In-class Exercise 5.3
Linear programming
AMATH 301
University of Washington
Jakob Kotas

A linear program has an objective function of the form:

$$\min \sum_{i=1}^{n} c_i x_i$$

where x_i are the variables and c_i are constant coefficients. We focus solely on the minimization case, as the maximization problem can always be converted into a minimization problem (maximizing $f(x_1, x_2, ..., x_n)$ is equivalent to minimizing $-f(x_1, x_2, ...x_n)$.)

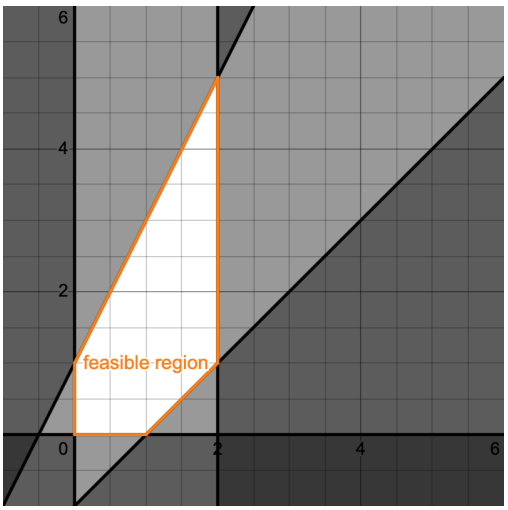
Linear inequality constraints can be put in the form:

$$\sum_{i=1}^{n} a_i x_i \leq b$$

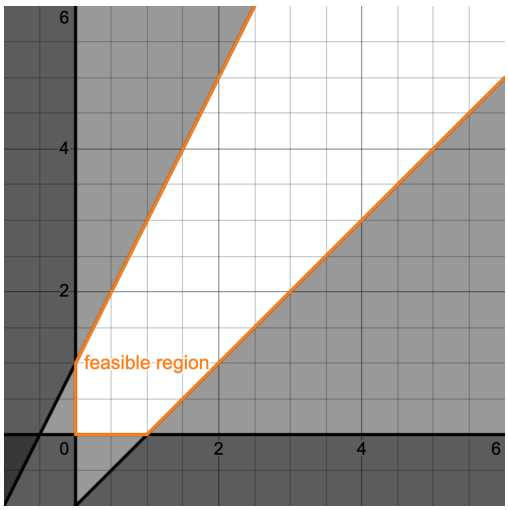
Linear equality constraints of the form:

$$\sum_{i=1}^{n} a_i x_i = b$$

Consider a linear program with two variables. In 2D, the feasible region (set of points which satisfy the constraints) will be a "convex" polygon. We say a polygon is convex if there does not exist a set of two points within the polygon such that the whole line segment between those two points is contained within the polygon. An equivalent definition is that the polygon's interior angles are all $\leq 180°$. Feasible regions can be bounded, meaning they fit in a finite box $a \leq x \leq b$, $c \leq y \leq d$, or unbounded. Examples in 2D are shown below, but these concepts generalize to higher dimensions (convex polytope).



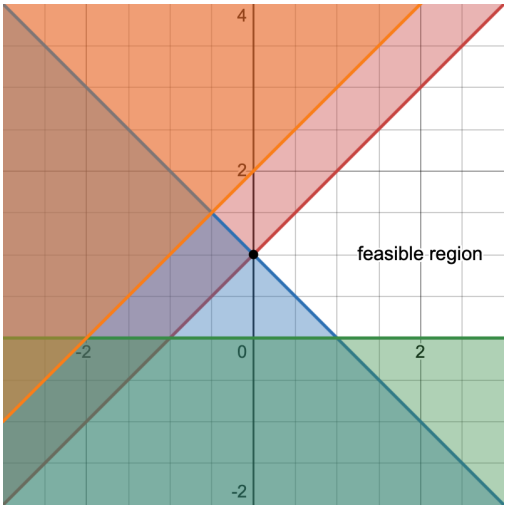
Bounded feasible region

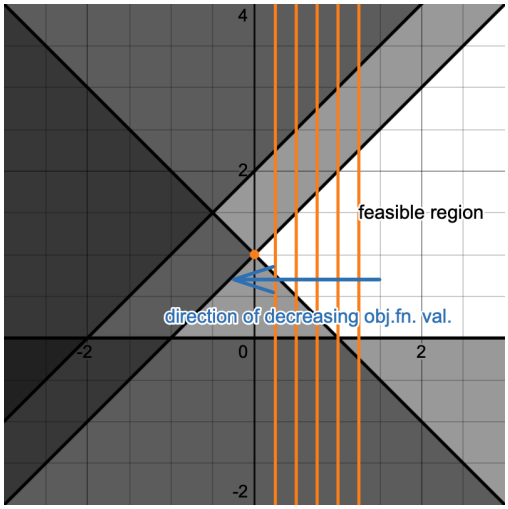


Unbounded feasible region

Within the feasible region (domain), we wish to find a point where the objective function is minimized. The set of such points is guaranteed to include a vertex of the feasible region (basic feasible point), so we restrict our search to vertices.

In the left picture below, we wish to minimize x within the feasible region. This occurs at $(0, 1)$. An inequality constraint is active if the two sides are equal at the solution. The red and blue constraints are active in this picture. A constraint is redundant if removing it does not affect the feasible region at all. The orange constraint is redundant in this picture. The green constraint is not active at the solution, but it is also not redundant.





In the right picture above (same constraints and objective function), you can visualize moving toward the solution by plotting lines of constant objective function value: $x = c$. By moving from line to line, you can see where the solution is (last point(s) within the feasible region when moving leftward).

1. We seek to decide the number of units of different foods to consume every day so we meet the minimum daily requirement (MDR) of different nutrients at minimum cost.

	wheat	rye	MDR
carbs/unit	5	7	10
protein/unit	4	1	4
vitamins/unit	2	1	3
cost/unit	0.6	0.4	

Let x be the number of units of wheat and y be the number of units of rye to consume every day. The LP becomes

$$\min 0.6x + 0.4y$$

subject to

$$5x + 7y \geq 10$$

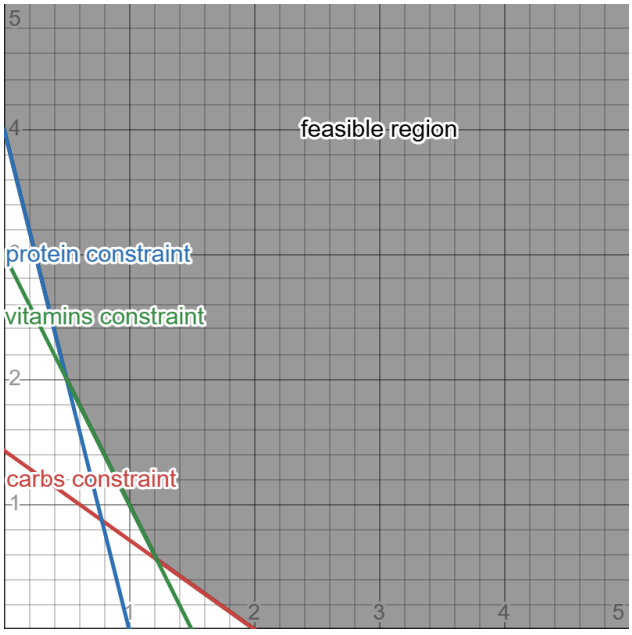
$$4x + y \geq 4$$

$$2x + y \geq 3$$

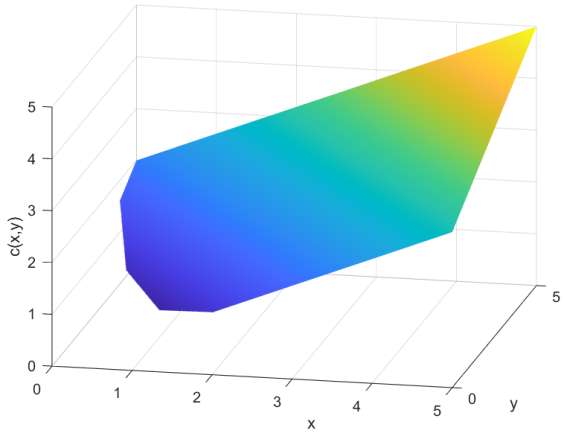
$$x \geq 0$$

$$y \geq 0$$

Notice the feasible region is unbounded, as x and y can be arbitrarily large. First we plot the feasible region alone, defined as the region above the three lines in the plot:



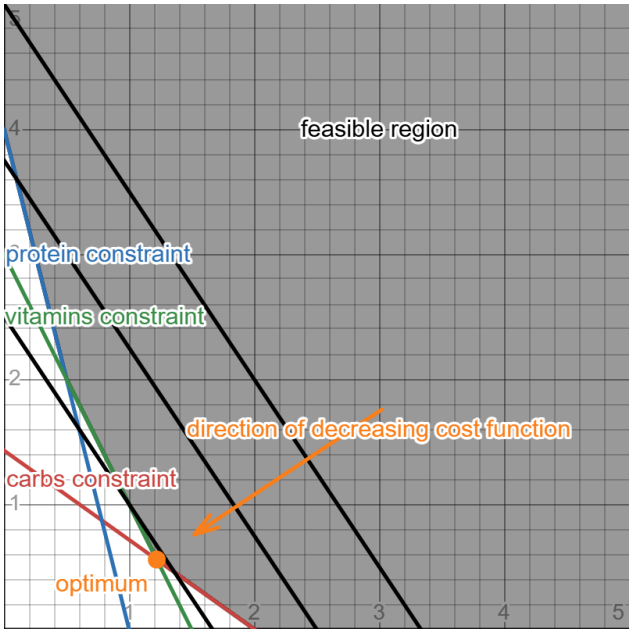
We see that none of the constraints are redundant and there are 4 non-degenerate vertices at $(0, 4)$, $(0.5, 2)$, $\approx (1.222, 0.556)$, $(2, 0)$. Since the feasible region is unbounded, the cost could be unbounded, or it could reach an optimum at one of the vertices. (If there is a tie between two vertices, the whole line segment attaching the vertices are optima). We can use a third dimension to plot the value of the cost function $z = c(x, y) = 0.6x + 0.4y$ and then choose the point with the lowest z-coordinate.



Visually it appears that the optimum is at $\approx (1.222, 0.556)$. Another way that we can arrive at this answer without resorting to 3D graphs is to plot contours of the cost function:

$$c(x, y) = 0.6x + 0.4y = C$$

for various values of C within the feasible region. These are parallel lines. As C decreases, the lines sweep through the feasible region until they are just about to exit. We can visually see which vertex (or vertices) is the last one standing.



2. We present a more complex problem in 4 dimensions. Say we have four metals A,B,C,D.

metal	density	carbon %	phosphorus %	price $/kg
A	6500	0.2	0.05	2
B	5800	0.35	0.015	2.5
C	6200	0.15	0.065	1.5
D	5900	0.11	0.1	2.0

We want to make an alloy at minimum cost having the following requirements:

range	density	carbon %	phosphorus %
min	5950	0.1	0.045
max	6050	0.3	0.055

We will decide the amount of x_A, x_B, x_C, x_D per 1 kg of alloy. The LP is:

$$\min 2x_A + 2.5x_B + 1.5x_C + 2x_D$$

subject to

$$6500x_A + 5800x_B + 6200x_C + 5900x_D \geq 5950$$
$$6500x_A + 5800x_B + 6200x_C + 5900x_D \leq 6050$$
$$0.2x_A + 0.35x_B + 0.15x_C + 0.11x_D \geq 0.1$$
$$0.2x_A + 0.35x_B + 0.15x_C + 0.11x_D \leq 0.3$$
$$0.05x_A + 0.015x_B + 0.065x_C + 0.1x_D \geq 0.045$$
$$0.05x_A + 0.015x_B + 0.065x_C + 0.1x_D \leq 0.055$$
$$x_A + x_B + x_C + x_D = 1$$
$$x_A \geq 0$$
$$x_B \geq 0$$

$$x_C \geq 0$$
$$x_D \geq 0$$

(Why don't we need the constraints $x_A, x_B, x_C, x_D \leq 1$? Because it would be redundant.) Now we are in too high of a dimensional space to see graphically where the vertices are, much less which one would minimize the objective function value.

(a) Use `linprog`, in the `scipy.optimization` library, to solve the LP.

The LP must be put into a specific format that the solver can understand. First we write the objective function as the product of a row vector containing the weights and a column vector containing the variables:

$$\begin{bmatrix} 2 & 2.5 & 1.5 & 2 \end{bmatrix} \begin{bmatrix} x_A \\ x_B \\ x_C \\ x_D \end{bmatrix} = 2x_A + 2.5x_B + 1.5x_C + 2x_D$$

The row vector of weights is an input to the LP solver. We assume that the problem is a minimization problem; if it is a maximization problem, we must append a negative to the objective function.

The inequality constraints are placed into a matrix, multiplied by a column vector of variables, with the right side being another column vector of values. We assume that all inequality constraints are $\leq$. Any inequality constraints that are $\geq$ must be multiplied through by -1 to turn them into $\leq$ constraints.

$$\begin{bmatrix} -6500 & -5800 & -6200 & -5900 \\ 6500 & 5800 & 6200 & 5900 \\ -0.2 & -0.35 & -0.15 & -0.11 \\ 0.2 & 0.35 & 0.15 & 0.11 \\ -0.05 & -0.015 & -0.065 & -0.1 \\ 0.05 & 0.015 & 0.065 & 0.1 \end{bmatrix} \begin{bmatrix} x_A \\ x_B \\ x_C \\ x_D \end{bmatrix} \leq \begin{bmatrix} -5950 \\ 6050 \\ -0.1 \\ 0.3 \\ -0.045 \\ 0.055 \end{bmatrix}$$

The matrix and right-side vector are inputs to the LP solver.

Finally the equality constraints are placed into another matrix, multiplied by a column vector of variables, with the right side being a column vector of values. For our problem the matrix is only one row since we have only one equality constraint.

$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_A \\ x_B \\ x_C \\ x_D \end{bmatrix} = \begin{bmatrix} 1 \end{bmatrix}$$

Again the matrix and right-side vector are inputs.

Lastly, we specify bounds for each individual variable.

$$0 \leq x_A$$
$$0 \leq x_B$$

$$0 \leq x_C$$

$$0 \leq x_D$$

For this problem we have only lower bounds, but upper bounds can be handled as well.

(b) Reformulate the problem by introducing slack variables to change the inequality constraints to equality constraints. Show that `linprog` yields the same result.

Since there are 6 inequality constraints (not counting the non-negativity constraints), we introduce 6 slack variables $x_{S1}, x_{S2}, \cdots, x_{S6}$. The new objective function is:

$$\begin{bmatrix} 2 & 2.5 & 1.5 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_A \\ x_B \\ x_C \\ x_D \\ x_{S1} \\ x_{S2} \\ x_{S3} \\ x_{S4} \\ x_{S5} \\ x_{S6} \end{bmatrix}$$

The equality constraints are:

$$\begin{bmatrix} -6500 & -5800 & -6200 & -5900 & 1 & 0 & 0 & 0 & 0 & 0 \\ 6500 & 5800 & 6200 & 5900 & 0 & 1 & 0 & 0 & 0 & 0 \\ -0.2 & -0.35 & -0.15 & -0.11 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0.2 & 0.35 & 0.15 & 0.11 & 0 & 0 & 0 & 1 & 0 & 0 \\ -0.05 & -0.015 & -0.065 & -0.1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0.05 & 0.015 & 0.065 & 0.1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_A \\ x_B \\ x_C \\ x_D \\ x_{S1} \\ x_{S2} \\ x_{S3} \\ x_{S4} \\ x_{S5} \\ x_{S6} \end{bmatrix} = \begin{bmatrix} -5950 \\ 6050 \\ -0.1 \\ 0.3 \\ -0.045 \\ 0.055 \\ 1 \end{bmatrix}$$

And the bounds for each variable are:

$$\begin{bmatrix} x_A \\ x_B \\ x_C \\ x_D \\ x_{S1} \\ x_{S2} \\ x_{S3} \\ x_{S4} \\ x_{S5} \\ x_{S6} \end{bmatrix} \geq 0$$

3. LPs are very useful in the area of resource allocation. Let's look at the following example and try to formulate it as an LP. This example is from
`http://www.me.utexas.edu/~jensen/ORMM/instruction/powerpoint/or_models_09/03_lp2.ppt`

A large company has a toll-free hotline that is open 24-7 to answer questions regarding a new product. The following table summarizes the number of full-time equivalent (FTE) employees that must be on duty in each time block.

$Interval$	$Time$	$FTEs$
1	$0-4$	15
2	$4-8$	10
3	$8-12$	40
4	$12-16$	70
5	$16-20$	40
6	$20-0$	35

A large company may hire both full-time and part-time employees. The former work 8-hour shifts and the latter work 4-hour shifts; their respective hourly wages are \$15.20 and \$12.95. Employees may start work only at the beginning of 1 of 6 intervals.

Part-time employees can only answer 5 calls in the time a full-time employee can answer 6 calls. (i.e., a part-time employee is only 5/6 of a full-time employee.)

At least two-thirds of the employees working at any one time must be full-time employees.

We let x_t be the number of FTEs that begin the day at the start of interval t and work for 8 hours. Let y_t be the number of part time employees that begin the day at interval t. We are trying to minimize the total wages given to all employees while fulfilling the constraints.

$$\min 8 \times 15.20 \times (x_1 + ... + x_6) + 4 \times 12.95 \times (y_1 + ... + y_6)$$

subject to

$$x_6 + x_1 + \frac{5}{6}y_1 \geq 15$$

$$x_1 + x_2 + \frac{5}{6}y_2 \geq 10$$

$$x_2 + x_3 + \frac{5}{6}y_3 \geq 40$$

$$x_3 + x_4 + \frac{5}{6}y_4 \geq 70$$

$$x_4 + x_5 + \frac{5}{6}y_5 \geq 40$$

$$x_5 + x_6 + \frac{5}{6}y_6 \geq 35$$

$$y_1 \leq \frac{1}{2}(x_1 + x_6)$$

$$\vdots$$

$$y_6 \leq \frac{1}{2}(x_5 + x_6)$$

$$x_t \geq 0, \;\; y_t \geq 0 \;\; \forall t$$

(a) Solve using `linprog`.

Objective function to be minimized:

$$\begin{bmatrix} 121.6 & 121.6 & 121.6 & 121.6 & 121.6 & 121.6 & 51.8 & 51.8 & 51.8 & 51.8 & 51.8 & 51.8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix}$$

Inequality constraints:

$$\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & -1 & -\frac{5}{6} & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & -\frac{5}{6} & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & -\frac{5}{6} & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & -\frac{5}{6} & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & -\frac{5}{6} & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & -\frac{5}{6} \\ -\frac{1}{2} & 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} \leq \begin{bmatrix} -15 \\ -10 \\ -40 \\ -70 \\ -40 \\ -35 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

(b) Is there anything suspicious about the result?

The optimum contains non-integer values, which are not physically meaningful in the context of our problem. This problem is more properly formulated as an "integer program," not a linear program, because the output vector should be integer-valued. `linprog` cannot solve integer programs. IPs in general are much harder than LPs and it's not as simple as it sounds– i.e., you cannot just round the value from `linprog` to the nearest integer. This is a whole different class of problems with its own algorithms. However we will not cover them in this class.