Solving Linear Programs Using the Simplex Method (Manual)

GáborRétvári
E-mail: retvari@tmit.bme.hu

The GNU Octave Simplex Solver Implementation

As part of the course material two simple GNU Octave/MATLAB based simplex implementations are available for download, one for the primal and one for the dual simplex, which can be of great help in learning the use of simplex tableau and checking one's solution steps. The programs will solve a linear program, either using the primal or the dual simplex method, from a user-specified initial basis and output the sequence of simplex tableau encountered along the way. Note that the implementations are pretty rudimentary at this point as no code for finding an initial basis, checking numerical instability, preventing cycling, etc., is available at this point; contributions are welcome!

The primal and dual simplex implementations are available as separate scripts:

- psimplex.m: primal simplex,
  http://lendulet.tmit.bme.hu/~retvari/courses/VITMD097/psimplex.m
- dsimplex.m: dual simplex,
  http://lendulet.tmit.bme.hu/~retvari/courses/VITMD097/dsimplex.m

An experimental implementation of the two-phase primal simplex method using the artificial variables technique for searching initial basis is also available (thanks to János Beluzsár) at: http://lendulet.tmit.bme.hu/~retvari/courses/VITMD097/psimplex_mod.m

The code was written for the GNU Octave numerical computation framework but, theoretically, it should work in MATLAB as well. After downloading place the scripts into the working directory or under any path where GNU Octave finds them (see: Octave info pages: addpath).

Success of the installation can be checked as follows:

```
octave> help psimplex
PSIMPLEX - simplex algorithm for maximization problems in standard form

 [A, subs, x, z] = psimplex(c, A, b, subs, varargin)

 -------------------------------------------------------------------------
 DESCRIPTION
 -------------------------------------------------------------------------

 The simplex algorithm for the LP problem

                          max z = c*x
```

```
Subject to: Ax  = b
             x >= 0

The fifth input parameter is optional. If it is set to 'y', then the
initial and final tableaux are displayed to the screen.
```

Using the Software

We show the usage of the software through the process of solving the below linear program:

$$\begin{array}{lrcrcr} \max & -x_1 & + & 2x_2 & & \\ \text{s.t.} & 3x_1 & + & 4x_2 & \leq & 12 \\ & 2x_1 & - & x_2 & \leq & 12 \\ & x_1, & & x_2 & \geq & 0 \end{array}$$

The scripts expect the linear program as a *maximization* problem written in the *standard form*. Correspondingly, introduce slack variables to obtain the standard form of the linear program:

$$\begin{array}{lrcrcrcrcr} \max & -x_1 & + & 2x_2 & & & & & & \\ \text{s.t.} & 3x_1 & + & 4x_2 & + & x_3 & & & = & 12 \\ & 2x_1 & - & x_2 & & & + & x_4 & = & 12 \\ & x_1, & & x_2, & & x_3, & & x_4 & \geq & 0 \end{array}$$

The scripts take as input the objective vector $\boldsymbol{c}^T$, the constraint matrix $\boldsymbol{A}$, and the right-hand side vector $\boldsymbol{b}$.

$$\boldsymbol{c}^T = \begin{bmatrix} -1 & 2 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} 3 & 4 & 1 & 0 \\ 2 & -1 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 12 \\ 12 \end{bmatrix}$$

In addition, an initial basis also needs to be specified, since currently the code does not implement a starting basis searching technique. Naturally, the primal simplex requires a primal feasible and the dual simplex requires a dual feasible initial basis, feasibility is checked upon calling. In the above example, the columns of the slack variables constitute a primal feasible initial basis: $B = \{3, 4\}$.

What remained to be done is to specify these parameters in the format of `GNU Octave` as arguments for the `psimplex` function. Take note that $\boldsymbol{c}^T$ is a row-vector; size constraints are explicitly enforced by the programs.

```
octave> c = [-1 2 0 0];
octave> A = [3 4 1 0;2 -1 0 1];
octave> b=[12;12];
octave> psimplex(c, A, b, [3,4], 1)
     ------------------------------------------------

              Tableaux of the Simplex Algorithm

     ------------------------------------------------

      Initial problem, bringing to tableau form
```

```
A =

    3    4    1    0   12
    2   -1    0    1   12
    1   -2   -0   -0    0

 Press any key to continue ...
```

In the first step the script checks the size of the input parameters and prints the initial table (not necessarily a valid simplex tableau yet). The display format is special, as the objective row of the tableau (row zero) is shown as the last row of the displayed matrix due to code organization and simplicity considerations. The RHS will remain in the last column and the current objective function value is shown in the last column of the last line. The above matrix therefore corresponds to the below table.

	z	x_1	x_2	x_3	x_4	RHS
z	1	1	-2	0	0	0
x_3	0	3	4	1	0	12
x_4	0	2	-1	0	1	12

In the second step, the script through a sequence of pivot operations brings the table into the required form for the primal simplex tableau: basic variable columns will form an identity matrix and the corresponding reduced costs in the objective row will all be zeroed out. During this process, the script checks whether the initial basis is nonsingular and primal feasible.

```
     ------------------------------------------------

                  Initial tableau

A =

    3    4    1    0   12
    2   -1    0    1   12
    1   -2    0    0    0


                  Initial basis

subs =

   3   4

 Press any key to continue ...
```

In our case the initial table does not change since our basis was already an identity matrix and the corresponding reduced costs were zero.

During the next steps the script iteratively computes the optimal simplex tableau through a sequence of improving pivots using Dantzig's pivot rule.

```
    ------------------------------------------------

         2 enters, 3 leaves the basis
A =

    0.75000    1.00000    0.25000    0.00000    3.00000
    2.75000    0.00000    0.25000    1.00000   15.00000
    2.50000    0.00000    0.50000    0.00000    6.00000

         current basis:
subs =

   2   4

 Press any key to continue ...
```

In every iteration the script determines the entering and the leaving variables and performs the pivot. Recall that the display format is special; the above matrix as the conventional simplex tableau form is as follows:

	z	x_1	x_2	x_3	x_4	RHS
z	1	2.5	0	0.5	0	6
x_2	0	0.75	1	0.25	0	3
x_4	0	2.75	0	0.25	1	15

During the iteration the script checks whether the current pivot is valid and halts if unboundedness is encountered. Degeneration and cycling are not checked, therefore the script may enter an infinite loop during the iteration. Finally, if the current tableau is primal optimal then the script terminates and outputs the optimal tableau.

```
    ------------------------------------------------
                Optimal tableau, end
ans =

    0.75000    1.00000    0.25000    0.00000    3.00000
    2.75000    0.00000    0.25000    1.00000   15.00000
    2.50000    0.00000    0.50000    0.00000    6.00000
```

In our case the optimal objective function value is 6 and the optimal solution can be read from the displayed matrix: the value of nonbasic variables x_1 and x_3 is, by assumption, zero, while the indices of the basic variables are $B = \{2, 4\}$, so reading the RHS column in the corresponding rows yields $x_2 = 3$, $x_4 = 15$.

If only in the solution is interesting but not the steps of the iteration and the intermediate simplex tableau, then simply omit the last argument when calling `psimplex`:

```
psimplex(c, A, b, [3,4])
```

If the results should be saved in variables for further use:

```
octave> [T, subs, x, z] = psimplex(c, A, b, [3,4]);
```

In this case the script stores the optimal tableau in `T`, the column indices of the optimal basis in `subs`, the optimal solution in `x`, and the optimal objective function value in `z`.

The script implementing the dual simplex method, `dsimplex`, works similarly.

(Note: contributions are welcome, you can get a better grade at the exam! You may write code to improve the display format for the simplex tableau, implement cycling prevention, etc., write me an email if you'd like to contribute!)