

DM545/DM871
Linear and Integer Programming

Revised Simplex Method

Marco Chiarandini

Department of Mathematics and Computer Science

October 5, 2026

1. Revised Simplex Method

2. Efficiency Issues

Complexity of single pivot operation in standard simplex:

- entering variable $O(n)$
- leaving variable $O(m)$
- updating the tableau $O(mn)$

Problems with this:

- Time: we are doing operations that are not actually needed
Space: we need to store the whole tableau: $O(mn)$ floating point numbers
- Most problems have sparse matrices (many zeros)
sparse matrices are typically handled efficiently
the standard simplex has the 'Fill in' effect: sparse matrices are lost
- accumulation of Floating Point Errors over the iterations

1. Revised Simplex Method

2. Efficiency Issues

Revised Simplex Method

Several ways to improve wrt pitfalls in the previous slide, requires matrix description of the simplex.

$$\begin{aligned} \max \quad & \sum_{j=1}^n c_j x_j \\ & \sum_{j=1}^n a_{ij} x_j \leq b_i \quad i = 1..m \\ & x_j \geq 0 \quad j = 1..n \end{aligned}$$

$$\begin{aligned} \max \quad & \mathbf{c}^T \mathbf{x} \\ & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \\ & A \in \mathbb{R}^{m \times (n+m)} \\ & \mathbf{c} \in \mathbb{R}^{(n+m)}, \mathbf{b} \in \mathbb{R}^m, \mathbf{x} \in \mathbb{R}^{n+m} \end{aligned}$$

$$\max\{\mathbf{c}^T \mathbf{x} \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$$

At each iteration the simplex moves from a basic feasible solution to another.

For each basic feasible solution:

- $N = \{1 \dots n\}$
- $B = \{n+1 \dots n+m\}$ basis
- $A_N = [\mathbf{a}_1 \dots \mathbf{a}_n]$
- $A_B = [\mathbf{a}_{n+1} \dots \mathbf{a}_{n+m}]$ basis matrix
- $\mathbf{x}_N = 0$
- $\mathbf{x}_B \geq 0$

$$\left[\begin{array}{cc|c|c} & & & \\ & A_N & A_B & \mathbf{0} & \mathbf{b} \\ \hline \mathbf{c}_N^T & & \mathbf{c}_B^T & 1 & 0 \end{array} \right]$$

$$A\mathbf{x} = A_N\mathbf{x}_N + A_B\mathbf{x}_B = \mathbf{b}$$

$$A_B\mathbf{x}_B = \mathbf{b} - A_N\mathbf{x}_N$$

Basic feasible solution $\iff A_B$ is non-singular

$$\mathbf{x}_B = A_B^{-1}\mathbf{b} - A_B^{-1}A_N\mathbf{x}_N$$

for the objective function:

$$z = \mathbf{c}^T \mathbf{x} = \mathbf{c}_B^T \mathbf{x}_B + \mathbf{c}_N^T \mathbf{x}_N$$

Substituting for $\mathbf{x}_B$ from above:

$$\begin{aligned} z &= \mathbf{c}_B^T (A_B^{-1} \mathbf{b} - A_B^{-1} A_N \mathbf{x}_N) + \mathbf{c}_N^T \mathbf{x}_N = \\ &= \mathbf{c}_B^T A_B^{-1} \mathbf{b} + (\mathbf{c}_N^T - \mathbf{c}_B^T A_B^{-1} A_N) \mathbf{x}_N \end{aligned}$$

Collecting together:

$$\begin{aligned} \mathbf{x}_B &= A_B^{-1} \mathbf{b} - A_B^{-1} A_N \mathbf{x}_N \\ z &= \mathbf{c}_B^T A_B^{-1} \mathbf{b} + (\mathbf{c}_N^T - \underbrace{\mathbf{c}_B^T A_B^{-1} A_N}_{\bar{A}}) \mathbf{x}_N \end{aligned}$$

In tableau form, for a basic feasible solution corresponding to B we have:

$$\left[\begin{array}{c|c|c|c} A_B^{-1} A_N & I & \mathbf{0} & A_B^{-1} \mathbf{b} \\ \hline \mathbf{c}_N^T - \mathbf{c}_B^T A_B^{-1} A_N & 0 & 1 & -\mathbf{c}_B^T A_B^{-1} \mathbf{b} \end{array} \right]$$

We do not need to compute all elements of $\bar{A}$

Example

$$\begin{aligned}
 \max \quad & x_1 + x_2 \\
 \text{s.t.} \quad & -x_1 + x_2 \leq 1 \\
 & x_1 \leq 3 \\
 & x_2 \leq 2 \\
 & x_1, x_2 \geq 0
 \end{aligned}$$

Initial tableau

x1	x2	x3	x4	x5	-z	b
-1	1	1	0	0	0	1
1	0	0	1	0	0	3
0	1	0	0	1	0	2
1	1	0	0	0	1	0

$$\begin{aligned}
 \max \quad & x_1 + x_2 \\
 \text{s.t.} \quad & -x_1 + x_2 + x_3 = 1 \\
 & x_1 + x_4 = 3 \\
 & x_2 + x_5 = 2 \\
 & x_1, x_2, x_3, x_4, x_5 \geq 0
 \end{aligned}$$

After two iterations

x1	x2	x3	x4	x5	-z	b
1	0	-1	0	1	0	1
0	1	0	0	1	0	2
0	0	1	1	-1	0	2
0	0	1	0	-2	1	3

Basic variables x_1, x_2, x_4 . Non basic: x_3, x_5 . From the **initial tableau**:

$$A_B = \begin{bmatrix} -1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad A_N = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \quad x_B = \begin{bmatrix} x_1 \\ x_2 \\ x_4 \end{bmatrix} \quad x_N = \begin{bmatrix} x_3 \\ x_5 \end{bmatrix}$$

$$c_B^T = [1 \ 1 \ 0] \quad c_N^T = [0 \ 0]$$

- **Entering variable:**

in standard we look at tableau, in revised we need to compute: $\mathbf{c}_N^T - \mathbf{c}_B^T \mathbf{A}_B^{-1} \mathbf{A}_N$

1. find $\mathbf{y}^T = \mathbf{c}_B^T \mathbf{A}_B^{-1}$ (by solving $\mathbf{y}^T \mathbf{A}_B = \mathbf{c}_B^T$, the latter can be done more efficiently)
2. calculate $\mathbf{c}_N^T - \mathbf{y}^T \mathbf{A}_N$

Step 1:

$$[y_1 \ y_2 \ y_3] \begin{bmatrix} -1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = [1 \ 1 \ 0]$$

$$\mathbf{y}^T A_B = \mathbf{c}_B^T$$

$$[1 \ 1 \ 0] \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & -1 \end{bmatrix} = [-1 \ 0 \ 2]$$

$$\mathbf{c}_B^T A_B^{-1} = \mathbf{y}^T$$

Step 2:

$$[0 \ 0] - [-1 \ 0 \ 2] \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} = [1 \ -2]$$

$$\mathbf{c}_N^T - \mathbf{y}^T A_N$$

(Note that they can be computed individually: $\mathbf{c}_j - \mathbf{y}^T \mathbf{a}_j > 0$)

Let's take the first we encounter $\mathbf{x}_3$

- **Leaving variable**

we increase variable by largest feasible amount θ

$$\text{R1: } x_1 - x_3 + x_5 = 1$$

$$x_1 = 1 + x_3 \geq 0$$

$$\text{R2: } x_2 + x_5 = 2$$

$$x_2 = 2 \geq 0$$

$$\text{R3: } -x_3 + x_4 - x_5 = 2$$

$$x_4 = 2 - x_3 \geq 0$$

$$\mathbf{x}_B = \mathbf{x}_B^* - A_B^{-1} A_N \mathbf{x}_N$$

$$\mathbf{x}_B = \mathbf{x}_B^* - \mathbf{d}\theta$$

$\mathbf{d}$ is the column of $A_B^{-1} A_N$ that corresponds to the entering variable, ie, $\mathbf{d} = A_B^{-1} \mathbf{a}$ where $\mathbf{a}$ is the entering column

3. Find θ such that $\mathbf{x}_B$ stays positive:

Find $\mathbf{d} = A_B^{-1} \mathbf{a}$ (by solving $A_B \mathbf{d} = \mathbf{a}$)

Step 3:

$$\begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \Rightarrow \mathbf{d} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \Rightarrow \mathbf{x}_B = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix} - \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \theta \geq 0$$

$$2 - \theta \geq 0 \Rightarrow \theta \leq 2 \rightsquigarrow x_4 \text{ leaves}$$

- So far we have done computations, but now we save the pivoting update. The update of A_B is done by replacing the leaving column by the entering column

$$x_B^* = \begin{bmatrix} x_1 - d_1\theta \\ x_2 - d_2\theta \\ \theta \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 2 \end{bmatrix} \quad A_B = \begin{bmatrix} -1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

- Many implementations depending on how $y^T A_B = c_B^T$ and $A_B d = a$ are solved. They are in fact solved from scratch.
- many operations saved especially if many variables!
- special ways to call the matrix A from memory
- better control over numerical issues since A_B^{-1} can be recomputed.

1. Revised Simplex Method

2. Efficiency Issues

Solving the two Systems of Equations

$A_B \mathbf{x} = \mathbf{b}$ solved without computing A_B^{-1}
(costly and likely to introduce numerical inaccuracy)

Recall how the inverse is computed:

For a 2×2 matrix

the matrix inverse is

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$A^{-1} = \frac{1}{|A|} \begin{bmatrix} d & -c \\ -b & a \end{bmatrix}^T = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

For a 3×3 matrix

the matrix inverse is

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$A^{-1} = \frac{1}{|A|} \begin{bmatrix} + \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & - \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} & + \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \\ - \begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} & + \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & - \begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} \\ + \begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} & - \begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} & + \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{bmatrix}^T$$

Eta Factorization of the Basis

At the k th iteration of the simplex, let $B_k := A_{B_k}$,

B_k be the matrix with col $\mathbf{p}$ differing from B_{k-1}

Column $\mathbf{p}$ is the column $\mathbf{a}$ appearing in $B_{k-1}\mathbf{d} = \mathbf{a}$ solved in the third step of the revised

Hence:

$$B_k = B_{k-1}E_k$$

E_k is the **eta matrix** differing from identity matrix in only one column, which is set equal to $\mathbf{d}$

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

No matter how we solve $\mathbf{y}^T B_{k-1} = \mathbf{c}_B^T$ and $B_{k-1}\mathbf{d} = \mathbf{a}$, their update always relays on $B_k = B_{k-1}E_k$ with E_k available. Plus when initial basis by slack variable $B_0 = I$ and $B_1 = E_1, B_2 = E_1 E_2 \cdots$:

$$B_k = E_1 E_2 \cdots E_k \quad \text{eta factorization}$$

$$((((\mathbf{y}^T E_1) E_2) E_3) \cdots) E_k = \mathbf{c}_B^T,$$

$$(E_1(E_2 \cdots E_k \mathbf{d})) = \mathbf{a},$$

$$\mathbf{u}^T E_4 = \mathbf{c}_B^T, \mathbf{v}^T E_3 = \mathbf{u}^T, \mathbf{w}^T E_2 = \mathbf{v}^T, \mathbf{y}^T E_1 = \mathbf{w}^T$$

$$E_1 \mathbf{u} = \mathbf{a}, E_2 \mathbf{v} = \mathbf{u}, E_3 \mathbf{w} = \mathbf{v}, E_4 \mathbf{d} = \mathbf{w}$$

Solve the systems $\mathbf{y}^T E_1 E_2 E_3 E_4 = [1 \ 2 \ 3]$ and $E_1 E_2 E_3 E_4 \mathbf{d} = [1 \ 2 \ 3]^T$ with

$$E_1 = \begin{bmatrix} 1 & 3 & 0 \\ 0 & 0.5 & 0 \\ 0 & 4 & 1 \end{bmatrix} \quad E_2 = \begin{bmatrix} 2 & 0 & 0 \\ 1 & 1 & 0 \\ 4 & 0 & 1 \end{bmatrix} \quad E_3 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \quad E_4 = \begin{bmatrix} -0.5 & 0 & 0 \\ 3 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

We use backward transformation and solve the sequence of linear systems:

$$\mathbf{u}^T E_4 = [1 \ 2 \ 3], \quad \mathbf{v}^T E_3 = \mathbf{u}^T, \quad \mathbf{w}^T E_2 = \mathbf{v}^T, \quad \mathbf{y}^T E_1 = \mathbf{w}^T$$

$$\mathbf{u}^T \begin{bmatrix} -0.5 & 0 & 0 \\ 3 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = [1, 2, 3]$$

Since the eta matrices have always one 1 in two columns then the solution can be read up easily. From the third column we find $u_3 = 3$. From the second column, we find $u_2 = 2$. Substituting in the first column, we find $-0.5u_1 + 3 \cdot 2 + 1 \cdot 3 = 1$, which yields $u_1 = 16$. The next system is:

$$\mathbf{v}^T \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} = [16, 2, 3]$$

From the first column we get $v_1 = 16$, from the second column $v_2 = 2$ from the last column $v_1 + 3v_2 + v_3 = 3$, which yields $v_3 = -19$. The next:

$$\mathbf{w}^T \begin{bmatrix} 2 & 0 & 0 \\ 1 & 1 & 0 \\ 4 & 0 & 1 \end{bmatrix} = [16, 2, -19]$$

- Solving $\mathbf{y}^T B_k = \mathbf{c}_B^T$ also called backward transformation (BTRAN)
- Solving $B_k \mathbf{d} = \mathbf{a}$ also called forward transformation (FTRAN)
- E_i matrices can be stored by only storing the column and the position
- If sparse columns then can be stored in compact mode, ie only nonzero values and their indices

Worth to consider also the case of $B_0 \neq I$:

$$B_k = B_0 E_1 E_2 \dots E_k \quad \text{eta factorization}$$

$$\begin{aligned} ((((\mathbf{y}^T B_0) E_1) E_2) \dots) E_k &= \mathbf{c}_B^T \\ (B_0 (E_1 \dots E_k \mathbf{d})) &= \mathbf{a} \end{aligned}$$

We need an LU factorization of B_0

To solve the system $A\mathbf{x} = \mathbf{b}$ by Gaussian Elimination we put the A matrix in row echelon form by means of elementary row operations. Each row operation corresponds to multiply left and right side by a lower triangular matrix L and a permutation matrix P . Hence, the method:

$$\begin{aligned} A\mathbf{x} &= \mathbf{b} \\ L_1 P_1 A\mathbf{x} &= L_1 P_1 \mathbf{b} \\ L_2 P_2 L_1 P_1 A\mathbf{x} &= L_2 P_2 L_1 P_1 \mathbf{b} \\ &\vdots \\ L_m P_m \dots L_2 P_2 L_1 P_1 A\mathbf{x} &= L_m P_m \dots L_2 P_2 L_1 P_1 \mathbf{b} \end{aligned}$$

thus

$$U = L_m P_m \dots L_2 P_2 L_1 P_1 A \quad \text{triangular factorization of } A$$

where U is an upper triangular matrix whose entries in the diagonal are ones. (if A is nonsingular such triangularization is unique)

(see numerical example in [Va] sc 8.1)

We can compute the triangular factorization of B_0 before the initial iterations of the simplex:

$$L_m P_m \dots L_2 P_2 L_1 P_1 B_0 = U$$

We can then rewrite U as

$$U = U_m U_{m-1} \dots, U_1$$

Hence, for $B_k = B_0 E_1 E_2 \dots E_k$:

$$L_m P_m \dots L_2 P_2 L_1 P_1 B_k = U_m U_{m-1} \dots U_1 E_1 E_2 \dots E_k$$

Then $\mathbf{y}^T B_k = \mathbf{c}_B^T$ can be solved by first solving:

$$((((\mathbf{y}^T U_m) U_{m-1}) \dots) E_k = \mathbf{c}_B^T$$

$$B_k = \underbrace{(L_m P_m \dots L_1 P_1)}_L^{-1} \underbrace{U_m \dots E_k}_U$$

and then replacing

$$\mathbf{y}^T \text{ by } ((\mathbf{y}^T L_m P_m) \dots) L_1 P_1$$

$$\mathbf{y} L^{-1} U = \mathbf{c}$$

$$\mathbf{w} U = \mathbf{c}$$

$$\mathbf{w} = \mathbf{y} L^{-1} \implies \mathbf{y} = L \mathbf{w}$$

LU decomposition, the matrix form of Gaussian elimination.

$$\mathbf{A} = \mathbf{LU}$$

$$\mathbf{PA} = \mathbf{LU}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

If the diagonal of $\mathbf{U}$ is 1 and $\mathbf{A}$ is nonsingular, then there exists only one LU factorization

- it can be rewritten as $\mathbf{A}_B = \mathbf{LU}$, $\mathbf{L}$ lower triangular matrix, $\mathbf{U}$ upper triangular matrix, ie, LU-factorization
- $\mathbf{LUx} = \mathbf{b}$: setting $\mathbf{y} = \mathbf{Ux}$ then
 1. $\mathbf{Ly} = \mathbf{b}$ can be solved easily by forward substitution
 2. $\mathbf{Ux} = \mathbf{y}$ can be solved easily by backward substitution

- Tableau method is unstable: computational errors may accumulate. Revised method has a natural control mechanism: we can recompute A_B^{-1} at any time
- Commercial and freeware solvers differ from the way the systems $\mathbf{y}^T A_B = \mathbf{c}_B^T$ and $A_B \mathbf{d} = \mathbf{a}$ are resolved

- Dual simplex with steepest descent (largest increase)
- Linear Algebra:
 - Dynamic LU-factorization using Markowitz threshold pivoting (Suhl and Suhl, 1990)
 - sparse linear systems: Typically these systems take as input a vector with a very small number of nonzero entries and output a vector with only a few additional nonzeros.
- Presolve, ie problem reductions: removal of redundant constraints, fixed variables, and other extraneous model elements.
- dealing with degeneracy, stalling (long sequences of degenerate pivots), and cycling:
 - bound-shifting (Paula Harris, 1974)
 - Hybrid Pricing (variable selection): start with partial pricing, then switch to devex (approximate steepest-edge, Harris, 1974)
- A model that might have taken a year to solve 10 years ago can now solve in less than 30 seconds (Bixby, 2002).

- Ellipsoid method: cannot compete in practice but polynomial time (Khachyian, 1979)
- **Interior point algorithm(s)** (Karmarkar, 1984) competitive with simplex and polynomial in some versions
 - iterate through points interior to the feasibility region
 1. Start at an interior point of the feasible region
 2. Move in a direction that improves the objective function value at the fastest possible rate
 3. Transform the feasible region to place the current point at the center of it
 - because of patents reasons, also known as **barrier algorithm**
 - one single iteration is computationally more intensive than the simplex
 - particularly competitive in presence of many constraints (eg, for $m = 10,000$ may need less than 100 iterations)
 - bad for post-optimality analysis $\rightsquigarrow$ crossover algorithm to convert a sol of barrier method into a basic feasible solutions for the simplex
 - usually fastest for large, difficult models but numerically sensitive.

A method is a general template that lacks details to be fully defined. An algorithm is a well defined procedure that terminates.

- Simplex method
- Interior-Point methods
- Network Simplex
- Sifting methods
- Lagrangian relaxation
- Decomposition methods:
 - Dantzig Wolfe decomposition
 - Benders decomposition
- First-order methods: Primal-Dual Hybrid Gradient Method (matrix factorization free, good for GPUs)

- Bottleneck: Matrix factorization to solve linear equations:
 - simplex: LU-factorization
 - interior point method: Cholesky factorization
- Constraint matrix is extremely sparse in practice while the factorization is much denser. Hence, more memory than just storing the LP instance itself. This may lead to out-of-memory errors when solving large instances.
Factorization may still take a long time for large instances.
- Highly challenging to take advantage of modern computing architectures, such as GPUs or distributed systems, to solve the linear equations in these two methods.
- For simplex, multiple cores are generally employed only in the “pricing” step that selects variables entering or leaving the basis, but the speedups are generally minimal after two or three cores
- For interior point methods, multiple threads can be used for factorization but speedups are generally negligible after at most six shared-memory cores

How Large Problems Can We Solve?

Very large model

	Rows	Columns	Nonzeros
Original size	5034171	7365337	25596099
After presolve	1296075	2910559	10339042

Solution times were as follows:

Very large model—solution times

Version	Algorithm		
	Barrier	Dual	Primal
CPLEX 5.0	8642.6	350000.0	71039.7
CPLEX 7.1	5642.6	6413.1	1880.0