University of Illinois at Urbana-Champaign
Department of Mechanical and Industrial Engineering

IE411 Optimization of Large-Scale Linear Systems

Fall 2021

Coding project: A Revised Simplex Method Code (Due December 14, 2021)

In this project, you are asked to form groups with no more than **three** students in a group to implement your own revised simplex method code in Matlab (or Python if you prefer).

The input to your code is the following linear program:

$$\begin{aligned} \min \quad & cx \\ s.t. \quad & Ax = b \\ & x \geq 0, \end{aligned}$$

where A is a $m \times n$ matrix, b is a column vector in $\Re^m$, c is a row vector in $\Re^n$.

Step One

The first step in the project is to implement a Matlab function called simplex_step for executing a single step of the revised simplex method. We will keep track of the current basic feasible vector with three variables: iB, iN, and xB. The vector iB will hold the indices of the current set of basic variables, iN will hold the indices of the current set of nonbasic variables, and xB will hold the values of the basic variables.

The function simplex_step should be placed in a file simplex_step.m and it should have the calling sequence:

```
function [istatus,iB,iN,xB, Binv] = simplex_step(A,b,c,iB,iN,xB,Binv,irule)
%
% Take a single simplex method step for the linear program
%
%     min  cx
%     s.t.   Ax=b
%              x>=0,
%
% where A is an (m,n) matrix.
%
% That is, given a basic feasible vector described by the variables iB,iN,xB return the values
% of iB,iN, and xB corresponding to the adjacent basic feasible vector arrived at via a
% simplex method step.
%
```

IE411 Optimization of Large-Scale Linear Systems

Fall 2021

```
%      Input Parameters:
%
% A - (m,n) constraint matrix

% b - (m,1) POSITIVE vector appearing in the constraint equation above

% c - (1,n) vector giving the coefficients of the objective function
%
% iB - (1,m) integer vector specifying the indices of the basic
%        variables at the beginning of the simplex step
% iN - (1,n-m) integer vector specifying the indices of the nonbasic
%         variables at the beginning of the simplex step
% xB - (m,1) vector specifying the values of the basic
%         variables at the beginning of the simplex step
% Binv – (m,m) inverse matrix of the basis B
%
% irule - integer parameter speciying which pivot rule to use:
%       irule = 0 indicates that the smallest coefficient rule should be
%             used
%       irule = 1 indicates that Bland's rule should be used
%
%            Output Parameters:
%
% istatus - integer parameter reporting on the progress or lake thereof
%             made by this function
%         istatus = 0 indicates normal nondegenerate simplex method step
%             completed
%         istatus = 16 indicates the program is unbounded
%         istatus = -1 indicates an optimal feasible vector has been
%             found
%
% iB - integer vector specifying the m indices of the basic variables
%        after the simplex step
% iN - integer vector specifying the n-m indices of the nonbasic
%        variables after the simplex step
% xB - vector of length m specifying the values of the basic
%        variables after the simplex step
```

%

Make sure that your calling sequence for the function is exactly correct, including ordering of the input and output parameters, dimensions and choice of integers return parameters.

It is quite easy to verify that your function is working correctly. Please make sure that it performs the correct operations and reports optimal functions and unboundedness correctly.

Step Two

The second step in the project is to implement a Matlab function for performing initialization as described in the two-phase method. The function should be called simplex_init and it should be placed in the file simplex_init.m. The calling sequence for this function is:

```
function [istatus,iB,iN,xB] = simplex_init(A,b,c)
%
% Attempt to find a basic feasible vector for the linear program
%
%    min    cx
%    s.t.   Ax=b
%           x>=0,
%
% where A is a (m,n) matrix.
%
%       Input Parameters:
%
% A - (m,n) constraint matrix
% b - (m,1) vector appearing in the constraint equation above
% c - (1,n) vector giving the coefficients of the objective function
%
%       Output Parameters:
%
% istatus - integer parameter reporting the result of the initialization procedure
%      istatus = 0 indicates a basic feasible vector was found
%      istatus = 4 indicates that the initialization procedure failed
%      istatus = 16 indicates that the problem is infeasible
%
```

```
% iB - integer vector of length m specifying the indices of the basic variables
% iN - integer vector of length n-m specying the indices of the nonbasic variables

% xB - vector of length m specifying the values of the basic variables
%
```

Again, please make sure that your function conforms exactly to the calling sequence given above.

Step Three

The final step in the project is the implementation of a Matlab function simplex_method which used the preceding two functions in order to compute a solution to a linear program.

The calling sequence for the function simplex_method, which should reside in the file simplex_method.m, is as follows:

```
function [istatus,X,eta,iB,iN,xB] = simplex_method(A,b,c,irule)
%
% Find a basic optimal solution for the linear program
%
%    min   cx
%    s.t.    Ax=b
%              x>=0,
%
% where A is an (m,n) matrix.
%
%         Input Parameters:
%
% A - (m,n) constraint matrix
% b - (m,1) a vector appearing in the constraint equation above
% c - (1,n) vector giving the coefficients of the objective function
%
% irule - integer parameter speciying which pivot rule to use:
%      irule = 0 indicates that the smallest coefficient rule should be used
%      irule = 1 indicates that Bland's rule should be used
%
```

IE411 Optimization of Large-Scale Linear Systems

Fall 2021

```
%       Output Parameters:
%

% istatus - integer parameter reporting the results obtained by this function
%       istatus = 0 indicates normal completion (i.e., a solution has been found and reported)
%       istatus = 4 indicates the program is infeasible
%       istatus = 16 indicates the program is feasible but our initialization procedure has failed
%       istatus = 32 indicates that the program is unbounded
%
% X - vector of length n specifying the solution
% eta - the minimum value of the objective function
% iB - integer vector specifying the m indices of the basic variables after the simplex step
% iN - integer vector specifying the n-m indices of the nonbasic variables after the simplex step
% xB - vector of length m specifying the values of the basic variables after the simplex step
%
```

Grading

We will generate linear program instances to test whether your code returns the correct solutions. You will receive a full mark if the test is passed. To facilitate the development of your code, some test files are provided, which should be self-explanatory.

Extra credit: Alternative implementations

In the above implementation, the inverse of the basis is derived by carrying out the pivot. For sparse problems (problems with a lot of zero elements in A), an alternative implementation is to store the inverse of the basis as the product of elementary matrices. This leads to fewer storage and computational burden, and it provides greater numerical stability by reducing accumulated round-off errors. You will get extra credit if you can also implement this approach and compare it with the tableau pivoting implementation. Even better if you can carry out the LU factorization approach. For information of the alternative implementations, please read Bazaraa, Jarvis and Sherali, *Linear Programming and Network Flows (fourth edition),* Chapter 5.

Submission: Please complete the following report and submit it with your code in a zip file through compass (one submission for each group only)

University of Illinois at Urbana-Champaign
Department of Mechanical and Industrial Engineering

IE411 Optimization of Large-Scale Linear Systems

Fall 2021

In the case the simplex_step input is not consistent with the test code which does not have BInv, you can choose one of the two options:
1. Remove BInv from the input of simplex_step. This of course means you cannot pass the information to the next pivot unless you define BInv as a global variable.
2. Keep BInv as an input of simplex_step. You will then change the test code to accommodate that.
Another thing I would like to mention is the failure of the initialization procedure, which means that the first phase ends up with an artificial variable still in the basis but taking value zero when we stop the first phase at an optimal solution.

Team members:

1. Did your code pass all the test cases?

2. Does your code pass the inverse of the basis to the next pivot in your implementation or not?

3. Did your implement the alternative approach of storing the inverse of the basis as the product of elementary matrices?

4. Did you implement the LU factorization approach?