SimPy User Guide

Tristan Gaeta

October 2023

1 What Is SimPy?

SimPy is a simple and fast python library that can solve linear programs (LPs) using the simplex method. SimPy uses the two phase method to solve LPs. All problems are converted to canonical form and use a phase-1 LP solution to find an initial basic feasible solution.

2 Representing Linear Programs

To use SimPy, an LP must be represented in vector form. SimPy is capable of solving LPs with any combination of equality and weak inequality constraints. That is, any LP that can be represented in the following vector form.

$$\begin{aligned} \min / \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & A_{eq}\mathbf{x} = \mathbf{b}_{eq} \\ & A_{leq}\mathbf{x} \leq \mathbf{b}_{leq} \\ & A_{geq}\mathbf{x} \geq \mathbf{b}_{geq} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

LPs that only use one kind of constraint (not including the variable constraint) are said to be in one of *canonical*, *standard*, or *normal* form depending on the kind of constraint used. When solving an LP, SimPy will first convert the LP to canonical form by introducing slack and surplus variables for each inequality constraint.

2.1 Cannonical Form

We define an LP to be in *canonical* form if it is written as:

$$\begin{aligned} \min / \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

2.2 Standard Form

We define an LP to be in *standard* form if it is written as:

$$\begin{array}{rc} \min / \max & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array}$$

2.3 Normal Form

We define an LP to be in *normal* form if it is written as:

$$\begin{array}{rc} \min / \max & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} & A\mathbf{x} \geq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array}$$

3 How to Use SimPy

To solve an LP using SimPy we use the `simplex()` function. Use the following line of code to import the `simplex()` function from the library.

```
>>> from simpy import simplex
```

The `simplex()` function takes the following arguments.

simplex(A, b, c, form='canonical', minimize=False, print_steps=False, **krargs)

Argument	Description
A	The $m \times n$ matrix that is the constraint matrix for this LP
b	The m dimensional vector that is the constraint vector for this LP
c	The n dimensional vector that defines the objective function of this LP
form *(optional)*	The form that this LP is in. Acceptable values are **canonical**, **standard**, **normal**, **eq**, **leq**, and **geq**. Default is **canonical**.
minimize *(optional)*	When set to true this LP will be solved as a minimization problem
print_steps *(optional)*	When set to true each iteration of the algorithm will be printed.
kwargs *(optional)*	Any additional keyword arguments. Providing the arguments **A_eq** and **b_eq**, **A_leq** and **b_leq**, and **A_geq** and **b_geq** allow the user to have additional types of constraints.

The return value of the `simplex()` function will be a tuple where the first entry is the optimal value of the LP (as a float) and the second entry is the solution that yields that optimal value (as an NumPy ndarray).

3.1 Example 1

Suppose we want to solve the following LP.

$$\begin{array}{rc} \max & 2x_1 + 4x_2 \\ \text{s.t.} & 4x_1 + 6x_2 \leq 120 \\ & 2x_1 + 6x_2 \leq 72 \\ & x_2 \leq 10 \\ & x_1, x_2 \geq 0 \end{array}$$

This LP has an optimal value of 64 at the point $(x_1, x_2) = (24, 4)$. Lets examine how we can solve this LP using SimPy. First we must represent this LP in vector notation:

$$\begin{array}{rc} \max & \begin{bmatrix} 2 & 4 \end{bmatrix} \mathbf{x} \\ \text{s.t.} & \begin{bmatrix} 4 & 6 \\ 2 & 6 \\ 0 & 1 \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} 120 \\ 72 \\ 10 \end{bmatrix} \\ & \mathbf{x} \geq \mathbf{0} \end{array}$$

Because all of the constraints in this LP are less than or equal to constraints, it is in standard form. Thus, we can solve it using SimPy as follows.

```
>>> from simpy import simplex
>>>
>>> a = [[4, 6],
>>>      [2, 6],
>>>      [0, 1]]
>>> b = [120, 72, 10]
>>> c = [2, 4]
>>>
>>> simplex(a, b, c, form='standard')
(64.0, array([24.,  4.]))
```

We can see that SimPy yields the correct solution of 64 at the point $(x_1, x_2) = (24, 4)$.

3.2 Example 2

SimPy is also capable of solving problems with several types of inequalities. Consider the following LP.

$$\begin{array}{rc} \min & 0.3x_1 + 0.2x_2 \\ \text{s.t.} & 6x_1 + 2x_2 \geq 6 \\ & 5x_1 + 15x_2 \geq 15 \\ & x_1 + x_2 \geq 2 \\ & 0.6x_1 + x_2 \leq 3 \\ & x_1, x_2 \geq 0 \end{array}$$

It has an optimal solution of 0.45 at the point $(x_1, x_2) = (0.5, 1.5)$. Let's solve it using SimPy with the following code sample.

```
>>> from simpy import simplex
>>>
>>> # define >= constraints
>>> a1 = [[6, 2],
>>>       [5, 15],
>>>       [1, 1]]
>>> b1 = [6, 15, 2]
>>>
>>> # define <= constraint
>>> a2 = [[0.6, 1]]
>>> b2 = [3]
>>>
>>> c = [0.3, 0.2]
>>>
>>> simplex(a1,b1,c,form='geq',minimize=True,A_leq=a2,b_leq=b2)
(0.45000000000000007, array([0.5, 1.5]))
```

Notice that the solution is subject to floating point rounding errors, however we do achieve the correct solution of $(x_1, x_2) = (0.5, 1.5)$.

3.3 Expected Errors

SimPy will raise an exception when a given LP is either infeasible or unbounded. For example consider the following code snippets.

```
>>> a = [[1]]
>>> b = [5]
>>> c = [1]
>>> simplex(a,b,c,form='geq')
ValueError: This problem is unbounded.
>>>
>>> a = [[1],
>>>      [1]]
>>> b = [1,2]
>>> c = [1]
>>> simplex(a,b,c)
ValueError: This problem is infeasible.
```