# CHEMICAL ENGINEERING 4G03

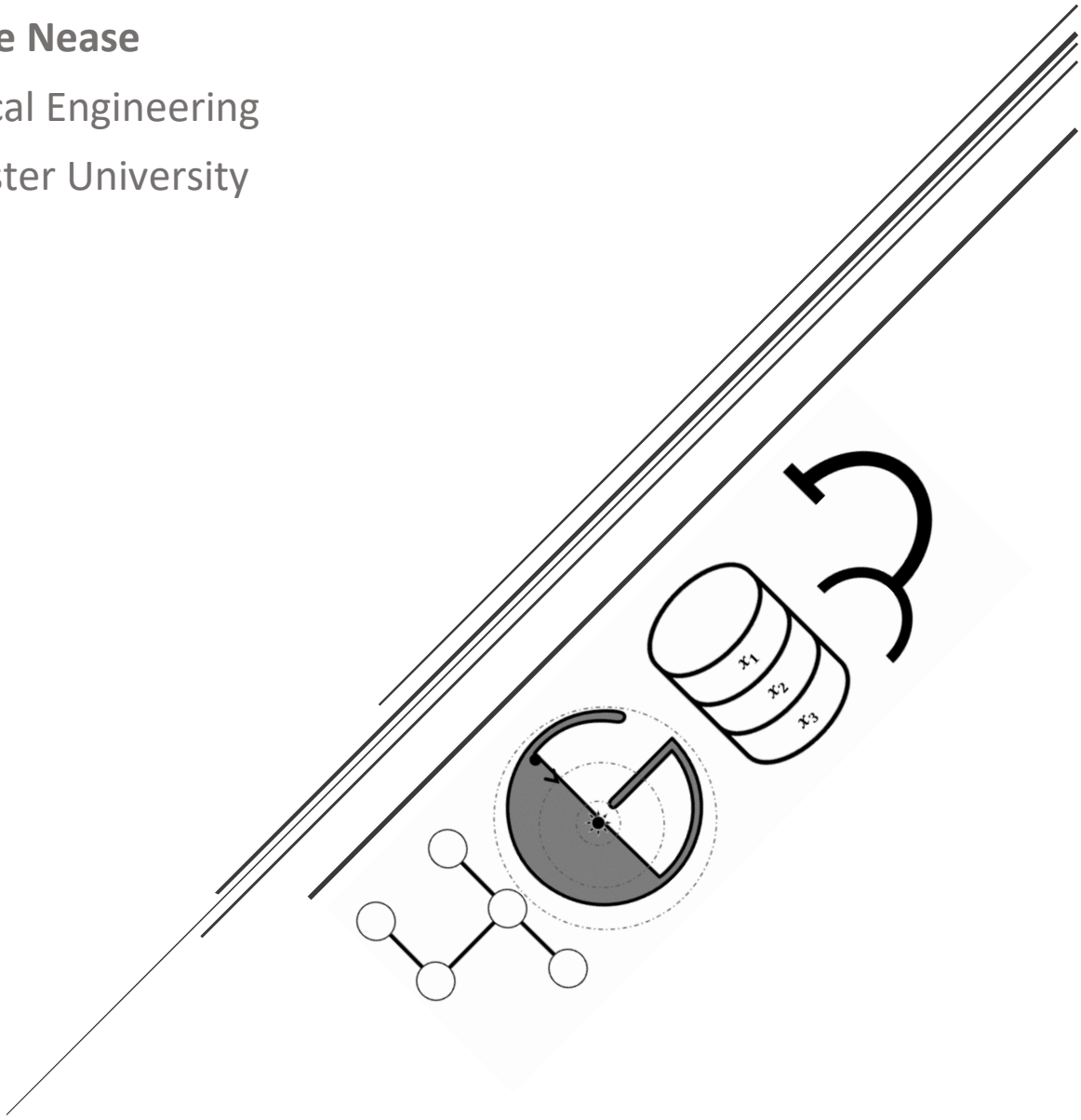
## Module 04

## Linear Programming (II) – Simplex Search

**Dr. Jake Nease**

Chemical Engineering

McMaster University




Updated February 7, 2023

# Outline of Module

This module consists of the following topics. This is a continuation of module 03, so we will be using those concepts extensively here (especially the standard form representation of LPs).



# Suggested Readings

**Rardin (1st edition)**: Chapter 5.2-5.7

**Rardin (2nd edition)**: Chapter 5.2-5.7

I would particularly recommend that you review the concepts of basic solutions for this section. It is in section 5.2 (toward the end) of the text and will be discussed here. Please let me know if you are not clear on the concept of basic solutions.

… Heads-up, we will likely have to **continue this lesson into tutorials**.

# The Concept of the Simplex Search and Basic Solutions

Near the end of the previous module, we came up with a general idea to find the optimum of a linear program through the combination of geometric insight and numerical methods. As a review, the general procedure we want to pursue is as follows:

- Consider only adjacent extreme points when finding an improving direction
- Move along the edge that yields the greatest improvement
- Continue moving until we hit another extreme point
- Check again if there is another edge that will yield an improvement in the objective function. If so, continue, else, stop.

This is called the **Simplex Search** and is the mostly widely used solution method for linear programs because it works and the total number of iterations scales **finitely** with problem size (that is, there is some finite maximum to the number of guesses it will take for us to find the global optimum). Recall that we are looking for the *global* optimum because all linear programs are *convex*.

However, even though this is a good idea we face the following issues:

1. How do we characterize **extreme feasible points** without actually drawing them?
2. How do we **compute the move** from one extreme feasible point to another?
3. If I can't visualize the feasible region, how do I find an **initial feasible extreme point**?

These are the questions that we are going to answer in this section. Once we are done, we will have the basis of the simplex algorithm down pat!

Recall the general formulation of a linear program in **standard form:**

$$\begin{aligned} \min_{\boldsymbol{x}} \phi = \boldsymbol{c}^T \boldsymbol{x} \quad & \leftarrow \text{Objective Function} \\ s.t. \quad & \leftarrow \text{"Subject to"} \\ A\boldsymbol{x} = b \quad & \leftarrow \text{ONLY Equality Constraints} \\ x \geq 0 \quad & \leftarrow \text{Variable Bounds} \end{aligned}$$

### Basic Solution to a Linear Program

A **basic solution** $\mathcal{B}$ to a linear program *in standard form* is obtained by fixing *just enough variables* to the value **0** so that the model's equality constraints can be solved *uniquely* for the remaining variables.

- The variables fixed to 0 are called **nonbasic variables** of the basic solution.
- The variables obtained by solving the equalities for a given set of nonbasic variables are called the **basic variables** of the basic solution.
- It is important to recall that a solution $x^{(k)}$ is a collection of variables, NOT the objective function value *at the solution*.
- Finally, recall that our LP in standard for has $n$ variables and $m$ independent equality constraints. We therefore require $(n - m)$ nonbasic variables at each basic solution.

**Class Workshop – Identifying Basic Solutions**

Consider the following linear program. Formulate the program in standard form and compute the basic solution corresponding to $x_1$ **and** $x_2$ **basic**.

$$\begin{aligned} \min_x \phi &= 9x_1 + 4x_2 \\ \text{s.t.} & \\ x_1 + x_2 &= 1 \\ 3x_1 - 2x_2 &\leq 8 \\ x_1 &\geq 0 \\ x_2 &\leq 0 \end{aligned}$$

**Workshop Solution – Identifying Basic Solutions**

## *Existence of Basic Solutions*

One must be wary of coming up with any combination of basic variables. It is **not possible** to come up with a basic solution with any set of variables declared basic. In fact:

> *A basic solution exists **if and only if** the system of equality constraints corresponding to a set of basic variables has a **unique solution**.*

Thinking back to linear algebra, what does this mean? Well, it means that the solution to our system of constraints $Ax = b$ with nonbasic variables removed has a **unique solution** for the vector $x$. This means that our matrix $A$ must **be square** (same number of rows and columns) and (the following are equivalent statements):

- Be full-rank.
- Be invertible.
- Have a nonzero determinant (how do you compute the determinant again…?).

We can thus develop some **necessary** (to claim) and **sufficient** (to prove) conditions for a system of $m$ linear equations $Ax = b$ to have a **unique** solution. It is *necessary* that the system is square, and *sufficient* if the system has a nonzero determinant (and is thus invertible).

**Class Workshop – Existence of Basic Solutions**

Consider the following linear constraints in standard form. Do basic solutions exist for the following sets of basic variables? Why or why not?

1. $\mathcal{B} = \{x_1, x_2\}$
2. $\mathcal{B} = \{x_1\}$
3. $\mathcal{B} = \{x_2, x_3\}$

$$\begin{aligned} 4x_1 - 8x_2 - x_3 &= 15 \\ x_1 - 2x_2 &= 10 \\ x_i &\geq 0 \qquad \forall i \end{aligned}$$

**Workshop Solution – Existence of Basic Solutions**

## *Basic Feasible Solutions and Extreme Points*

OK, so what do basic solutions do for us exactly? Well, they are able to identify the positions in which *just enough constraints are active* to denote the intersection of $n - m$ constraints. Why is this useful? Well, it turns out that such intersections that also are at the **extrema of our feasible region** (corner points) coincide with the basic feasible solutions of any linear program! We now can use the simplex search algebraically!

**Basic Feasible Solution**

A **basic feasible solution** to a linear program is a basic solution that satisfies **all non-negativity constraints** (it also satisfies all other constraints… why?).

**FUNDAMENTAL RESULT**: The complete set of *basic feasible solutions* to ANY linear program in standard form are *EXACTLY* the collection of *ALL* extreme points of its feasible region.

**Class Workshop – Basic Feasible Solutions I**

Is the basic solution $\mathcal{B} = \{x_2, x_3\}$ in the previous workshop **feasible**? You may label it as feasible/infeasible in the workshop solution section above.

This is a significant result indeed! Since we have algebraic tests to check for basic feasible solutions, we are thus able to characterize all of the extreme points in a linear program **algebraically**. That is to say, we

can define all locations of **potential optima** (recall that any extreme point of a linear program is a potential optimum) without actually visualizing the region, which is a significant result and will allow us to solve linear programs much faster and (more importantly!) in > 3D problems.

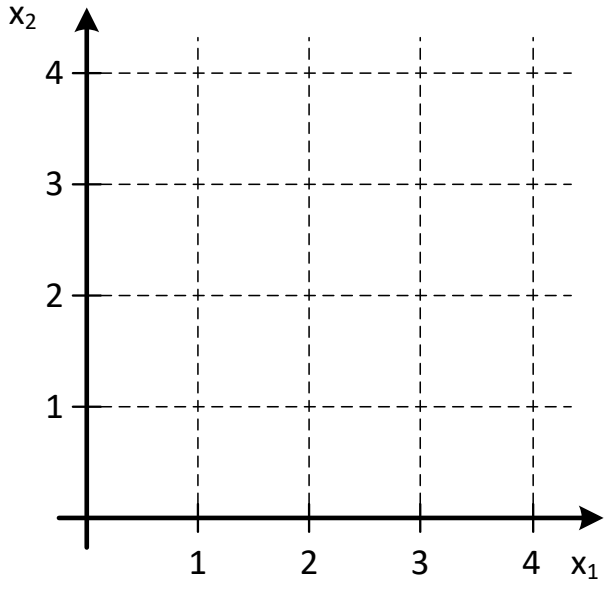
## Class Workshop – Basic Feasible Solutions II

Consider the following set of linear optimization program constraints and answer the questions:

1. Plot the feasible region and indicate the extreme points.
2. Rewrite the problem in standard form using $x_j : j > 2$ as slack variables (it is assumed they will appear in the same order as the constraints).
3. Compute the basic solutions for the following sets of basic variables, and determine which of these are basic feasible solutions. Indicate on the feasible region the locations of each basic solution, even if they are infeasible.
   a. $\mathcal{B} = \{x_3, x_4, x_5\}$
   b. $\mathcal{B} = \{x_1, x_2, x_4\}$
   c. $\mathcal{B} = \{x_1, x_2, x_5\}$

$$\begin{aligned} -\boldsymbol{x_1} + \boldsymbol{x_2} &\geq \boldsymbol{0} \\ \boldsymbol{x_1} &\leq \boldsymbol{2} \\ \boldsymbol{x_2} &\leq \boldsymbol{3} \\ \boldsymbol{x_i} &\geq \boldsymbol{0} \qquad \forall\, \boldsymbol{i} \end{aligned}$$

## Workshop Solution – Basic Feasible Solutions II



# The Simplex Algorithm

The simplex algorithm is an application of the **improving search** as we learned way back in Module 01. However, it is adapted to handle the specific nature of linear programs with elegance and simplicity. This is the part where we learn how to compute the **feasible improving directions** and **step lengths** of the simplex algorithm. In order to do this as simply as possible, we must re-write our LP in standard form into what is called the **Simplex Tableau** (or "standard display).

### Simplex Tableau

The Simplex Tableau is a standard "block form" display of the variables $x$, coefficient matrix and vector $A$ and $b$, and cost coefficient vector $c$:

| | $x_1$ | $x_2$ | … | $x_n$ | |
|---|---|---|---|---|---|
| $\min \phi = \boldsymbol{c} \rightarrow$ | $c_1$ | $c_2$ | … | $c_n$ | $\boldsymbol{b}$ |
| $\boldsymbol{A}$ | $a_{1,1}$ | $a_{1,2}$ | … | $a_{1,n}$ | $b_1$ |
| | $a_{2,1}$ | $a_{2,2}$ | ⋱ | $a_{2,n}$ | $b_2$ |
| | ⋮ | … | ⋱ | ⋮ | ⋮ |
| | $a_{m,1}$ | … | … | $a_{m,n}$ | $b_m$ |

For example, the Simplex Tableau for the class workshop above with the objective function $\max \phi = x_1 + x_2$ is as follows:

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | |
|---|---|---|---|---|---|---|
| $\max \phi = \boldsymbol{c} \rightarrow$ | 1 | 1 | 0 | 0 | 0 | $\boldsymbol{b}$ |
| $\boldsymbol{A}$ | −1 | 1 | −1 | 0 | 0 | 0 |
| | 1 | 0 | 0 | 1 | 0 | 2 |
| | 0 | 1 | 0 | 0 | 1 | 3 |

## *Initializing the Simplex Algorithm*

OK, so all we need now is an initial solution. Recall that our Simplex algorithm wants to travel from one **extreme point** to another, and thus we need to start at an extreme point. Luckily for us, the extreme points of all LPs are exactly equal to their **basic feasible solutions**!

### Simplex Initialization

The Simplex Search begins at an **extreme point** of the feasible region. In other words, it begins at a **basic feasible solution** to the model in standard form.

For example, we begin the simplex search for this example by using the basic feasible solution $\mathcal{B}^{(0)} = \{x_2, x_4, x_5\}$ to find our initial point $x^{(0)}$ using the tableau as shown below. I must make the following additions/calculations:

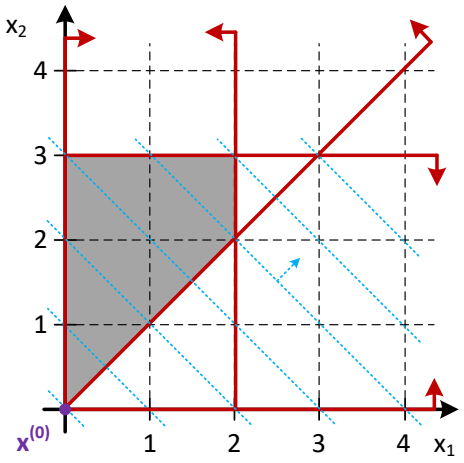
- I add a new row that shows the current set of basic (not fixed to zero) and nonbasic (fixed to zero) variables. I then solve the *SQUARE* system of equations $Ax = b$ using *only the basic variables*. In the case of the initial guess, this gives me the iterate $x^{(0)}$.
- I add one final row to the Tableau that shows the current values of the variables for the given iterate. Notice that the nonbasic variables (columns with "N") are zero by default.

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | |
|---|---|---|---|---|---|---|
| $\max \phi = \boldsymbol{c} \rightarrow$ | 1 | 1 | 0 | 0 | 0 | $\boldsymbol{b}$ |
| $\boldsymbol{A}$ | −1 | 1 | −1 | 0 | 0 | 0 |
| | 1 | 0 | 0 | 1 | 0 | 2 |
| | 0 | 1 | 0 | 0 | 1 | 3 |
| $\mathcal{B}^{(0)}$ | N | B | N | B | B | ← List Basic Variables Here |
| $x^{(0)}$ | 0 | 0 | 0 | 2 | 3 | $c^T x^{(0)} = 0$ |

Determine values for basic variables (set nonbasic variables to 0 and solve Ax = b)

Compute ϕ for current iterate (basic/nonbasic)

The iteration $x^{(0)}$ as shown in the Simplex Tableau above can readily be seen to be the point in the bottom-left corner of the feasible region as shown in the figure below:




## *Determining the Simplex Direction*

Now that we know where our initial point it, we need to determine the direction that results in traveling along one of the two constraints (in this 2D example) that define our current basic feasible solution (extreme point). How do we do this? Well! Recall that our inequality constraints as they define our feasible region have been reformulated as equality constraints (through the addition of slack and surplus variables). Thus, while we are trying to identify the edge directions that join the current extreme point to an adjacent one, we can remember this fact:

- The active constraints at a basic feasible solution correspond to the **non-negativity constraints on all nonbasic variables**. What does this *mean*? Well, if all variables are positive ($x_i \geq 0$) in our standard program, and we set 2 of our variables to be nonbasic (*ie*, fixed them to zero), we have *forced the non-negativity constraints for the nonbasic variables to be active*.

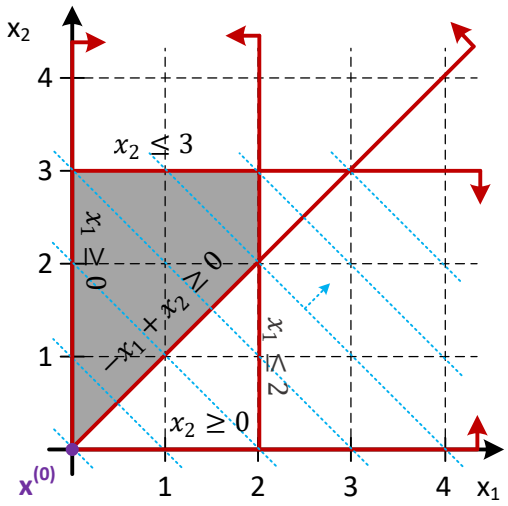
In the specific case above, with the initial basic feasible solution $\mathcal{B} = \{x_2, x_4, x_5\}$, we are forcing $x_1 = 0$ and $x_3 = 0$. Moreover, **recall that $x_3$** was the **invented surplus** subtracted from the constraint:

$$-x_1 + x_2 \geq 0$$

In other words, if I make the variable $x_3 > 0$, I am ***relaxing*** the inequality constraint above. Furthermore, if I make the variable $x_1 > 0$, I am ***relaxing*** the constraint $x_1 \geq 0$.

Here is the important part: If I change either $x_1$ or $x_3$ to be greater than zero (which, in effect, would add them to the basis), I must maintain the activity of the other constraint. In other words, at the current basic feasible solution I am **permitted two moves, both of which will take me along a constraint defining the edge of the feasible region:**

1. I can take some step $\Delta x_1$, which means I *deactivate the* $x_1 \geq 0$ *constraint* and thus am no longer stuck on the $x_2$ axis. This is equivalent to travelling along the $-x_1 + x_2 \geq 0$ constraint as shown in the figure below.
2. I can take some step $\Delta x_3$, which means I *deactivate the* $-x_1 + x_2 \geq 0$ *constraint* and thus am no longer stuck on the 45° line defining that constraint. This is equivalent to travelling along the $x_1 \geq 0$ constraint (along the $x_2$ axis) as shown in the figure below.




THIS EXAMPLE allows us to conclude that, from **any basic feasible solution**, all possible search paths correspond to **taking a step in the direction of all nonbasic variables**!

## Simplex Direction

Let $\mathcal{B}$ be the current basic feasible solution, and consider the **nonbasic variable** $x_j \notin \mathcal{B}$:

$$\Delta x_j = 1 \quad \& \quad \Delta x_k = 0 \quad \forall x_k \notin \mathcal{B}, k \neq j$$

Is a potential Simplex Search direction that deactivates *one* constraint at a basic feasible solution and moves along the corresponding *edge*.

Candidate Simplex directions must also maintain the **feasibility** of the constraints. In other words, if my original constraints in standard form were satisfied, the constraints after my move $\Delta x$ must also be satisfied, or:

$$\left.\begin{aligned} Ax &= b \\ A(x + \Delta x) &= b \end{aligned}\right\} \Rightarrow \boxed{\boldsymbol{A\Delta x = 0}}$$

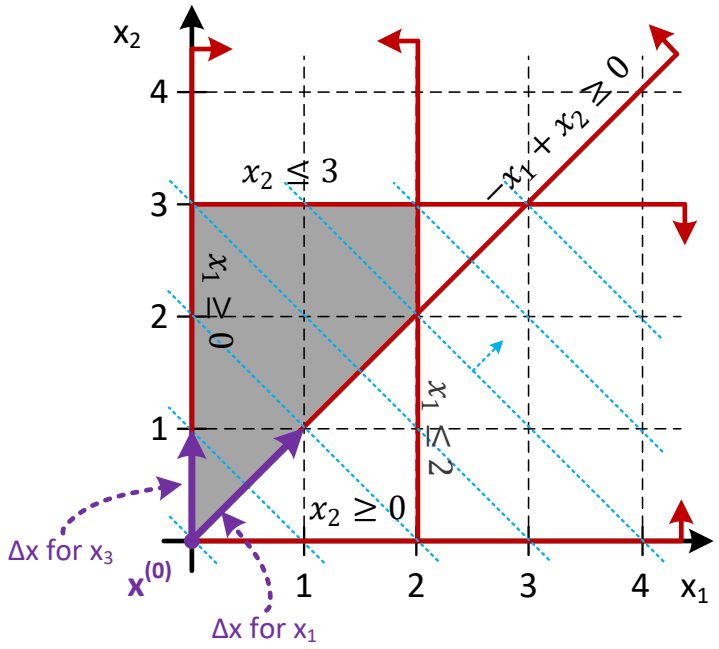
**Fundamental Result: All Simplex Search directions** are constructed by increasing a single nonbasic variable by 1, leaving the other nonbasic variables the same (*ie*, 0), and computing the (unique) corresponding change in *basic variables* necessary to ensure $A\Delta x = 0$.

**Class Workshop – Simplex Directions**
Compute the Simplex Search directions $\Delta x$ by adding $x_1$ and $x_3$ to the basis.

**Workshop Solution – Simplex Directions**

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | |
|---|---|---|---|---|---|---|
| $\max \phi = \boldsymbol{c} \rightarrow$ | 1 | 1 | 0 | 0 | 0 | $\boldsymbol{b}$ |
| $\boldsymbol{A}$ | $-1$ | 1 | $-1$ | 0 | 0 | 0 |
| | 1 | 0 | 0 | 1 | 0 | 2 |
| | 0 | 1 | 0 | 0 | 1 | 3 |
| $\mathcal{B}^{(0)}$ | N | B | N | B | B | |
| $x^{(0)}$ | 0 | 0 | 0 | 2 | 3 | $c^T x^{(0)} = 0$ |
| $\Delta x$ for $x_1$ | **1** | | **0** | | | |
| $\Delta x$ for $x_3$ | **0** | | **1** | | | |

We can readily see how these changes will be reflected on our plot by only looking at the changes for the **non-slack/surplus variables** ($x_1$ and $x_2$ in this case), as shown in the figure below.



## *Selecting a Direction to Follow*

Selecting a Simplex Search direction to follow is actually pretty straightforward. All we need to do it see if our $\Delta x$ values result in an *improving objective function*. Recall that the objective function is:

$$\phi = c^T x$$

Therefore we can compute the *change* in objective function for some step $\Delta x$ by simply computing:

$$\Delta\phi = c^T \Delta x \qquad \bar{\boldsymbol{\phi}} = \boldsymbol{c^T \Delta x}$$

The value $\bar{\phi}$ is known as the **reduced cost** of a move in the direction $\Delta x$. This will let us determine which of our Simplex directions are *improving*.

**Improving Simplex Direction**

A candidate Simplex Search direction $\Delta x$ obtained by increasing a nonbasic variable $x_j$ is **IMPROVING** if:

- $\bar{\phi} > 0$ for a **maximization** problem.
- $\bar{\phi} < 0$ for a **minimization** problem.

**Class Workshop – Improving Simplex Directions**

Determine if the Simplex Search directions $\Delta x$ by adding $x_1$ and $x_3$ to the basis are improving.

**Workshop Solution – Simplex Directions**

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | |
|---|---|---|---|---|---|---|
| $\max \phi = \boldsymbol{c} \rightarrow$ | 1 | 1 | 0 | 0 | 0 | $\boldsymbol{b}$ |
| $\boldsymbol{A}$ | $-1$ | 1 | $-1$ | 0 | 0 | 0 |
| | 1 | 0 | 0 | 1 | 0 | 2 |
| | 0 | 1 | 0 | 0 | 1 | 3 |
| $\mathcal{B}^{(0)}$ | N | B | N | B | B | |
| $x^{(0)}$ | 0 | 0 | 0 | 2 | 3 | $c^T x^{(0)} = 0$ |
| $\Delta x$ for $x_1$ | **1** | | **0** | | | $\bar{\phi}_1 =$ |
| $\Delta x$ for $x_3$ | **0** | | **1** | | | $\bar{\phi}_3 =$ |

It is worth noting here that there is a chance *all* simplex directions are improving (looking at our feasible region plot, it should be obvious that this is possible). It is also (more likely) possible that some Simplex directions improve while other do not (think about what happens after one iteration?).

When performing the Simplex Search, we only need to choose **any one direction** $\Delta x$ that results in an improving objective function (it actually does not change the outcome no matter which one you pick… think WHY!). The next final that has to be asked is: HOW FAR do we follow this direction? Recall that our basic improving search method required that we follow any direction $\Delta x$ for some length $\alpha$. Once we determine this step length, we can proceed to the next iteration of the Simplex Search (our algorithm is complete).

## *Determining Simplex Direction Step Length*

When addressing the step length for any direction $\Delta x$, we have a very useful property of the set of basic feasible solutions that we can employ. **The limit to the maximum allowable step size $\alpha$ can only come from violating a non-negativity constraint**. Now, ask yourself: WHY is this so? I will give you some space to write it out:

When determining the maximum allowable step, we consider **only the basic variables that are decreasing for the current $\Delta \boldsymbol{x}$**. We only consider *decreasing* elements because we want to see which one of them will become zero (and thus activate its non-negativity constraint) first. We are now able to compute the maximum allowable step:

### Maximum Allowable Step

The maximum allowable step for the Simplex Search in the direction $\Delta x$ is limited to the *decreasing* basic variable that will become 0 first when travelling in that direction:

$$\alpha = \min\left\{\frac{x_j^{(k)}}{-\Delta x_j} : \Delta x_j < 0\right\}$$

### Class Workshop – Making the Next Move

Assume we have selected $\Delta x$ for $x_1$ in the examples above. Determine the maximum allowable step size in this direction and compute the next iteration $x^{(1)}$.

### Workshop Solution – Making the Next Move

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | |
|---|---|---|---|---|---|---|
| $\max \phi = \boldsymbol{c} \rightarrow$ | 1 | 1 | 0 | 0 | 0 | $\boldsymbol{b}$ |
| $\boldsymbol{A}$ | $-1$ | 1 | $-1$ | 0 | 0 | 0 |
| | 1 | 0 | 0 | 1 | 0 | 2 |
| | 0 | 1 | 0 | 0 | 1 | 3 |
| $\mathcal{B}^{(0)}$ | N | B | N | B | B | |
| $x^{(0)}$ | 0 | 0 | 0 | 2 | 3 | $c^T x^{(0)} = 0$ |
| $\Delta x$ for $x_1$ | **1** | **1** | **0** | **$-1$** | **$-1$** | $\bar{\phi}_1 = 2$ |
| $\frac{x^{(0)}}{-\Delta x}$ | | | | | | $\alpha =$ |

## *The Final Step: Updating the Basis*

And finally, we may proceed to the next iteration of our Simplex Search by determining a **new basic feasible solution**. This is actually quite easy. Since we increased the value of one nonbasic variable via the move $\Delta x$ AND we ended up fixing one basic variable to zero due to the step size $\alpha$, we simply swap the first variable into the basic solution and the second variable out!

**Updating the Basic Feasible Solution**

After each step of the Simplex Search:

1. The **NONBASIC** variable changed to determine the search direction becomes **BASIC**
2. The **BASIC** variable (or, if there were more than one, *any one of them*) that was the limiting factor when selecting the step size becomes **NONBASIC**.

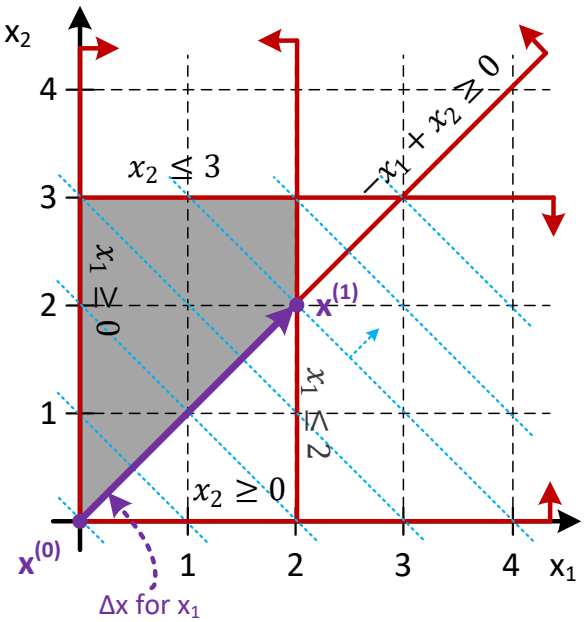
**Class Workshop – Updating the Basic Feasible Solution**

Determine the new basic feasible solution $\mathcal{B}^{(1)}$ for our example.

**Workshop Solution – Updating the Basic Feasible Solution**

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| $\mathcal{B}^{(0)}$ | N | B | N | B | B |
| $\mathcal{B}^{(1)}$ | | | | | |

Well, that's it! I know it might seem a little tedious, but I promise that it is actually not so bad once you go through it a couple of times. What follows now is the official algorithm statement. Feel free to check out the sections above for a refresher on any of the steps as you need them. For your interest, the plot of our problem with the first iteration is below.



# Algorithm Statement: Rudimentary Simplex Search

1. **Initialize**
   Choose any starting basic feasible solution $\mathcal{B}^{(0)}$.
   Construct the starting point $x^{(0)}$ with nonbasic variables fixed to 0.
   Define the iteration index $k = 0$.

2. **Determine Simplex Direction**
   Construct candidate directions $\Delta x_j$ associated with increasing each nonbasic variable $x_j$.
   Compute the reduced cost $\bar{\phi} = c^T \Delta x$ for each direction.
   If no Simplex direction is *improving*, **STOP** → The current solution $x^{(k)}$ is **globally optimal**.
   Otherwise, select *any* improving direction $\Delta x$ and assign its associated nonbasic variable $x_e$.

3. **Compute Step Size**
   If there is no limit on feasible moves in the direction $\Delta x$, **STOP** → Model is **unbounded**.
   Otherwise, determine $\alpha = \min\left\{\frac{x_j^{(k)}}{-\Delta x_j} : \Delta x_j < 0\right\}$.
   Denote the basic variable used to determine $\alpha$ as $x_l$.

4. **Update**
   Compute the new solution point: $x^{(k+1)} = x^{(k)} + \alpha \Delta x$.
   Compute the new basic solution: $\mathcal{B}^{(k+1)} = \mathcal{B}^{(k)} \cup \{x_e\} \setminus \{x_l\}$ ($x_e$ enters basis, $x_l$ leaves).
   Increment $k = k + 1$ and return to step (2).

There you have it!! Let's continue our search from before and finish this bad boy off.

**Class Workshop – Continuing the Simplex Search**
Continue the Simplex Search for our ongoing example to locate the optimum.

**Workshop Solution – Continuing the Simplex Search**

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | |
|---|---|---|---|---|---|---|
| $\max \phi = \boldsymbol{c} \rightarrow$ | 1 | 1 | 0 | 0 | 0 | $\boldsymbol{b}$ |
| $\boldsymbol{A}$ | $-1$<br>1<br>0 | 1<br>0<br>1 | $-1$<br>0<br>0 | 0<br>1<br>0 | 0<br>0<br>1 | 0<br>2<br>3 |
| $\mathcal{B}^{(1)}$ | B | B | N | N | B | |
| $x^{(1)}$ | 2 | 2 | 0 | 0 | 1 | $c^T x^{(1)} = 4$ |
| $\Delta x$ for $x\_$ | | | | | | $\bar{\phi}\_ =$ |
| $\Delta x$ for $x\_$ | | | | | | $\bar{\phi}\_ =$ |
| $\frac{x^{(1)}}{-\Delta x}$ | | | | | | $\alpha =$ |

# Locating an Initial Basic Feasible Solution

OK, so we now know how to solve any linear program via the very clever method known as the Simplex Search. However, there should still be *one* question that needs answering… "How do I find that initial basic feasible solution to START the Simplex Search?"

This is a very good and a very important question, because without the ability to graphically interpret a scenario, we are left with the problem that a basic solution is rather easy to come by (we set $n-m$ variables to be nonbasic and solve for the rest), but a basic *FEASIBLE* solution is a little more tricky. We do not want to just guess-and-check combinations of basic solutions until we happen to find a feasible one, because this could take very, very long (think why?). There is a better method. As a matter of fact, that better method involves the Simplex Search. I would like to make a proposal to you:

> *One possible method of finding an initial basic FEASIBLE solution is to set up a completely different linear program that attempts to satisfy the constraints of the original problem by setting a NEW set of variables, originally basic, to nonbasic (thus eliminating the new variables from the problem completely). If I can easily find an initial basic feasible solution to this new problem, I can use the Simplex Search to find its optimum, which corresponds to a basic FEASIBLE solution to the original problem.*

To which you will probably say something along the lines of:

OK, here's the thing – it was a mouthful, but it is **actually not as hard as it may seem.** Let's see how we do it, shall we?

## *Artificial Linear Programs*

Recall our LP in standard form:

$$\min_{\boldsymbol{x}} \phi = \boldsymbol{c}^T\boldsymbol{x} \quad \leftarrow \text{Objective Function}$$

$$s.t. \quad \leftarrow \text{"Subject to"}$$

$$A\boldsymbol{x} = b \quad \leftarrow \text{ONLY Equality Constraints}$$

$$x \geq 0 \quad \leftarrow \text{Variable Bounds}$$

Our **BASIC FEASIBLE SOLUTION** had the following characteristics:

- $(n-m)$ elements of $x$ are set to 0 (nonbasic variables)
- The remaining $m$ variables must therefore be non-negative and satisfy $Ax = b$ (basic variables)

Here is an **idea**: What if I just increase the number of variables by $m$ (that is, add one new variable per constraint). The resulting LP will be *different*, BUT I can choose all of those new fictitious variables to be **BASIC**, and set ALL of the original variables in my problem to be non-basic (just fix them to 0). This is a basic feasible solution to my problem, but the objective does not depend on my new variables at all. This is not a problem, because I can define a **new objective function** that is just the sum of my fictitious variables. Here is the key concept:

**If my new objective function attempts to minimize the fictitious variables, I KNOW that the optimum of this new LP is $\phi = 0$ and corresponds to all of my fictitious variables being 0. However, I also know that the values for the non-fictitious variables, since they are subject to the same constraints as the artificial LP, represent a basic feasible solution to the original LP.**

## Artificial Linear Program

An artificial LP is constructed so that it **minimizes the sum of the constraint violations** for the equality constraints in the original LP. It does this by ADDING one new variable to each constraint:

**Original Standard Form Constraints**

$$\begin{array}{cccc} a_{1,1}x_1 + & \dots & +a_{1,n}x_n & = b_1 \\ a_{2,1}x_1 + & \dots & +a_{2,n}x_n & = b_2 \\ \vdots & \ddots & \vdots & = \vdots \\ a_{m,1}x_1 + & \dots & +a_{m,n}x_n & = b_m \end{array}$$

**NEW Standard Form Constraints**

$$\begin{array}{ccccccccc} a_{1,1}x_1 + & \dots & +a_{1,n}x_n & \pm \boldsymbol{x_{n+1}} & & & & = & b_1 \\ a_{2,1}x_1 + & \dots & +a_{2,n}x_n & & \pm \boldsymbol{x_{n+2}} & & & = & b_2 \\ \vdots & \ddots & \vdots & & & \ddots & & = & \vdots \\ a_{m,1}x_1 + & \dots & +a_{m,n}x_n & & & & \pm \boldsymbol{x_{n+m}} & = & b_m \end{array}$$

- The artificial variable $x_{n+j}$ is **ADDED** if $b_j \geq 0$.
- The artificial variable $x_{n+j}$ is **SUBTRACTED** if $b_j < 0$.

The new objective function is now:

$$\boxed{\min_{\boldsymbol{x}} \boldsymbol{v} \triangleq \boldsymbol{x_{n+1}} + \boldsymbol{x_{n+2}} + \cdots + \boldsymbol{x_{n+m}}}$$

- This artificial LP is already in standard form.
- $\mathcal{B}^{(0)} = \{x_{n+1} \dots x_{n+m}\}$ is a **basic feasible solution** (do you see why)?
- If we solve the artificial LP, is it **guaranteed to have an optimal objective function of 0** if the original problem is feasible.
- The location of the optimum $x^\star$ for the artificial LP represents a basic feasible solution to the original LP. OR, $x_A^\star \triangleq \mathcal{B}_O$ (optimal solution to **A**rtificial is a basic solution to **O**riginal).
- We may use the optimal solution to the artificial program as the **initial point** for the original LP!

The introduction of this method allows us to finally derive an algorithm that will work for ANY LP without the *a priori* need of a basic feasible solution.

# Algorithm Statement: Two-Phase Simplex Search

1. **PHASE I**
   Derive the artificial LP model.
   Apply the Simplex Search algorithm to the artificial LP.

2. **Infeasibility**
   If Phase I terminates with an optimal objective $v^\star > 0$, **STOP**. The original LP is **INFEASIBLE**.
   Otherwise, use the optimum of the artificial LP $x^\star$ as the initial basic solution to the original LP.

3. **PHASE II**
   Apply the Simplex Search to the original LP with the given $\mathcal{B}^{(0)}$ to solve for the optimum or prove that the problem is unbounded.

## Class Workshop – Formulating Artificial LPs

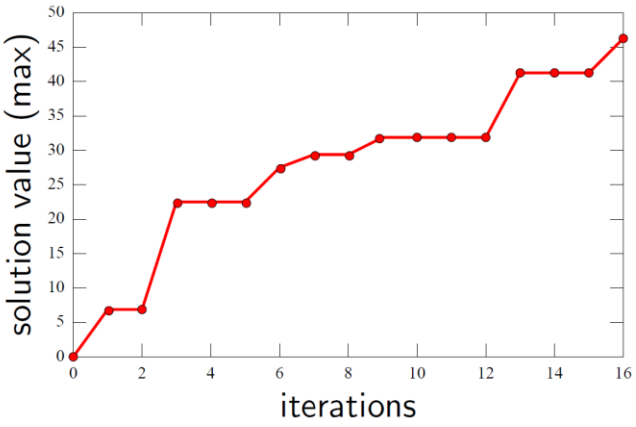
Consider the following LP in standard form. Formulate the artificial LP to find an initial basic feasible solution to the LP below and identify the artificial LP's initial basic feasible solution.

$$\begin{aligned} \max_{x} \boldsymbol{\phi} \quad &= \quad \boldsymbol{x_1 + x_2} \\ \boldsymbol{-x_1 + x_2 - x_3} \quad &= \quad \boldsymbol{0} \\ \boldsymbol{x_1 + x_4} \quad &= \quad \boldsymbol{2} \\ \boldsymbol{x_2 + x_5} \quad &= \quad \boldsymbol{3} \\ \boldsymbol{x_i} \quad &\geq \quad \boldsymbol{0} \qquad\qquad \forall\, \boldsymbol{i = 1 \ldots 5} \end{aligned}$$

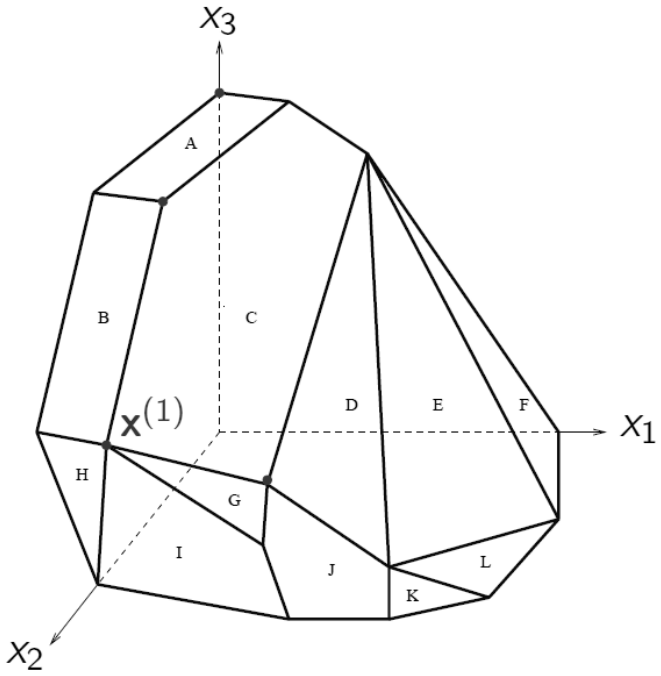
## Workshop Solution – Formulating Artificial LPs

## Iteration Degeneracy

It is not always guaranteed that the Simplex Search will locate a new extreme point with a better objective function value at each iteration. In fact, it is more common that the Simplex Search will encounter periods of no advance (note that it can NEVER get worse). For example, see the figure below (courtesy of Dr. Chachuat) that shows the value of $\phi$ during a series of maximization iterations using the Simplex Search:



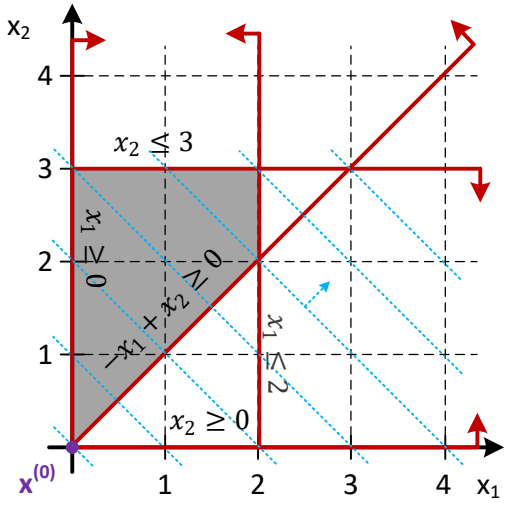

This happens because, like we saw in the previous module, it is possible for us to have a corner point defined by **more than the minimum required number of active constraints**. Recall this figure:




The point $x^{(1)}$ requires only $n = 3$ constraints to define it, but it occurs at the intersection of five constraints! If our direction of improving objective value happens to run along a set of two intersecting constraints **not corresponding to our basis**, we will result in a maximum allowable step length of $\alpha = 0$. This is what is called a **degenerate** solution. For example, at $x^{(1)}$ in the figure above, our improving direction may go along the edge defined by constraints B and C. However, if our basis corresponds to the intersection of constraints H, I, and G, it will not permit us to move along the improving edge. Is this going to break the Simplex Search? Well, maybe. But it is extremely unlikely.

**Degeneracy**
A basic feasible solution to a standard-form LP is known as **degenerate** if the non-negativity constraints for some BASIC variables are active (in other words, more variables are 0 than only the nonbasic variables).

For example, check out the figure of our ongoing example:




If I chose the initial basic feasible solution $\mathcal{B} = \{x_3, x_4, x_5\}$ with $x_1$ and $x_2$ nonbasic, it corresponds to the same initial point $x^{(0)}$ as before. However, since my nonbasic variables correspond to the constraints on the $x_1$ and $x_2$ axes, I can't take a step along either of those constraints! This corresponds to the Tableau:

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | |
|---|---|---|---|---|---|---|
| $\max \phi = \boldsymbol{c} \rightarrow$ | 1 | 1 | 0 | 0 | 0 | $\boldsymbol{b}$ |
| $\boldsymbol{A}$ | −1 | 1 | −1 | 0 | 0 | 0 |
| | 1 | 0 | 0 | 1 | 0 | 2 |
| | 0 | 1 | 0 | 0 | 1 | 3 |
| $\mathcal{B}^{(0)}$ | N | N | B | B | B | |
| $x^{(0)}$ | 0 | 0 | 0 | 2 | 3 | $c^T x^{(0)} = 0$ |
| $\Delta x$ for $x_1$ | **1** | **0** | **−1** | **−1** | **0** | $\bar{\phi}_1 = 1$ |
| $\frac{x^{(0)}}{-\Delta x}$ | – | – | **0** | **2** | – | $\boldsymbol{\alpha = 0}$ |
| $\mathcal{B}^{(1)}$ | **B** | N | **N** | B | B | |
| $x^{(1)}$ | 0 | 0 | 0 | 2 | 3 | $c^T x^{(1)} = 0$ |

The objective function value did **not change** in this example, nor did the solution $x$. In this case, the direction $\Delta x$ for $x_1$ is in fact an improving direction, but the Simplex Search may not proceed in this direction because it will result in the violation of the constraint for $x_3$. This is not an issue; we may update the basis after taking a step of length $\alpha = 0$ and proceed as normal with the next iteration.

This leads us to conclude that unless we are very unlucky, the Simplex Search will **naturally work its way out of a degenerate solution** if we treat a zero-length iteration as a normal iteration. See the following rules for dealing with degeneracy.

**Rules for Dealing with Degeneracy**
If a degenerate solution results in a step size of $\alpha = 0$:

- Swap out the basic/nonbasic variables as if a positive step had been taken.
- Continue computations as if a positive step was taken.

The **ONLY** time this will fail is if we keep swapping the *same* basic/nonbasic variables over and over again. This is referred to as **cycling**. The good news is that most practical optimization problems will not run into this issue. Moreover, commercial solvers using Simplex-based methods (such as CPLEX) are very clever and will remember recent basic feasible solutions and their objectives, and will avoid re-entering the same basis it has already visited, thus avoiding the problem entirely. With this added layer of safety, we can conclude the following

**Finite Convergence**
The Simplex Search, with all step sizes $\alpha > 0$ **or** logic to prevent cycling, is GUARANTEED to converge on the optimum (or prove unboundedness) after **finitely** many iterations (no tolerance required).

## Conclusions

WHEW! It was a long road, but here we are. We have finally algebraically derive a method that will solve all linear programs, even without the presence of an initial basic feasible solution, in a finite number of iterations. It is a little hairy, I agree, but the good news is that, like all numerical methods, we only really need to understand it **once** to be able to apply it generally for all problems.

Hopefully you found this blend of linear algebra, geometry, and linear programming principles as interesting as I do! For perspective, it is safe to assume I would not ask you to complete a Simplex Search by hand on a test (assignments for sure though!). I could ask you to perform one step, plot what would happen, or ask you qualitative questions about it, however.

*~~ END OF MODULE ~~*

\|T|/