MATLAB Solution of Linear Programming Problems

The simplex method is included in MATLAB using *linprog* function. All is needed is to have the problem expressed in the terms of MATLAB definitions. Appendix A contains a list of possible options.

Example 3.5:

Maximize, $P = 4x_1 + 16x_2$

Subject to, $2x_1 + 3x_2 \leq 16$

$4x_1 + x_2 \leq 24$

$x_2 \leq 2.5$

$x_i \geq 0$ i = 1, 2

The problem is changed into,

Minimize, $f = -4x_1 - 16x_2$

Subject to, $2x_1 + 3x_2 \leq 16$

$4x_1 + x_2 \leq 24$

$x_2 \leq 2.5$

$x_i \geq 0$ i = 1, 2

Code #1:

```
close all
clear all

%objectve function: input coefficients
f=[-4;-16];

%Inequality constraints. The should be in the form [A]{x}={b}
A=[2 3; 4 1; 0 1];
b=[16;24;2.5];

%Lower and upper bounds of variables
lb=zeros(2,1);
ub=[1e4;1e4];

%Add empty matrices for coefficients of equality constraints and initial
%guess
Aeq=[];
beq=[];

x0=[];
%Specify search options:
%Use the dual-simplex algorithm since simplex will be removed after this
version
%Display the results of all iterations
options=optimoptions('linprog','Algorithm','Simplex','Display','iter')
[x,fval,exitflag,output] = linprog(f,A,b,Aeq,beq,lb,ub,x0,options)
```

Output of Code #1:

```
Lin_Prog_Example

options =
  linprog options:

   Options used by current Algorithm ('simplex'):
   (Other available algorithms: 'active-set', 'dual-simplex', 'interior-point', 'interior-point-legacy')

   Set properties:
           Algorithm: 'simplex'
             Display: 'iter'

   Default properties:
         MaxIterations: '100*numberOfVariables'
    OptimalityTolerance: 1.0000e-06

   Show options not used by current Algorithm ('simplex')

Warning: The 'simplex' algorithm will be removed in a future release. To avoid this warning or a future
error, choose a different algorithm:
'interior-point' or 'dual-simplex'.

> In linprog (line 282)
  In Lin_Prog_Example (line 25)
The default starting point is feasible, skipping Phase 1.

Phase 2: Minimize using simplex.
     Iter          Objective          Dual Infeasibility
                   f'*x               A'*y+z-w-f
      0               0                 16.4924
      1             -40                      4
      2             -57                      0
Optimization terminated.

x =
    4.2500
    2.5000

fval =
   -57

exitflag =
     1

output =
         iterations: 2
          algorithm: 'simplex'
```

```
      cgiterations: []
           message: 'Optimization terminated.'
    constrviolation: 0
     firstorderopt: 0
```

It can be noticed that the solution progressed along the same path as the solution we did manually.
We will now follow the suggestion of MATLAB and use the Dual –Simplex algorithm (Appendix B), which is a variation of the Simplex. This algorithm is used in *Code 2* below.

Code #2:

```
close all
clear all

%objectve function: input coefficients
f=[-4;-16];

%Inequality constraints. The should be in the form [A]{x}={b}
A=[2 3; 4 1; 0 1];
b=[16;24;2.5];

%Lower and upper bounds of variables
lb=zeros(2,1);
ub=[1e4;1e4];

%Add empty matrices for coefficients of equality constraints and initial
%guess
Aeq=[];
beq=[];

x0=[];
%Specify search options:
%Use the dual-simplex algorithm since simplex will be removed after this
version
%Display the results of all iterations
options=optimoptions('linprog','Algorithm','Dual-Simplex','Display','iter')
[x,fval,exitflag,output] = linprog(f,A,b,Aeq,beq,lb,ub,x0,options)
```

Output of Code #2:

```
options =

 linprog options:

  Options used by current Algorithm ('dual-simplex'):
  (Other available algorithms: 'active-set', 'interior-point', 'simplex')

  Set by user:
    Algorithm: 'dual-simplex'
      Display: 'iter'

  Default:
   Diagnostics: 'off'
       MaxIter: '10*(numberOfEqualities+numberOfInequalities+numberOfVariables)'
       MaxTime: Inf
    Preprocess: 'basic'
        TolCon: 1.0000e-04
        TolFun: 1.0000e-07


LP preprocessing removed 0 inequalities, 0 equalities,
0 variables, and 0 non-zero elements in 0.00 secs.

   Iter     Time         Fval  Primal Infeas    Dual Infeas
     0     0.01  -2.000000e+05   6.843095e+04   0.000000e+00
     3     0.01  -5.700000e+01   0.000000e+00   0.000000e+00

Optimal solution found.


x =

    4.2500
    2.5000


fval =

   -57
```

Homework: Solve 3.99, 3.53 using MATLAB

Appendix A. LINPROG Options

http://www.mathworks.com/help/optim/ug/linprog.html#buus4rk-1_1

`options` — Optimization optionsoutput of `optimoptions` | structure as `optimset` returns

Optimization options, specified as the output of `optimoptions` or a structure as `optimset` returns.

Some options apply to all algorithms, and others are relevant for particular algorithms. See Optimization Options Reference for detailed information.

All Algorithms

`Algorithm`	Choose the optimization algorithm:
	`'interior-point-legacy'` (default)
	`'interior-point'`
	`'dual-simplex'`
	`'active-set'`
	`'simplex'`
	For information on choosing the algorithm, see Linear Programming Algorithms.

`Diagnostics`	Display diagnostic information about the function to be minimized or solved. Choose `'off'` (default) or `'on'`.
`Display`	Level of display (see Iterative Display): `'final'` (default) displays just the final output. `'off'` or `'none'` displays no output. `'iter'` displays output at each iteration. The `'iter'` option is unavailable for the `'active-set'` algorithm.
`LargeScale` Use `Algorithm` instead	Use the `'interior-point-legacy'` algorithm when set to `'on'` (default). Use a medium-scale algorithm when set to `'off'`(see `Simplex` in `options`). For more information, see Choosing the Algorithm.
`MaxIter`	Maximum number of iterations allowed, a positive integer. The default is: `85` for the `'interior-point-legacy'` algorithm `200` for the `'interior-point'` algorithm `10*(numberOfEqualities + numberOfInequalities + numberOfVariables)` for the `'dual-simplex'` algorithm `10*numberOfVariables` for the `'simplex'` algorithm `10*max(numberOfVariables, numberOfInequalities + numberOfBounds)` for the `'active-set'` algorithm See Tolerances and Stopping Criteria and Iterations and Function Counts.
`TolFun`	Termination tolerance on the function value, a positive scalar. The default is: `1e-8` for the `'interior-point'` algorithm `1e-7` for the `'dual-simplex'` algorithm `1e-6` for the `'interior-point'` and `'simplex'` algorithms

	The option is not used for the `'active-set'` algorithm `TolFun` measures dual feasibility tolerance.

interior-point Algorithm

`Preprocess`	Level of LP preprocessing prior to algorithm iterations. Specify `'basic'` (default) or `'none'`.
`TolCon`	Feasibility tolerance for constraints, a scalar from `1e-10` through `1e-3`. `TolCon` measures primal feasibility tolerance. The default is `1e-6`.

Dual-Simplex Algorithm

`MaxTime`	Maximum amount of time in seconds that the algorithm runs. The default is `Inf`.
`Preprocess`	Level of LP preprocessing prior to dual simplex algorithm iterations. Specify `'basic'` (default) or `'none'`.
`TolCon`	Feasibility tolerance for constraints, a scalar from `1e-10` through `1e-3`. `TolCon` measures primal feasibility tolerance. The default is `1e-4`.

Simplex Algorithm

`Simplex` Use `Algorithm` instead	If `'on'`, and if `LargeScale` is `'off'`, `linprog` uses the simplex algorithm. The simplex algorithm uses a built-in starting point, ignoring the starting point `x0` if supplied. The default is `'off'`, meaning `linprog` uses an active-set algorithm.

Appendix B. Dual-Simplex Algorithm

https://www.me.utexas.edu/~jensen/ORMM/supplements/methods/lpmethod/S1_dualsimplex.pdf

LP Methods.S1
Dual Simplex Algorithm

In the tableau implementation of the primal simplex algorithm, the right-hand-side column is always nonnegative so the basic solution is feasible at every iteration. For purposes of this section, we will say that the basis for the tableau is *primal feasible* if all elements of the right-hand side are nonnegative. Alternatively, when some of the elements are negative, we say that the basis is *primal infeasible*. Up to this point we have always been concerned with primal feasible bases.

For the primal simplex algorithm, some elements in row 0 will be negative until the final iteration when the optimality conditions are satisfied. In the event that all elements of row 0 are nonnegative, we say that the associated basis is *dual feasible*. Alternatively, if some of the elements of row 0 are negative, we have a *dual infeasible* basis.

As described, the primal simplex method works with primal feasible, but dual infeasible (nonoptimal) bases. At the final (optimal) solution, the basis is both primal and dual feasible. Throughout the process we maintain primal feasibility and drive toward dual feasibility.

In this section, a variant of the primal approach, known as the dual simplex method, is considered that works in just the opposite fashion. Until the final iteration, each basis examined is primal infeasible (some negative values on the right-hand side) and dual feasible (all elements in row 0 are nonnegative). At the final (optimal) iteration the solution will be both primal and dual feasible. Throughout the process we maintain dual feasibility and drive toward primal feasibility. For a given problem, both the primal and dual simplex algorithms will terminate at the same solution but arrive there from different directions.

The dual simplex algorithm is most suited for problems for which an initial dual feasible solution is easily available. It is particularly useful for reoptimizing a problem after a constraint has been added or some parameters have been changed so that the previously optimal basis is no longer feasible.

We will have much more to say about duality and the relationship between primal and dual solutions in Chapter 5; however, in this section, we are principally concerned with the mechanics of implementing the dual simplex method in the tableau format. We will see that the dual simplex algorithm is very similar to the primal simplex algorithm.

Algorithm

With reference to the tableau, the algorithm must begin with a basic solution that is dual feasible so all the elements of row 0 must be nonnnegative. The iterative step of the primal simplex algorithm first selects a variable to enter the basis and then finds the variable that must leave so that primal feasibility is maintained. The dual simplex method does the opposite; it first selects a variable to leave the basis and then finds

the variable that must enter the basis to maintain dual feasibility. This is the principal difference between the two methods. The algorithm below assumes a basic solution is described by a tableau.

Step 1 (*Initialization*)

Start with a dual feasible basis and let $k = 1$. Create a tableau for this basis in the simplex form. If the right-hand side entries are all nonnegative, the solution is primal feasible, so stop with the optimal solution.

Step 2 *(Iteration k*)

a. *Select the leaving variable*. Find a row, call it *r*, with a negative right-hand-side constant; i.e., $\bar{b}_r < 0$. Let row *r* be the pivot row and let the leaving variable be $x_{B(r)}$. A common rule for choosing *r* is to select the most negative RHS value; i.e.,

$$\bar{b}_r = \min\{\bar{b}_i : i = 1,\ldots,m\}.$$

b. *Determine the entering variable*. For each negative coefficient in the pivot row, compute the negative of the ratio between the reduced cost in row 0 and the structural coefficient in row *r*. If there is no negative coefficient, $\bar{a}_{rj} < 0$, stop; there is no feasible solution.

Let the column with the minimum ratio, designated by the index *s*, be the pivot column; let x_s is the entering variable. The pivot column is determined by the following ratio test.

$$\frac{-\bar{c}_s}{\bar{a}_{rs}} = \min\left\{\frac{-\bar{c}_j}{\bar{a}_{rj}} : \bar{a}_{rj} < 0,\ j = 1,\ldots,n\right\}$$

c. *Change the basis*. Replace $x_{B(r)}$ by x_s in the basis. Create a new tableau by performing the following operations (these are the same as for the primal simplex algorithm).

Let $\bar{\mathbf{a}}_i$ be the vector of the *i*th row of the current tableau, and let $\bar{\mathbf{a}}_i^{\text{new}}$ be the *i*th row in the new tableau. Let $\bar{b}_i$ be the RHS for row *i* in the current tableau, and let $\bar{b}_i^{\text{new}}$ be the RHS of the new tableau.

Let $\bar{a}_{is}$ be the element in the *i*th row of the pivot column *s*.

The pivot row in the new tableau is

$$\bar{\mathbf{a}}_r^{\text{new}} = \bar{\mathbf{a}}_r / \bar{a}_{rs} \text{ and } \bar{b}_r^{\text{new}} = \bar{b}_r / \bar{a}_{rs}.$$

The other rows in the new tableau are

$$\bar{\mathbf{a}}_i^{\text{new}} = \left(-\bar{a}_{is} \times \bar{\mathbf{a}}_r^{\text{new}}\right) + \bar{\mathbf{a}}_i \text{ and}$$

$$\bar{b}_i^{\text{new}} = \left(-\bar{a}_{is} \times \bar{b}_r^{\text{new}}\right) + \bar{b}_i \text{ for } i = 0, 1, \ldots, m, \; i \neq r$$

(These operations have the effect of pricing out the pivot column. Its replacement will have a single 1 in row r and a zero in all other rows as required by the simplex form.)

Step 3 (*Feasibility test*)

If all entries on the right-hand side are nonnegative the solution is primal feasible, so stop with the optimal solution. Otherwise, put $k \leftarrow k + 1$ and return to Step 2.

Examples

An Easy Dual Feasible Starting Solution

The simplest situation arises when there is an obvious dual feasible basis that can be used to initialize the algorithm. Consider the following problem.

$$\begin{aligned} \text{Maximize } z &= -5x_1 - 35x_2 - 20x_3 \\ \text{subject to } & x_1 - x_2 - x_3 \leq -2 \\ & -x_1 - 3x_2 \leq -3 \\ & x_1 \geq 0, \; x_2 \geq 0, \; x_3 \geq 0 \end{aligned}$$

Step 1: Adding slack variables x_4 and x_5, leads to the first tableau which is primal infeasible but dual feasible.

Row	Basic	Coefficients						RHS
		z	x_1	x_2	x_3	x_4	x_5	
0	z	1	5	35	20	0	0	0
1	x_4	0	1	-1	-1	1	0	-2
2	x_5	0	-1	-3	0	0	1	-3

Iteration 1: Row 2 is selected as the pivot row so x_5 leaves the basis. The ratio test indicates that x_1 is to enter the basis. The tableau shows the ratio calculations.

Row	Basic	Coefficients						RHS
		z	x_1	x_2	x_3	x_4	x_5	
0	z	1	5	35	20	0	0	0
1	x_4	0	1	-1	-1	1	0	-2
2	x_5	0	-1	-3	0	0	1	-3
	Ratio	---	5	11.67	---	---	---	

After the pricing out operations to obtain the simplex form for x_1 we get the following tableau. The feasibility test at Step 3 fails because the basis is not yet primal feasible so we return to Step 2. We show the row and column selection steps for the next iteration.

Row	Basic	Coefficients						RHS
		z	x_1	x_2	x_3	x_4	x_5	
0	z	1	0	20	20	0	5	-15
1	x_4	0	0	-4	-1	1	1	-5
2	x_1	0	1	3	0	0	-1	3
	Ratio	---	---	5	20	---	---	

Iteration 2: Row 1 is selected as the pivot row so x_4 leaves the basis and x_2 enters. This leads to the tableau below that still has a negative RHS value so we return to Step 2.

Row	Basic	Coefficients						RHS
		z	x_1	x_2	x_3	x_4	x_5	
0	z	1	0	0	15	5	10	-40
1	x_2	0	0	1	0.25	-0.25	-0.25	1.25
2	x_1	0	1	0	-0.75	0.75	-0.25	-0.75
	Ratio	---	---	---	20	---	40	

Iteration 3: Selecting row 2 as the pivot row, x_1 leaves the basis and x_3 enters. The updated tableau given below is both primal and dual feasible indicating that the optimal solution has been obtained. The algorithm terminates.

Row	Basic	Coefficients						RHS
		z	x_1	x_2	x_3	x_4	x_5	
0	z	1	20	0	0	20	5	-55
1	x_2	0	0.333	1	0	0	-0.33	1
2	x_3	0	-1.33	0	1	-1	0.333	1

Restarting after Changing the Right-Hand-Side Constants

A primary use of the dual simplex algorithm is to reoptimize a problem after it has been solved and one or more of the RHS constants is changed. This is illustrated with the following problem. The optimal tableau is also shown with x_{s1}, x_{s2}, and x_{s3} as slacks.

$$\begin{aligned} \text{Maximize } z = \quad & 2x_1 + 3x_2 \\ \text{subject to} \quad & -x_1 + x_2 \le 5 \\ & x_1 + 3x_2 \le 35 \\ & x_1 \le 20 \\ & x_1 \ge 0,\ x_2 \ge 0 \end{aligned}$$

Row	Basic	Coefficients						RHS
		z	x_1	x_2	x_{s1}	x_{s2}	x_{s3}	
0	z	1	0	0	0	1	1	55
1	x_2	0	0	1	0	0.33	-0.3	5
2	x_1	0	1	0	0	0	1	20
3	x_{s1}	0	0	0	1	-0.3	1.33	20

Changing the right-hand-side constants will only change the entries in the last column of the tableau. In particular, if we change b_2 from 35 to 20 and b_3 from 20 to 26 in the original problem statement, the RHS vector in the above tableau for the current basis **B** becomes

$$\mathbf{x}_B = \bar{\mathbf{b}} = \mathbf{B}^{-1}\mathbf{b}^{new} = \begin{pmatrix} -1 & 1 & 1 \\ 1 & 3 & 0 \\ 1 & 0 & 0 \end{pmatrix}^{-1} \begin{pmatrix} 5 \\ 20 \\ 26 \end{pmatrix} = \begin{pmatrix} 26 \\ -2 \\ 33 \end{pmatrix} \text{ with } z = 46.$$

Consequently, when $\mathbf{b}^{new} = (5, 20, 26)^T$ replaces $\mathbf{b}^{old} = (5, 35, 20)^T$, we get the tableau below which is a candidate for the dual simplex algorithm. Note that $\bar{b}_2$ appears in row 1 and $\bar{b}_1$ appears in row 2 in the tableau because row 2 corresponds to x_1 and row 1 to x_2.

Row	Basic	Coefficients						RHS
		z	x_1	x_2	x_{s1}	x_{s2}	x_{s3}	
0	z	1	0	0	0	1	1	55
1	x_2	0	0	1	0	0.33	-0.3	-2
2	x_1	0	1	0	0	0	1	26
3	x_{s1}	0	0	0	1	-0.3	1.33	33
	Ratio	---	---	---	---	---	3	

Changing the RHS values does not effect the reduced costs so the entries in row 0 remain nonnegative; however, a negative value for $\bar{b}_2$ indicates that the basic solution is now infeasible. It is clear from the tableau that x_2 will leave the basis and x_{s3} will enter at the next iteration.

Row	Basic	Coefficients						RHS
		z	x_1	x_2	x_{s1}	x_{s2}	x_{s3}	
0	z	1	0	3	0	2	0	40
1	x_{s3}	0	0	-3	0	-1	1	6
2	x_1	0	1	3	0	1	0	20
3	x_{s1}	0	0	4	1	1	0	25

Adding a Constraint

Using the previous problem, we now add the constraint $x_2 \geq 10$. The solution in the optimal tableau, $x_1 = 20$ and $x_2 = 5$, does not satisfy this constraint, so action must be taken to incorporate it into the tableau. First we subtract a slack variable x_{s4} to get the equality

$$x_2 - x_{s4} = 10$$

and then multiply it by –1 to achieve the correct form. A row corresponding to this constraint and a column corresponding to the slack variable are added to the current tableau resulting in the modified tableau below.

Row	Basic	Coefficients							RHS
		z	x_1	x_2	x_{s1}	x_{s2}	x_{s3}	x_{s4}	
0	z	1	0	0	0	1	1	0	55
1	x_2	0	0	1	0	0.333	-0.33	0	5
2	x_1	0	1	0	0	0	1	0	20
3	x_{s1}	0	0	0	1	-0.33	1.333	0	20
4	x_{s4}	0	0	-1	0	0	0	1	-10

To regain the simplex form for column x_2, we must add row 1 to row 4. Now the tableau is in the simplex form.

Row	Basic	Coefficients							RHS
		z	x_1	x_2	x_{s1}	x_{s2}	x_{s3}	x_{s4}	
0	z	1	0	0	0	1	1	0	55
1	x_2	0	0	1	0	0.333	-0.33	0	5
2	x_1	0	1	0	0	0	1	0	20
3	x_{s1}	0	0	0	1	-0.33	1.333	0	20
4	x_{s4}	0	0	0	0	0.333	-0.33	1	-5
	Ratio	---	---	---	---	---	3	---	

As expected, the solution is dual feasible but not primal feasible. The only negative RHS value appears in row 4 so x_{s4} must leave the basis. The entering variable is x_{s3}, the only candidate with a negative entry in the pivot row. The optimum tableau is shown below.

Row	Basic	Coefficients							RHS
		z	x_1	x_2	x_{s1}	x_{s2}	x_{s3}	x_{s4}	
0	z	1	0	0	0	2	0	3	40
1	x_2	0	0	1	0	0	0	-1	10
2	x_1	0	1	0	0	1	0	3	5
3	x_{s1}	0	0	0	1	1	0	4	0
4	x_{s3}	0	0	0	0	-1	1	-3	15