

PROGRAMMING PROJECT TWO: A SIMPLE SIMPLEX METHOD CODE
(DUE FRIDAY MAY 22, 2008)

INTRODUCTION

The object of this project is to implement your own simplex method code in Matlab. There are two big complications and one minor complication in “industrial strength” linear programming code: numerical stability and the preservation of sparsity are the major problems (and often contrary goals) and initialization is a minor problem. As novices, our first goal is to get a basic implementation working without worrying too much about these complications.

THE OBJECTIVE

You will be implementing three Matlab functions which together solve linear programs of the form

$$(1) \quad \begin{array}{ll} \text{min:} & cx \\ \text{subject to:} & Ax = b \\ & x \geq 0, \end{array}$$

where A is a $m \times n$ matrix, x is a row vector of length n , and c is a column vector of length n . You may not assume that A has any special form; in particular, you should not assume that A is a constraint matrix arising from the introduction of slack variables of the form $(A \ I)$.

You will be implementing the “revised simplex method,” the algorithm described in Chapter 6 of the textbook, rather than the more familiar tableau method. The two algorithms perform equivalent operations in the same order, but differ in what they store and in the nitty gritty details of the computations at each step. Although the tableau method is easier for human beings, there are numerous advantages to using the revised simplex method for a computer implementation.

A MORE GENERAL INITIALIZATION PROCEDURE

The first initialization procedure we studied, which uses an auxillary linear program to find a basic feasible vector, is inadequate for the initialization of problems of the form (1). We could use the dual initialization method we studied, but that would require the implementation of the dual simplex method, which we would like to avoid. Instead, we will introduce another, slightly more general, initialization procedure. The downside will be that the new procedure will not always work.

Beginning with a problem of the form (1), we introduce m new variables y_1, \dots, y_m and solve the auxillary problem

$$(2) \quad \begin{array}{ll} \text{min:} & y_1 + y_2 + \dots + y_m \\ \text{subject to:} & Ax + y = b \\ & x \geq 0 \\ & y \geq 0. \end{array}$$

It is clear that we can always find a basic feasible vector for the new auxillary problem (in particular, we choose the y_j to be the basic variables, and the corresponding columns of the constraint matrix are the identity).

The original problem has an initial basic feasible vector if and only if the problem (2) has a solution such that $y_1 = y_2 = \dots = y_m = 0$.

There is, however, one complication. If we arrive at a solution of (2) for which all of the basic variables are x 's, then we have a initial basic feasible vector for the initial problem. If, on the other hand, some of the y_j 's are basic feasible for our solution of (2), then we would have to perform further manipulations to get a basic feasible vector for the original problem. We won't worry about this. Our initialization procedure will attempt to find the solution to the auxillary problem (2). If we find such a solution and the basic variables include the y 's, we will simply report that our initialization procedure failed.

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] = simplex_step(A,b,c,iB,iN,xB,irule)
%
% Take a single simplex method step for the linear program
%
%   min:    c*x
%   ST:     Ax=b
%           x>=0,
%
% where A is an (m,n) matrix.
%
% That is, given a basic feasible vector 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.
%
%                               Input Parameters:
%
% A - (n,m) 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 specyng 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
%
% irule - integer parameter speciyng 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. You do not need to verify in your code that the inputs to the function are valid.

STEP TWO

The second step in the project is to implement a Matlab function for performing initialization. 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
%
%      max:    c*x
%      ST:    Ax=b
%            x>=0,
%
% where A is a (m,n) matrix.
%
%                Input Parameters:
%
% A - (n,m) 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 specifying 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. Note your procedure is allowed to fail only under the conditions discussed in the section “A More General Initialization Procedure.”

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:    c*x
%      ST:     Ax=b
%             x>=0,
%
% where A is an (m,n) matrix.
%
%             Input Parameters:
%
% A - (n,m) constraint matrix
% b - (m,1) POSITIVE vector appearing in the constraint equation above
% c - (1,n) vector giving the coefficients of the objective function
%
% irule - integer parameter specifying 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 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
%
```

EXTRA CREDIT: FOOLPROOF INITIALIZATION

Using any technique you would like, modify your function `simplex_init` so that it will not fail unless the linear program is infeasible. Solving a modified dual of the input problem is one way to do this. Modifying the algorithm we used to initialize is another.