Linear Programming on TI-89: A Tutorial for simplex( )

Yanchao Liu
Department of Industrial and System Engineering
University of Wisconsin-Madison
yliu67@wisc.edu

Students of industrial engineering (IE) might encounter such moments from time to time when a small-scale linear programming (LP) model needs to be solved. Some may reach for a computer. But what if the computer or the right software is not available at that particular moment? Even if a computer equipped with the most advanced LP solver is at one's disposal, typing up the model correctly in the required format would typically cost more time than the importance of the model deserves. Annoyed by this very fact as an IE student myself, I wrote up a small LP solver that implements the two-phase revised simplex algorithm and runs on a TI-89 calculator[1]. If you have a TI-89 (or higher) calculator as many other engineering students do, take a few minutes to read this tutorial, and a handy LP solver will be at your figure tips.

The solver is written in C language and the executable file is simplex.89z. Simply transfer this file to your TI-89 calculator and the program will be installed and ready to use. On the calculator, it will be displayed as an AMS program named simplex. The file size is 9831 bytes. For general information about installing programs on a TI-89 calculator, consult the calculator manual or the TI website.

Step 1: Write Your LP Model in Standard Form

Here is the definition of LP standard form:

> Linear programs in **standard form** (1) have only equality main constraints; (2) have only nonnegative variables, and (3) have objective function and main constraints simplified so that variables appear at most once, on the left-hand side, and any constant term (possibly zero) appears on the right-hand side. [1]

And the tips for converting inequalities:

> Main inequality constraints of a given linear program can be converted into nonnegativities by adding distinct, nonnegative, zero-cost slack variables in every such $\leq$ inequality and subtracting such slack variables in every main $\geq$. [1]

IMPORTANT: The solver always attempts to **maximize** the objective function, thus if your model has a minimization objective function, you need to convert it to a maximization one by negating all the cost coefficients in the objective function. See the example in the figures below.

[1] The solver is compatible with any TI device that runs the Advanced Mathematics Software (AMS), of which TI-89 is the one I own.

$$
\begin{aligned}
&\max\ 12x_1 + 9x_2 \\
&s.t.\quad x_1 \le 1000 \\
&\qquad x_2 \le 1500 \\
&\qquad x_1 + x_2 \le 1750 \\
&\qquad 4x_1 + 2x_2 \le 4800 \\
&\qquad x_1, x_2 \ge 0
\end{aligned}
$$

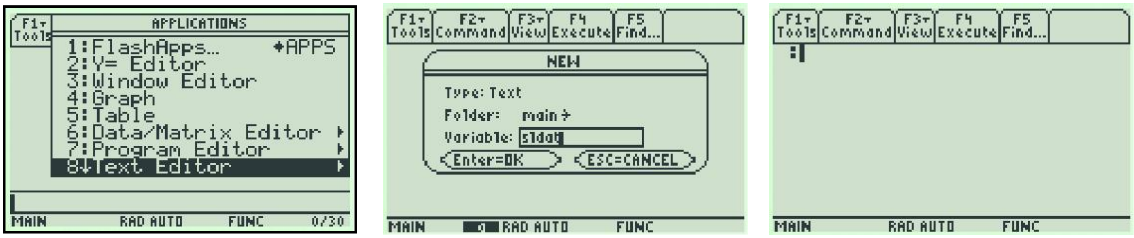
Figure 1 Original Model

$$
\begin{aligned}
&\max\ 12x_1 + 9x_2 \\
&s.t.\quad x_1 + x_3 = 1000 \\
&\qquad x_2 + x_4 = 1500 \\
&\qquad x_1 + x_2 + x_5 = 1750 \\
&\qquad 4x_1 + 2x_2 + x_6 = 4800 \\
&\qquad x_1, x_2, x_3, x_4, x_5, x_6 \ge 0
\end{aligned}
$$

Figure 2 Standard-form Model

Step 2: Enter Data in the sldat File

Create a text file using the Text Editor program on your TI-89. Name it *sldat* and place it in the same folder as simplex is in. See figures below.



Suppose that in your standard-form model there are *n* variables (including the slack variables) and *m* equality constraints, the number of pieces of datum you need to enter is $2 + m \times n + m + n$, and they must be entered following this sequence: **n, m, *A*, *b*, *c***, where A, b and c are the coefficient matrix and vectors in the LP canonical form:

$$
\begin{aligned}
&\max c^T x \\
&s.t. Ax = b
\end{aligned}
$$

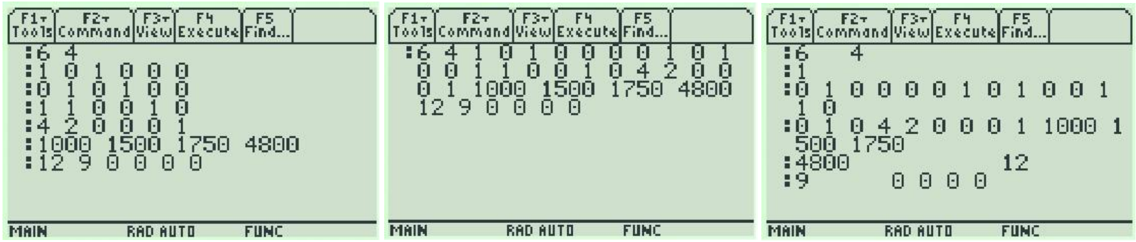
Take the model in Figure 2 for example. There are 6 variables and 4 constraints, and

$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 4 & 2 & 0 & 0 & 0 & 1 \end{bmatrix}$, $b = [1000\ \ 1500\ \ 1750\ \ 4800]$, $c = [12\ \ 9\ \ 0\ \ 0\ \ 0\ \ 0]$, so the input sequence should be:

6 4 1 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 4 2 0 0 0 1 1000 1500 1750 4800 12 9 0 0 0 0

(n = 6, m = 4; ***A*** = 1 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 4 2 0 0 0 1; ***b*** = 1000 1500 1750 4800; ***c*** = 12 9 0 0 0 0)

As long as the sequential order is observed, you have all the freedom to align the data entries (using [SPACE] or [ENTER] key on your TI-89). For example, any of the following fashions of *sldat* for the above model is acceptable.

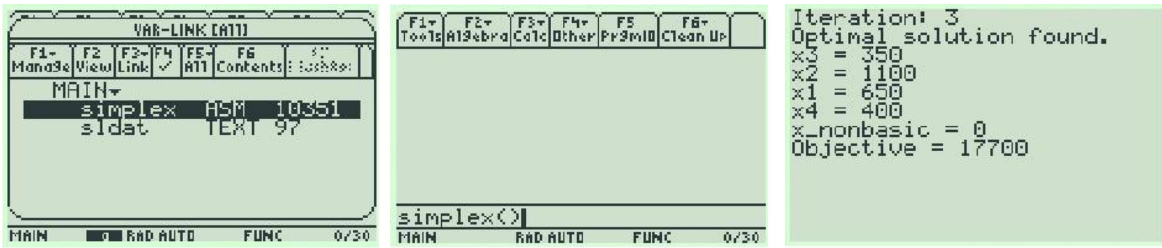


However, you still need to be very cautious about what you type in the *sldat* file. The solver will do some data validation, but oftentimes it just runs off with whatever it is provided and may report unreliable results. **Garbage in, garbage out!** Here are a few tips about how to make a valid *sldat* file.

- Never use non-numeric symbols except for the negative sign (-) and the decimal point (.).
- On the calculator, the negative sign and the minus sign are different. Be careful!
- Do not omit zeros (0) in the end of a vector, such as in the ***c*** vector.

Step 3: Run simplex() and Browse the Results

Find simplex in the VAR-LINK list or just type in *simplex()* on the home screen, then press [ENTER] to run. The results will be displayed on the screen. If the optimal solution (other than infeasible or unbounded) is found, the results will also be saved in the text file named *slout*, which can be browsed by the Text Editor after the program has ended. Be aware that the saved results will be updated (overwritten) by new optimal results of later runs.



Regardless of the notations in your model, the solver assumes the variable name *x* and the index set {1, 2, … } in the result display. Thus, *x1* corresponds to the first variable in the input model, *x2* the second, and so forth. Of an optimum solution, only the basic variable values will be displayed and all the non-basic variables take the value 0.

NOTE: Due to the memory restrictions on TI-89, the solver accepts up to 50 variables and 49 constraints.

FINAL WORDS: I have tested the program to the best that I can to ensure its robustness: no memory leak, no system collapse, etc. Nevertheless, assembly programs may cause serious problems to the device that runs them. I am not responsible for any consequence of using the *simplex()* solver. Feedbacks and critiques are welcome.

Reference

[1] Ronald L. Rardin, *Optimization in Operations Research*, Prentice Hall, 1997.