ZewailCity Of Science And Technology

Linear and Non-linear Programming

Math 404

Revised Simplex Method

Hajar Fawzi 201900537

Contents

1 Motivation

Linear Programming Problems (LPP) are encountered in many domains where optimal resource distribution is crucial. The original simplex method has some limitations in solving the LPP which motivated the development of new techniques with better features. One of those techniques is the Revised Simplex Method. The limitations of the original simplex method were seen in solving large problems with many variables and constraints, where it requires a large amount of computation time and computer storage. The limitations of the original simplex method were not only seen in the time and storage but also degeneracy(the solutions are located on the edges of the feasible region) and cycling(revisiting the same points without change or progress). The problem with the degeneracy and cycling is that degeneracy could cause an inefficient way to get the optimal solution, cycling can stop the algorithm from converging. The revised simplex method is a developed version of the original simplex method which aims to overcome these limitations.

2 Theory

The modifications done on the original simplex method were summarized in two main points: improving pivot rules and modifying only the elements of the inverted matrices not the elements of the whole simplex table as the original simplex method. The theory behind the revised simplex method includes some concepts such as the duality of the LPP and the feasible regions' geometry. The first rule to start using the revised simplex is the linear programming duality. The duality theorem states that the optimal solution for the dual problem is greater than or equal to the optimal solution for the primal problem. To show the duality consider the following LPP :

$$Maximize : z = c^T x \tag{1}$$

$$Subject.to : Ax = b, x \geq 0 \tag{2}$$

Applying the duality:

$$Minimize : f = b^T x \tag{3}$$

$$Subject.to : A^T y \geq c,\ y \geq 0 \tag{4}$$

The revised simplex method moves from a basic(non-zero) feasible point to another where these points are the edges of the feasible region. The main modification in to reach the revised simplex method is the pivot operations where it selects a leaving and an entering variable to apply the operation while ensuring feasibility and optimality. This method also includes an optimality test that is applied by reducing the costs in both problems(Duality and Primal). An iterative process starts till racing the optimal solution and this happens when all costs are non-negative. The revised simplex method handles the degeneracy and cycling process through some applied strategies.

3 Algorithm

As stated before, The revised simplex method has criteria to work including initialization, the optimality test, selecting the variables, the pivot operations, the feasibility test, repetition and finally Termination. The objective function $z =< c^T, x >$ in the revised simplex method is treated as a constraint, where z is treated as an unrestricted variable. The revised simplex method has two standard forms: standard form I and standard form II. The first form depends on slack and surplus variables for the initial identity matrix causing the identity matrix as the initial matrix. The second form depends also on the artificial variables for the initial identity matrix causing the usage of the two-phase method. Standard Form I: Our LPP is as follows:

$$Maximize : z = c^T x \tag{5}$$

$$Subject.to : Ax = b, x \geq 0 \tag{6}$$

Z is the objective function, c^T, x belongs to R^n and b belongs to R^m A dimension is m x n Now we have n+1 variables $(x_1, x_2, \ldots, z)$ and m+1 equations (Constraints and the objective function). Let $x_B = B^-1 * b$ the basic feasible solution to start with. Where B is starting basis submatrix of A. Now the new system of equations would be :

$$Ax + Oz = b \tag{7}$$

$$-c^T x + z = 0 \tag{8}$$

Where $x \geq 0$, z is unrestricted variable Reformulate the equations using B and x_B to get

$$Bx_B + Oz = b \tag{9}$$

$$-c^T x + z = 0 \tag{10}$$

Assume : $\hat{B} = \begin{pmatrix} B & O \\ -c_B & 1 \end{pmatrix}$, $\hat{x_B} = \begin{pmatrix} x_B \\ z \end{pmatrix}$ and $\hat{b} = \begin{pmatrix} b \\ 0 \end{pmatrix}$, So $\hat{B}\hat{x_B} = \hat{b}$ if x_B is the basic feasible solution of the LPP given, $\hat{x_B}$ is a also basic feasible solution. Now assume that $\hat{B}^-1 = \begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix}$ as known in Linear algebra when a matrix is multiplied by its inverse the answer is the 1 or identity matrix(if they are matrices of matrices). This gives

$$\hat{B}^-1\hat{B} = \begin{pmatrix} I & O \\ O & 1 \end{pmatrix} \tag{11}$$

now substitute in 11 with the matrices B and $\hat{B}^-1$ to get the following:

$$\begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix} \begin{pmatrix} B & O \\ -c_B & 1 \end{pmatrix} = \begin{pmatrix} I & O \\ O & 1 \end{pmatrix} \tag{12}$$

By multiplying the matrices we get the following equations:

$$A_1 B - A_2 c_B = I \tag{13}$$

$$A_1 * O - A_2 * 1 = O \tag{14}$$

$\therefore A_2 = O$

$$A_3 B - A_4 c_B = O \tag{15}$$

$$A_3 * O - A_4 * 1 = 1 \tag{16}$$

$\therefore A_4 = 1$

By substituting with both $A_4 = 1$ and $A_2 = O$ in equations 13 and 15 we get :

$$A_1 = B^-1 \tag{17}$$

$$A_3 = c_B B^-1 \tag{18}$$

Now we replace the different A values in the matrix of $\hat{B}^-1$:

$$\therefore \hat{B}^-1 = \begin{pmatrix} B^-1 & O \\ c_B B^-1 & 1 \end{pmatrix} \tag{19}$$

Assume we have $\hat{A} = \begin{pmatrix} A \\ -c \end{pmatrix}$ and $\hat{y} = \hat{B}^{-}1\hat{A}$ substitute with both $\hat{B}^{-}1\hat{A}$ as we already got them :

$$\therefore \hat{y} = \begin{pmatrix} B^{-}1 & O \\ c_B B^{-}1 & 1 \end{pmatrix} \begin{pmatrix} A \\ -c \end{pmatrix} \tag{20}$$

$$= \begin{pmatrix} B^{-}1A \\ c_B B^{-}1A - c \end{pmatrix} \tag{21}$$

Substituting with $y = B^{-}1A$ we get

$$\hat{y} = \begin{pmatrix} y \\ c_B y - c \end{pmatrix} \tag{22}$$

The dimensions of c_B is 1*m while that of y is (m+1)*n and c is 1*n, now we need another expression for $c_B y - c$ so let us substitute according to the given dimensions:

$$c_B y - c = (c_{B1}, c_{B2}, c_{B3}, ..., c_{Bm}) \begin{pmatrix} y_{11} & y_{12} & y_{13} & ... & y_{1n} \\ y_2 1 & y_2 2 & y_2 3 & ... & y_2 n \\ y_3 1 & y_3 2 & y_3 3 & ... & y_3 n \\ ... & ... & ... & ... & .... \\ y_m 1 & y_m 2 & y_m 3 & ... & y_m n \end{pmatrix} - (c_1, c_2, c_3, ..., c_n) \tag{23}$$

lets name $\sum c_{Bi} y_{i1}$ as z_1, $\sum c_{Bi} y_{i2}$ as z_2 where $i = (1, 2, 3, ..., m)$ and so on.

now we get

$$c_B y - c = (z_1 - c_1, z_2 - c_2, z_3 - c_3, ..., z_n - c_n) \tag{24}$$

substitute with 24 in 22 to get the new expression of $\hat{y}$

$$\hat{y} = \begin{pmatrix} y_1 & y_2 & y_3 & \cdots & y_n \\ z_1 - c_1 & z_2 - c_2 & z_3 - c_3 & ... & z_n - c_n \end{pmatrix} \tag{25}$$

As we know before $c_B B^{-1} A - c = \begin{pmatrix} c_B B^{-1} & 1 \end{pmatrix} \begin{pmatrix} A \\ -c \end{pmatrix}$ using 21 and 22 we've found that $c_B B^{-}1A - c = c_B y - c$ which we found that it also equals $z_j - c_j$ where $j = (1, 2, 3, ..., n)$

$$\therefore z_j - c_j = \begin{pmatrix} c_B B^{-1} & 1 \end{pmatrix} \begin{pmatrix} A \\ -c \end{pmatrix} \tag{26}$$

4 Procedures of solving LPP using Revised simplex

- Reformulate the problem using slack and surplus variables (canonical form) and make sure it is in the maximization form.
- Start the iterations using slack and surplus,-z as basic variables
- Start with $B = I_m$ as the Basic feasible solution $\hat{B} = \begin{pmatrix} B & O \\ -c_B & 1 \end{pmatrix}$
- Get the inverse using our formulas: $\hat{B}^-1 = \begin{pmatrix} B^-1 & O \\ c_B B^-1 & 1 \end{pmatrix}$
- Get $\hat{A}$ from $\hat{A} = \begin{pmatrix} A \\ -c \end{pmatrix}$
- Get $\hat{b}$ from $\hat{b} = \begin{pmatrix} b \\ 0 \end{pmatrix}$
- Get the net evaluations $z_j - c_j$ where $j = (1, 2, 3, ..., n)$ $z_j - c_j = \begin{pmatrix} c_B B^{-1} & 1 \end{pmatrix} \begin{pmatrix} A \\ -c \end{pmatrix}$
- check the signs of all $z_j - c_j$, if their values are non-negative so the solution is optimum.
- if there is a negative value, determine the most negative one of $z_j - c_j$ and select the corresponding $\hat{y}$ to be the entering variable.
- Get the least non negative ratio $\hat{x_B}$ and selected $\hat{y}$ to get the pivot
- if all $\hat{y}_i \leq 0$ the optimum solution is unbounded
- start combining results to form the table of the revised simplex
- Start the pivot operations to update B^-1
- update $\hat{x_B} = \hat{B}^-1\hat{b}$
- Keep iterating till reaching the optimum solution.

5 Application

5.1 Example 1

Example

Maximize the objective function:

$$\text{Max } Z = 3X_1 + 5X_2$$

subject to the constraints:

$$\begin{aligned} X_1 &\leq 4 \\ 2X_2 &\leq 12 \\ 3X_1 + 2X_2 &\leq 18 \\ X_1, X_2 &\geq 0 \end{aligned}$$

Standard form of constraints:

$$\begin{aligned} X_1 + S_1 &= 4 \\ 2X_2 + S_2 &= 12 \\ 3X_1 + 2X_2 + S_3 &= 18 \\ X_1, X_2, S_1, S_2, S_3 &\geq 0 \end{aligned}$$

The basic variables $\mathbf{x}_B$ are calculated as:

$$\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix} = \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix}$$

The cost associated with the basic variables $\mathbf{c}_B$ is:

$$\mathbf{c}_B = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

The value of the objective function Z is:

$$Z = \mathbf{c}_B^T \mathbf{x}_B = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix} = 0$$

First Iteration

Step 1

Determine the entering variable, X_j, with the associated vector P_j.

- Compute $Y = c_B B^{-1}$
- Compute $Z_j - c_j = YP_j - c_j$ for all non-basic variables.

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

$$Z_1 - c_1 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix} - 3 = -3$$

and similarly for $Z_2 - c_2 = -5$

Therefore, X_2 is the entering variable.

Step 2

Determine the leaving variable, X_r, with the associated vector P_r.

- Compute $\mathbf{x}_B = B^{-1}\mathbf{b}$ (current R.H.S.)
- Compute current constraint coefficients of entering variable: $\mathbf{q}' = B^{-1}P_j$

X_r is associated with

$$\theta = \min \left\{ \frac{\mathbf{x}_B}{\mathbf{q}'}, \mathbf{q}' > 0 \right\}$$

$$\mathbf{x}_B = \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix}, \quad \mathbf{q}' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix}$$

$$\theta = \min \left\{ -, \frac{12}{2}, \frac{18}{2} \right\} = \frac{12}{2}$$

S_2 leaves Determine the new B^{-1}.

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 1 \end{bmatrix}, \quad B^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & -1 & 1 \end{bmatrix}$$

Solution after one iteration:

$$\mathbf{x}_B = B^{-1}\mathbf{b} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \\ 6 \end{bmatrix}$$

Proceed to Step 1 of the second iteration.

Second Iteration

Step 1

Compute $Y = c_B B^{-1}$

$$Y = \begin{bmatrix} 0 & 5 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & \frac{5}{2} & 0 \end{bmatrix}$$

Compute $Z_j - c_j = YP_j - c_j$ for all non-basic variables (X_1 and S_2):

$$X_1 : Z_1 - c_1 = \begin{bmatrix} 0 & \frac{5}{2} & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix} - 3 = -3$$

$$S_2 : Z_4 - c_4 = \begin{bmatrix} 0 & \frac{5}{2} & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} - 0 = \frac{5}{2}$$

Therefore, X_1 enters the basis.

Step 2

Determine the leaving variable.

$$\mathbf{x}_B = \begin{bmatrix} 4 \\ 6 \\ 6 \end{bmatrix}, \quad \mathbf{q}' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix}$$

$$\theta = \min \left\{ \frac{4}{1}, -, \frac{6}{3} \right\} = \frac{6}{3}$$

Therefore, S_3 leaves.

Step 3

Determine new B^{-1}:

$$B = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 0 & 2 & 3 \end{bmatrix}, \quad B^{-1} = \begin{bmatrix} 1 & \frac{1}{3} & -\frac{1}{3} \\ 0 & \frac{1}{2} & 0 \\ 0 & -\frac{1}{3} & \frac{1}{3} \end{bmatrix}$$

Solution after two iterations:

$$\mathbf{x}_B = B^{-1}\mathbf{b} = \begin{bmatrix} 1 & \frac{1}{3} & -\frac{1}{3} \\ 0 & \frac{1}{2} & 0 \\ 0 & -\frac{1}{3} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \\ 2 \end{bmatrix}$$

Go to step 1.

Step 1 (second iteration)

Compute $Y = c_B B^{-1}$

$$Y = \begin{bmatrix} 0 & 5 & 3 \end{bmatrix} B^{-1} = \begin{bmatrix} 0 & \frac{3}{2} & 1 \end{bmatrix}$$

Compute $Z_j - c_j = YP_j - c_j$ for all non-basic variables (S_2 and S_3):

$$S_2: \quad Z_4 - c_4 = \begin{bmatrix} 0 & \frac{3}{2} & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} - 0 = \frac{3}{2}$$

$$S_3: \quad Z_5 - c_5 = \begin{bmatrix} 0 & \frac{3}{2} & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - 0 = 1$$

No negatives. Therefore stop.

Optimal solution

$$S_1^* = 2, \quad X_2^* = 6, \quad X_1^* = 2, \quad Z^* = c_B \mathbf{x}_B = \begin{bmatrix} 0 & 5 & 3 \end{bmatrix} \begin{bmatrix} 2 \\ 6 \\ 2 \end{bmatrix} = 36$$

5.2 Example 2

$$\begin{array}{lrcrcrcl}
\text{max} & 3x_1 & - & 2x_2 & - & 3x_3 & & \\
\text{s.t.} & x_1 & - & x_2 & - & x_3 & \leq & 1 \\
 & 7x_1 & - & 8x_2 & - & 11x_3 & \leq & 2 \\
 & 2x_1 & - & 2x_2 & - & 3x_3 & \leq & 1 \\
 & x_1 & , & x_2 & , & x_3 & \geq & 0
\end{array}$$

Adding slack variables x_4, x_5, x_6 gives the following linear program (P'):

Maximize $c^T x$ subject to $Ax = b$, $x \geq 0$,

where

$$P = \begin{bmatrix} 1 & -1 & -1 & 1 & 0 & 0 \\ 7 & -8 & -11 & 0 & 1 & 0 \\ 2 & -2 & -3 & 0 & 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix},$$

and

$$c = \begin{bmatrix} 3 & -2 & -3 & 0 & 0 & 0 \end{bmatrix}^T$$

Start from feasible basis $B = \{4, 5, 6\}$.

Iteration 1:

Given $B = \{4, 5, 6\}$, and $x_B^* = \begin{bmatrix} 2 & 1 & 0 \end{bmatrix}^T$, $A_B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

Solve $A_B^T y = c_B$ to get $y = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$.

Compute $\tilde{c}_1 = c_1 - A_1^T y = 3 - [172]y = 3 > 0$. x_1 enters.

Solve $A_B d = A_1$ to get $d = \begin{bmatrix} 1 & 7 & 2 \end{bmatrix}^T$.

Let $t = \min\left\{\frac{2}{1}, \frac{1}{7}, \frac{0}{2}\right\} = \frac{2}{7}$. x_5 leaves.

Update $x_1^* = t = \frac{2}{7}$, $x_4^* = 1 - (1)\left(\frac{2}{7}\right) = \frac{5}{7}$, $x_6^* = 1 - (0)\left(\frac{2}{7}\right) = 1$.

Iteration 2:

Given $B = \{1, 4, 6\}$, and $x_B^* = \begin{bmatrix} \frac{2}{7} & \frac{5}{7} & 1 \end{bmatrix}^T$, $A_B = \begin{bmatrix} 1 & 1 & 0 \\ 7 & 0 & 0 \\ 2 & 0 & 1 \end{bmatrix}$.

Solve $A_B^T y = c_B$ to get $y = \begin{bmatrix} 0 \\ \frac{3}{7} \\ 0 \end{bmatrix}$.

Compute $\tilde{c}_2 = c_2 - A_2^T y = -2 - [-1 - 8 - 2]y = \frac{10}{7} > 0$. x_2 enters.

Solve $A_B d = A_2$ to get $d = \begin{bmatrix} -\frac{8}{7} \\ \frac{1}{7} \\ \frac{2}{7} \end{bmatrix}$.

Let $t = \min\left\{\frac{5}{7}/\frac{1}{7}, 1/\frac{2}{7}\right\} = \frac{3}{2}$. x_6 leaves.

Update $x_2^* = t = \frac{3}{2}$, $x_1^* = \frac{2}{7} - \left(-\frac{8}{7}\right)\left(\frac{3}{2}\right) = 2$, $x_4^* = \frac{5}{7} - \left(\frac{1}{7}\right)\left(\frac{3}{2}\right) = \frac{1}{2}$.

Iteration 3:

Given $B = \{1,2,4\}$, and $x_B^* = \begin{bmatrix} 2 \\ \frac{3}{2} \\ \frac{1}{2} \end{bmatrix}$, $A_B = \begin{bmatrix} 1 & -1 & 1 \\ 7 & -8 & 0 \\ 2 & -2 & 0 \end{bmatrix}$.

Solve $A_B^T y = c_B$ to get $y = \begin{bmatrix} -2 \\ -1 \\ 5 \end{bmatrix}$.

Compute $\tilde{c}_3 = c_3 - A_3^T y = -3 - [-1 - 11 - 3]y > 0$. x_3 enters.

Solve $A_B d = A_3$ to get $d = \begin{bmatrix} -1 \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$.

Let $t = \min\left\{2/-1, \frac{3}{2}/\frac{1}{2}, \frac{1}{2}/\frac{1}{2}\right\} = 1$. x_4 leaves.

Update $x_3^* = t = 1$, $x_1^* = 2 - (-1)(1) = 3$, $x_2^* = \frac{3}{2} - \left(\frac{1}{2}\right)(1) = 1$.

5.3 Example 3

Max $Z = 2x_1 + x_2$

Subject to $3x_1 + 4x_2 \leq 6$

$6x_1 + x_2 \leq 3$ and $x_1, x_2 \geq 0$

By applying same steps we get the optimal solution to be:

$$x_1 = \frac{2}{7}, \quad x_2 = \frac{9}{7}$$

$$\text{Max } Z = \frac{13}{7}$$

6 Python Implementation

6.1 Implementing step by step

[239]:
```
from scipy.optimize import linprog
```

Starting with an example:

[240]:
```
A = np.array([[1, 0],
              [0, 2],
              [3, 2]])
b = np.array([4, 12, 18])
print(b)
c = np.array([3,5])
```

```
[ 4 12 18]
```

initializations

[241]:
```
m, n = A.shape # m is constraint count
    B = np.eye(m, dtype=int)
    c_B=np.zeros(m)
    print(c_B)
    x_B = np.linalg.solve(B, b)
    print(x_B)
    Z=np.dot(c_B, x_B)
    print(Z)
    C = np.hstack((c, c_B))
    print(C)
```

```
[0. 0. 0.]
[ 4. 12. 18.]
0.0
[3. 5. 0. 0. 0.]
```

6.1.1 Step 1

[242]:
```
Y = np.linalg.solve(B.T, c_B)
print(Y)
diff_z_c = np.dot(Y, P[:, 0]) - C[0]
print(diff_z_c)
P = np.hstack((A, B))
print(P)
min_diff_z_c_index = None
min_diff_z_c_value = np.inf

for idx, value in enumerate(c_B):
    diff_z_c = np.dot(Y, P[:, idx]) - C[idx]

    if diff_z_c < min_diff_z_c_value:
        min_diff_z_c_value = diff_z_c
        min_diff_z_c_index = idx
    if not np.any(diff_z_c < 0):
        # Objective function value
        Z = np.dot(c_B, x_B)
        break
entering_index=min_diff_z_c_index

print(f"Smallest diff_z_c and entering variable: {min_diff_z_c_value} at␣
 ↪entering_index: {entering_index+1}")
print(f"The entering variable is X{entering_index+1}")
```

```
[0. 0. 0.]
-3.0
[[1 0 1 0 0]
 [0 2 0 1 0]
 [3 2 0 0 1]]
Smallest diff_z_c and entering variable: -5.0 at entering_index: 2
The entering variable is X2
```

6.1.2 Step 2

[243]:
```
alpha = np.linalg.solve(B, P[:, entering_index])
print(P[:, entering_index])
print(alpha)
alpha[alpha <= 0] = np.nan
leaving_index = np.nanargmin(x_B / alpha)
print(f"The leaving is S{leaving_index+1}")
```

```
[0 2 2]
[0. 2. 2.]
The leaving is S2
```

6.1.3 Step 3

[244]:
```
B[:, leaving_index] = P[:, entering_index]
#B_indices[leavingSlack_index] = entering_index
x_B = np.linalg.solve(B, b)
print(x_B)
c_B[leaving_index] = C[entering_index]
print(c_B)
```

```
[4. 6. 6.]
[0. 5. 0.]
```

Now we know the process, Lets write it in a function to be easier to use

6.2 Revised Simplex Function

[245]:
```
import numpy as np
def revised_simplex_method(A, b, c):
   #initializations
    m, n = A.shape
    B = np.eye(m, dtype=int)
    # keep track of the indices of the basic variables
    index = np.arange(n, n+m)
    #print(index)
    #print(index)
```

```
c_B=np.zeros(m)
C=np.hstack((c, c_B))
x_B = np.linalg.solve(B, b)
Z=np.dot(c_B, x_B)
P = np.hstack((A, B))
#boolean array to represent the set of basic and non-basic variables
J= np.hstack((np.zeros(n), np.ones(m))).astype(bool)
#print(J)
while True:
  #step 1:
  print("Step 1")
  Y = np.linalg.solve(B.T, c_B)
  # diff_z_c = np.dot(Y, P[:, 0]) - C[0]
  #min_diff_z_c_index = None
  #min_diff_z_c_value = np.inf
  #for idx, value in enumerate(c_B):
  #diff_z_c = np.dot(Y, P[:, idx]) - C[idx]
  #if diff_z_c < min_diff_z_c_value:
  #  min_diff_z_c_value = diff_z_c
  #  min_diff_z_c_index = idx
  #if not np.any(diff_z_c < 0):
    # Objective function value
  #  Z = np.dot(c_B, x_B)
  # break
  #entering_index=min_diff_z_c_index
  #print(Y)
  #print(P)
  #print("This is c",C)
  #print("HEEEEEEEE")
  diff_z_c = (Y @ P[:, ~J]) - C[~J]
  if np.all(diff_z_c >= 0):
        # Objective function
        Z = np.dot(c_B, x_B)
        break
    entering_index=np.argmin(diff_z_c)
    for i, value in enumerate(J):
        if value == 0:
           entering_index -= 1  # decrease entering_index for each occurrence of 0
        if entering_index == -1:
          entering_index = i
          break



    print(f"Smallest Variable corresponding to entering variable: {np.
↪min(diff_z_c)} at entering_index: {entering_index+1}")
    print(f"The entering variable is X{entering_index+1}")
    #step 2:
    print("Step 2")
    alpha = np.linalg.solve(B, P[:, entering_index])
    alpha[alpha <= 0] = np.nan
    leaving_index = np.nanargmin(x_B / alpha)



    for i in range(len(J)):
       if i == entering_index:
          J[i] = True



    print(f"The leaving is S{leaving_index+1}")
    #step 3:
    print("Step 3")
    B[:, leaving_index] = P[:, entering_index]
    #print(index)
    index[leaving_index] = entering_index
    #print(index)
    x_B = np.linalg.solve(B, b)
    c_B[leaving_index] = C[entering_index]
```

```python



    #To make sure the values are ordered according to the variables
    Results = np.zeros(n+m)
    for i, value in zip(index, x_B):
        Results[i] = value
    #printing Results
    for i, value in enumerate(Results):
            print("--------------\n")
            print(f"X{i + 1} = {value:.4f} ")
    print("--------------\n")
    print(f"f  = {Z:.4f}\n")
```

6.3 Examples:

6.3.1 Problem 1:

$$
\begin{aligned}
\text{Max } Z &= 3X_1 + 5X_2 \\
\text{s.t.} \quad X_1 &\leq 4 \\
2X_2 &\leq 12 \\
3X_1 + 2X_2 &\leq 18 \\
X_1, X_2 &\geq 0
\end{aligned}
$$

[310]:
```python
# Example usage:
A = np.array([[1, 0],
              [0, 2],
              [3, 2]])
b = np.array([4, 12, 18])
c = np.array([3,5])

revised_simplex_method(A, b, c)
```

```
Step 1

Smallest Variable corresponding to entering variable: -5.0 at entering_index: 2

The entering variable is X2

Step 2

The leaving is S2

Step 3

Step 1

Smallest Variable corresponding to entering variable: -3.0 at entering_index: 1

The entering variable is X1

Step 2

The leaving is S3

Step 3

Step 1

--------------


X1 = 2.0000

--------------


X2 = 6.0000

--------------


X3 = 2.0000

--------------


X4 = 0.0000

--------------


X5 = 0.0000

--------------


f  = 36.0000
```

[311]:
```python
import time
bnd = [(0, float("inf")),  (0, float("inf"))]
start_time = time.time()
res_ = linprog(c= -np.ones(len(c)) * c, A_ub=A, b_ub=b, bounds=bnd, method="revised␣
 ↪simplex")
print("Optimal solution:")
print(res_.x)
print("Optimal value:")
print(res_.fun)
```

```
Optimal solution:
[2. 6.]
Optimal value:
-36.0

<ipython-input-311-291486b28f20>:4: DeprecationWarning: `method='revised
simplex'` is deprecated and will be removed in SciPy 1.11.0. Please use one of
the HiGHS solvers (e.g. `method='highs'`) in new code.
  res_ = linprog(c= -np.ones(len(c)) * c, A_ub=A, b_ub=b, bounds=bnd,
method="revised simplex")
```

6.3.2 Problem 2:

Max $Z = 2x_1 + x_2$

Subject to $3x_1 + 4x_2 \leq 6$

$6x_1 + x_2 \leq 3$ and $x_1, x_2 \geq 0$

[313]:
```python
# Example usage:
A = np.array([[3, 4],
              [6, 1]])
b = np.array([6, 3])
c = np.array([2,1])
```

```
revised_simplex_method(A, b, c)
```

```
Step 1

Smallest Variable corresponding to entering variable: -2.0 at entering_index: 1

The entering variable is X1

Step 2

The leaving is S2

Step 3

Step 1

Smallest Variable corresponding to entering variable: -0.6666666666666667 at

entering_index: 2

The entering variable is X2

Step 2

The leaving is S1

Step 3

Step 1

--------------


X1 = 0.2857

--------------


X2 = 1.2857

--------------


X3 = 0.0000

--------------


X4 = 0.0000

--------------


f  = 1.8571
```

```
[314]: import time
       bnd = [(0, float("inf")),  (0, float("inf"))]
       start_time = time.time()
       res_ = linprog(c= -np.ones(len(c)) * c, A_ub=A, b_ub=b, bounds=bnd, method="revised␣
        ↪simplex")
       print("Optimal solution:")
       print(res_.x)
       print("Optimal value:")
       print(res_.fun)
```

```
Optimal solution:
[0.28571429 1.28571429]
Optimal value:
-1.8571428571428572

<ipython-input-314-291486b28f20>:4: DeprecationWarning: `method='revised
simplex'` is deprecated and will be removed in SciPy 1.11.0. Please use one of
the HiGHS solvers (e.g. `method='highs'`) in new code.
  res_ = linprog(c= -np.ones(len(c)) * c, A_ub=A, b_ub=b, bounds=bnd,
method="revised simplex")
```

6.3.3 Problem 3:

$$\begin{array}{lccccccc}
\text{max} & 3x_1 & - & 2x_2 & - & 3x_3 & & \\
\text{s.t.} & x_1 & - & x_2 & - & x_3 & \leq & 1 \\
 & 7x_1 & - & 8x_2 & - & 11x_3 & \leq & 2 \\
 & 2x_1 & - & 2x_2 & - & 3x_3 & \leq & 1 \\
 & x_1 & , & x_2 & , & x_3 & \geq & 0
\end{array}$$

```
[316]: # Example usage:
       A = np.array([[1 ,-1, -1],
                     [7 ,-8 ,-11],
              [2 ,-2, -3]])
b = np.array([1,2,1])
c = np.array([3,-2,-3])


revised_simplex_method(A, b, c)
```

```
Step 1
Smallest Variable corresponding to entering variable: -3.0 at entering_index: 1
The entering variable is X1
Step 2
The leaving is S2
Step 3
Step 1
Smallest Variable corresponding to entering variable: -1.7142857142857144 at
entering_index: 3
The entering variable is X3
Step 2
The leaving is S1
Step 3
Step 1
Smallest Variable corresponding to entering variable: -1.0 at entering_index: 2
The entering variable is X2
Step 2
The leaving is S3
Step 3
Step 1
--------------

X1 = 3.0000
--------------

X2 = 1.0000
--------------
```

```
X3 = 1.0000

--------------


X4 = 0.0000

--------------


X5 = 0.0000

--------------


X6 = 0.0000

--------------


f  = 4.0000
```

[318]:
```python
import time
bnd = [(0, float("inf")),  (0, float("inf")),(0, float("inf"))]
start_time = time.time()
res_ = linprog(c= -np.ones(len(c)) * c, A_ub=A, b_ub=b, bounds=bnd, method="revised␣
 ↪simplex")
print("Optimal solution:")
print(res_.x)
print("Optimal value:")
print(res_.fun)
```

```
Optimal solution:
[3. 1. 1.]
Optimal value:
-4.000000000000001

<ipython-input-318-7eb04f27489e>:4: DeprecationWarning: `method='revised
simplex'` is deprecated and will be removed in SciPy 1.11.0. Please use one of
the HiGHS solvers (e.g. `method='highs'`) in new code.
  res_ = linprog(c= -np.ones(len(c)) * c, A_ub=A, b_ub=b, bounds=bnd,
method="revised simplex")
```

7 References:

- Admin. (2022, April 6). Revised Simplex Method (Introduction, steps and example). BYJUS. https://byjus.com/maths/revised-simplex-method/

- Shah, P. N. (n.d.). Revised Simplex method Standard form-1: Example-1. https://cbom.atozmath.com/example/CBOM/Simplex.aspx?q=rsmq1=E1

- A, S. (2017, March 21). Simplex Method for solution of L.P.P (With Examples) | Operation Research. Engineering Notes India. https://www.engineeringenotes.com/linear-programming/simplex-method-for-solution-of-l-p-p-with-examples-operation-research/15373