Basic Matlab implementation of the Simplex matrix algorithm

Nasser M. Abbasi

June 22, 2020 Compiled on September 9, 2023 at 9:34pm

Contents

List of Figures

List of Tables

1 Introduction

This is a description of a Matlab function called `nma_simplex.m` that implements the matrix based simplex algorithm for solving standard form linear programming problem. It supports phase one and phase two.

The function solves (returns the optimal solution x^* of the standard linear programming problem given by

$$\min_x J(x) = c^T x$$

Subject to

$$Ax = b$$
$$x \geq 0$$

The constraints have to be in standard form (equality), which results after adding any needed surplus and/or slack variables. The above is equivalent to Matlab's A_{eq}, b_{eq} used with the standard command `linprog`.

This function returns the final tableau, which contains the final solution. It can print all of the intermediate tableau generated and the basic feasible solutions generated during the process by passing an extra flag argument. These are generated as it runs through the simplex algorithm.

The final tableau contains the optimal solution x^* which can be read directly from the tableau. Examples below illustrate how to call this function and how to read the solution from the final tableau.

The tableau printed on the screen have this format

A	b
c^T	

The optimal x^* is read directly by looking at the columns in A that make up the identity matrix. With the debug flag set, the optimal x^* is also displayed on the screen.

For example, if the final tableau was the following

$$\begin{pmatrix} 1 & 0 & 0.4 & -0.2 & 1.4 \\ 0 & 1 & -0.2 & 0.6 & 3.8 \\ 0 & 0 & 0.6 & 2.2 & 24.6 \end{pmatrix}$$

Then $x_1 = 1.4$ and $x_2 = 3.8$. This means the optimal solution is

$$x^* = \begin{pmatrix} 1.4 \\ 3.8 \\ 0 \\ 0 \end{pmatrix}$$

The function accepts A, b, c and a fourth parameter which is a flag (true or false). If the flag is true, then each tableau is printed as the algorithm searches for the optimal x solution, and it also prints each x found at each step.

The following are few example showing how to use this function to solve linear programming problems, and comparing the answer to Matlab's linprog to verify they are the same.

2 Examples

2.1 Example 1

A vendor selling rings and bracelets. A ring has 3 oz. gold., 1 oz. silver. Bracelet has 1 oz. gold, 2 oz. silver. Profit on a ring is \$4 and the profit on bracelet is \$5. Initially we have 8 oz. gold and 9 0z silver. How many rings and bracelets to produce to maximize profit?

Note: we need integer LP to solve this. But for now we can ignore this to illustrate the use of this function.

$$\begin{aligned} x_1 &= \text{number of rings} \\ x_2 &= \text{number of bracelets} \\ J(x) &= 4x_1 + 5x_2 \end{aligned}$$

Since we want to maximize $J(x)$, then we change the sign

$$J(x) = -4x_1 - 5x_2$$

With $x_i \geq 0$. The constraints are $3x_1 + x_2 \leq 8$ and $x_1 + 2x_2 \leq 9$. We start by converting to standard linear programming model by adding slack variables. This results in standard model

$$\min_x c^T x = \min_x \begin{pmatrix} -4 & -5 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

subject to

$$Ax = b$$

$$\begin{pmatrix} 3 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 8 \\ 9 \end{pmatrix}$$

Here is the call and result returned which is the final tableau

```
A=[3,1,1,0;
   1,2,0,1];
b=[8,9];
c=[-4,-5,0,0];
format short;
nma_simplex(A,b,c,false)
```

Which returns

```
ans =

    1.0000         0    0.4000   -0.2000    1.4000
         0    1.0000   -0.2000    0.6000    3.8000
         0         0    0.6000    2.2000   24.6000

solution_vector =

    1.4000
```

```
    3.8000
         0
         0
```

We see from above that the optimal x is

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1.4000 \\ 3.8000 \\ 0 \\ 0 \end{pmatrix}$$

Since $J(x) = 4x_1 + 5x_2$ then we calculate $J(x) = 4(1.4) + 5(3.8)$ which gives the optimal objective function of 24.4 dollars. This mean the vendor should make 1.4 rings and 3.8 bracelets for maximum profit.

The same solution using Matlab's linprog is

```
A=[3,1,1,0;
   1,2,0,1];
b=[8,9];
c=[-4,-5,0,0];
format short;
options = optimset('LargeScale','off','Simplex','on');
[X,FVAL,EXITFLAG,OUTPUT]=linprog(c,[],[],A,b,zeros(size(c)),[],[],options)
```

which gives

```
Optimization terminated.

X =
    1.4000
    3.8000
         0
         0

FVAL =
  -24.6000

EXITFLAG =
     1

OUTPUT =
         iterations: 0
          algorithm: 'simplex'
       cgiterations: []
            message: 'Optimization terminated.'
    constrviolation: 8.8818e-16
      firstorderopt: 8.8818e-16
```

Which is the same.

2.2 Example 2

Given

$$A_{eq} = \begin{pmatrix} 2 & 2 & 1 & 0 & 0 \\ 2 & 2 & 0 & -1 & 0 \\ -1.5 & 1 & 0 & 0 & -1 \end{pmatrix}$$

And

$$b_{eq} = \begin{pmatrix} 10 & 4 & 0 \end{pmatrix}$$

And

$$c^T = \begin{pmatrix} \frac{1}{30} & \frac{1}{15} & 0 & 0 & 0 \end{pmatrix}$$

The following gives the solution

```
A=[2,2,1,0,0;
   2,2,0,-1,0;
  -1.5,1,0,0,-1];
b=[10,4,0];
c=[1/30,1/15,0,0,0];

nma_simplex(A,b,c,false);
```

With the output printed on the console as

```
tab =

         0         0    1.0000    1.0000         0    6.0000
    1.0000         0         0   -0.2000    0.4000    0.8000
         0    1.0000         0   -0.3000   -0.4000    1.2000
    0.0333    0.0667         0         0         0         0


solution_vector =

    0.8000
    1.2000
    6.0000
         0
         0
```

From the above, we see that the solution is

$$\begin{pmatrix} 0.8 \\ 1.2 \\ 6 \\ 0 \\ 0 \end{pmatrix}$$

Given the optimal solution, the optimal objective function is now known.

To see each step and each x solution found, set the last argument to true.

```
nma_simplex(A,b,c,true);
```

Here is the result

```
>>>>Current tableau [phase one]
    2.0000    2.0000    1.0000         0         0    1.0000         0         0   10.0000
    2.0000    2.0000         0   -1.0000         0         0    1.0000         0    4.0000
   -1.5000    1.0000         0         0   -1.0000         0         0    1.0000         0
         0         0         0         0         0    1.0000    1.0000    1.0000         0

**********************
Current tableau [phase one]
    2.0000    2.0000    1.0000         0         0    1.0000         0         0   10.0000
    2.0000    2.0000         0   -1.0000         0         0    1.0000         0    4.0000
   -1.5000    1.0000         0         0   -1.0000         0         0    1.0000         0
   -2.5000   -5.0000   -1.0000    1.0000    1.0000         0         0         0         0

pivot row is 3
current basic feasible solution is
     0
     0
     0
     0
     0
    10
     4
     0

**********************
Current tableau [phase one]
    5.0000         0    1.0000         0    2.0000    1.0000         0   -2.0000   10.0000
    5.0000         0         0   -1.0000    2.0000         0    1.0000   -2.0000    4.0000
   -1.5000    1.0000         0         0   -1.0000         0         0    1.0000         0
  -10.0000         0   -1.0000    1.0000   -4.0000         0         0    5.0000         0
```

```
pivot row is 2
current basic feasible solution is
    0.8000
    1.2000
         0
         0
         0
    6.0000
         0
         0

**********************
Current tableau [phase one]
         0         0    1.0000    1.0000         0    1.0000   -1.0000         0    6.0000
    1.0000         0         0   -0.2000    0.4000         0    0.2000   -0.4000    0.8000
         0    1.0000         0   -0.3000   -0.4000         0    0.3000    0.4000    1.2000
         0         0   -1.0000   -1.0000         0         0    2.0000    1.0000    8.0000

pivot row is 1
current basic feasible solution is
    0.8000
    1.2000
    6.0000
         0
         0
         0
         0
         0

**********************
Current tableau [phase one]
         0         0    1.0000    1.0000         0    1.0000   -1.0000         0    6.0000
    1.0000         0         0   -0.2000    0.4000         0    0.2000   -0.4000    0.8000
         0    1.0000         0   -0.3000   -0.4000         0    0.3000    0.4000    1.2000
         0         0         0         0         0    1.0000    1.0000    1.0000   14.0000

**********************
Current tableau [phase two]
         0         0    1.0000    1.0000         0    6.0000
    1.0000         0         0   -0.2000    0.4000    0.8000
         0    1.0000         0   -0.3000   -0.4000    1.2000
    0.0333    0.0667         0         0         0         0
```

Using Matlab linprog

```
A=[2,2,1,0,0;
   2,2,0,-1,0;
```

```
  -1.5,1,0,0,-1];
b=[10,4,0];
c=[1/30,1/15,0,0,0];
options = optimset('LargeScale','off','Simplex','on');
[X,FVAL,EXITFLAG,OUTPUT]=linprog(c,[],[],...
                  A,b,zeros(size(c)),[],[],options)
```

Which gives

```
Optimization terminated.

X =
    0.8000
    1.2000
    6.0000
         0
         0

FVAL =
    0.1067


EXITFLAG =
     1

OUTPUT =
         iterations: 0
          algorithm: 'simplex'
       cgiterations: []
            message: 'Optimization terminated.'
    constrviolation: 0
      firstorderopt: 0
```

Which is the same solution.

2.3 Example 3

minimize $2x_1 + 3x_2$ subject to

$$\begin{aligned} 4x_1 + 2x_2 &\geq 12 \\ x_1 + 4x_2 &\geq 6 \\ x_i &\geq 0 \end{aligned}$$

We convert the problem to standard form, which results in minimize $2x_1 + 3x_2$ subject to

$$\begin{aligned} 4x_1 + 2x_2 - x_3 &= 12 \\ x_1 + 4x_2 - x_4 &= 6 \end{aligned}$$

with $x_i \geq 0$. Therefore

$$A = \begin{pmatrix} 4 & 2 & -1 & 0 \\ 1 & 4 & 0 & -1 \end{pmatrix}$$

And

$$b = \begin{pmatrix} 12 \\ 6 \end{pmatrix}$$

And $c^T = \begin{pmatrix} 2 & 3 & 0 & 0 \end{pmatrix}$. To solve this using `nma_simplex` the commands are

```
A=[4,2,-1,0;
  1,4,0,-1];
b=[12,6];
c=[2,3,0,0];
nma_simplex(A,b,c,true);
```

And the solution is

```
>>>>Current tableau [phase one]
     4     2    -1     0     1     0    12
     1     4     0    -1     0     1     6
     0     0     0     0     1     1     0

**********************
Current tableau [phase one]
     4     2    -1     0     1     0    12
     1     4     0    -1     0     1     6
    -5    -6     1     1     0     0     0

pivot row is 2
current basic feasible solution is
         0
    1.5000
         0
         0
    9.0000
         0

**********************
Current tableau [phase one]
    3.5000         0   -1.0000    0.5000    1.0000   -0.5000    9.0000
    0.2500    1.0000         0   -0.2500         0    0.2500    1.5000
   -3.5000         0    1.0000   -0.5000         0    1.5000    9.0000

pivot row is 1
current basic feasible solution is
```

```
    2.5714
    0.8571
         0
         0
         0
         0

**********************
Current tableau [phase one]
    1.0000         0   -0.2857    0.1429    0.2857   -0.1429    2.5714
         0    1.0000    0.0714   -0.2857   -0.0714    0.2857    0.8571
         0         0         0         0    1.0000    1.0000   18.0000

**********************
Current tableau [phase two]
    1.0000         0   -0.2857    0.1429    2.5714
         0    1.0000    0.0714   -0.2857    0.8571
    2.0000    3.0000         0         0         0

solution_vector =

    2.5714
    0.8571
         0
         0
```

We see from the last tableau that $x_1 = 2.5714$ and $x_2 = 0.8671$. The optimal x is also printed in the display since the the debug flag is true.

The optimal objective function is

$$\begin{aligned} J(x) &= 2(2.5714) + 3(0.8671) \\ &= 7.7441 \end{aligned}$$

2.4 Example 4

maximize $3x_1 + 5x_2$ subject to

$$\begin{aligned} x_1 + 5x_2 &\leq 40 \\ 2x_1 + x_2 &\leq 20 \\ x_1 + x_2 &\leq 12 \\ x_i &\geq 0 \end{aligned}$$

Introducing slack and surplus variables and converting to standard form gives

minimize $-3x_1 - 5x_2$ subject to

$$\begin{aligned} x_1 + 5x_2 + x_3 &= 40 \\ 2x_1 + x_2 + x_4 &= 20 \\ x_1 + x_2 + x_5 &= 12 \\ x_i &\geq 0 \end{aligned}$$

Therefore

$$A = \begin{pmatrix} 1 & 5 & 1 & 0 & 0 \\ 2 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

And

$$b = \begin{pmatrix} 40 \\ 20 \\ 12 \end{pmatrix}$$

And $c^T = \begin{pmatrix} -2 & -5 & 0 & 0 & 0 \end{pmatrix}$. To solve this using `nma_simplex`

```
A=[1,5,1,0,0;
   2,1,0,1,0;
   1,1,0,0,1];

b=[40,20,12];
c=[-3,-5,0,0,0];
nma_simplex(A,b,c,true);
```

And the solution is

```
>>>>Current tableau [phase one]
     1     5     1     0     0     1     0     0    40
     2     1     0     1     0     0     1     0    20
     1     1     0     0     1     0     0     1    12
     0     0     0     0     0     1     1     1     0

**********************
Current tableau [phase one]
     1     5     1     0     0     1     0     0    40
     2     1     0     1     0     0     1     0    20
     1     1     0     0     1     0     0     1    12
    -4    -7    -1    -1    -1     0     0     0     0

pivot row is 1
current basic feasible solution is
     0
```

```
     8
     0
     0
     0
     0
    12
     4

**********************
Current tableau [phase one]
    0.2000    1.0000    0.2000         0         0    0.2000         0         0    8.0000
    1.8000         0   -0.2000    1.0000         0   -0.2000    1.0000         0   12.0000
    0.8000         0   -0.2000         0    1.0000   -0.2000         0    1.0000    4.0000
   -2.6000         0    0.4000   -1.0000   -1.0000    1.4000         0         0   56.0000

pivot row is 3
current basic feasible solution is
     5
     7
     0
     0
     0
     0
     3
     0

**********************
Current tableau [phase one]
         0    1.0000    0.2500         0   -0.2500    0.2500         0   -0.2500    7.0000
         0         0    0.2500    1.0000   -2.2500    0.2500    1.0000   -2.2500    3.0000
    1.0000         0   -0.2500         0    1.2500   -0.2500         0    1.2500    5.0000
         0         0   -0.2500   -1.0000    2.2500    0.7500         0    3.2500   69.0000

pivot row is 2
current basic feasible solution is
     5
     7
     0
     3
     0
     0
     0
     0

**********************
Current tableau [phase one]
         0    1.0000    0.2500         0   -0.2500    0.2500         0   -0.2500    7.0000
```

```
         0         0    0.2500    1.0000   -2.2500    0.2500    1.0000   -2.2500    3.0000
    1.0000         0   -0.2500         0    1.2500   -0.2500         0    1.2500    5.0000
         0         0    0.0000         0   -0.0000    1.0000    1.0000    1.0000   72.0000

pivot row is 3
current basic feasible solution is
     0
     8
     0
    12
     4
     0
     0
     0

**********************
Current tableau [phase one]
    0.2000    1.0000    0.2000         0         0    0.2000         0         0    8.0000
    1.8000         0   -0.2000    1.0000         0   -0.2000    1.0000         0   12.0000
    0.8000         0   -0.2000         0    1.0000   -0.2000         0    1.0000    4.0000
    0.0000         0    0.0000         0         0    1.0000    1.0000    1.0000   72.0000

**********************
Current tableau [phase two]
    0.2000    1.0000    0.2000         0         0    8.0000
    1.8000         0   -0.2000    1.0000         0   12.0000
    0.8000         0   -0.2000         0    1.0000    4.0000
   -3.0000   -5.0000         0         0         0         0

pivot row is 1
current basic feasible solution is
     0
     8
     0
    12
     4

**********************
Current tableau [phase two]
    0.2000    1.0000    0.2000         0         0    8.0000
    1.8000         0   -0.2000    1.0000         0   12.0000
    0.8000         0   -0.2000         0    1.0000    4.0000
   -2.0000         0    1.0000         0         0   40.0000

pivot row is 3
current basic feasible solution is
     5
```

```
     7
     0
     3
     0

**********************
Current tableau [phase two]
         0    1.0000    0.2500         0   -0.2500    7.0000
         0         0    0.2500    1.0000   -2.2500    3.0000
    1.0000         0   -0.2500         0    1.2500    5.0000
         0         0    0.5000         0    2.5000   50.0000

solution_vector =

     5
     7
     0
     3
     0
```

We see that $x_1 = 5$ and $x_2 = 7$ and $x_4 = 3$ with all others zero. We always read the solution from the identity matrix inside the final tableau. All other x_i are zero. Hence the optimal solution is

$$\begin{pmatrix} 5 \\ 7 \\ 0 \\ 3 \\ 0 \end{pmatrix}$$

And the corresponding optimal objective function is $3x_1 + 5x_2 = 3(5) + 5(7) = 50$

Result using Matlab linprog is

```
A=[1,5,1,0,0;
   2,1,0,1,0;
   1,1,0,0,1];

b=[40,20,12];
c=[-3,-5,0,0,0];

options = optimset('LargeScale','off','Simplex','on');
[X,FVAL,EXITFLAG,OUTPUT]=linprog(c,[],[], A,b,zeros(size(c)),[],[],options)
```

Gives

```
Optimization terminated.

X =
```

```
     5
     7
     0
     3
     0


FVAL =

   -50


EXITFLAG =

     1


OUTPUT =

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

2.5 Example 5

maximize $2x_1 + 5x_2$ subject to

$$\begin{aligned} x_1 &\leq 4 \\ x_2 &\leq 6 \\ x_1 + x_2 &\leq 8 \\ x_i &\geq 0 \end{aligned}$$

Introducing slack and surplus variables and converting to standard form we now have the following problem

minimize $-2x_1 - 5x_2$ subject to

$$\begin{aligned} x_1 + x_3 &= 4 \\ x_2 + x_4 &= 6 \\ x_1 + x_2 + x_5 &= 8 \\ x_i &\geq 0 \end{aligned}$$

Therefore

$$A=\begin{pmatrix}1 & 0 & 1 & 0 & 0\\ 0 & 1 & 0 & 1 & 0\\ 1 & 1 & 0 & 0 & 1\end{pmatrix}$$

And

$$b=\begin{pmatrix}4\\ 6\\ 8\end{pmatrix}$$

And $c^T=\begin{pmatrix}-2 & -5 & 0 & 0 & 0\end{pmatrix}$. To solve this using `nma_simplex`

```
A=[1,0,1,0,0;
   0,1,0,1,0;
   1,1,0,0,1];

b=[4,6,8];
c=[-2,-5,0,0,0];
nma_simplex(A,b,c,false);
```

And the solution is

```
>>>>Current tableau [phase one]
     1     0     1     0     0     1     0     0     4
     0     1     0     1     0     0     1     0     6
     1     1     0     0     1     0     0     1     8
     0     0     0     0     0     1     1     1     0

**********************
Current tableau [phase one]
     1     0     1     0     0     1     0     0     4
     0     1     0     1     0     0     1     0     6
     1     1     0     0     1     0     0     1     8
    -2    -2    -1    -1    -1     0     0     0     0

pivot row is 1
current basic feasible solution is
     4
     0
     0
     0
     0
     0
     6
     4
```

```
**********************
Current tableau [phase one]
     1     0     1     0     0     1     0     0     4
     0     1     0     1     0     0     1     0     6
     0     1    -1     0     1    -1     0     1     4
     0    -2     1    -1    -1     2     0     0     8

pivot row is 3
current basic feasible solution is
     4
     4
     0
     0
     0
     0
     2
     0

**********************
Current tableau [phase one]
     1     0     1     0     0     1     0     0     4
     0     0     1     1    -1     1     1    -1     2
     0     1    -1     0     1    -1     0     1     4
     0     0    -1    -1     1     0     0     2    16

pivot row is 2
current basic feasible solution is
     2
     6
     2
     0
     0
     0
     0
     0

**********************
Current tableau [phase one]
     1     0     0    -1     1     0    -1     1     2
     0     0     1     1    -1     1     1    -1     2
     0     1     0     1     0     0     1     0     6
     0     0     0     0     0     1     1     1    18

**********************
Current tableau [phase two]
     1     0     0    -1     1     2
     0     0     1     1    -1     2
```

```
     0     1     0     1     0     6
    -2    -5     0     0     0     0

pivot row is 3
current basic feasible solution is
     0
     6
     2
     0
     0

**********************
Current tableau [phase two]
     1     0     0    -1     1     2
     0     0     1     1    -1     2
     0     1     0     1     0     6
    -2     0     0     5     0    30

pivot row is 1
current basic feasible solution is
     2
     6
     2
     0
     0

**********************
Current tableau [phase two]
     1     0     0    -1     1     2
     0     0     1     1    -1     2
     0     1     0     1     0     6
     0     0     0     3     2    34

solution_vector =

     2
     6
     2
     0
     0
```

We see that $x_1 = 2$ and $x_2 = 6$ and $x_3 = 2$ with all others zero. We always read the solution from the identity matrix inside the final tableau. All other x_i are zero. Hence the optimal

solution is

$$\begin{pmatrix} 2 \\ 6 \\ 2 \\ 0 \\ 0 \end{pmatrix}$$

And the corresponding optimal objective function is $2x_1 + 5x_2 = 2(2) + 5(6) = 34$

Result using Matlab's linprog

```
>> A=[1,0,1,0,0;
   0,1,0,1,0;
   1,1,0,0,1];

b=[4,6,8];
c=[-2,-5,0,0,0];
>> options = optimset('LargeScale','off','Simplex','on');
>> [X,FVAL,EXITFLAG,OUTPUT]=linprog(c,[],[], A,b,zeros(size(c)),[],[],options)
```

Gives

```
Optimization terminated.

X =
     2
     6
     2
     0
     0


FVAL =

   -34


EXITFLAG =

     1


OUTPUT =

         iterations: 0
          algorithm: 'simplex'
      cgiterations: []
            message: 'Optimization terminated.'
```

```
    constrviolation: 0
      firstorderopt: 0
```

2.6 References

1. Lecture notes, ECE 719 optimal systems, Univ. Wisconsin, Madison spring 2016 by Professor B. Ross Barmish
2. An introduction to Optimization. Edwin Chong, Stanislaw Zak. Wiley publication, NY 1996.

3 Source code listing

```
function tab = nma_simplex(A,b,c,debug)
%function [A,b,c]=nma_simplex(A,b,c)
%This function implments the simplex matrix algorithm.
%It accepts A_eq and b_eq and c as defined in standard
%documentation and generates all the simplex tableaus, and
%returns the final tableau which the user can read from it the
%minimum value of the objective function and the optimal x vector
%directly.
%
%It runs both phase one and phase two automatically.
%
%The input is
%
%A: This is the Ax=b matrix. This is for simplex standard
%   form only. The caller must convert all inequalites to
%   equalities first by using slack and suprluse variables. This
%   is what is called the Aeq matrix in Matlab documenation.
%   This function does not support Ax<b form. A has to be in
%   standard form
%
%b: Vector. This is the right hand side of Ax=b.
%
%c: Vector. This is from minimize  J(x) = c'x. As defined in
%   standard Matlab documentations.
%
%debug: flag. Set to true to see lots of internal steps.
%
%Returns:
%
%This function returns the final tableau. It has the form
%
%  [ A | b  ]
%  [ c | J  ]
```

```
%
% UPDATES
%
% Version May 12,  2016   original
% Version June 17, 2020   updated to print the final solution vector
%                         to make it easier to read.
% by Nasser M. Abbasi
% Free for use.

validate_input(A,b,c);

[A,b] = make_phase_one(A,b,debug);
tab   = simplex(A,b,c,debug,'phase two')
print_solution_vector(tab);
end
%==========================
function [A,b] = make_phase_one(A,b,debug)
[m,n]              = size(A);
tab                = zeros(m+1,n+m+1);
tab(1:m,1:n)       = A;
tab(end,n+1:end-1) = 1;
tab(1:m,end)       = b(:);
tab(1:m,n+1:n+m)   = eye(m);

if debug
    fprintf('>>>>Current tableau [phase one]\n');
    disp(tab);
end

for i = 1:m %now make all entries in bottom row zero
    tab(end,:) = tab(end,:)-tab(i,:);
end

tab = simplex(tab(1:m,1:n+m),tab(1:m,end),tab(end,1:n+m),...
                                                debug,'phase one');
%if tab(end,end) ~=0
%   error('artificial J(x) is not zero at end of phase one.');
%end

A = tab(1:m,1:n);
b = tab(1:m,end);

end
%==================================
function tab = simplex(A,b,c,debug,phase_name)
[m,n]        = size(A);
tab          = zeros(m+1,n+1);
```

```
tab(1:m,1:n) = A;
tab(m+1,1:n) = c(:);
tab(1:m,end) = b(:);

keep_running = true;
while keep_running
    if debug
        fprintf('**********************\n');
        fprintf('Current tableau [%s] \n',phase_name);
        disp(tab);
    end

    if any(tab(end,1:n)<0)%check if there is negative cost coeff.
        [~,J] = min(tab(end,1:n)); %yes, find the most negative
        % now check if corresponding column is unbounded
        if all(tab(1:m,J)<=0)
          error('problem unbounded. All entries <= 0 in column %d',J);
        %do row operations to make all entries in the column 0
        %except pivot
        else
            pivot_row = 0;
            min_found = inf;
            for i = 1:m
                if tab(i,J)>0
                    tmp = tab(i,end)/tab(i,J);
                    if tmp < min_found
                        min_found = tmp;
                        pivot_row = i;
                    end
                end
            end
            if debug
                fprintf('pivot row is %d\n',pivot_row);
            end
            %normalize
            tab(pivot_row,:) = tab(pivot_row,:)/tab(pivot_row,J);
            %now make all entries in J column zero.
            for i=1:m+1
                if i ~= pivot_row
                    tab(i,:)=tab(i,:)-sign(tab(i,J))*...
                                abs(tab(i,J))*tab(pivot_row,:);
                end
            end
        end
        if debug  %print current basic feasible solution
            fprintf('current basic feasible solution is\n');
            disp(get_current_x());
```

```
        end
    else
        keep_running=false;
    end
end

    %internal function, finds current basis vector
    function current_x = get_current_x()
        current_x = zeros(n,1);
        for j=1:n
            if length(find(tab(:,j)==0))==m
                idx= tab(:,j)==1;
                current_x(j)=tab(idx,end);
            end
        end
    end
end
%==========================
function validate_input(A,b,c)
if ~ismatrix(A)
    error('A must be matrix');
end

if ~isvector(b)
    error('b must be vector');
end
if ~isvector(c)
    error('c must be vector');
end

[m,n]=size(A);
if rank(A) <m
    error('Rank A must be equal to number of rows in A');
end

if length(b) ~= m
    error('b must have same size as number of rows of A');
end
if length(c) ~= n
    error('c must have same size as number of columns of A');
end
end

%================================
function print_solution_vector(tab)
%tab(1:m,1:n) = A;
```

```
[nRow,nCol] = size(tab);

A = tab(1:nRow-1,1:nCol-1);
b = tab(1:nRow-1,nCol);

q = A ~= 0;
q = find(sum(q,1)==1); %find all columns with one non-zero entry;

solution_vector = zeros(nCol-1,1);

for n=1:length(q)
    j = find(A(1:nRow-1,q(n))==1);
    if isempty(j)
       solution_vector(q(n)) = 0;
    else
       solution_vector(q(n)) = b(j);
    end;
end;

solution_vector

end
```