

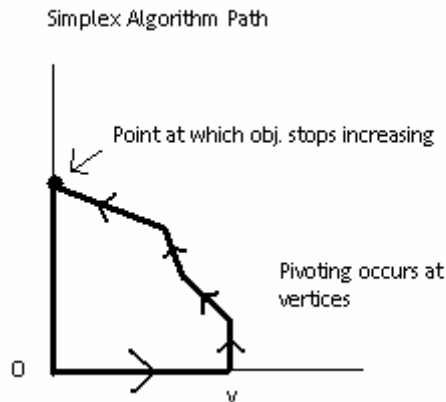
27. Linear Programming Part II (More Theory & Application)

27.1 The Simplex Algorithm

The simplex algorithm can be described geometrically as the process to get an origin in the x variables that obeys all of our constraints.

You start at the origin O , and consider the edges of the feasible region that meet it. You will choose some edge E having vertices at O and at some vertex V . You will also need to make sure that you chose an edge where if you move along it from O to V , that the objective function is increasing. The procedure chooses edges where the objective function increases along it if you move away from the origin.

You then repeat this procedure, starting from that vertex V , until you cannot proceed any further.



Algebraically, you do all this by changing the basis you use in the equation space, from the original form associated with the origin O to the form associated with V . This algebraic step is called a **pivot** and is repeated until the objective function is maximized (you cannot pivot to increase the objective function and maintain your constraints).

In order to describe the algorithm we answer the following questions to enhance your understanding of what is happening in linear programming.

1. What does an edge of the feasible region containing the origin correspond to?
2. How do you find the other end of the edge?
3. What does it mean for the objective function to be greater at the other end of the edge than at the origin?
4. What is the effect of a pivot operation on the equations and objective function?
5. What happens if the objective function decreases or stays the same along every edge from the current origin?
6. What happens if the edge we choose to move along increases the objective function but the edge itself increases infinitely in one direction?

7. How can you get started if the origin is not a vertex of the feasible region and how do you recognize this condition?

We will give qualitative answers to all these questions, consider an example, and then discuss implementation of the algorithm in section 27.5.

27.2 Addressing the Questions

Point 1- The origin is the point where all of the unchosen variables are 0. It is a vertex of our feasible region when all the constraints are obeyed there. The obeyed constraints correspond to the condition that all the b_j are non-negative. If some of the b_j are 0, that means that the origin is a degenerate point for our equations. Since some of the s variables will be 0 there along with all the x variables that will mean that more than n variable = 0 constraints pass through the origin.

We call the **unchosen** variables **x variables** to avoid introducing additional notation. That is what they are at the origin. An edge is what you get if you relax the condition $x_K = 0$ for some x variable allowing the x_k to increase, when there is no degeneracy at the origin.

Point 2- Since we require that all the other x_k remain at 0 on this edge, then the equations that hold on that edge are all of the form $A_{jK}x_K + s_j = b_j$.

As you increase x_K on this edge, you will meet the j constraint when $s_j = 0$, or $x_K = b_j/A_{jK}$. Thus the vertex at the other end of the edge comes from the smallest value of b_j/A_{jK} among all possible j for which this ratio is positive, and if we call the index corresponding to that smallest value J , it is the point with coordinates

$$x_k = 0 \text{ for } k \text{ not } K, \text{ and } x_K = b_J/A_{JK}, \text{ and } s_J = 0.$$

Notice that when A_{jK} is negative, increasing x_K causes s_j to increase, which means you are moving away from the constraint when you increase x_K . Thus we can only hit constraints here for which A_{jK} is positive.

Point 3- The value of the objective function at this vertex will be $v_K b_J/A_{JK}$ more than it was at the origin in the unchosen variables. We can recognize when this is an increase easily because we know that b_j and A_{jK} are positive, so the condition for being allowed to pivot on variable x_K is that v_K must be positive.

Point 4- The pivot operation adds x_K to the list of chosen variables and removing s_J from that list.

This consists of two steps: one is using equation J to eliminate x_K from all the other equations and the objective function. Equation J is the equation you pivoted on.

The other is dividing equation J by A_{JK} so that the coefficient of x_K becomes 1 in it. We can describe the first step most succinctly if we define A_{j0} to be $-b_j$ and A_{0k} to be v_k . Then we remove x_K everywhere but in equation J by, for all other j including 0, replacing A_{jk} for all k other than K by $A_{jk} - A_{jK}A_{JK}/A_{JK}$, and adding new terms $-s_jA_{jK}/A_{JK}$ into the j -th equation for j not J. (the x_K terms will be gone from them.)

Some people like to rename s_j as an additional x_K , and vice versa since at the next step s_j will be an un-chosen variable and x_K a chosen one. We will not do this.

Point 5- *If all the v 's are non-positive, then the current origin is a maximum feasible point for the objective function.* Every other feasible point can be described by positive values for the unchosen variables, and the objective function is obviously no greater at any such point. If any of the coefficients in the objective function are 0 then there will be at least an edge of maximum points.

Point 6- If there is a K such that v_K is positive and every A_{jK} is 0 or negative, then the objective function can increase without limit if you let x_K increase. *The problem then has no maximum, since the objective function has no upper bound.*

Point 7- Suppose the origin is not a feasible vertex, since some of the b_j are negative. *We create a new problem, by adding a new dimension, that is a new variable x_{n+1} .*

We do this in such a way that the origin in all the variables, x_1, \dots, x_{n+1} is feasible, and when x_{n+1} is 1 we are back to our original problem.

We also add a constraint stating $x_{n+1} \leq 1$ and a second objective function consisting of x_{n+1} which is to be maximized.

If we maximize x_{n+1} under these circumstances, one of two things will happen; either we will find a vertex for which x_{n+1} is 1, in which case we can ignore x_{n+1} , and are back to our original problem with a feasible current origin, or we find that x_{n+1} gets stuck at some value less than 1 and cannot be increased beyond it. This means that the original problem had no feasible region whatsoever, since any feasible point in the original problem provides a solution to the new problem with $x_{n+1} = 1$.

How do we do this? We add an equation and a second objective function as indicated, and also add terms $c_j (x_{n+1}-1)$ to equation j for each j for which b_j is negative, with c_j chosen arbitrarily so that $b_j + c_j$ is positive.

Obviously these terms all disappear when x_{n+1} is 1, and the significant effect of adding them is to replace b_j by $b_j + c_j$ in the constant term of our equations, which makes the new origin feasible.

Thus we can force the (new) origin to be feasible by adding one new variable, one new constraint, and one new objective function.

People often find the optimum of the new objective function first, which produces a feasible start to the original problem, and then go on to pivot in it.

You can, however, work on both objective functions simultaneously. One interesting way is to start with objective function $-x_{n+1}$ for which the origin is the optimal solution, and then rotate the objective function as $-x_{n+1}\cos(t) + \sum v_k x_k \sin t$, moving your origin every time the current one ceases to be optimal, until you have rotated 180 degrees, at which point you should be at the solution to your problem.

Thus the origin will not be optimal as soon as t becomes positive, when any positive v_k will allow pivoting on any edge on which the corresponding x_k becomes positive.

In general, a pivot will become possible along an edge when the objective function becomes normal to that edge. This will happen only once during the rotation on any edge.

27.3 Technical Points and Handling Special Cases

There are also some technical questions that arise in special cases we will need to address the following technical points and special cases.

1. What do you do differently if some of your equations are equalities instead of inequalities?
2. What do you do differently if some of your variables need not be positive?
3. What do you do if your current origin is a degenerate vertex, which means that more than n constraints are obeyed at it?
4. How can we set this algorithm up on a spreadsheet?

Point 1- We here consider the effects of having equality constraints, variables that do not need be positive, degeneracy on the simplex algorithm, and also alternative schemes for pivoting when several v_k are positive and you can pivot on multiple variables.

If at the beginning a constraint is an equality, there is little point in introducing a slack variable for it. In fact, you can choose any variable that appears in it, and make it into a slack variable for the equation by dividing by the coefficient of that variable in the equation, and using that equation to eliminate the variable from all the other equations.

The only problem in doing this is that it may cause some b_j to become negative, but that can be handled as described above.

Point 2- When an x variable, say x_k , need not be positive, then the value 0 has no particular significance for it, so that it is unnecessary to look at the origin in it. Thus you can solve any equation x_k appears in for it, and eliminate x_k from the other equations, and then put the indicated equation aside and not pivot on it, because the sign of the b term in the equation has no significance when the slack variable in it can be positive or negative.

Again, eliminating x_k from other equations can cause some b_j to become negative. You should therefore first take care of variables that need not be positive and equality constraints. After dealing with those, you can then handle any b_j that are negative, and then begin to pivot.

Point 3- The current origin will be degenerate when some b_j are 0. If this happens, and A_{jK} is positive in any equation for which this is so, then your pivot will cause you to stay put, since the distance that you move along the pivot edge is b_j/A_{jK} when you pivot.

This opens up the possibility that if you are not careful you can pivot around a circle and never get out of it. This is impossible otherwise, since the objective function otherwise will increase by a finite amount at each pivot.

To avoid this possibility you can change any b_j that are 0 into some very small positive numbers, which will break the degeneracy. This will have the effect of splitting the degenerate vertex into two or more vertices by moving slightly aside the extra constraints that passed through it.

If the optimal solution is elsewhere, you will eventually leave the degenerate vertex, and then you can ignore the change you made to break it up, since you will never return to it.

If you do this, the objective function will always increase by something at each pivot, and you will never pivot around a cycle.

Point 4- Section 27.5 has the implementation of the simplex algorithm on a spreadsheet.

27.4 Suitable pivot variables

There are a number of possible ways that you can choose to pivot when you have a choice of different values of k for which v_k is positive.

One possible choice is the k value which has the greatest objective function slope. This is the variable with largest value of v_k .

Another sensible choice is the k values for which the objective function at the other end of the edge pivoted on is greatest. This means that it is the k such that $v_k b_{J(k)k} / A_{J(k)k}$ is greatest, where $J(k)$ is the index of the constraint equation on the other end of the x_k edge.

There are other reasonable choices as well. You can use the rotating objective function idea described above for use when you have introduced a second objective function to gain feasibility. Or you could pick a variable x_k with positive v_k at random.

The simplex algorithm allows any choice of pivot variable x_K as long as v_K is positive.

27.5 Implementing the Simplex Algorithm

Suppose we are given a linear program (LP) and wish to solve it using the simplex algorithm. There is unfortunately very little in life more unrewarding than performing the additions and multiplications necessary to do a few pivots by hand.

In addition to the tedium of it, our human propensity for occasional careless error means that we will end up doing something wrong 9 out of 10 times, and this wasted effort is simply not worthwhile.

Fortunately, we can perform a pivot on a spreadsheet by entering and judiciously copying only two simple instructions, no matter how big the problem is. We simply need to get the first command to be copied correct!

Moreover, if we are willing to use additional space we can have the spreadsheet indicate which equation is to be pivoted on for each x_K and how much the objective function will increase if we choose that variable to remove from the basis.

We first note how the problem is traditionally set up.

To set it up we create what is called a **tableau**. This is a matrix in which each variable, x_k or s_j and $-b$ defines a column and each equation and the objective function defines a row. The equations that we will be using are the constraints in our linear program (our inequalities).

The entries in this tableau-matrix are the coefficients of the corresponding variable (or the number $-b_j$ in the $-b$ column) in the equation corresponding to the row (or the objective function).

Consider for example the LP defined by the following two inequalities...

$$2x_1 - x_2 + x_3 \leq 2$$

$$x_1 + 2x_2 - x_3 \leq 1$$

We define and add slack variables whose positivity is equivalent to these constraints by...

$$s_1 + 2x_1 - x_2 + x_3 = 2$$

$$s_2 + x_1 + 2x_2 - x_3 = 1$$

with the objective function to be maximized...

$$x_1 + x_2 + 2x_3$$

We represent this problem by the following tableau:

Original Tableau	s_1	s_2	x_1	x_2	x_3	$-b$
Constraint 1	1	0	2	-1	1	-2
Constraint 2	0	1	1	2	-1	-1
Objective Function	0	0	1	1	2	0

Here the last row represents the objective function.

Since all the tableau's components are positive we can pivot on any of x_1 , x_2 , or x_3 .

If we pivot on x_1 then we perform the pivot on the second row and the objective function will increase by 1 after the pivot at the new origin (we will explain how to choose pivot points near the end of this section). If we do so on x_2 the result will be the same. However, if we desire to pivot on x_3 the pivot point will be in the first row and the objective function will go up by 4 after the pivot. Please look at the following tableaus as examples of pivoting.

Pivot x_1 Row 2	s_1	s_2	x_1	x_2	x_3	$-b$
Constraint 1	1	-2	0	-5	3	0
Constraint 2	0	1	1	1	-1	-1
Objective Function	0	-1	0	0	3	1

Pivot x_2 Row 2	s_1	s_2	x_1	x_2	x_3	-b
Constraint 1	1	-2	0	-5	3	0
Constraint 2	0	1	1	1	-1	-1
Objective Function	0	-1	0	0	3	1

Pivot x_3 Row 1	s_1	s_2	x_1	x_2	x_3	-b
Constraint 1	1	0	2	-3	1	-2
Constraint 2	1	1	3	-2	0	-3
Objective Function	-2	0	-3	7	0	4

We therefore choose x_3 and s_1 as the variables whose roles are exchanged in the pivot operation.

It can be described by two steps. The first step is to eliminate x_3 from everywhere except the first row. This involves adding the first row to the second, and subtracting twice it from the third, or objective function row.

In general if the pivot takes place in row Y and column Z you replace the entry A_{yz} in every other row by $A_{yz} - A_{yZ}A_{YZ}/A_{YZ}$.

The second step involves dividing the pivot row by A_{YZ} . Here that entry is 1 and the step is trivial.

On a spreadsheet you can implement these two steps by writing the instruction

$$=A_{11} - A_{1Z}A_{Y1}/A_{YZ}$$

directly below the first column of the tableau (where 11 refers to the name of the square containing the leftmost top element of the tableau and YZ is the name of the pivot square) and copying it to a new tableau with the same rows and columns as the original one.

This will perform the first step of the pivot. The second step involves entering $=A_{Y1}/A_{YZ}$ in the first entry of the new pivot row and copying or filling to the right into the entire row.

In our case the new tableau will look like

Pivot x_3 Row 1	s_1	s_2	x_1	x_2	x_3	-b
Constraint 1	1	0	2	-1	1	-2
Constraint 2	1	1	3	1	0	-3
Objective Function	-2	0	-3	3	0	4

At this point the only remaining pivot possible is on the column x_2 and second row. The objective function increases by 9 to 13 and we add the second row to the first, getting

Pivot x_2 Row 2	s_1	s_2	x_1	x_2	x_3	-b
Constraint 1	2	1	5	0	1	-5
Constraint 2	1	1	3	1	0	-3
Objective Function	-5	-3	-3	-12	0	13

Now the fact that all the variable entries in the objective function row are negative means that we have our solution.

Any variables with negative values in the objective function row are set to zero in order to solve the linear program. s_1 , s_2 , and x_1 are zero because they have negative values in their objective function entry. Solving we get $x_3 = 5$ and $x_2 = 3$. Thus the solution is $s_1 = s_2 = x_1 = 0$, $x_3 = 5$, $x_2 = 3$ and the objective function maximum is 13.

You will notice that on a spreadsheet the complexity of applying the algorithm depends only on the number of pivots necessary to get to a solution. Every pivot requires entering two instructions and copying or filling into whatever rows or columns are necessary.

Notice also that there is symmetry between rows and columns in the instruction for changing the tableau in all rows except the pivot row. Also the condition for a solution is symmetric between the variable entries in the objective function and the entries in the $-b$ column: both must be non-positive in order for us to be at a solution to the problem.

If you want the spreadsheet to tell you which row you have to pivot on you can enter $=\text{if}(A_{11}>0,b_1/A_{11},0)$ to the right of the top row of the tableau and copy it into a shape similar (in other words a same sized matrix) to the tableau to its right. (Here you can address b_1 as $-A_{1\$}(n+m+1)$).

The smallest positive entry in each column tells which row must contain the pivot square if you pivot on the corresponding column.

Here is what the tableau for this problem might look like on the spreadsheet.

linear program						Which pivot should I choose?				
s_1	s_2	x_1	x_2	x_3	-b	s_1	s_2	x_1	x_2	x_3
1	0	2	-1	1	-2	2	0	1	0	2
0	1	1	2	-1	-1	0	1	1	0.5	0
0	0	1	1	2	0	0	0	0	0	0

1	0	2	-1	1	-2
1	1	3	1	0	-3
-2	0	-3	3	0	4

2	0	1	0	2
3	3	1	3	0

2	1	5	0	1	-5
1	1	3	1	0	-3
-5	-3	-12	0	0	13

The formulae in the last two columns of the tableau and the first column of the pivot choosing helper look like

x3	-b	which pivot should I choose?
1	-2	=IF(A3>0,-\$F3/A3,0)
-1	-1	=IF(A4>0,-\$F4/A4,0)
2	0	=IF(A5>0,-\$F5/A5,0)
		=IF(A6>0,-\$F6/A6,0)
=E3/\$E3	=F3/\$E3	=IF(A7>0,-\$F7/A7,0)
=E4-E\$3*\$E4/\$E\$3	=F4-F\$3*\$E4/\$E\$3	=IF(A8>0,-\$F8/A8,0)
=E5-E\$3*\$E5/\$E\$3	=F5-F\$3*\$E5/\$E\$3	
=E7-E\$8*\$D7/\$D\$8	=F7-F\$8*\$D7/\$D\$8	
=E8/\$D8	=F8/\$D8	
=E9-E\$8*\$D9/\$D\$8	=F9-F\$8*\$D9/\$D\$8	

You have probably wondered why we switch over from b to -b in the tableau and in incorporating b into the coefficient (A) matrix.

One reason is that by doing so the constant terms are moved to the same side of the equations as the variables, and can be treated in exactly the same way.

When the constant term is left on the other side of the equations, and there is no analogue of this in the objective function, a strange minus sign creeps into the change in the constant term of the objective function relative to the change of the constant terms of the equations that is thoroughly confusing.

27.6 How well does the Simplex Algorithm perform?

The simplex algorithm involves moving from a feasible origin along edges of the feasible region on which the objective function increases, until you reach the maximum point for that function or find a direction in which it can increase unboundedly.

It does a remarkably good job in practice in solving problems of operations research. It has been applied to solve problems with thousands of variables and hundreds of thousand constraints, with results that are claimed to be excellent.

A single pivot requires at most a looking at x_1, \dots, x_n variables to find a positive v_k , and examining $m * j$ values to find the minimum positive b_j/A_{jk} . Once these are decided on, performing the pivot involves updating the $m + 1$ equations each having $n + 1$ terms in it.

Thus each pivot takes on the order of $m * n$ operations in all.

The key issue in the speed of the algorithm is then the number of pivots that will be needed to solve the given problem.

Here practitioners say that the number of pivots needed is generally on the order of the smaller of n or m . On the other hand, nobody has ever been able to prove that in worst case, the number of pivots needed is bounded by any fixed power of n .

In fact, for most of the obvious pivoting rules I believe that you can find polytopes for that require a number of pivots that is exponential in n and m .

Any feasible polytope defines a **graph**, whose vertices are the vertices of the polytope and whose edges are its edges. We can define the *distance between two vertices as the length (number of edges) of the shortest path between them in this graph*. The **diameter** of the polytope is then the maximum distance among pairs of vertices.

We do not even know whether the maximum diameter of a polytope in n variables defined by $m + n$ constraints (as we have been considering here) is bounded by a constant multiplied by a finite power of these variables.

You could start off with the origin at one end of a high diameter polytope and the optimum at the other, and it will take you a number of pivots at least equal to the diameter to get to the optimum.

Thus proof that the simplex algorithm requires only a number of pivots bounded by a constant times a finite power of $n * m$ would establish a new bound on the maximum diameter of polytopes.

Nevertheless practitioners seem to find the simplex algorithm a thoroughly reliable and efficient tool for solving linear programs.

This situation has led in two directions. First to other approaches to the LP problem and in fact several algorithms have been developed which can be proven to get to the solution in a number of steps bounded by a power of the parameters.

Also attempts have been made recently to devise reasonable alternative criteria for analyzing the efficiency of algorithms. There have been some interesting developments in this direction here at M.I.T. in the last few years.

27.7 Duality

The LP problem has a remarkable symmetry that is very useful in many ways.

If we transpose the coefficient matrix, including $A_{0k} = c_k$ and $A_{j0} = -b_j$, we get the dual LP

Let us return to our original LP example. It had inequalities

$$2x_1 - x_2 + x_3 \leq 2$$

$$x_1 + 2x_2 - x_3 \leq 1$$

and objective function to be maximized:

$$x_1 + x_2 + 2x_3.$$

As a tableau it became

Original Tableau	s_1	s_2	x_1	x_2	x_3	-b
Constraint 1	1	0	2	-1	1	-2
Constraint 2	0	1	1	2	-1	-1
Objective Function	0	0	1	1	2	0

Now suppose we introduce a new variable y_j corresponding to each of our inequalities, and define inequalities corresponding to the variables x_j .

Here we would get $2y_1 + y_2 + 1$ on the left from the x_1 variable. $2y_2 - y_1 + 1$ from the x_2 variable and $y_1 - y_2 + 2$ from the variable x_3 .

Notice that we here include a contribution on a par with that from each of the other rows coming from the objective function.

We now introduce the dual constraints, that these combinations must be positive, that is, greater than 0.

We here get

$$2y_1 + y_2 + 1 \geq 0$$

$$2y_2 - y_1 + 1 \geq 0$$

$$y_1 + y_2 + 2 \geq 0$$

and now we want to maximize $\Sigma (-b_j y_j)$ which means minimizing $\Sigma (b_j y_j)$.

In general, the dual of the LP defined by

$$\Sigma_k A_{jk} x_k \leq b_j, \text{ maximize } \Sigma (c_k x_k),$$

is,

$$\text{Minimize } \Sigma (b_j y_j) \text{ subject to constraints } \Sigma_j y_j A_{jk} \geq -b_k.$$

Notice that the inequalities are reversed, the indices of variables and equations are reversed and maximization has switched to minimization in going from the original problem to its dual.

It is customary to introduce a dual slack variable, t_k in the dual inequalities so that they become equalities.

Then for each index there are two variables: the original variables pair with the dual slack variables; and the slack variables and dual variables pair together.

27.8 Relevance of the Dual Linear Program

The original LP, which is usually called the primal problem, and the dual have an intimate relation with one another based on the following facts:

1. The interrelation is intrinsic to the equations themselves and does not depend on the form of the basis used to describe the equations. Thus, it is preserved by a change in basis such as a pivot in either problem.
2. You can determine the solution to one problem immediately from the solution to the other.

In particular, one of each pair of variables must be 0 at the pair of solutions, which means the basis variables in the dual must include the partners of the non-basis variables in the primal, and vice versa.

Furthermore the $-v_k$ at the solution in the primal are the values of the t_k at the solution of the dual, (Exercise: Figure out the dual statement to this)

3. The value of the objective function in the primal and dual at solutions are always the same. An unbounded primal means an unfeasible dual and vice versa
4. The value of the dual objective function at any feasible point in the dual is greater than the value of the primal objective function at any feasible primary point.

Not only are these statements true and useful, but there are ways to switch from the primal to the dual problem to make things easier.

Exercises

Exercise 1 Here is a linear program, with variables x_1 to x_4 and constraints as follows

$$3x_1 - 2x_2 + x_3 + x_4 \leq 3$$

$$x_1 + 2x_2 - 2x_3 + 2x_4 \leq 3$$

$$3x_1 - x_2 + 3x_3 - x_4 \leq 1$$

Each x_j is non-negative and we want to maximize $x_1 + x_2 + x_3 + x_4$.

Set up a tableau for this program (with 8 columns, one for each s one for each x and one for $-b$) and use the simplex algorithm to perform enough pivots to find the solution. Read off the solution. (The values of the original x variables and of the objective function at the point at which all c 's become negative.)

Exercise 2 Write down the dual linear program to the previous one and deduce the solution to the dual problem from your solution above. (this is the value of the y 's and of the dual objective function at its solution point.)

Exercise 3 Here is another linear program. Our variables are as before except that x_4 need not be positive. The constraints are now

$$3x_1 - 2x_2 + x_3 + x_4 \leq 0$$

$$x_1 + 2x_2 - 2x_3 + x_4 \leq 0$$

$$-3x_1 - x_2 + 3x_3 + x_4 \leq 0$$

$$x_1 + x_2 + x_3 = 1$$

Maximize x_4 . Write down the dual to this LP.

Treat x_1 as a slack variable for the last equality (by using it to eliminate x_1 everywhere else). Also perform a pivot on the first equation and the variable x_4 (ignoring the signs of the b 's). If some of your b 's other than the one for which b_4 is a slack are now negative, add a new variable so that the origin in all 5 variables is feasible.

Perform pivots to find the solution to this, and deduce the solution of the dual.

Exercise 4 State and indicate a proof of the duality theorem of linear programming.

Exercise 5 Explain with an example what you do when the origin is not a feasible point for your constraints to make it one.

Exercise 6 Solve the following linear program on a spreadsheet:

$$\text{Maximize: } x_1 + x_2 + x_3 + x_4$$

Subject to the constraints:

$$3x_1 - 2x_2 + x_3 - x_4 \leq 2$$

$$x_1 + 2x_2 - x_3 + x_4 \leq 7$$

$$x_1 - x_2 - x_3 \leq 1$$

$$x_3 + x_4 \leq 2$$

all x 's are positive. Give the optimal objective function and the values of the x 's that achieve it. State the dual problem and give its solution as well.

- Jonathan Lii