

Clemson University
Mathematical and Statistical Sciences

Summer I Project Report

Simple Pivot Tool for Bounded Simplex Method



Sandra Annie Tsiorintsoa
with: Rakhi Goswami

Supervised by: Dr. Matthew Saltzman
June 23, 2020

Contents

1	Introduction	2
2	Literature Survey	2
2.1	Linear programming problem in standard form	2
2.1.1	Dictionary of linear programming problem in standard form:	2
2.1.2	Optimality condition:	3
2.1.3	Infeasibility:	3
2.1.4	Unboundedness:	3
2.2	Bounded linear programming problem	3
2.2.1	Dictionary of linear programming problem in standard form:	4
2.2.2	Optimality condition:	4
2.2.3	Infeasibility:	4
2.2.4	Unboundedness:	4
3	Methodology	4
3.1	Analysis	4
3.2	Design	5
3.3	Implementation and Pivoting mechanism	8
3.4	Pivoting mechanism	10
3.5	Removed functions	15
3.6	Other modified and new functions	15
4	Summary and Further works	22
5	Learning Outcomes and Collaboration	23

1 Introduction

The Simplex method was introduced by Dantzig in 1947 to solve the problems of linear programming. Practical implementations of the simplex method has been discussed by many researchers. Robert J. Vanderbei at Princeton University developed an important tool to do pivoting of simplex method for linear programming problem in standard form. It is called [Simplex Pivot Tool](#) with title "Simple Pivot Tool. Most of the time, real world linear programming problems are not directly in standard form. If the problem have bounds for all or part of the variables, we call it bounded linear programming problem or linear programming in general form. Even all linear programming problems can be transform into standard form, we can deal with a problem directly without converting it, using an appropriate adaptation of the simplex method. We call it the bounded simplex method.

Aims and Objectives

Talking about practical implementations, this project aims to build a useful pivot tool for bounded simplex method by using a suitable modification of [Simplex Pivot Tool](#).

Robert Vanderbei used the JavaScript language to develop the tool. Regarding learning objectives, we are willing to enhance our skills of programming with JavaScript. Also, we intend to learn and get used to the mechanism of bounded simplex method, and its contrast to simplex method in standard form.

2 Literature Survey

2.1 Linear programming problem in standard form

In standard form, linear programming problem is given by:

$$\begin{aligned} \max \quad & \mathbf{c}^T \mathbf{x} \\ \text{Subject to: } & \mathbf{Ax} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned}$$

where A be a full rank $m \times n$ matrix, $m \leq n$

2.1.1 Dictionary of linear programming problem in standard form:

$$\begin{aligned} \zeta &= \mathbf{c}_B^T B^{-1} \mathbf{b} & + & (\mathbf{c}_N^T - \mathbf{c}_B^T B^{-1} N) \mathbf{x}_N \\ \mathbf{x}_B &= B^{-1} \mathbf{b} & - & B^{-1} N \mathbf{x}_N \end{aligned}$$

$$\mathbf{x}_B, \mathbf{x}_N \geq \mathbf{0}.$$

There are three possible statues for a dictionary: Optimal, Infeasible, or Unbounded.

2.1.2 Optimality condition:

If $\mathbf{c}_N^T - \mathbf{c}_B^T B^{-1} N \leq \mathbf{0}$ and $B^{-1} \mathbf{b} \geq \mathbf{0}$, then the current dictionary is optimal.

2.1.3 Infeasibility:

If $\exists 0 \leq i \leq m, B^{-1} \mathbf{b}_i < 0$ and $B^{-1} A_{i.} \leq \mathbf{0}$ then the current dictionary is infeasible.

2.1.4 Unboundedness:

If $\exists 0 \leq j \leq n, \mathbf{c}_j - \mathbf{c}_B^T B^{-1} A_{.j} > 0$ and $B^{-1} A_{.j} \geq \mathbf{0}$ then the current dictionary is unbounded.

The [Simplex Pivot Tool](#) is displayed using dictionary.

Current Dictionary

maximize	$\zeta =$													
				0	x_1	+	0	x_2	+	0	x_3	+	0	x_4
subject to:	$w_1 =$	0	-	0	x_1	-	0	x_2	-	0	x_3	-	0	x_4
	$w_2 =$	0	-	0	x_1	-	0	x_2	-	0	x_3	-	0	x_4
	$w_3 =$	0	-	0	x_1	-	0	x_2	-	0	x_3	-	0	x_4

$x_1, x_2, x_3, x_4, w_1, w_2, w_3 \geq 0$

Figure 1: Java applet of [Simplex Pivot Tool](#) using dictionary.

2.2 Bounded linear programming problem

General linear programming is of the form:

$$\begin{aligned} \max \quad & \mathbf{c}^T \mathbf{x} \\ \text{Subject to:} \quad & \mathbf{Ax} \leq \mathbf{b} \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \end{aligned}$$

2.2.1 Dictionary of linear programming problem in standard form:

$$\begin{aligned}\zeta &= \mathbf{c}_B^T B^{-1} \mathbf{b} + (\mathbf{c}_U^T - \mathbf{c}_B^T B^{-1} U) \mathbf{x}_U + (\mathbf{c}_L^T - \mathbf{c}_B^T B^{-1} L) \mathbf{x}_L \\ \mathbf{x}_B &= B^{-1} \mathbf{b} - B^{-1} U \mathbf{x}_U - B^{-1} L \mathbf{x}_L \\ &\quad l_U \leq \mathbf{x}_U = u_U \\ &\quad l_L \leq \mathbf{x}_L \leq u_L \\ &\quad l_B \leq \mathbf{x}_B \leq u_B\end{aligned}$$

where, \mathbf{x}_U : nonbasic variables at their upper bound, \mathbf{x}_L : nonbasic variables at their lower bound, \mathbf{x}_B : basic variables.

There are three possible statues for a dictionary: Optimal, Infeasible, or Unbounded.

2.2.2 Optimality condition:

If $\mathbf{c}_L^T - \mathbf{c}_B^T B^{-1} L \leq \mathbf{0}$ and $\mathbf{c}_U^T - \mathbf{c}_B^T B^{-1} U \geq \mathbf{0}$ and $l_B \leq \mathbf{x}_B \leq u_B$, then the current dictionary is optimal.

2.2.3 Infeasibility:

If $\exists 0 \leq i \leq m$, $[x_{B_i} \leq l_{B_i} \text{ and } B^{-1} A_{i.} \leq \mathbf{0}]$ or $[x_{B_i} \geq u_{B_i} \text{ and } B^{-1} A_{i.} \geq \mathbf{0}]$

2.2.4 Unboundedness:

If $\exists j$, $\mathbf{c}_j - \mathbf{c}_B^T B^{-1} L_{.j} > 0$ and $B^{-1} A_{.j} \leq \mathbf{0}$, or $\mathbf{c}_j - \mathbf{c}_B^T B^{-1} U_{.j} < 0$ and $B^{-1} U_{.j} \geq \mathbf{0}$, then the current dictionary is unbounded.

3 Methodology

For the modification, we first need to know how we are going to implement the bounded simplex method using dictionary looking at the [Simplex Pivot Tool](#) Java applet. Then, we analyze how do they develop the [Simplex Pivot Tool](#) using JavaScript. Once we have an overview of what the code does, we note down what we should add, remove, keep or modify from their code to build our extended one.

From [Simplex Pivot Tool](#), we need to view the page source to grab the JavaScript code and all referred code sources like .css and .js from the html code source.

3.1 Analysis

There are two code sources [simplex.css](#) and [sprintf.js](#) that have been referred in the [simple.html](#):

- [simplex.css](#) is about the style of the [simple.html](#). We modify it and name it as "boundedsimplex.css".
- [sprintf.js](#) is covered by copyright. We didn't use it in this project. It used once in the original code ([simple.html](#)), when it comes to "decimal" format.

There are about 40 functions created in "[simple.html](#)". Among them, about 20 are considered as main functions that we should not remove. We removed 2 functions, about 20 remain the same and we added 8 new functions that we will elaborate shortly.

3.2 Design

The display of [Simplex Pivot Tool](#) uses dictionary. We also use dictionary for "Bounded Simplex Pivot Tool". We need to add two more rows and three more columns to [1](#).

The two additional rows are for Lower-bound and Upper-bound values of each nonbasic variables. The three more columns are basic variables' values, lower and upper bound for each of them.

Our display looks like:

Current Dictionary

maximize:	ζ	=	0	+	0	x_1	+	0	x_2	+	0	x_3	+	0	x_4	vb	lb	ub	
subject to:	w_1	=	0	-	0	x_1	-	0	x_2	-	0	x_3	-	0	x_4	=	0	0	0
	w_2	=	0	-	0	x_1	-	0	x_2	-	0	x_3	-	0	x_4	=	0	0	0
	w_3	=	0	-	0	x_1	-	0	x_2	-	0	x_3	-	0	x_4	=	0	0	0
lowerbound:					0	x_1		0	x_2		0	x_3		0	x_4				
upperbound:					0	x_1		0	x_2		0	x_3		0	x_4				

$0 \leq w_1 \leq 0; 0 \leq w_2 \leq 0; 0 \leq w_3 \leq 0$
 $0 \leq x_1 \leq 0; 0 \leq x_2 \leq 0; 0 \leq x_3 \leq 0; 0 \leq x_4 \leq 0$

Figure 2: Java applet Bounded' simplex Pivot Tool.

This dictionary in [2](#) is read as:

- entries of column vb are values of basic variables w_1, w_2, w_3 after plug in the value of nonbasic variables at their lower bound or upper bound. Column lb consists of lower bound of basic variables, and ub consists of upper bound of basic variables.
- the additional rows "lowerbound" and "upperbound" involve values of lower bound and upper bound of nonbasic variables x_1, \dots, x_4 respectively.

next, we are going to see all functions in [simple.html](#) to build the display.

Function "start()" and "toGrid()" in [simple.html](#) design those displays. Hence, we had to modify them. We added lines to start() and carried the modification on toGrid(), setFormat(v), genrand() and so on.

```

1 add("text", " " ,1 ,0,"readonly","sign","e" );
2 add("button" ,"vb" ,1 ,0,"readonly","butvar" ,"vb"+0 );
3 add("text", " " ,1 ,0,"readonly","equals","e" );
4 add("button", "lb",1 ,0,"","butvar" ,"lb"+0);
5 add("text", " " ,1 ,0,"readonly","sign" ,"e" );
6 add("button", "ub",1 ,0,"","butvar" ,"ub"+0);
7 add("text", " " ,1 ,0,"readonly","sign","e" );

```

Listing 1: Added lines in function start(). Add new elements with id to the html of the zeroth row of the dictionary.

```

1 add("text", " = " ,1 ,i,"readonly","equals","e"+i );
2 add("text" ,"0" ,1 ,i,"","val" ,"vb"+i);
3 add("text", " " ,1 ,i,"readonly","equals","e"+i );
4 add("text", "0",1 ,i,"","val" ,"lb"+i);
5 add("text", " " ,1 ,i,"readonly" ,"sign" ,"e" );
6 add("text", "0",1 ,i,"","val" ,"ub"+i);
7 add("text", " " ,1 ,i,"readonly","sign","e" );

```

Listing 2: Added lines in function start(). Add new elements with id to the html through first to m^{th} row of the dictionary.

```

1 add("text", "",1,m+1,"readonly","sign","mb" +(m+1));
2 document.getElementById("mb" +(m+1)).style.textAlign = "right";
3 add("text", " " ,1,m+1,"readonly","sign","e0" );
4 add("text", " " ,3,m+1,"readonly","sign","e0" );
5 add("text", " " ,5,m+1,"readonly","sign","e0" );
6
7 for(j=1;j<=n;j++){
8 add("text" ," " ,3,m+1,"readonly","sign" ,"minus" +(m+1)+"," +
   j );
9 add("text" ,"0" ,5,m+1,"" ,"val" ,"l" +(m+1)+"," +j
   );
10 add("button",xlabel[j]+"",3,m+1,"","butvar" ,"x" +(m+1)+"," +j);
11 }
12
13 add("text", " " ,1 ,m+1,"readonly","sign","e" );
14 add("text" ," " ,1 ,m+1,"readonly","sign" ,"vb" +(m+1) );
15 add("text", " " ,1 ,m+1,"readonly","equals","e" );
16 add("text", "",1 ,m+1,"readonly","sign" ,"lb" +(m+1));
17 add("text", " " ,1 ,m+1,"readonly","sign" ,"e" );
18 add("text", "",1 ,m+1,"readonly","sign" ,"ub" +(m+1));
19 add("text", " " ,1 ,m+1,"readonly","sign","e" );

```

Listing 3: Added lines in function start(). Add new elements with id to the html for the lowerbound row $((m+1)^{st}$ row) of the dictionary.

Function add() add elements to the Html DOM with attributes: type (text, button, input,etc.), name, width, row, rw (read or write only), clas (developped in "boundedsimplex.css" for style, color, etc), id.

For instance: add("text"," ",1,m+1,"readonly","sign","e"): this add an element with id: "e", type: "text", name: " ", width: 1, row: m+1, rw: "readonly", clas:"sign".

```

1 add("text"," ",1,m+2,"readonly","sign","mb"+(m+2));
2 document.getElementById("mb"+(m+2)).style.textAlign = "right";
3 add("text"," ",1,m+2,"readonly","sign","e0");
4 add("text"," ",3,m+2,"readonly","sign","e0");
5 add("text"," ",5,m+2,"readonly","sign","e0");
6
7 for (j=1; j<=n; j++) {
8 add("text"," ",3,m+2,"readonly","sign","minus"+(m+2)+" "+j);
9 add("text","0",5,m+2,"","val","u"+(m+2)+" "+j);
10 add("button",xlabel[j]+"",m+2,"","butvar","x"+(m+2)+" "+j);
11 }
12
13 add("text"," ",1,m+2,"readonly","sign","e");
14 add("text"," ",1,m+2,"readonly","sign","vb"+(m+2));
15 add("text"," ",1,m+2,"readonly","equals","e");
16 add("text"," ",1,m+2,"readonly","sign","lb"+(m+2));
17 add("text"," ",1,m+2,"readonly","sign","e");
18 add("text"," ",1,m+2,"readonly","sign","ub"+(m+2));
19 add("text"," ",1,m+2,"readonly","sign","e");

```

Listing 4: Added lines in function start(). Add new elements with id to the html for the lowerbound row $((m+2)^{th}$ row) of the dictionary.

From nonnegativity constraints in 1 to the bounded variables constraints in 2, we added the following lines to start() and toGrid().

From	to
$x_1, x_2, x_3, x_4, w_1, w_2, w_3 \geq 0$	$0 \leq w_1 \leq 0; 0 \leq w_2 \leq 0; 0 \leq w_3 \leq 0$ $0 \leq x_1 \leq 0; 0 \leq x_2 \leq 0; 0 \leq x_3 \leq 0; 0 \leq x_4 \leq 0$

```

1 add("button", "", 80, m+3, "readonly", "val", "basicvars");
2 add("button", "", 80, m+4, "readonly", "val", "nonbasicvars");

```

Listing 5: Added lines in function start(). Add new elements with id to the html for the $(m+3)^{rd}$ and $(m+4)^{th}$ row of the dictionary (bounded constraints).


```

1  for(j=1;j<=n;j++){
2  nnv1 += l[j] + " \u2264" + document.getElementById("x0,"+j).value +
    "\u2264 " +u[j];
3      if(j<n){nnv1 += " "; }
4      }
5
6  for(i=1;i<=m;i++){
7  nnv += lb[i] + " \u2264" + document.getElementById("w"+i).value + " \
    \u2264" +ub[i];
8      if(i<m){nnv+=" "; }
9      }
10 if (listlen == 0) {
11     element = document.getElementById("basicvars");
12     element.value = nnv;
13     element.style.color = "gray";
14 }
15 if (listlen == 0) {
16     element = document.getElementById("nonbasicvars");
17     element.value = nnv1;
18     element.style.color = "gray";
19 }

```

Listing 6: *Added lines in function toGrid(). Develop the bounded constraints..*

Code lines in 21 and 3 aim to design the bounded constraints in figure 2. The first two lines in (21) is doing: $l[j] \leq x_j \leq u[j]$ and $lb[i] \leq w_i \leq ub[i]$.

3.3 Implementation and Pivoting mechanism

It is worth mentioning that there are several kinds of method developed in [Simplex Pivot Tool](#): Primal, Dual, Lexicographic, Klee Minty, Generic and in the [Advanced Pivot Tool](#) there are methods with Phase I for instance, Primal with x0. We exclude all but not Primal for now. Thus, every code lines that include them will just be deleted.

Simple Pivot Tool

So general: Constraints = , Variables = . Labels:

Seed =

Or: Select an Exercise from *Linear Programming: Foundations and E*

Number format: Zero Visibility:

Current Dictionary

maximize $\zeta =$ x_1 + x_2 + x_3 + x_4

subject to:

$w_1 =$	<input type="text" value="0"/>	-	<input type="text" value="0"/> x_1	-	<input type="text" value="0"/> x_2	-	<input type="text" value="0"/> x_3	-	<input type="text" value="0"/> x_4
$w_2 =$	<input type="text" value="0"/>	-	<input type="text" value="0"/> x_1	-	<input type="text" value="0"/> x_2	-	<input type="text" value="0"/> x_3	-	<input type="text" value="0"/> x_4
$w_3 =$	<input type="text" value="0"/>	-	<input type="text" value="0"/> x_1	-	<input type="text" value="0"/> x_2	-	<input type="text" value="0"/> x_3	-	<input type="text" value="0"/> x_4

$x_1, x_2, x_3, x_4, w_1, w_2, w_3 \geq 0$

Figure 3: Simple pivot Tool java applet with Primal, Dual, Lexicographic, Klee Minty, Generic method choices.

Since we have "vb" which contained values of basic variables, we need to calculate them when we implement the values of non basic variables at their upper or lower bounds. Hence, we create a function called "vbval()" in 20 to do that. Other than that, we need to indicate which nonbasic variables are at their upper or lower bounds. For that one, we indicate variables at their upper bounds with colored upper bound value, "purple", and those at their lower bounds with colored lower bound value, "orange". Candidate entering variable is indicated by pink color on the top row of the dictionary. Values with pink color on "vb" column indicates that corresponding basic variable of that value is outside its bound. To better understand that we display an example in figure 4. The example in 4 means:

- basic variables w_1, w_2, w_3 are between 0 and ∞ ,
- nonbasic variables: $-7 \leq x_1 \leq 4, 0 \leq x_2 \leq 1, -5 \leq x_3 \leq 5, -3 \leq x_4 \leq 1$. x_1, x_2 are at their lower bound that is $x_1 = -7, x_2 = 0$. x_3, x_4 are at their upper bound that is $x_3 = 5, x_4 = 1$. And then column "vb", $w_1 = 23, w_2 = -7, w_3 = 36$ is calculated using function "vbval()" in 20 after implementing $x_1 \cdots x_4$.
- Nonbasic variable x_1, x_4 are candidates to enter and the basic variable $w_2 = -7$ is outside its bound.

Current Dictionary																			
maximize:	ζ	=	0	+	14	x_1	+	-9	x_2	+	1	x_3	+	-7	x_4	vb	lb	ub	
subject to:	w_1	=	12	-	5	x_1	-	-8	x_2	-	4	x_3	-	-8	x_4	=	23	0	Infinity
	w_2	=	13	-	-5	x_1	-	1	x_2	-	-2	x_3	-	1	x_4	=	-7	0	Infinity
	w_3	=	7	-	3	x_1	-	4	x_2	-	-2	x_3	-	6	x_4	=	38	0	Infinity
lowerbound:					-7	x_1		0	x_2		-5	x_3		-3	x_4				
upperbound:					4	x_1		1	x_2		8	x_3		1	x_4				

Figure 4: Example of implementation, indication of nonbasic variables at their lower bounds (x_1 and x_2), at their upper bounds (x_3 and x_4).

```

1 function vbval(i){
2   var uz=[];var lz=[];
3   for (j=1; j<=n; j++)
4     { element=document.getElementById("l"+(m+1)+","+j);
5       var element1=document.getElementById("u"+(m+2)+","+j);
6       if(element.className=="orangeval")
7         {lz[j]=j;}
8       if( element1.className=="purpleval")
9         {uz[j]=j;}
10    }
11   var uf=uz.filter(x=>x!=null);
12   var lf=lz.filter(x=>x!=null);
13   var vbj=0; var vb=0;
14     for (j of lf) {
15       vbj+=a[i][j]*l[j];}
16     for (k of uf) {
17       vb+=a[i][k]*u[k];}
18   return b[i]+vbj+vb;
19 }

```

Listing 7: Added lines in function toGrid(). Develop the bounded constraints..

In the function "vbval()" of the code 20 above, we first create two arrays to extract the indices of those with color orange (for lz) and purple (for uz). For example, in figure 4, we will get $lz=[1,2,"",""]$ and $uz=[""," ",3,4]$. To drop the ones with null (" ") value, we use filter to do that, line 11 and 12 in 20 just say filter uz and lz (respectively) by getting all values that are not null (" "). Hence, we will get $lf=[1,2]$ and $uf=[3,4]$. And then we just calculate values of basic variables using their formulas: $\mathbf{x}_B = B^{-1}\mathbf{b} - B^{-1}U\mathbf{x}_U - B^{-1}L\mathbf{x}_L$. We are almost there, since the display is ready to use, we can think about the mechanism of pivoting.

3.4 Pivoting mechanism

Here is the heart of this project, to be able to pivot using our Java applet. Pivoting in simplex method and bounded simplex method are quite similar. The only difference is for bounded simplex we have 2 choices for leaving variables: they can leave at its lower

or upper, while for simplex method they just become nonbasic variables. For entering variable in bounded simplex method, we have 3 choices:

- if it is at its lower bound in the current dictionary, in the very next dictionary, it can be increased to its upper bound or a basic variable will leave.
- if it is at its upper bound in the current dictionary: on the very next iteration, it can decrease to its lower bound or one basic variable will leave.

Whereas, entering variable in simplex method will just become a basic variable.

For those mechanisms, we need to find a way for users to have those choices. In [Simplex Pivot Tool](#) when user clicked on a nonbasic variable, it will pivot on that row and column position of the clicked nonbasic variable. For instance in figure 5, x_2 enters the basis and w_1 leaves when the user clicked x_2 in the current dictionary.

Current Dictionary									
maximize	ζ	=							
subject to:	w_1	=							
	w_2	=							
	w_3	=							
$x_1, x_2, x_3, x_4, w_1, w_2, w_3 \geq 0$									
Next Dictionary									
maximize	ζ	=							
subject to:	x_2	=							
	w_2	=							
	w_3	=							
$x_1, x_2, x_3, x_4, w_1, w_2, w_3 \geq 0$									







Figure 5: Pivot on x_2 and w_1 in [Simplex Pivot Tool](#).

In "Bounded' Simplex Pivot tool", we can fire an event sound to tell the user to choose where the leaving variable goes to. When user click on a nonbasic variable in the matrix of the dictionary, the background color of the clicked variable becomes blue cyan and there is a 3 seconds waiting time for that to be back to the original background (gray) and also waiting for a user to click their choice. Users need to click on "lb" if they want it to go to its lower bound and "ub" for going to its upper bound. Following are event sounds that user will hear m4a or m4a.

When "lb" or "ub" got clicked, their background color turns to be blue cyan and there is a delay of 1 second for that to get back to its original state (gray). To be able to pivot, user need to choose finite value of the upper or lower bound of the chosen nonbasic variable. And the coefficient of their nonbasic variable choice need to be nonzero. If those requirements are met and once one of "lb" or "ub" got clicked, pivoting as in [Simplex Pivot Tool](#) will follow with the chosen status of the exiting variable. Following in figure 6 is an example.

Current Dictionary																			
maximize:	ζ	=	0	+	1	x_1	+	-6	x_2	+	1	x_3	+	0	x_4	vb	lb	ub	
subject to:	w_1	=	10	-	-1	x_1	-	-5	x_2	-	-1	x_3	-	5	x_4	=	-49	0	Infinity
	w_2	=	4	-	0	x_1	-	2	x_2	-	0	x_3	-	-4	x_4	=	40	0	Infinity
	w_3	=	8	-	4	x_1	-	2	x_2	-	5	x_3	-	-2	x_4	=	2	0	Infinity
lowerbound:					-2	x_1		-8	x_2		-7	x_3		-1	x_4				
upperbound:					5	x_1		1	x_2		8	x_3		5	x_4				
Current Dictionary with x2 and lb got clicked																			
maximize:	ζ	=	0	+	1	x_1	+	-6	x_2	+	1	x_3	+	0	x_4	vb	lb	ub	
subject to:	w_1	=	10	-	-1	x_1	-	-5	x_2	-	-1	x_3	-	5	x_4	=	-49	0	Infinity
	w_2	=	4	-	0	x_1	-	2	x_2	-	0	x_3	-	-4	x_4	=	40	0	Infinity
	w_3	=	8	-	4	x_1	-	2	x_2	-	5	x_3	-	-2	x_4	=	2	0	Infinity
lowerbound:					-2	x_1		-8	x_2		-7	x_3		-1	x_4				
upperbound:					5	x_1		1	x_2		8	x_3		5	x_4				
Next Dictionary																			
maximize:	ζ	=	0	+	1	x_1	+	3	w_2	+	1	x_3	+	-12	x_4	vb	lb	ub	
subject to:	w_1	=	20	-	-1	x_1	-	5/2	w_2	-	-1	x_3	-	-5	x_4	=	51	0	Infinity
	x_2	=	2	-	0	x_1	-	1/2	w_2	-	0	x_3	-	-2	x_4	=	12	-8	1
	w_3	=	4	-	4	x_1	-	-1	w_2	-	5	x_3	-	2	x_4	=	-38	0	Infinity
lowerbound:					-2	x_1		0	w_2		-7	x_3		-1	x_4				
upperbound:					5	x_1		Infinity	w_2		8	x_3		5	x_4				

Figure 6: Pivoting processes of "Bounded' Simplex Pivot Tool"

If the lower bound (respectively upper bound) of the chosen leaving variable is "infinity" (∞) but the user clicked on "lb" (respectively "ub"), there will be a complain sound says that it is not allowed. Here are the sounds users will hear.  or . or . We fail to pivot around variables with zero coefficient, so the applet will fire a sound like  or  or .

To be able to develop those pivoting mechanism we had to add and modify some functions in [Simplex Pivot Tool](#). In function pivot() we just added lines to switch up bounds of leaving and entering variable, as in 7.

```
1 lcol=l[col];
2 ucol=u[col];
3 l[col]=lb[row];
4 lb[row]=lcol;
5 u[col]=ub[row];
6 ub[row]=ucol;
```

Listing 8: *Added lines in function pivot(). Switch up bounds of leaving and entering variable..*

We introduced couple of functions such as lbgotChoice() in 9, ubgotChoice() in 9, yellowtrue() in 7, lbgotClicked() in 9, ubgotClicked() in 9.

Those new functions incorporate in pivot mechanism. When user click on "lb" or "ub", lbgotChoice() and ubgotChoice() are executed respectively. Those two functions are used as detectives if "lb" or "ub" got clicked. "lb" or "ub" becomes blue cyan when got clicked and will return on its initial color (gray) after 1 second. Those information will be used in yellowtrue(): if "lb" or "ub" becomes cyan, lbgotClicked() or ubgotClicked() will be respectively executed.

```
1 function lbgotChoice (row,col) {
2 for (var i=0; i<100; i++) blist[i] = "lb";
3 document.getElementById("lb"+0).className="clickvar";
4     setTimeout(function(){
5         //Max timeout
6         document.getElementById("lb"+0).className="butvar";
7     }, 1000);
8 }
```

Listing 9: *New function lbgotChoice(). Detect if "lb" got clicked.*

```
1 function ubgotChoice() {
2 for (var i=0; i<100; i++) blist[i] = "ub";
3 document.getElementById("ub"+0).className="clickvar";
4     setTimeout(function(){
5         //Max timeout
6         document.getElementById("ub"+0).className="butvar";
7     }, 1000);
8 }
```

Listing 10: *New function ubgotChoice(). Detect if "ub" got clicked.*

```

1 function yellowtrue(row,col){
2 if(document.getElementById("lb"+0).className=="clickvar"){
3 lbgotClicked(row,col);}
4 else if(document.getElementById("ub"+0).className=="clickvar"){
5 ubgotClicked(row,col);}
6 else {return;}}

```

Listing 11: New function yellowtrue(). Test if "lb" or "ub" got clicked then pivot according to the choice of "lb" or "ub"..

```

1 function lbgotClicked(row,col) {
2 if(lb[row]!=-Infinity){
3     document.getElementById("l"+(m+1)+", "+col).className="orangeval";
4     pivot(row,col);
5     document.getElementById("u"+(m+2)+", "+col).className="val";
6 }
7 else{finiteB.play();}
8 }

```

Listing 12: New function lbgotClicked(). If the lower bound of the chosen basic variable is finite, the leaving variable goes to its lower bound when pivoting. If the lower bound of the chosen basic variable is infinity an event sound m4a will be played.

```

1 function ubgotClicked(row,col) {
2 if(ub[row]!=-Infinity){
3     document.getElementById("u"+(m+2)+", "+col).className="purpleval";
4     pivot(row,col);
5     document.getElementById("l"+(m+1)+", "+col).className="val";
6 }
7 else{finiteB.play();}
8 }

```

Listing 13: New function ubgotClicked(). If the upper bound of the chosen basic variable is finite, the leaving variable goes to its upper bound when pivoting. If the upper bound of the chosen basic variable is infinity an event sound m4a will get fired.

The function gotClicked(), in [Simplex Pivot Tool](#) is acting as when users click on a non-basic variable, it will just pivot around that variable position. In "Bounded Simplex Pivot Tool", we need to add an event sound to let users choose where the leaving variable goes to and then after the choice is made we can pivot. We need to record the status of clicked nonbasic variable because we need those information in function Undo().

```

1 if (Math.random() <= 0.5) {
2     bark = new Audio("fail-trombone.mp3");
3     finiteB= new Audio("finitebound_pls.m4a");
4     pivotF=new Audio("you're stuck.m4a");
5     meg = new Audio("Home_Run.mp3");
6     dothat= new Audio("lb or ub.m4a");
7     } else if (0.5< Math.random() <= 1){
8     bark = new Audio("Tryagain.mp3");
9     finiteB= new Audio("Down to earth.m4a");

```

```

10 pivotF=new Audio("can'tpivot.m4a");
11 meg = new Audio("deduction.mp3");
12 dothat=new Audio("Chooselbub.m4a");
13 }
14 else{bark = new Audio("Doomed.m4a");
15 finiteB= new Audio("Doomed.m4a");
16 pivotF=new Audio("Doomed.m4a");}

```

Listing 14: *Added lines in start() for event sounds.*

3.5 Removed functions

We deleted the functions `reset()` and `makeitshowid()` from the [Simplex Pivot Tool](#). The reason why they got removed are: we didn't see where the function `reset()` got used in [Simplex Pivot Tool](#) and the main thing is it didn't affect our needs. The `makeitshowid()` assign all zero values to a class "val" (in the `simplex.css`, the color is gray). Thus, we removed `makeitshowid()` since if a variable is at its zero lower bound or at its zero upper bound we need them to have orange color or purple color not gray.

3.6 Other modified and new functions

Functions to detect if the current dictionary is optimal, infeasible or unbounded is defined as `ck_opt()` in 31, `ck_p_infeas()` in 48 and `ck_p_unbdd()` in 38 respectively. Those functions have been changed upon optimality, infeasibility and unboundedness condition in bounded simplex method. The function `toGrid()` is the engine of the entire code, most of other functions call back to `Grid()`. Thus, obviously we modified it, we added couple of lines to it, the modification is displayed in 42. If the current dictionary is optimal, infeasible, or unbounded and the user click on the button Optimal, Infeasible or Unbounded respectively, an event sound `mp3` or `mp3` will get fired. Else if the current dictionary is not optimal, not infeasible, or not unbounded and the user click on the button Optimal, Infeasible or Unbounded respectively, an event sound `mp3` or `mp3` or `m4a` will get fired.

```

1 function ck_opt()
2 {
3     var i, j;
4     var optflag=true;
5
6     for (i=1; i<=m; i++) {
7         if (document.getElementById("vb"+i).className=="pinkval") {
8             optflag = false;
9         }
10    }
11
12    for (j=mmn+1; j<=n; j++) {element=document.getElementById("c"+j);
13        if (element.className=="pinkval") {

```



```

14     optflag = false;
15 }
16 }
17 for (j=1; j<=n; j++) {element=document.getElementById("l"+(m+1)+","+j
    );element1=document.getElementById("u"+(m+2)+","+j);
18     if (element1.className=="val" && element.className=="val") {
19         if( (l[j]!=0 || u[j]!=0) && (l[j]>0 || u[j]<0)) {
20             optflag = false;
21         }
22     }
23 }
24 if (optflag == true) {
25     meg.play();
26 } else {
27     bark.play();
28 }
29 }

```

Listing 15: *Modified ck_opt()*.

```

1 function ck_p_infeas()
2 {
3     var i, j;
4     var p_infeas_flag1=false;
5     var p_infeas_flag2=false;
6
7     for (i=1; i<=m; i++) {
8         vb[i] = fracDivide(document.getElementById("vb"+i).value);
9     }
10    for (j=mmn+1; j<=n; j++) {
11        c[j] = fracDivide(document.getElementById("c"+j).value);
12    }
13    for (i=1; i<=m; i++) {
14        for (j=1; j<=n; j++) {
15
16            a[i][j] =
17                -fracDivide(document.getElementById("a"+i+","+j).value);
18        }
19    }
20    for (i=1; i<=m; i++) {
21        if ((vb[i] < 1e-10*lb[i])) {
22            p_infeas_flag1 = true;
23            for (j=mmn+1; j<=n; j++) {
24                if (a[i][j] > 0.00001) {
25                    p_infeas_flag1 = false;
26                    break;
27                }
28            }
29        }
30        if ((vb[i] > 1e-10*ub[i])) {
31            p_infeas_flag2 = true;
32            for (j=mmn+1; j<=n; j++) {

```

```

33         if (a[i][j] < -0.00001) {
34             p_infeas_flag2 = false;
35             break;
36         }
37     }
38 }
39 if (p_infeas_flag1 == true || p_infeas_flag2 == true) break;
40 }
41 if (p_infeas_flag1 == true || p_infeas_flag2 == true) {
42     // reset();
43     meg.play();
44 } else {
45     bark.play();
46 }
47 }

```

Listing 16: *Modified ck_p_infeas() according to infeasibility condition of bounded simplex.*

```

1
2     for (j=mmn+1; j<=n; j++) {
3         if (document.getElementById("l"+(m+1)+","+j).className=="orangeval"
4             && c[j] > 1e-10){
5             d_infeas_flag=true;
6             for (i=1; i<=m; i++) {
7                 if (a[i][j] > 0.00001) {
8                     d_infeas_flag = false;
9                     break;
10                }
11            }
12            if (document.getElementById("u"+(m+2)+","+j).className=="
13                purpleval" && c[j] < -1e-10){
14                d_infeas_flag=true;
15                for (i=1; i<=m; i++) {
16                    if (a[i][j] < -0.00001) {
17                        d_infeas_flag = false;
18                        break;
19                    }
20                }
21                if (d_infeas_flag == true) {
22                    for (i=1; i<=m; i++) {
23                        if ((vb[i] < 1e-10*lb[i]) || (vb[i] > 1e-10*ub[i])) {
24                            d_infeas_flag = false;
25                            break;
26                        }
27                    }
28                }
29                if (d_infeas_flag == true) break;
30            }
31            if (d_infeas_flag == true) {
32                // reset();
33                meg.play();
34            } else {

```

```

33     bark.play();
34
35     }
36 }

```

Listing 17: *Modified lines in ck_p_unbdd().*

Following are added lines to toGrid(). Those lines code the fact that a variable cannot be at its lower bound and its upper at the same time. If a user enter small value of upper bound of a variable than the lower bound of that variable, those will automatically switched up. Every time the dictionary get updated, the value of basic variable "vb" column is updated too.

```

1  for (i=1;i<=n;i++) {
2      element = document.getElementById("l"+(m+1)+","+i);
3  if( element.className=="orangeval" )
4      { document.getElementById("u"+(m+2)+","+i).className="val "
5      }
6  for (i=1;i<=n;i++) {
7      element= document.getElementById("u"+(m+2)+","+i);
8  if( element.className=="purpleval" )
9      { document.getElementById("l"+(m+1)+","+i).className="val "
10     ;}
11 }
12 for (i=1;i<=n;i++) {
13     element = document.getElementById("l"+(m+1)+","+i);
14     element1= document.getElementById("u"+(m+2)+","+i);
15     if( (element.value== Infinity) )
16     { element.className="val ";}
17     if( (element1.value== Infinity) ){ element1.className="val "}
18     if(u[i]<=l[i]){tmp=u[i];u[i]=l[i];l[i]=tmp;}
19 }
20 for(i=1;i<=m;i++){
21     vb[i]=vbval(i);
22     if(ub[i]<=lb[i]){tmp=ub[i];ub[i]=lb[i];lb[i]=tmp;}

```

Listing 18: *Added lines to toGrid().*

In the function setExercise(), we just added one set up example (we took it from Math8130 note) and we removed all exercises from [Simplex Pivot Tool](#).

User can click on variables on the row of "lowerbound" and "upperbound" to set up variables at its lower bound or at its upper bound upon their choice. That is developed with two more new functions lowgotChoice() and upgotChoice(). The function toGrid() is called inside of those added functions for them to get synchronously executed while updates happen. Users choices will be recorded by lowlist[i] and uplist[i] for later use in function Undo().

```

1 function lowgotChoice(col) {
2
3
4     collist[listlen] = col;
5     if (listlen < 99) { listlen++; } else {listlen=0;}
6     for (var i=0; i<100; i++) rowlist[i] = -1;
7     if((document.getElementById("l"+(m+1)+","+col).className=="orangeval"
8         ")|| (l[col]==-Infinity )){
9 for (var i=0; i<100; i++) lowlist[i] = "lco";document.getElementById(
10    "l"+(m+1)+","+col).className="val";}
11     else if ((document.getElementById("l"+(m+1)+","+col).className=="
12         val")&& (document.getElementById("u"+(m+2)+","+col).className=="
13         val")){
14 for (var i=0; i<100; i++) lowlist[i] = "lcv";document.getElementById(
15    "l"+(m+1)+","+col).className="orangeval";}
16 else if ((document.getElementById("l"+(m+1)+","+col).className=="val"
17     )&& (document.getElementById("u"+(m+2)+","+col).className=="
18     purpleval")){
19 for (var i=0; i<100; i++) lowlist[i] = "lc";document.getElementById("
20    l"+(m+1)+","+col).className="orangeval";document.getElementById("u
21    "+(m+2)+","+col).className="val";}
22 toGrid();}
23
24 function upgotChoice(col) {
25
26
27     collist[listlen] = col;
28     if (listlen < 99) { listlen++; } else {listlen=0;}
29     for (var i=0; i<100; i++) rowlist[i] = -1;
30     if((document.getElementById("u"+(m+2)+","+col).className=="
31         purpleval")|| (u[col]==Infinity )){
32 for (var i=0; i<100; i++) uplist[i] = "uco";document.getElementById("
33    u"+(m+2)+","+col).className="val";}
34     else if ((document.getElementById("u"+(m+2)+","+col).className=="
35         val")&& (document.getElementById("l"+(m+1)+","+col).className=="
36         val")){
37 for (var i=0; i<100; i++) uplist[i] = "ucv";document.getElementById("
38    u"+(m+2)+","+col).className="purpleval";}
39 else if ((document.getElementById("u"+(m+2)+","+col).className=="val"
40     )&& (document.getElementById("l"+(m+1)+","+col).className=="
41     orangeval")){
42 for (var i=0; i<100; i++) uplist[i] = "uc";document.getElementById("u
43    "+(m+2)+","+col).className="purpleval";document.getElementById("l"
44    +(m+1)+","+col).className="val";}
45 toGrid();}

```

Listing 19: *New functions. User can click on variables on the row of "lowerbound" and "upperbound" to set up variables at its lower bound or at its upper bound upon their choice.*

The Hint() and Undo() functions also got modified. In function Hint(), if all basic variables are within their bounds we do the ratio test of candidate exiting variable at their lower bound or upper bound. If there are basic variables that are outside their bounds, they are

candidate to exit the basis. Candidate variables will have background color green for 300 milliseconds developed by setTimeout(). After 300 milliseconds the variable will go back to its initial background (gray).

```

1 var colov=[];
2 for (i=1; i<=m; i++) {
3     element=document.getElementById("vb"+i);
4     colov[i]=window.getComputedStyle(element).backgroundColor;
5 }
6 var coloc=[];
7 for (i=1; i<=n; i++) {
8     element=document.getElementById("c"+i);
9     coloc[i]=window.getComputedStyle(element).backgroundColor;}
10 if ( colov.indexOf("rgb(255, 170, 255)")!= -1) {
11     for (i=1; i<=m; i++){
12         element=document.getElementById("vb"+i);
13         if ( element.className==="pinkval" ){
14             for (j=1; j<=n; j++) {
15                 var element1 =document.getElementById("x"+i+","+j);
16                 var element2 =document.getElementById("u"+(m+2)+"+","+j");
17                 var element3 =document.getElementById("l"+(m+1)+"
18                 ,"+j);
19                 if((c[j] > 1e-10 && element3.className=="
20                 orangeval")||(c[j] < 1e-10 && element2.
21                 className=="purpleval")){
22                     element1.className = "greenvar";
23                 }
24             }
25         }
26     }
27 }
28
29 setTimeout(function(){ for(i=1;i<=m;i++){for(j=1;j<=n;j++){document.
30     getElementById("x"+i+","+j).className= "butvar";}}},300);

```

Listing 20: Added lines in Hint().

Function Undo() also got modified, we introduced 4 new variables: blist[listlen] records where the leaving variable goes, nblist[listlen] records where was the entering variable, lowlist[listlen] records if a nonbasic variable changed its "lowerbound" state, uplist[listlen] records if a nonbasic variable changed its "upperbound" state.

```

1      var row, col, bsc, nbsc, lowl, upl;
2  if (listlen > 0) { listlen--; } else { listlen = 99; }
3      row = rowlist[listlen];
4      col = collist[listlen];
5          bsc= blist[listlen];
6      nbsc=nblist[listlen];
7          lowl= lowlist[listlen];
8      upl=uplist[listlen];
9      if (row !=-1) {
10         pivot(row,col);
11         if ((bsc=="ub" && nbsc=="u") || (bsc=="lb" && nbsc=="u")) {
12             document.getElementById("u"+(m+2)+","+col).className="purpleval
";
13 document.getElementById("l"+(m+1)+","+col).className="val";
14         }
15         else if ((bsc=="lb" && nbsc=="l") || (bsc=="ub" && nbsc=="l")) {
16             document.getElementById("l"+(m+1)+","+col).className="orangeval
";
17 document.getElementById("u"+(m+2)+","+col).className="val";
18         }
19     }
20 if (col!=-1 || col== -1) {
21     if (upl=="uc"){
22         document.getElementById("l"+(m+1)+","+col).className="orangeval";
23         document.getElementById("u"+(m+2)+","+col).className="
val";
24         toGrid();}
25 if (upl=="uco" && u[col] != Infinity){
26     document.getElementById("u"+(m+2)+","+col).className="
purpleval";
27     toGrid();}
28 if (upl=="ucv"){
29     document.getElementById("u"+(m+2)+","+col).className="
val";
30     toGrid();}
31 if (lowl=="lc"){
32     document.getElementById("u"+(m+2)+","+col).className="purpleval";
33     document.getElementById("l"+(m+1)+","+col).className="
val";
34     toGrid();}
35 if (lowl=="lco" && l[col] != -Infinity){
36     document.getElementById("l"+(m+1)+","+col).className="
orangeval";
37     toGrid();}
38 if (lowl=="lcv"){
39     document.getElementById("l"+(m+1)+","+col).className="

```

```

40         toGrid();}
41     }

```

Listing 21: Added lines in Undo().

Following in figure 7 is an example using Hint() function.

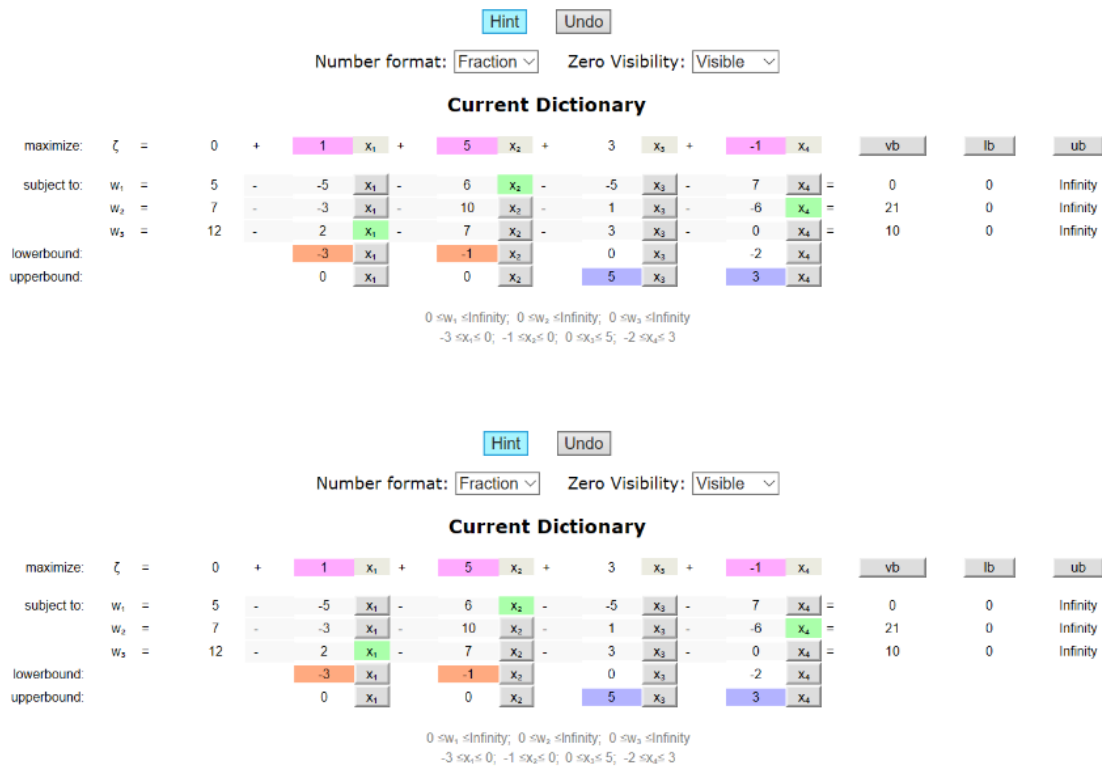


Figure 7: Example of bounded' simplex pivot tool using Hint().

4 Summary and Further works

In summary, J. Robert Vanderbei developed a tool for pivoting the simplex method which is a Java applet with name [Simplex Pivot Tool](#). Our goal in this project is to extend that tool for bounded simplex method and we named it "Bounded' Simplex Pivot Tool". After analyzing the "simplex.html" code source, a lot of functions has been modified, removed and added. We have build the basis foundation of the "Bounded' Simplex Pivot Tool". We

can pivot a bounded problem using that tool. We can also detect if a current dictionary is optimal, infeasible or unbounded.

We have just done the set up of the "Bounded Simplex Pivot Tool". There are still a lot to get improved. For instance, other method like Dual, Lexicographic, Klee-Minty, Generic. We can also make it more accessible, those event sounds can get combined with popup messages on the clicked variable using [Bootstrap JavaScript, popover or tooltip](#) or we can just improve the waiting time in function gotClicked(). "Bounded Simplex Pivot Tool" requires at least one finite bound. Thus, we can extend it to have free variables, that is, it is allowed to have non finite lower and upper bounds at the same time. That will use the notion of super bases: once a free variable enter the basis it will never exit. We need to add that condition into our code, in toGrid() function or gotclicked(). There is an advanced version ([Advanced Pivot Tool](#)) of the [Simplex Pivot Tool](#), in that advanced version there are methods with Phase I (e.g: Primal with x_0), so after we finish to develop the simple one we can move to the advanced.

5 Learning Outcomes and Collaboration

I have learned a lot from this project. I didn't know JavaScript before, I got the overview of JavaScript, Html, Css languages from [Tutorial for JavaScript, Html, Css, etc.](#) Hence, my programming skills using JavaScript, Html, Css languages are improved because of this project. I also reviewed and get more used to bounded simplex method. Doing this project, I use gitHub for my first time. However, I still need to learn about dual simplex method. About the collaboration, we didn't have chance to meet often. Thus, what is covered in this report is what I've done.