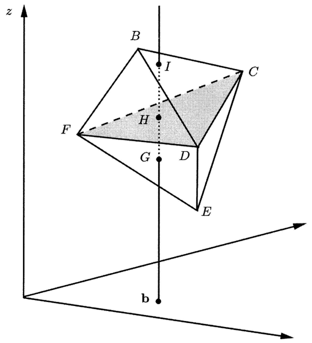
Column geometry and the simplex method



Example:

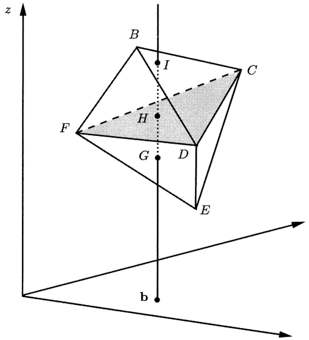
- The basic points C, D, F, determine a two-dimensional basic simplex.
- If point E is to become basic, we obtain a three-dimensional simplex (pyramid) with vertices C, D, E, F.

Column geometry and the simplex method



- The requirement line exits this $(m + 1)$-dimensional simplex through its top face and must therefore enter it by crossing some other face.
- This determines which one of the potential basic simplices will be obtained after the change of basis.

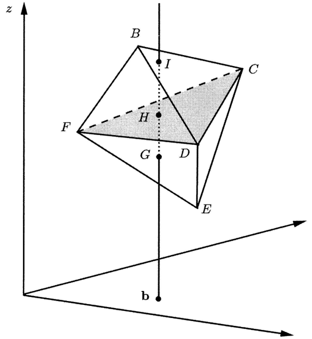
Column geometry and the simplex method



Example:

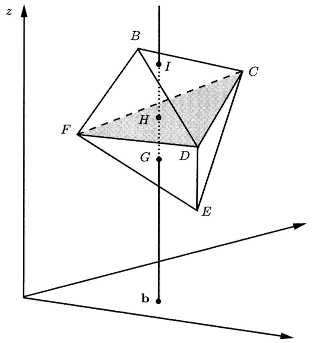
- The requirement line exits the pyramid through its top face with vertices C, D, F.
- It enters the pyramid through the face with vertices D, E, F.
- D, E, F is the new basic simplex and C exits the basis.

Column geometry and the simplex method



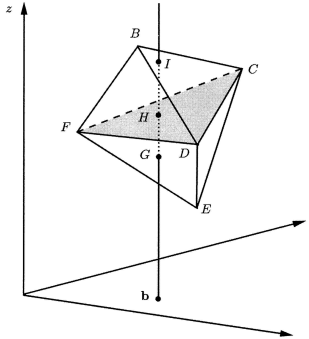
- We can now visualize pivoting through the following physical analogy.
- Think of the original basic simplex with vertices C, D, F, as a solid object anchored at its vertices.

Column geometry and the simplex method



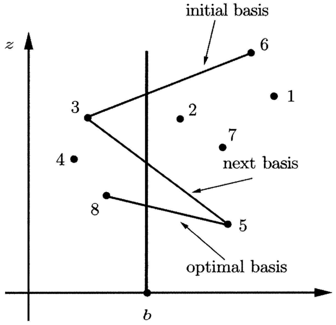
- Grasp the corner of the basic simplex at the vertex C leaving the basis, and pull the corner down to the new basic point E.
- While so moving, the simplex will hinge, or pivot, on its anchor and stretch down to the lower position.

Column geometry and the simplex method



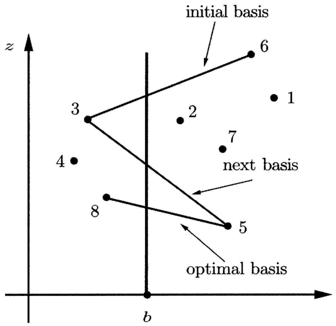
- The terms "simplex" and "pivot" associated with the simplex method have their roots in this column geometry.

Example 3.10



- In this problem we have $m = 1$.
- We use the following pivoting rule: choose a point (A_i, c_i) below the dual plane to become basic, whose vertical distance from the dual plane is largest.
- Exercise 3.30: this is identical to the pivoting rule that selects an entering variable with the most negative reduced cost.

Example 3.10



- Initial basic simplex: $3, 6$.
- Next basic simplex: $3, 5$.
- Next basic simplex: $5, 8$.

3.7 Computational efficiency of the simplex method

Computational efficiency of the simplex method

The computational efficiency of the simplex method is determined by two factors:

(a) The computational effort at each iteration.

(b) The number of iterations.

Computational efficiency of the simplex method

- The computational requirements of each iteration have already been discussed in Section 3.3.
- For example, the full tableau implementation needs $O(mn)$ arithmetic operations per iteration.
- The same is true for the revised simplex method in the worst case.
- We now turn to a discussion of the number of iterations.

The number of iterations in the worst case

The number of iterations in the worst case

- The number of extreme points of the feasible set can increase exponentially with the number of variables and constraints.
- However, it has been observed in practice that the simplex method typically takes only O(m) pivots to find an optimal solution.
- Unfortunately, however, this practical observation is not true for every linear programming problem.
- We will describe shortly a family of problems for which an exponential number of pivots may be required.

The number of iterations in the worst case

- Recall that for nondegenerate problems, the simplex method always moves from one vertex to an adjacent one, each time improving the value of the cost function.
- We will now describe a polyhedron that has an exponential number of vertices, along with a path that visits all vertices, by taking steps from one vertex to an adjacent one that has lower cost.
- Once such a polyhedron is available, then the simplex method – under a pivoting rule that traces this path – needs an exponential number of pivots.

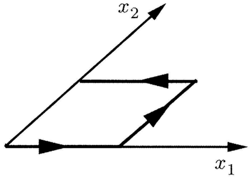
The number of iterations in the worst case

- Consider the unit cube in $\mathbb{R}^n$, defined by the constraints

$$0 \leq x_i \leq 1, \qquad i = 1, \ldots, n.$$

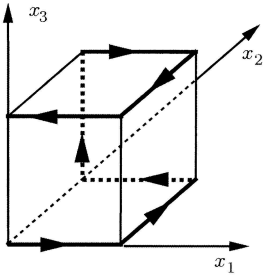
- The unit cube has 2^n vertices: all binary vectors.
- Furthermore, there exists a path that travels along the edges of the cube and which visits each vertex exactly once; we call such a path a <u>spanning path</u>.
- Let's see how a spanning path can be constructed.

The number of iterations in the worst case



This is a spanning path p_2 in the two-dimensional cube.

The number of iterations in the worst case



A spanning path p_3 in the three-dimensional cube can be obtained as follows:

- Split the three-dimensional cube into two two-dimensional cubes (one in $x_3 = 0$ and one in $x_3 = 1$).
- Follow path p_2 in one of them.
- Move to the other cube and follow p_2 in the reverse order.

This construction generalizes and provides a recursive definition of a spanning path for the general n-dimensional cube.

The number of iterations in the worst case

Let us now introduce the cost function $-x_n$.

- Half of the vertices of the cube have zero cost and the other half have a cost of -1.
- Thus, the cost does not decrease strictly with each move along the spanning path.

We do not yet have the desired example!

The number of iterations in the worst case

- However, we choose some $\epsilon \in (0, 1/2)$ and consider the perturbation of the unit cube defined by the constraints

$$\begin{aligned} \epsilon \leq x_1 \leq 1, & \\ \epsilon x_{i-1} \leq x_i \leq 1 - \epsilon x_{i-1}, & \qquad i = 2, \ldots, n. \end{aligned}$$

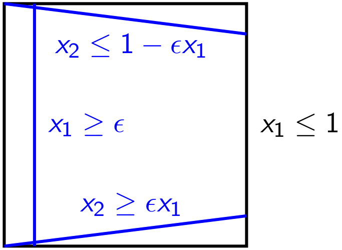
- Then it can be verified that the cost function decreases strictly with each move along a suitably chosen spanning path. Which one?

The number of iterations in the worst case

- However, we choose some $\epsilon \in (0, 1/2)$ and consider the perturbation of the unit cube defined by the constraints

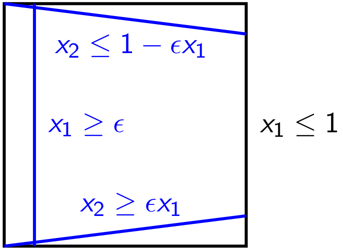
$$\epsilon \leq x_1 \leq 1,$$
$$\epsilon x_{i-1} \leq x_i \leq 1 - \epsilon x_{i-1}, \qquad i = 2, \ldots, n.$$

- Then it can be verified that the cost function decreases strictly with each move along a suitably chosen spanning path. Which one?



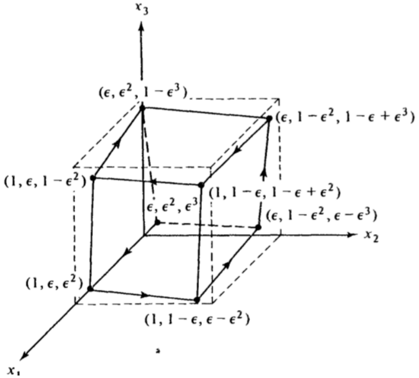
The number of iterations in the worst case

- If we start the simplex method at the first vertex on that spanning path and if our pivoting rule is to always move to the next vertex on that path, then the simplex method will require $2^n - 1$ pivots.
- We summarize this discussion in the following theorem.



The number of iterations in the worst case

- If we start the simplex method at the first vertex on that spanning path and if our pivoting rule is to always move to the next vertex on that path, then the simplex method will require $2^n - 1$ pivots.
- We summarize this discussion in the following theorem.



x_3
$(\epsilon, \epsilon^2, 1-\epsilon^3)$
$(\epsilon, 1-\epsilon^2, 1-\epsilon+\epsilon^3)$
$(1, \epsilon, 1-\epsilon^2)$
$(1, 1-\epsilon, 1-\epsilon+\epsilon^2)$
$\epsilon, \epsilon^2, \epsilon^3$
$(\epsilon, 1-\epsilon^2, \epsilon-\epsilon^3)$
x_2
$(1, \epsilon, \epsilon^2)$
$(1, 1-\epsilon, \epsilon-\epsilon^2)$
x_1

The number of iterations in the worst case

Theorem 3.5

Consider the linear programming problem of minimizing $-x_n$ subject to the constraints

$$\begin{aligned} \epsilon \le x_1 &\le 1, \\ \epsilon x_{i-1} \le x_i &\le 1 - \epsilon x_{i-1}, \qquad i = 2, \ldots, n. \end{aligned}$$

Then:

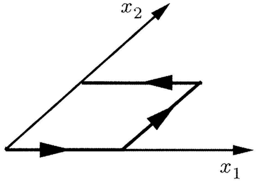
(a) The feasible set has 2^n vertices.

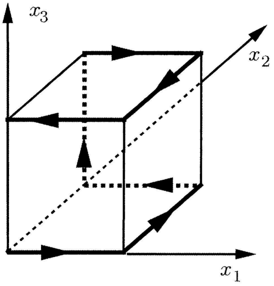
(b) The vertices can be ordered so that each one is adjacent to and has lower cost than the previous one.

(c) There exists a pivoting rule under which the simplex method requires $2^n - 1$ changes of basis before it terminates.

Proof: Exercise 3.32.

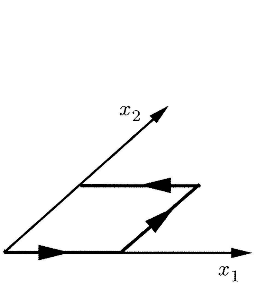
The number of iterations in the worst case

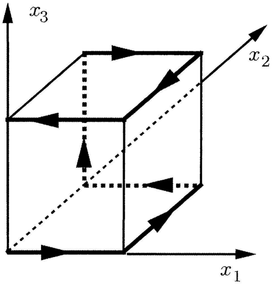




- We observe in the figure the first and the last vertex in the spanning path are adjacent.
- This property persists in the perturbed polyhedron as well.
- Thus, with a different pivoting rule, the simplex method could terminate with a single pivot.

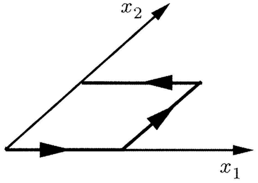
The number of iterations in the worst case

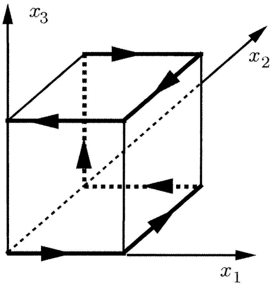




- We are thus led to the following question: is it true that for every pivoting rule there are examples where the simplex method takes an exponential number of iterations?
- For several popular pivoting rules, such examples have been constructed.

The number of iterations in the worst case



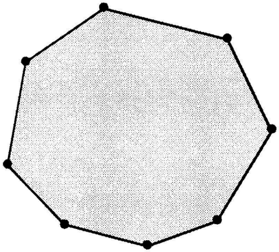
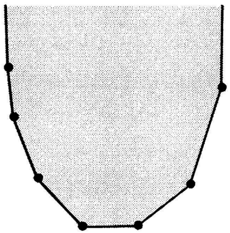


- However, these examples cannot exclude the possibility that some other pivoting rule might fare better.
- This is one of the most important open problems in the theory of linear programming.
- In the next subsection, we address a closely related issue.

The diameter of polyhedra and the Hirsch conjecture

The diameter of polyhedra and the Hirsch conjecture

- The preceding discussion leads us to the notion of the diameter of a polyhedron P, which is defined as follows.
- Suppose that from any vertex of the polyhedron, we are only allowed to jump to an adjacent vertex.
- We define the $\underline{\text{distance } d(x, y)}$ between two vertices x and y as the minimum number of such jumps required to reach y starting from x.

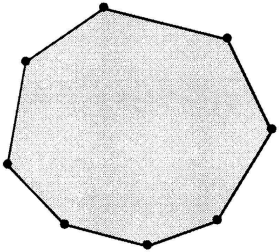



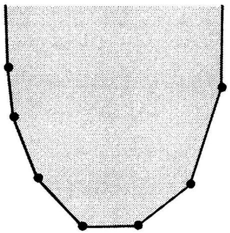
The diameter of polyhedra and the Hirsch conjecture

- The diameter $D(P)$ of the polyhedron P is then defined as the maximum of $d(x, y)$ over all pairs (x, y) of vertices.
- For example, any polyhedron P in $\mathbb{R}^2$ that is represented in terms of m linear inequality constraints is such that:

$D(P) \leq \left\lfloor \frac{m}{2} \right\rfloor$
if P is bounded

$D(P) \leq m - 2$
if P is unbounded





The diameter of polyhedra and the Hirsch conjecture

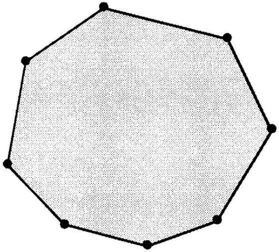
- Suppose that the feasible set P in a linear programming problem has diameter d.
- Let x and y be two vertices of P such that the distance between x and y is equal to d.
- If the simplex method is initialized at x, and if y happens to be the unique optimal solution, then at least d steps will be required.

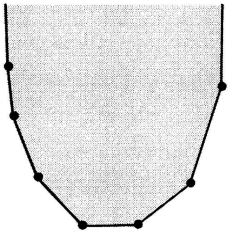
The diameter of polyhedra and the Hirsch conjecture

- We define $\underline{\Delta(n, m)}$ as the maximum of $D(P)$ over all polyhedra in $\mathbb{R}^n$ that are represented in terms of m linear inequality constraints.

$D(P) \leq \left\lfloor \frac{m}{2} \right\rfloor$
if P is bounded

$D(P) \leq m - 2$
if P is unbounded





- Therefore, $\Delta(2, m) = m - 2$.

The diameter of polyhedra and the Hirsch conjecture

- Now, if $\Delta(n, m)$ increases exponentially with n and m, this implies that there exist examples for which the simplex method takes an exponentially increasing number of steps, no matter which pivoting rule is used.
- Thus, in order to have any hope of developing pivoting rules under which the simplex method requires a polynomial number of iterations, we must first establish that $\Delta(n, m)$ grows with n and m at the rate of some polynomial.

The diameter of polyhedra and the Hirsch conjecture

- The practical success of the simplex method has led to the conjecture that $\Delta(n, m)$ does not grow exponentially fast.
- In fact, the following, much stronger, conjecture has been advanced:

Hirsch Conjecture (1957)
$\Delta(n, m) \leq m - n$.

The diameter of polyhedra and the Hirsch conjecture

- The practical success of the simplex method has led to the conjecture that $\Delta(n, m)$ does not grow exponentially fast.
- In fact, the following, much stronger, conjecture has been advanced:

Hirsch Conjecture (1957)
$\Delta(n, m) \leq m - n$.

Bad news: The Hirsch conjecture is false.

- Klee and Walkup, 1967. (Due to unbounded polyhedra.)
- Santos, 2010. (Only considering polytopes.)

The diameter of polyhedra and the Hirsch conjecture

- Even though the Hirsch conjecture is false, we do not know whether the growth of $\Delta(n, m)$ is polynomial or exponential.
- The following (weaker) conjecture has been advanced:

Polynomial Hirsch Conjecture
$\Delta(n, m)$ is bounded above by a polynomial of n and m.

- Despite the significance of $\Delta(n, m)$, we are far from establishing the polynomial Hirsch conjecture.

The diameter of polyhedra and the Hirsch conjecture

- Even though the Hirsch conjecture is false, we do not know whether the growth of $\Delta(n, m)$ is polynomial or exponential.
- The following (weaker) conjecture has been advanced:

Polynomial Hirsch Conjecture
$\Delta(n, m)$ is bounded above by a polynomial of n and m.

- Despite the significance of $\Delta(n, m)$, we are far from establishing the polynomial Hirsch conjecture.

- Question: If the polynomial Hirsch conjecture is true, then is the simplex method a polynomial-time algorithm?

The diameter of polyhedra and the Hirsch conjecture

- Regarding upper bounds, it has been established that the worst-case diameter grows slower than exponentially.
- But the available upper bound grows faster than any polynomial.
- In particular, the following bound is known:

$$\Delta(n, m) \leq m^{1+\log_2 n} = (2n)^{\log_2 m}.$$

The average case behavior of the simplex method

The average case behavior of the simplex method

- Our discussion has been focused on the worst-case behavior of the simplex method, but this is only part of the story.
- Even if every pivoting rule requires an exponential number of iterations in the worst case, this is not necessarily relevant to the typical behavior of the simplex method.
- For this reason, there has been a fair amount of research aiming at an understanding of the average behavior of the simplex method.

The average case behavior of the simplex method

- On average $O(n)$ iterations seem to suffice.
- This has been supported by so-called probabilistic analysis, though it is very hard to come up with a representative probabilistic model for random feasible LP-instances.
- The smoothed analysis of the simplex method shows that bad instances are very non-dense in the set of all possible instances, since tiny random perturbations of the coefficients gives a polynomial number of iterations in expectation.

The average case behavior of the simplex method

- The main difficulty in studying the average behavior of any algorithm lies in defining the meaning of the term "average."
- Basically, one needs to:
  1. Define a probability distribution over the set of all problems of a given size.
  2. Take the mathematical expectation of the number of iterations required by the algorithm, when applied to a random problem drawn according to the postulated probability distribution.
- Unfortunately, there is no natural probability distribution over the set of linear programming problems.
- Nevertheless, a fair number of positive results have been obtained for a few different types of probability distributions.

The average case behavior of the simplex method

- In one such result, a set of vectors $c, a_1, \ldots, a_m \in \mathbb{R}^n$ and scalars $b_1, \ldots, b_m$ is given.
- For $i = 1, \ldots, m$, we introduce either constraint

$$a_i' x \leq b_i \qquad \text{or} \qquad a_i' x \geq b_i,$$

  with equal probability.
- We then have 2^m possible linear programming problems, and suppose that L of them are feasible.
- Haimovich (1983) has established that under a rather special pivoting rule, the simplex method requires no more than $n/2$ iterations, on the average over those L feasible problems.
- This linear dependence on the size of the problem agrees with observed behavior.