A Presentation of the Simplex Method PhL 11.29.04

[This document was originally called "simplex attempt 4". It is very large in bytes, and it is unstable due to the presence of some Microsoft Equation Editor equations. Therefore, if you do edits on this document, be sure to save frequently because Word could crash at any time.]

1. **The Physical E_n Space, and the m inequality constraints.** We start out with some "physical" problem having some number n of independent variables or physical coordinates (x,y,z....). We can think of each such coordinate set as a point in a vector space E^n. Our problem has a set of m inequality constraints which appear as m Planar boundaries in this space.

An example of a constraint inequality in E^2 would be $5x-3y \le 4$ which is a lower half-plane bounded by the line $5x-3y = 4$.

In E^3 we might have $5x - 3y + 2z \le 8$. The boundary is then the plane $5x - 3y + 2z = 8$, having normal (5,-3,2) and passing within 8 units of the origin. The "half-plane" selected by the inequality is in this case half of 3D space.

In any dimension n, we will refer to the boundary equalities as **Planes** with capital P (n=2 line, n=3 plane, n> 3 hyperplane). Similarly, we will refer to the spaces into which these Planes divide E^n as **half-spaces**. For n=2, a half-space happens to be a half-plane because the entire space is a plane.

2. The n coordinate boundaries. In addition to the above m constraints, we have n more constraints of the form $x \ge 0$, $y \ge 0$ etc for the n physical coordinates. This is just another set of n boundary Planes, so we have a total of n+m bounding Planes.

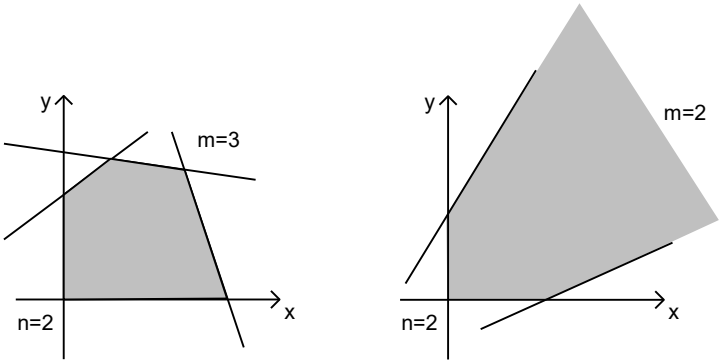
3. The Feasible Region. The space that is the intersection of the n+m "good side" half-spaces of these Planes is called **the feasible region** for the problem, because any point inside it is a "feasible" solution to our problem. Feasible just means that all n+m constraints are satisfied. (The problem will be stated soon.)

- For the moment, we disallow constraints which are equalities, we only allow inequalities. [See comment at the very end on this subject.]

- In theory we could have a boundary Plane which lies far away from the convex feasible region volume formed by the other Planes. If the entire feasible region is on the good side of such a Plane, then we can ignore the Plane. It comes from an inequality condition which we can ignore. The other conditions "cover" this condition. If we are on the bad side, then the feasible region is null. So from here on, we assume each of our n+m Planes forms a facet of the feasible region n-gon.

3. **The feasible region is convex**. Convex means that if you pick two points inside the region and join them by a line segment, that entire segment is contained in the region. It is probably easy to prove that a region which is the intersection of a set of half-spaces defined by Planes is convex for any dimension n.

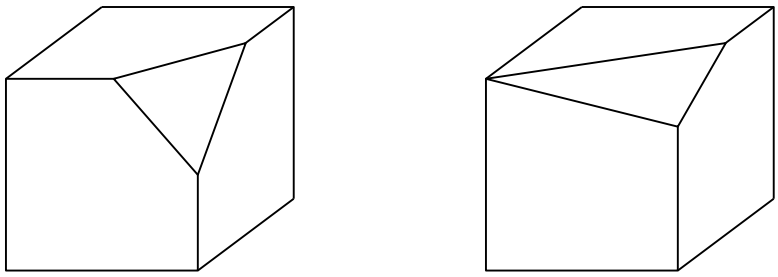
In 2D (n=2) the convex region is a convex polygon. It is a polygon because the edges are flat, and they are flat because the boundary Planes are lines. Each **vertex** of the polygon is a place where n=2 of the 2+m boundary Planes intersect. If we number the Planes p = 1,2,3...2+m, then we can label the vertices by pairs of these numbers like [1,2], [2,3]... [2+m,1]. The pair of numbers tells which two Planes intersect at that vertex. Counting the pairs, we see that there are 2+m vertices. Below on the left is an example with m=3, and we see that there are indeed 2+3 = 5 vertices,



On the right there are m=2 boundaries (other than the axes), and the feasible region is unbounded. In this case we expect there to be 2+2 = 4 vertices, and we regard the 4th vertex as being at infinity. Notice that the region is convex in both cases.

In the 2D case, we know there are $\binom{2+m}{2}$ unique pairs of Plane intersections, but many of these intersections are not vertices of the feasible region. In the above example on the left, there are $\binom{5}{2} = 10$ Plane intersections, but only 5 of them are vertices of our feasible region. We were able to count the vertices because they can be put into a linear order as shown above.

In 3D we can number the Planes p = 1,2,3...3+m. A vertex is a place where n=3 Planes intersect $[p_1, p_2, p_3]$. There are $\binom{m+3}{3}$ ways to pick a group of 3 Planes from m+3 Planes, but as in the 2D case not all these intersections will be vertices of the feasible region, most in fact will not be. A slightly distorted cube (non parallel sides) has m=3 so there will be $\binom{6}{3} = 20$ triple-Plane intersections, but only 8 of these form the vertices of the cube. One might wonder if there is a formula relating the number of vertices to n and m. In the 2D case it was 2+m as shown above. In the 3D case and for n>3, there is no such formula. Here is a counterexample with n=3. Consider the following cube on the left with a sanded off corner,



In E^3 we have n = 3, and there are m=4 constraint planes, so there n+m = 7 planes (think of the invisible back cube corner as the origin). The cube had 8 vertices, but shaving off the corner converted 1 vertex into 3, so the above left object has 8-1+3 = 10 vertices. If we sand differently, however, we can get the picture on the right which still has m=4 constraint planes, but now there are only 9 vertices instead of 10. By further distortions of the sanding plane, we could lower this to 8 or 7 vertices.

Presumably a hypercube (or a slightly distorted one so we avoid parallel sides) is always a possible shape in E^n for a feasible region (for some problem). It has 2n faces, meaning the intersection of 2n

Planes, and since we have been writing this as n+m, we get m = n. This hypercube has 2^n vertices, so a problem with n=10 variables (x,y,z...) and m=10 constraints could then have a feasible region with 2^{10} = 1,024 vertices. If we have to investigate each vertex for some reason, this might take a lot of work.

- When we start out doing problems, we will assume that the origin of the E^n space is in fact one of the vertices of our feasible region volume, so that the point (x,y,...) = (0,0...) satisfies all the constraints.

4. **Moving between adjacent vertices**. In the general E^n case a vertex is the intersection of n Planes. A vertex is a *point* in E^n. The total number of constraint Planes is n+m. If we number them 1 through n+m, then we can describe each vertex by an Planes n-tuplet like [3,4,7.....] where of course all the little Plane numbers must be different. If we are sitting at a particular vertex, there are n other vertices we can reach by sliding along an "edge" of the feasible region to that neighboring vertex. The n-tuplet of each of these neighboring vertices differs by a change in exactly one coordinate in the n-tuplet.

For example, in E^3 we might have a vertex [1,2,3]. If we move to one of the 3 neighboring vertices, we might find ourselves at [1,4,3] because, while we still lie on planes 1 and 3, we have moved from plane 2 to plane 4. We think of this motion as a plane swap $2 \leftrightarrow 4$. By doing a carefully planned sequence of swaps like this, we can travel through all the vertices of the feasible volume.

This last statement is generally true in E^n. All vertices are accessible from a given starting vertex by doing some sequence of Plane swaps. Remember that by the word vertex we mean a vertex that is on the feasible region, not some intersection of Planes that is out in the boonies somewhere.

5. Our Problem. So far we have n coordinates (x,y,z...) that form E^n , and we have n+m constraint Planes which bound our feasible region. Now we come along with some function M of the n coordinates (known as **the objective**) that we want to maximize or minimize. Perhaps M = 3x-2y+5z + 6. The final constant 6 makes no difference, but we include it just to be general. We can worry only about maximizing things if we want, since to minimize M we could maximize M' = - M. So our problem is to find the point or points in the feasible region which **maximize** the objective M. It is as simple as that.

6. A Fancy Theorem says that there is (at least) one vertex of the feasible region where M will take its maximum value over the entire feasible region. The maximum might occur at several vertices and on a surface containing them.

In 2D, the truth of this theorem is easy to see because M is a line, and as you drag a line parallel to itself to "raise M", the last point of contact as you leave a convex polygon will be a vertex. If the line happens to be parallel to an edge connected to that vertex, then the maximum will occur at two vertices and along the line segment connecting them.

In 3D, M is a plane, and we reach a similar conclusion. There could be 2 degenerate solution vertices, in which case the line segment joining them is also a solution. If the departing M plane happens to align with a polygon facet of the feasible region, and if that polygon has k vertices, then all k vertices and the polygon they border are solution points, that is, are points where M is a maximum subject to the n+m constraints which define the feasible region.

7. Solving the Problem. A simple way to solve our problem is to compute the coordinates of all the vertices, then evaluate M at each vertex and then pick the maximum value obtained. This is essentially what the simplex method does. It does this by starting at some vertex where it evaluates M. It then selects a Plane swap to move to a neighboring vertex such that M will increase. It knows the "direction" in which

M increases and can make an intelligent choice about which neighboring vertex to move to by this swap. It then thinks again at the next vertex in the same way. It finally reaches a vertex where it knows that all directions from that vertex will make M *decrease*. At this point, the problem is solved. One reason this works is that the feasible region is convex. For example, if you are sitting on the top vertex of a tilted cube, and all directions are down from that vertex, you know you are at the highest vertex. You cannot have a local maximum that is not a global maximum because the cube is convex.

During its motion through vertices, the simplex method does not have to pass through all the vertices of the feasible region. Again, think of climbing from the bottom of a tilted cube to the top. Although the cube has 8 vertices, you could get from bottom to top by visiting only a total of 4 vertices. It is possible that the number of vertices visited by the simplex method will depend on how decisions are made along the way, but a finite number of visits (≤ the total number of vertices) will get the job done. If you are traveling up (direction of increasing M), you will never revisit a vertex because such a vertex will be "below" where you are.

The simplex method has to have a way to "get to" a starting point vertex from which it can then start climbing to increasing M by doing Plane swaps to move to adjacent vertices.

8. Slack Variables.

(A) Consider the inequality $x - 2y \leq 7$. We can introduce a variable $u \geq 0$ to "take up the slack" so this becomes an equality: $x - 2y + u = 7$. Notice that when $x = y = 0$, we have $u = 7$ which is > 0 as required.

(B) What about $ax + by \leq -7$, where a and b have arbitrary signs. We could try $ax + by + u = -7$, but now when $x=y=0$, we have $u = -7$ which is not ≥ 0.

(C) What about $-x - 2y \leq 7$. Since $x \geq 0$ and $y \geq 0$, we know that $-x - 2y \leq 0$, so we would have a throw-away constraint here.

(D) What about $ax + by \geq 7$. We can replace this with $-ax - by \leq -7$, but then we have the same problem as in case (B) above.

For the moment, we will assume our constraints are all of the form $x - 2y \leq 7$, case (A), so we can introduce our $u \geq 0$ slack variable to get $x - 2y + u = 7$. At least one of the signs on the left hand side must be positive, the inequality direction must be ≤, and the number on the right must be positive. Once we learn how to handle constraints of this type, we can look at case (D) which is really the same as case (B). We have already thrown out case (C). In case (D) we will have to introduce a slack variable AND a so-called artificial variable.

9. Interpreting the slack variables. Assuming all m constraints are type (A) above, we introduce m slack variables (u,w,v....) as described above. Like the n physical variables (x,y,z...) the m slack variables are ≥ 0. In this way, we convert all our inequality conditions to equalities. Since we are allowed to scale an equation by any constant, we can imagine that our new equalities are scaled (before the slack variable is introduced) such that the sum of the squares of the coefficients is 1. In this case, it can be shown that the slack variable measures the distance of a physical point (x,y,z...) from the Plane defined by the equation into which that slack variable has been injected [See Note 2 at end for n=2.] . So now each of our m Planes has its own slack variable that tells us how far we are from that Plane in E^n. Notice that the physical variables (x,y...) also measure distance from their respective planes. For example, x is the distance of a point from the x=0 Plane. Thus, in some sense, all our n+m variables (x,y,z.... u,v,w....) are on an *equal footing*. This is a key idea.

10. Selecting a Set of Independent Variables in E^n. Barring the case of parallel Planes, we can choose to make any n of our n+m coordinates be the "independent" coordinates to locate a point in the feasible region in E^n. This set of n coordinates forms a basis in E^n. When we start, we use the physical coordinates (x,y,z...), but we could then perhaps swap y for u and use (x,u,z...) as the independent variable set. Any subset of n coordinates will do. Then the remaining m coordinates, whatever they are, are the dependent coordinates. It turns out that we won't really use any basis in E^n, but it seemed good to just take note that one can make a basis in E^n. What we really care about is the basis in E^m, see below.

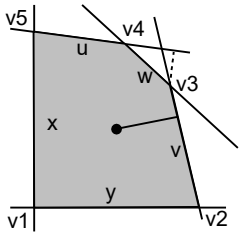
11. Selecting a Set of Independent Variables in E^m. We have said that we have a set of m slack variables (u,v,w..) For each point (x,y,z..) in the feasible region, we can figure out the values of all the slack coordinates (u,v,w...) from the equations in which we injected those slack variables. We can think of (u,v,w...) as a point in a *new space* E^m, which is (at the moment) the slack space. We than have a little mapping that goes $E^n \rightarrow E^m$. Either E^n or E^m could be the "larger space", since n and m are arbitrary positive integers. We do know, however, that each point in E^n will map into exactly one point in E^m. This is because of those equations just mentioned, which we can write as $\mathbf{u} = T\mathbf{x} + \mathbf{c}$ where T is a matrix of numbers, **u** is (u,v,w...) and **x** is (x,y,...). Notice that T is not square, it has n columns and m rows. We shall more to say soon about this equation.

First we want to talk about a **basis** in E^m, that is, a selection of independent vectors. An obvious choice is to pick the "axes" associated with the coordinates (u,v,w...) as the basis in E^m. This E^m is an abstract space so we can choose the basis vectors in E^m like this (in the case m=4)

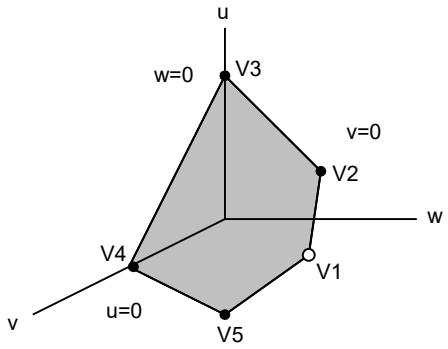
$$u = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad v = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad w = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad t = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

The variables we choose to be represented by unit vectors such as the above are called the **basic variables**, because we associate them with a simple basis in E^m . When we start a Simplex problem, all the basic variables are the slack variables, so the "physical" variables of the problem that we started with have to be called non-basic variables, although we might think them as being basic to our problem, so this terminology is a little confusing. The above unit vectors will eventually appear as columns in a Simplex table.

12. The mapping applied to the feasible region. Finally, we mention a fact which is interesting but perhaps not useful. Using the mapping $E^n \rightarrow E^m$ provided by our equation $\mathbf{u} = T\mathbf{x} + \mathbf{c}$ we can ask what the image in E^m is of our feasible region in E^n . Because the mapping is linear, we know that lines map into lines, planes into planes, and so on (Planes into Planes). Also, convexity is preserved. However, angles are not preserved, nor are the lengths of line segments. So the feasible region gets distorted in some manner. [See Note 3 at end.] The feasible region boundary is an n-dimensional surface in E^n and it maps into an n-dimensional surface in E^m if $m \geq n$, otherwise it is projected into something of m dimensions. As an example, a polygon in E^n for n= 2 maps into a distorted and tilted polygon in E^m for m=3:



E^2



E^3

The mapped polygon is contained entirely in the first "octant" of E^3 since all the slack variables (u,v,w) must be positive. All we can say in general is that the mapping of the feasible region in E^m is a distorted image of the original in E^n and lies in the first "octant" of E^n. All but one of the vertices of this mapped feasible region will lie on the axis walls of the E^m space (black dots above) because these walls *are* the mapping of the Planes in E^n where u=0, v=0, w=0 etc. In the original E^n space, all vertices except the one assumed at (x,y,z..) = (0,0..0) are located on at least one Plane whose slack variable is zero, so that vertex ends up on at least one wall of E^m. The one vertex in E^m which is the mapping of (x,y..) = 0 is floating out somewhere in the first octant of E^m (white dot).

13. The mapping equation. We have written this as **u** = T**x** + **c**. Here is an example with n=2 and m=3:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} -au & -bu \\ -av & -bv \\ -aw & -bw \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} cu \\ cv \\ cw \end{bmatrix} \qquad \mathbf{u} = T\mathbf{x} + \mathbf{c}$$

This is nothing more than a statement of the m constraints after the addition of the m slack variables. Suppose one of our constraints was au*x + bu*y ≤ cu (with constants au, bu and cu subject to the requirements of type (A) mentioned earlier.) We then insert the slack variable u to get au*x + bu*y + u = cu. We then solve for u to get u = - au*x - bu*y + cu. That is the first of the three equations shown above. The others derive in the same way from the other constraint inequalities turned into equalities by the addition of the slack variables. So this shows the linear mapping from $\mathbf{x} \in E^n$ to $\mathbf{u} \in E^m$.

14. Feasible region in E^{n+m}. Here is another way to write the above matrix equation:

$$\begin{bmatrix} au & bu & 1 & 0 & 0 \\ av & bv & 0 & 1 & 0 \\ aw & bw & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ u \\ v \\ w \end{bmatrix} = \begin{bmatrix} cu \\ cv \\ cw \end{bmatrix} \qquad S\,\mathbf{v} = \mathbf{c}$$

The vector **v** = (x,y,u,v,w) is in the space E^{n+m}. The m constraints written in this form can be interpreted as saying that all vectors **v** in E^{n+m} are allowed as long as they have projections equal to **c** under the

application of the matrix S. The set of points **v** that have projection **c** form an n dimensional volume inside E^{n+m}. So this projection, combined with the fact that all components of the vector must be non-negative, defines the feasible region in the space E^{n+m}.

Another view of the above equation is that x and y should be thought of as temporarily fixed parameters, and the only real variables are u,v,w. In this view, we think of S**v** = **c** as a mapping from E^m into E^m, which here is $E^3 \rightarrow E^3$. The columns of S can be regarded as vectors in E^3. There can only be m=3 linearly independent basis vectors in E^3 and we can regard the last three columns shown above as being a simple choice. [Compare to section 11 above.] They form an orthogonal basis in E^3. We might call this the (u,v,w) basis in E^3. Notice that if we select the parameters x=y=0, we find that u = cu, v = cv and w = vw. Thus, when we set all the "parameters" to 0, the right column **c** tells us the corresponding point (u,v,w) in E^3. In other words, the origin of the "parameter space" E^n maps to location **c** in the E^m space.

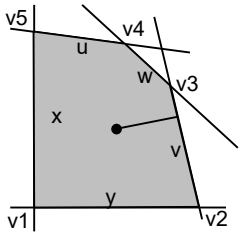
As a reminder, the reason the above matrix S contains an embedded 3x3 identity matrix is that the slack variables u,w,v were inserted into the constraint equations *one variable per equation*. That is to say,

```
au*x  +   bu*y   +u              = cu
av*x  +   bv*y        +v         = cv                    (14)
aw*x  +   bw*y              +w   = cw
```

As noted earlier, the variables (u,v,w...) in E^m are called the **basic variables** since they are associated with this basis of unit vectors, while the "parameter variables" (x,y...) in E^n care called the **non-basic variables**.

15. Doing a variable swap. We already discussed the idea of touring the vertices of the feasible region in E^n by swapping one parameter at a time in our Planes representation of a vertex. We represented such a vertex in E^n as a Planes n-tuple [p_1, p_2, ... p_n] where the arguments are just a permutation of the set of integers (1,2,3....n+m). This n-tuple tells which Planes are intersecting at this vertex. We noted that not all such n-tuples are actually vertices on the feasible volume, many of them are intersection points outside this volume. Instead of labeling the Planes by n+m integers, we might as well label them by our n+m variables in E^{n+m}. If plane 4 is really the Plane u=0, why not put u in the n-tuple instead of 4. In our example n=2 and m=3, our feasible region in E^2 might have its vertices labeled [x,y] or [x,u] or [v,w]. Notice the distinction between (x,y) meaning the coordinates of an arbitrary point in E^2 (perhaps inside the feasible region), and the meaning [x,y] which refers to that particular vertex of the feasible region which lies on the Planes x=0 and y=0. We could say that, at vertex [x,y], we have (x,y) = (0,0).

Suppose the vertex [x,y] is a vertex of the physical region (if not, see later). We can move to another vertex by doing a single variable swap. In such a swap, we trade ONE of the variables at this vertex with ONE variable that is not at the vertex. Only certain swaps take us to other vertices. Consider this example in E^2



where we have labeled the five Planes by the letters x,y,u,v,w. [The Plane (=line here) w means w=0.] We start at vertex v1 which is [x,y]. If we swap y ↔ u, we find ourselves at [x,u] which is vertex v5, so that is a good swap to do. If we had instead swapped y ↔ w, we would end up at [x,w] which is the intersection of the x and w Planes which you can see happens on the u axis above vertex v5, so this would be a bad swap. It is clear that by doing the *right* sequence of swaps, we can tour all the vertices of this feasible region. For example, if we are at v5 = [x,u], we could swap x ↔ w and get to v4 = [w,u].

We can associate these geometric swaps with changes in our basis in E^m . Let's reconsider the matrix equation we wrote earlier,

$$\begin{bmatrix} au & bu & 1 & 0 & 0 \\ av & bv & 0 & 1 & 0 \\ aw & bw & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ u \\ v \\ w \end{bmatrix} = \begin{bmatrix} cu \\ cv \\ cw \end{bmatrix} \qquad \text{S } \mathbf{v} = \mathbf{c} \qquad (15a)$$

and we assume the feasible region drawn above. If we start at vertex v1 = [x,y], then we have x=0 and y=0, and the above equation tells us that (u,v,w) =(cu,cv,cw). We associate this vertex [x,y] with the E^m basis (u,v,w), and those are the columns in matrix S which have the unit vectors. At this point, the basic variables are u,v,w and the non-basic variables are x,y. { Reminder: the u,v,w columns in the matrix S above are there because those variables appear *in isolation* in equations (14). }

Now what happens to matrix S when we move to vertex v5 = [x,u] by doing the swap y ↔ u. After such a swap, we will have x,u as the non-basic variables and y,v,w as the new basic variables. We *expect* our matrix equation to perhaps have the following new form:

$$\begin{bmatrix} * & 1 & * & 0 & 0 \\ * & 0 & * & 1 & 0 \\ * & 0 & * & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ u \\ v \\ w \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \qquad \text{S' } \mathbf{v} = \mathbf{c'} \qquad (15b)$$

where the *'s indicate new matrix element values of some sort. Our swap needs to move the unit vector (1,0,0) from the u column to the y column because we did y ↔ u. Now if we go to vertex v5 = [x,u], we have x=0 and u=0, and the above equation says (y,v,w) = (p,q,r) = **c'**, so we interpret the vector **c'** as the point in our new E^3 space (ie, new basis) to which the vertex v5 on the feasible region in E^2 is mapped in E^3. After the y ↔ u swap, (y,v,w) are the new basic variables.

All we have really said so far is that we expect a variable swap which moves us from one vertex to another vertex on the feasible region to be associated with a change in the form of matrix S in which one of the unit vectors moves to another column. We expect the other unit vectors to stay where they were (they were not part of the swap), and we expect that perhaps all other values in the matrix S will change.

The above "expectation" can be made more concrete by an example. We have seen how matrix equation (15a) above is a shorthand for equations (14) which we rewrite here,

```
au*x  +   bu*y   +u              = cu
av*x  +   bv*y        +v        = cv                    (15c)
aw*x  +   bw*y              +w   = cw
```

Suppose we solve the first equation for y to get something of the form $y = \alpha x + \beta u + \gamma$. We then substitute this in for y in the second pair of equations. Our three equations then have this form:

$$
\begin{aligned}
& y = \alpha x + \beta u + \gamma \\
& av^{*}x + bv^{*}(\alpha x + \beta u + \gamma) + v = cv \\
& aw^{*}x + bw^{*}(\alpha x + \beta u + \gamma) + w = cw
\end{aligned}
$$

We then rewrite the last two equations putting v and w on the left side and invent some new constants to get:

$$
\begin{aligned}
& y = \alpha x + \beta u + \gamma \\
& v = \alpha' x + \beta' u + \gamma' \\
& w = \alpha'' x + \beta'' u + \gamma''
\end{aligned}
$$

Notice that only (x,u) appear on the right side. If we put this equation into matrix form, we get

$$
\begin{bmatrix} -\alpha & 1 & -\beta & 0 & 0 \\ -\alpha' & 0 & -\beta' & 1 & 0 \\ -\alpha'' & 0 & -\beta'' & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \gamma \\ \gamma' \\ \gamma'' \end{bmatrix}
$$

and this is indeed in our expected form (15b) above. The point here is an important one: If you somehow move a unit vector from one column to another (while keeping the equations still true) it is *as if* you solved one of the equations for a variable and then replaced that variable in all the other equations. The new unit vectors will be for the variables that "stand in isolation" in their respective equations.

OK now we want to turn our animal around and start from the other direction. Suppose we start with matrix S associated with vertex [x,y] , which we repeat again,

$$
\begin{bmatrix} au & bu & 1 & 0 & 0 \\ av & bv & 0 & 1 & 0 \\ aw & bw & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ u \\ v \\ w \end{bmatrix} = \begin{bmatrix} cu \\ cv \\ cw \end{bmatrix} \qquad \mathrm{S}\,\mathbf{v} = \mathbf{c}
$$

What can we do to cause this to appear in the form above S'**v** = **c'** without breaking the truth of the equations? This matrix represents 3 equations. We are always allowed to scale an equation by a constant factor (ie, we can always multiply all numbers in a row, in both S and c, by a constant.). And of course we can add a multiple of one equation to another equation and end up with an equation that is still true. These two rules are sometimes called **elementary row operations**. Using these operations, it is easy to convert the above form to our desired form. The first step is to divide the first row by bu. (If bu = 0, then

the first constraint was of the form ax ≤ c which indicates that the u plane is parallel to the x plane which is a singular situation. Presumably we get around this with some sort of ε limit procedure.) This gives us

$$\begin{bmatrix} * & 1 & * & 0 & 0 \\ av & bv & 0 & 1 & 0 \\ aw & bw & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \alpha \\ cv \\ cw \end{bmatrix} \qquad \text{S } \mathbf{v} = \mathbf{c}$$

We then replace row2 by row2 - bv * row1 to clear out the bv position, and the same idea for all remaining rows, which then gives us

$$\begin{bmatrix} * & 1 & * & 0 & 0 \\ * & 0 & * & 1 & 0 \\ * & 0 & * & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} \qquad \text{S' } \mathbf{v} = \mathbf{c'}$$

which is our desired form. Since we know that this form represents a feasible region vertex, we know that the elementary operations will be successful in arriving at the above form.

16. Gauss-Jordan Elimination. This process of applying elementary row operations to move a unit vector from one column to another is associated with the term Gauss-Jordan elimination, and we digress momentarily on this subject. If we start with a general matrix with Ncols ≥ Nrows, like this one,

$$\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}$$

we use **Gauss elimination** (not always possible) to get the matrix into this form

$$\begin{bmatrix} 1 & * & * & * & * \\ 0 & 1 & * & * & * \\ 0 & 0 & 1 & * & * \end{bmatrix}$$

This is done by scaling the first row so the left end is a 1, then using that to clear out the rest of the first column. Then we scale the second row so it ends with a 1, and use that to clear out the rest of the second column, and so on. The matrix is then in "row echelon form" having a lower left triangle of zeros. This is all done with those elementary row operations. If the matrix is square and we have A**x** = **b**, then A really is in upper triangular form and one can solve A**x**=**b** for the components of **x** one at a time by working upwards from the last row, a process known as **back-substitution**.

In **Gauss-Jordan elimination**, you don't stop at the above form, but keep going to clear out the triangle above the diagonal of ones. For example, adding a multiple of the second row to the first can clear out the * above the middle 1. And doing this with the last row then clears out the two *'s above the third 1. When Gauss Jordan elimination is done, we have this form for the matrix:

$$\begin{bmatrix} 1 & 0 & 0 & * & * \\ 0 & 1 & 0 & * & * \\ 0 & 0 & 1 & * & * \end{bmatrix}$$

Our vertex/variable swap operation on our matrix S can be thought of as doing Gauss-Jordan on a selected set of columns.

For a square matrix A in A**x** = **b**, doing Gauss-Jordan gets us into the form I**x** = **b'** which is just **x** = **b',** so we have *solved* the matrix equation A**x** = **b** for x. In other words, Gauss-Jordan implements the back-substitution process within the matrix that one normally does after the simpler Gauss elimination.

Continuing on just a bit more here, consider again the equation A**x**= **b** which can be written A**x** - I**b** = **0**. For a 3x3 matrix A, we then have:

$$\begin{bmatrix} a11 & a12 & a13 & 1 & 0 & 0 \\ a21 & a22 & a23 & 0 & 1 & 0 \\ a31 & a32 & a33 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ -bx \\ -by \\ -bz \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

If we can successfully apply Gauss-Jordan to this system of 3 equations by doing our elementary row operations, we can change the above to

$$\begin{bmatrix} 1 & 0 & 0 & d11 & d12 & d13 \\ 0 & 1 & 0 & d21 & d22 & d23 \\ 0 & 0 & 1 & d31 & d32 & d33 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ -bx \\ -by \\ -bz \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

which says that Ix - Db = 0 or x = Db. Thus, matrix D must be the inverse of matrix A, and this is a standard and reasonably fast way to invert an invertible matrix.

17. Adding the Objective to the problem. At this point, we have the notion of wandering around from vertex to vertex of our feasible region in E^3 by doing variable swaps each of which causes some unit vector to move from a basic-variable column to a non-basic-variable column in our matrix S, causing that column to become a new basic variable column. We can actually implement the swap by doing the

appropriate Gauss-Jordan operation on our matrix S. Our underlying m equations are transformed into a new equivalent (ie, still true) set of equations. Only one step yet remains.

We are trying to maximize some quantity M = Ax + By + ... k which is a function of our starting point physical variables (x,y ...). We can write this as follows:

$$-Ax - By + M = k$$

and then we can just incorporate this equation with our other equations in a matrix equation that has one more row and one more column (the last column being the M column)

$$\begin{bmatrix} au & bu & 1 & 0 & 0 & 0 \\ av & bv & 0 & 1 & 0 & 0 \\ aw & bw & 0 & 0 & 1 & 0 \\ -A & -B & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ u \\ v \\ w \\ M \end{bmatrix} = \begin{bmatrix} cu \\ cv \\ cw \\ k \end{bmatrix}$$

This is a crucial step in understanding the Simplex method. In the last equation, M = Ax + By + k is the VALUE of the quantity we are trying to maximize. At the [x,y] vertex and we have x=0 and y=0 so

$$u = cu \qquad v = cv \qquad w = cw \qquad M = k$$

We are now working in a space $E^{n+m+1} = E^{2+3+1} = E^6$ where M is the new variable. The unit vectors in the matrix now have m+1 components. We can now do Gauss-Jordan to move a unit vector from one column to another to implement a variable swap (but we will never move the M column). Assume as before we want to do the swap $y \leftrightarrow u$. We get,

$$\begin{bmatrix} * & 1 & * & 0 & 0 & 0 \\ * & 0 & * & 1 & 0 & 0 \\ * & 0 & * & 0 & 1 & 0 \\ -r & 0 & -s & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ u \\ v \\ w \\ M \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ k' \end{bmatrix}$$

The last equation now says that M = rx + su + k. We are now expressing M in terms of the new non-basic variables (x,u) instead of the original ones (x,y). Now at the [x,u] vertex, we have x=0 and u=0, so those columns in the matrix do nothing to our equations, and we find that

$$y = \alpha \qquad v = \beta \qquad w = \gamma \qquad M = k'$$

If k' > k, then the vertex [x,u] is a *better solution* than the vertex [x,y] because M is larger at this second vertex and our problem is to maximize M. This is a key point.

If we somehow knew a proper sequence of variable swaps that would take us from our starting vertex to the vertex with the largest M, we would have our problem solved. We could just keep doing repeated iterations (so-called pivots) of the Gauss-Jordan process to implement that sequence of swaps (one at a time) and watch our M=k quantity get larger with each swap. We are crawling along the exoskel of our feasible region, always moving in the direction of increasing M.

But how exactly do we do this? In particular, we don't ever want to accidentally do a swap which takes us from a vertex to an intersection of Planes point which lies outside the feasible region. We don't want to fall off the wagon as we do our climb.

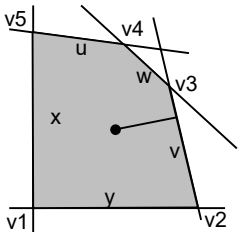
By the way, we will soon see that the matrix above together with the right hand column vector form the Simplex Tableau, just a table of numbers, which we will call a table and not a tableau.

18. Choosing the right sequence of vertices. Let us start at our usual starting position:

$$\begin{bmatrix} au & bu & 1 & 0 & 0 & 0 \\ av & bv & 0 & 1 & 0 & 0 \\ aw & bw & 0 & 0 & 1 & 0 \\ -A & -B & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ u \\ v \\ w \\ M \end{bmatrix} = \begin{bmatrix} cu \\ cv \\ cw \\ k \end{bmatrix}$$

We know that M = Ax + By ... + k and we are trying to maximize M. Our starting position is at the vertex [x,y] where x=0 and y=0, so at that vertex we have M = k. Since all these non-basic variables (x,y...) must be ≥ 0, increasing *any* of them will help our cause, provided its coefficient is positive in the expression M. It seems reasonable to choose to increase the non-basic variable which has the largest positive coefficient in the M equation. Suppose that coefficient is B. Then we will be wanting to increase y to y>0 , which means we want to swap our variable y with one of the basic variables (u,v,w....). Which one should we swap it with as we look for the vertex we are going to reach by increasing y? As we increase y, holding x=0, we are going to hit one of the Planes u,v,w... before any of the others. It is the plane that we hit first that determines our desired swap partner for y.

Let's look again at our simple E^2 example, where the label x indicates the Plane x=0, which happens to be the y axis:



If we start at vertex v1 = [x,y] in the lower left corner and we decide to increase y, we hit the u Plane first. So that is the partner to swap with y in this case, and we will then be moving to vertex [x,u] which is v5. By selecting the Plane we hit *first*, we avoid the problem of hitting a "vertex" which is outside the feasible region.

Now how do we identify this "first Plane hit" in terms of our matrix equation, which we repeat here

$$\begin{bmatrix} au & bu & 1 & 0 & 0 & 0 \\ av & bv & 0 & 1 & 0 & 0 \\ aw & bw & 0 & 0 & 1 & 0 \\ -A & -B & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ u \\ v \\ w \\ M \end{bmatrix} = \begin{bmatrix} cu \\ cv \\ cw \\ k \end{bmatrix}$$

Recall from an earlier discussion that cu, cv and cw are all ≥ 0 (our type A constraints). Consider the first equation which says u = -au*x - bu*y + cu. If we start at vertex [x,y] which is at (x,y) = (0,0), and we move in the direction of increasing y, this equation says u = -bu*y + cu. We will "hit the u Plane" u=0 when y has reached the value y = cu/bu, provided bu is positive. If bu is 0, changing y does not alter u at all, and if bu ≤ 0, then increasing y also increases u, so we will never hit the u=0 plane.

So assuming bu > 0, we compute y_1 = cu/bu, and we are safe relative to this plane if y ≤ y_1. We then repeat this with the next m-3 = 2 rows to get y_2 = cv/bv, y_3 = cw/bw. If one of these ratios is ∞ or negative, we ignore it, since these cases correspond to b=0 or b<0.

The Plane we hit first by increasing y is the row (and its associated variable like u) which gives the *smallest* y_i from the set of y_i that we have computed. This is the Big Idea in the simplex method. Suppose in fact that y_1 = cu/bu is the smallest ratio. If we set y = y_1, we will have u = 0, and so we know that the vertex we are arriving at is [x,u] where x=0 and u=0. Thus, our desired swap is y ↔ u.

We could alternatively compute *all* the ratios and then select the smallest *positive* y_i.

Let's now recap what we just said. First, we look in the bottom row of the matrix for the largest negative number, and we have assumed that number is -B, and this tells us *which* of the non-basic variables we want to swap (y in this case). We want to increase the variable y to some number y>0 because this will increase M. We then look at the column entries above the -B and compute the three ratios like bu/cu. We select the smallest positive ratio (call it y_1) and the row where this occurs corresponds to the basic variable we want to swap with our selected non-basic variable. In our example, we pick the first row because we assume it has the smallest positive ratio. This row is associated with variable u because if you write out the equation for this row, it contains the slack variable u.

So we have picked a column and then a row by our method. The matrix element at that location is known as **the pivot**. If we do our partial Gauss-Jordan based on that pivot (which means the pivot becomes a 1 and the rest of the column is cleared out), we end up with a unit vector moving from some other column into our pivot column. Let's do the first step of Gauss-Jordan for our example where we rescale the pivot row:

$$\begin{bmatrix} au/bu & 1 & 1/bu & 0 & 0 & 0 \\ av & bv & 0 & 1 & 0 & 0 \\ aw & bw & 0 & 0 & 1 & 0 \\ -A & -B & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ u \\ v \\ w \\ M \end{bmatrix} = \begin{bmatrix} cu/bu \\ cv \\ cw \\ k \end{bmatrix}$$

Now we do the rest of the process, and the result has this form:

$$\begin{bmatrix} au/bu & 1 & 1/bu & 0 & 0 & 0 \\ * & 0 & * & 1 & 0 & 0 \\ * & 0 & * & 0 & 1 & 0 \\ * & 0 & * & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ u \\ v \\ w \\ M \end{bmatrix} = \begin{bmatrix} cu/bu \\ * \\ * \\ k + B(cu/bu) \end{bmatrix}$$

Earlier we referred to the bottom element on the right side as k'. At our vertex [x,u] where x=0 and u=0 we now have M = k' = k + B(cu/bu) = k + B*y_1. Since B>0 and y_1>0, we confirm that k' > k, so our first hop on the exoskeleton has been successful.

At this point, we just iterate the same algorithm.

(1) Find the largest negative number in the last row to select the pivot column
(2) Find the smallest positive ratio to select the pivot row
(3) Do Gauss-Jordan relative to that pivot location.

When there are no more negative numbers in the last row, we are done, and we have arrived at the vertex (or one of several perhaps) that has the largest value of M.

This then is the Simplex Method. It is thus a "numerical" method or algorithm that solves our Problem. We have shown how one can understand the various swapping pivot iterations as steps along the exoskeleton of our feasible region, moving along a sequence of vertices toward larger M. Once the Simplex Method has been shown to be equivalent to this geometric view, we can dispose of the geometrical view (until, perhaps, something doesn't work, then we will want to understand things again.)

19. The Simplex Table

The Simplex table just a compact version of our matrix equation above. Since we never do anything with the last column in our matrix (the M column), there is no need to display it. And instead of displaying the vector which this matrix acts upon, we label the columns across the top of the table as reminders. The rightmost column is added to the right end of the table along with labels for the variables that each row represents.

Here then is the Simplex Table for the above problem with numbers appropriate to our starting position:

x	y	u	v	w		
au	bu	1	0	0	cu	=u
av	bv	0	1	0	cv	=v
aw	bw	0	0	1	cw	=w
-A	-B	0	0	0	k	=M

The rightmost two columns are called "the stub" and sometimes people put them on the left end like so:

		x	y	u	v	w	
u=	cu	au	**bu**	1	0	0	
v=	cv	av	bv	0	1	0	Table 1
w=	cw	aw	bw	0	0	1	
M=	k	-A	-B	0	0	0	

I prefer the stub on the right because I think of the table as the matrix equation S**v** = **c,** but I think most people put it on the left, so **c** = S**v**. Also, some people put the objective M equation as the first row instead of the last row. And many different names are use for the many variables, as you might imagine.

Now we can just use Courier text instead of a buggy equation editor to do the various Gauss-Jordan transformations. With the assumptions made earlier, here is what the table looks like after the first pivot operation, assuming the pivot element bu bolded above.

		x	y	u	v	w	
y=	*	*	1	*	0	0	
v=	*	*	0	*	1	0	Table 2
w=	*	*	0	*	0	1	
M=	k'	*	0	*	0	0	

where as was noted k' = k + B(cu/bu). Notice that the labels across the top have not changed, but a lot of the numbers in the table have changed. Notice also that the label on the top row has changed from u= to y=. This is simply because the unit vector is now in the y column and if we go to the vertex that this table is designed for, namely [x,u] where x=0 and u=0, then the upper left * is in fact what y will be equal to in our matrix equation. The labels on the left of the table (apart from the M) are always the current basic variables. The iteration which moved us from Table 1 to Table 2 was a swap $y \leftrightarrow u$. We have shown how this can be interpreted as a movement from one vertex to another on the feasible region, or as an exchange of basis vectors in the E^m space of the matrix column vectors.

Although the above discussion was specific to the case n=2 and m=3, the method applies for any n and m values. You label all n+m variables across the top, putting the non-basic ones on the left and then putting the basic ones to the right of those. The columns for the basic variables form an embedded m x m identity matrix (and had we kept the M column we would have an m+1 x m+1 identity matrix.) The rows are labeled on the left by the basic variables in the same order they appear across the top.

Once you have this starting position, you scan the bottom row for largest negative number, and that is the pivot column. Then you compute the ratios for all elements in the column above the bottom row and pick the smallest positive value, then you have your pivot row and hence the pivot point. Do Gauss-Jordan, and then iterate the algorithm until done.

In order to show that in many ways all n+m variables are on an equal footing, the variables are sometimes simply labeled x_1 through x_{n+m} so the starting table looks like this:

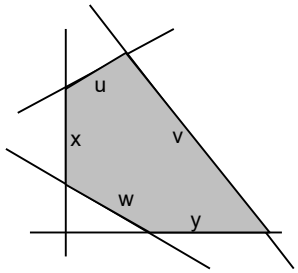
		<--- nonbasics = phys vars --->			<------ basics = slack vars ------->			
		x_1	x_2	...	x_n	x_{n+1}	x_{n+2}	...	x_{n+m}
$x_{n+1}=$	c_{n+1}	$a_{n+1,1}$	$a_{n+1,2}$		$a_{n+1,n}$	1	0		0
$x_{n+2}=$	c_{n+2}	$a_{n+2,1}$	$a_{n+1,2}$		$a_{n+1,n}$	0	1		0
. . . . .									
$x_{n+m}=$	c_{n+m}	$a_{n+m,1}$	$a_{n+1,2}$		$a_{n+1,n}$	0	0		1
M=	k	$-A_1$	$-A_2$		$-A_n$	0	0		0

where we have made up some row/column subscripts for the non-basic column entries in all but the last row. Note that $M = A_1x_1 + A_2x_2 + ... + A_nx_n + k$. Since we are trying to maximize M, we could as well set k=0 at the start. The right part of the table contains an m x m identity matrix.

20. Artificial Variables.

Using a starting table of the above form assumes that the Plane intersection $[x_1, x_2 .....x_n]$ is in fact a vertex of the feasible region in E^n. Again, this is where $x_1=0$, $x_2=0$, ... $x_n=0$. Once we are at this vertex, we know how to clamber up to an M-maximal vertex using the Gauss-Jordan pivots as pitons to swing us up from one vertex to the next until we get to the top of the mountain.

Suppose, however, the feasible region in E^n does not include the origin. Here is a simple example,



Here our problem is in E^2 and we have three constraints, but the w constraint is not of Type A. Instead of being a ≤ constraint, it is a ≥ constraint like so:

aw*x + bw*y ≥ cw

If we try to add a slack variable w ≥ 0 to this equation, we have to add it with a minus sign like so:

aw*x + bw*y - w = cw

This causes a "-1" to appear in the Simplex table in place of the previous +1, which puts the table into "non standard form". If we multiply that row by -1 to fix this, then the cw constant goes negative which is not allowed, as earlier discussed.

What we really want to do is "get to" one of the vertices of our feasible region so that we can start the usual mountain climbing process. Our usual starting point [x,y] is not possible. There are several methods available to locate a starting vertex. Here we will discuss a method known as the Big-M method, although we will use a big W variable instead of a big M variable since the M symbol is already in use.

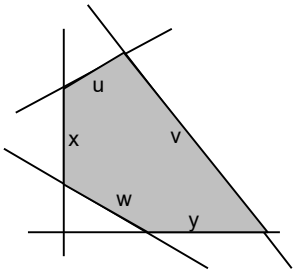
21. The Big-M Method.

In this method, we add an artificial extra axis to our E^n space to make it E^{n+1}. In our example, if we have only one ≥ constraint, we will be generalizing our feasible region from E^2 to E^3. The extra axis we will associate with a new "**artificial variable**" q. It is very much like a slack variable, and we add it into the constraint with a + sign so it will make the desired +1 in our Simplex table. Here we add the +q artificial variable to our constraint,

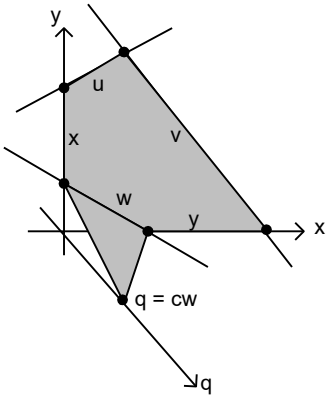
aw*x + bw*y - w + q = cw

For some fixed w ≥ 0, as we increase q, we can drive x and y down to x=0 and y=0 and still maintain the constraint. For w=0, here is partial attempt to draw the feasible region in (x,y,q) space:

in $E^{2:}$



in E^3 :



The original feasible region in E^2 is on the back wall q=0. As we map this thing into E^3, we get some strange 3D convex feasible region that is in fact an unbounded region. The other constraints that don't involve q (which is to say x,y,u,v) are just strips coming out toward the viewer from the back wall, which we have not drawn. For example, the u=0 line segment now becomes a strip on the u=0 plane which comes out perpendicular to the back wall. However, the w constraint becomes the little tilted triangle that is shown because it depends now on q. This is the w=0 Plane.

Here is the main point: the place where that little triangle hits the q axis is a *vertex* on the enhanced 3D feasible region. This is so because this point lies on the Planes x=0 and y=0 and w=0, and in E^3 this defines a point which is certainly a candidate vertex. We can see that by reducing q can get to two vertices of the enhanced feasible region (exoskel in E^3) which are also vertices of the original 2D feasible region. As we climb, however, we want to add a gravitational force to the problem that makes us climb eagerly toward the back wall. Once we get to a vertex on the back wall, we will be able to start with the normal algorithm, because we will be on a legal starting vertex. This is an extraordinarily clever idea, in my opinion. It provides a way to jump-start Simplex for ≥ constraints.

That gravitational force is where the Big-W comes in. We started off trying to maximize M = Ax + By + k. We change the problem so we are now going to maximize M' = M -qW. We imagine that W>0 is larger than all other numbers we will encounter in the problem. Since the enhanced feasible region has lots of vertices on q=0, we expect to quickly climb "up" to the back wall as we do the algorithm. Once we are on the back wall with q=0, then we are minimizing M' = M again. And we never climb downhill, so we will never come off the q=0 wall as we wander around on the 2D vertices. Once all the q type artificial variables are 0, we can forget about them.

Having stated the Big-W idea, we shall now see an example of how it works. Here was our simplex table for the problem where w had the simpler ≤ constraint, notice the bolded +1.

x	y	u	v	w			
au	bu	1	0	0	cu	=u	
av	bv	0	1	0	cv	=v	(1)
aw	bw	0	0	**1**	cw	=w	
-A	-B	0	0	0	k	=M	

Our problem is now that with the ≥ constraint, so this table looks like this (notice the bolded -1).

x	y	u	v	w			
au	bu	1	0	0	cu	=u	
av	bv	0	1	0	cv	=v	(2)
aw	bw	0	0	**-1**	cw	=w	
-A	-B	0	0	0	k	=M	

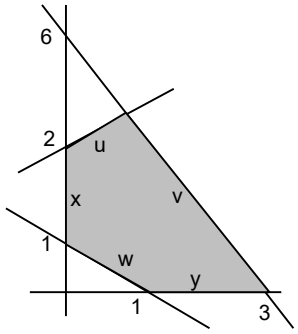
OK, so we add a new column for the new artificial variable q. We don't have any new constraint equations, we still only have m=3 of them, so there are no new rows to add. However, we will put an entry in our new q column which represents our new "gravitational" force pulling us to the back wall, and we rename our objective to be M'.

x	y	u	v	w	q			
au	bu	1	0	0	0	cu	=u	
av	bv	0	1	0	0	cv	=v	(3)
aw	bw	0	0	-1	1	cw	=w	
-A	-B	0	0	0	**W**	k	=**M'**	

Next, we need to clear out this new bottom item in the q column so that we will then have our three starting point unit vectors with all 0's in the last row, because this is our "standard form". To do this, add -W times the third row to the last row, and this does cause W's in other locations,

x	y	u	v	w	q			
au	bu	1	0	0	0	cu	=u	
av	bv	0	1	0	0	cv	=v	(4)
aw	**bw**	0	0	-1	1	cw	=w	
-A-aw*W	-B-bw*W	0	0	W	0	k-cw*W	=M'	

At this point, we are going to get in trouble relative to our graph above if we don't use some real numbers, so let's be very specific. The reason is that we cannot know where to pivot unless we have the actual numbers. Here is our picture with some numbers:



The constraints are these:

u:	$-x + y \le 2$	u line: $y = x + 2$
v:	$2x+y \le 6$	v line: $y = -2x +6$
w:	$x + y \ge 1$	w line: $y = -x + 1$

and then here is our table with real numbers (but we leave A and B and k general),

x	y	u	v	w	q			
-1	(1)	1	0	0	0	2	=u	
2	1	0	1	0	0	6	=v	(5)
1	1	0	0	-1	1	1	=w	
-A-1*W	-B-1*W	0	0	W	0	k-1*W	=M'	

This table is now in official starting point form because we have unit vectors in the columns u,v,q and their objective row entries are all 0. We are now going to start our climb toward the q=0 plane.

The bottom row has two offending (ie, large negative) entries on the left end (W is large > 0). Let's start with the leftmost one as our pivot column. The c/b ratios in this column are 2/-1 =-2, 6/2 = 3, and 1/1 = 1. The smallest positive ratio is 1, so we pivot on the third row where we have put a bolded 1.

Before doing this, we switch to the traditional convention of writing the last row as two rows just so we can keep the W and non-W terms separate. We also add informal row numbers on the left side. So rewrite the above as:

	x	y	u	v	w	q			
1	-1	(1)	1	0	0	0	2	=u	
2	2	1	0	1	0	0	6	=v	(6)
3	**1**	1	0	0	-1	1	1	=w	
4	-A	-B	0	0	0	0	k	=M'	
5	-W	-W	0	0	W	0	-W		check

Although the last two rows are really the same row of information, we can think of them as two different rows when it comes to doing the elementary row operations. This might not be obvious. Suppose we do an operation row4 <= row4 + α row3, and row5 <= row5 + β row3. Then this is the same as if we had done (row4 + row5) <= (row4 + row5) + (α+β) row3, and this is certainly a legal thing to do.

A. The Pivot. We pivot on the bolded 1 above in row 3, column x. Since it is already a 1, no need to scale the row. We then do our clearing operations one at a time (r1 means row 1):

Set: r1 <= r1 + r3:

	x	y	u	v	w	q			
1	0	(2)	1	0	-1	1	8	=u	
2	2	1	0	1	0	0	6	=v	(7)
3	**1**	1	0	0	-1	1	1	=w	
4	-A	-B	0	0	0	0	k	=M'	
5	-W	-W	0	0	W	0	-W		

Set: r 2 <= r2 - 2*r3

	x	y	u	v	w	q			
1	0	(2)	1	0	-1	1	8	=u	
2	0	-1	0	1	2	-2	4	=v	(8)
3	**1**	1	0	0	-1	1	1	=w	
4	-A	-B	0	0	0	0	k	=M'	
5	-W	-W	0	0	W	0	-W		

Set: r4 <= r4 + A*r3

	x	y	u	v	w	q			
1	0	(2)	1	0	-1	1	8	=u	
2	0	-1	0	1	2	-2	4	=v	(9)
3	**1**	1	0	0	-1	1	1	=w	
4	0	A-B	0	0	-A	A	k+A	=M'	
5	-W	-W	0	0	W	0	-W		

Set: r5 <= r5 + W*r3

	x	y	u	v	w	q			
1	0	(2)	1	0	-1	1	8	=u	
2	0	-1	0	1	2	-2	4	=v	(10)
3	**1**	1	0	0	-1	1	1	=w	
4	0	A-B	0	0	-A	A	k+A	=M'	
5	0	0	0	0	0	W	0		

B. Interpretation of the Resulting Table We are done with the pivot, and we see that we have done the swap x $\leftrightarrow$ q. The new basis is (x,u,v). If we set y=0 and q=0 and w=0, we get from these equations that u=8, v=4 and x=1. Since w=0, we know that x+y = 1 so y = 0. We have arrived at the vertex (x,y) = (1,0) in our feasible region. Since we have reached the promised q=0 back wall, we can now remove the q column (which contains some W information) from our problem to get:

	x	y	u	v	w			
1	0	2	1	0	-1	8	=u	
2	0	-1	0	1	2	4	=v	(4)
3	**1**	1	0	0	-1	1	=w	
4	0	A-B	0	0	-A	k+A	=M'	
5	0	0	0	0	0	0		

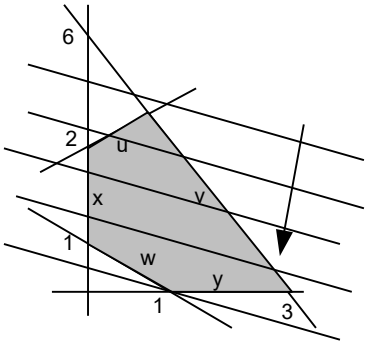
Now since our extra row 5 is all 0, we combine it back into row 4 to get

	x	y	u	v	w			
1	0	2	1	0	-1	8	=u	
2	0	-1	0	1	2	4	=v	(4)
3	**1**	1	0	0	-1	1	=w	
4	0	A-B	0	0	-A	k+A	=M'	

We are now at the START of a standard Simplex problem in the original space. Miraculously, it seems, there are no W's left anywhere in the table. We will discuss the reason for this in a moment. But first, let's finish the problem. Let's make our objective have this explicit form:

$$M = Ax + By + k = -x/2 - y + k. \quad \Rightarrow \quad y = -(1/2)x + k - M \quad A=-1/2 \; B = -1$$

We can plot these lines for various M on our earlier picture of the feasible region.



As the line moves down, M increases. The solution would appear to be the vertex we are currently sitting at, which is (x,y) = (1,0). Looking back at our simplex table bottom row as see that -A = 1/2 and A-B = 1/2 as well, so all values in the last row are positive so Simplex is in fact done! The solution value from the simplex table purports to be k+A which is k - 1/2. Is this correct?

$$M = Ax + By + k = -x/2 - y + k = -(1)/2 - (0) + k = k - 1/2 \qquad \text{yes.}$$

This concludes our example. The reader might notice that it is a real pain to do these elementary row operations by hand -- one mistake and you are cooked. There are various Java applets and Excel programs on the web to do such things, or you could just make our own little tool.

C. Why does W not appear in the table after the q column is removed? Once we arrive at a vertex on the q=0 plane and then delete the q=0 column, we have arrived at a valid vertex of the feasible region in the 2D space. We could have arrived at this table by other means. For example, we could see from the above graph that the point (x,y) = (1,0) is a valid vertex, we could compute the value of all the other variables at that vertex like u,v,w, and we could do elementary row operations on our starting table (including the -1 in the w column), and we could have arrived at the table we arrived at by the Big-M method. The resulting table obviously would not contain any W's anywhere since there *is* no W at that point. But there is only one Simplex table that is correct. If we got there by the Big-M method, then that table too must not contain any W's. To show by direct calculation that the W's don't appear anywhere at the end of the Big-M operations is very hard to do in the general case, but we know that it must be true if we have a valid problem that has a solution. The presence of residual W's in the non-artificial-variable part of the table would indicate that the original feasible region was null, I think.

22. The Two Phase Method.

In our treatment of the Big-M method, we carried along the variables A,B and k of our original objective "for the ride" We did need them at the end of course. If there is suspicion that there is no solution to the problem, then one can do the Big-M method up front with A = B = k = 0 so that M' = -qW. In this case, it suffices to set W = 1 so that M' = -q. In the general case, M' = - (sum of artificial variables). We then do exactly what we did above in the Big-M method to drive all the artificial variables out of the problem. If this is successful, then we take the newly arrived at Simplex table and we install the original-form M objective to replace the temporary objective M'. We do this with the original M coefficients like A,B,k. Then we do the elementary row operations on the last row (only) to get the table into standard form. We then arrive at the same result as the Big-M method which does both these steps at one time.

23. The general solution with artificial and slack variables.

We first get all constraints into our standard form with the constants positive, and we group them as being either ≤ constraints or ≥ constraints. Assume there are m1 constraints of the first kind and m2 of the second kind, with m1+m2 = m. We then add m slack variables as usual, and we add an additional m2 artificial variables. We add our gravitational adjustment M' = M - W * (sum of artificial variables) and add this to the table. We then do row operations to clear the W's out of the bottom rows of the columns corresponding to the artificial variables. This puts W's in other bottom row entries. We are then at a legal starting point. We look for a negative entry in the bottom row that is proportional to W. We use that column as a pivot. We do the ratio tests and locate the exact pivot point and execute the pivot. This results in an exchange of two variables between the basic set and the non-basic set because a unit vector has moved from one column to another. I think that if we choose columns with large W, then we will always end up swapping one of the artificial basic variables for one of the physical non-basic variables. If there are still large negative W terms in the bottom row, we do more pivots until they are gone.

When they are all gone, we will find that we have removed all artificial variable columns from the basis, call them q_1 q_2 .... q_{m2}. Each of these variables will have been swapped one at a time out of the basis. As we do this, non-basic (physical) variables are increasing from zero. At this point, we have "climbed" from our starting vertex in the E^{n+m2} space up to a vertex in the smaller E^n space which is the true space of our problem of interest.

We then delete all the q_i columns to make a reduced table that applies to the original E^n feasible region. We are amazed to find that there are no longer any W's anywhere in this table. So W is gone, the

artificial variables are gone, and we can then apply the regular simplex method from the starting point we arrive at. This starting point will be some vertex of the feasible region in E^n which is not the origin of E^n.

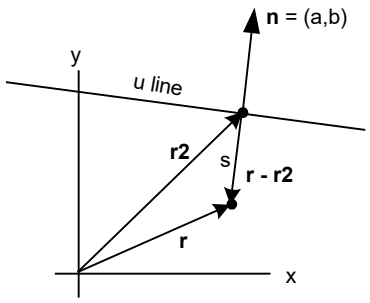
Getting to this starting point is sometimes called Phase I, and then doing the problem from there on is Phase II.

Note1: **Equality Constraints**. According to Mr. Lobos' ASU notes, you can handle an equality constraint by adding an artificial variable to it which gets driven out of the problem with the other ones. Here is his classification:

for each ≤ constraint:	add 1 slack variable
for each ≥ constraint	add 1 slack variable + 1 artificial variable
for each = constraint	add 1 artificial variable

I have not thought about the role of equality constraints, but probably everything just goes through as Mr. Lobos suggests. It does seem that an equality constraint would make the feasible region a lot smaller, lowering the dimension of its surface in E^n probably by 1 unit. I guess the single artificial variable added to the problem for that equality constraint makes it look like any other inequality constraint, but in the larger space with that fake variable added. When you finish up, your are then forced back on the "line". This then is in fact an artificial slack variable that is later removed.

Note 2: **Showing that a slack variables measures distance from a Plane**. Consider this picture:



What is the distance s from some arbitrary point **r** to the boundary line? By this we mean the distance of closest approach of the boundary line to point **r**. Here is a simple calculation to get the answer. Point $\mathbf{r_2}$ is selected on the line so that the vector $\mathbf{r_2 - r}$ is lined up with the line's (unit length) normal **n**. We then have

$$\mathbf{r_2 - r} = s\mathbf{n}$$

If we dot both sides with **n**, we know that $\mathbf{n} \bullet \mathbf{r_2} = c$ since $\mathbf{r_2}$ is on the line. So we get

$$c - \mathbf{n} \bullet \mathbf{r} = s \quad \Rightarrow \quad s = -ax - by + c \quad \Rightarrow \quad ax + by + s = c$$

This distance $s \geq 0$ is called "the slack variable" for this boundary line, and for the u boundary line it is called u in the discussion above. We will have similar slack variables w and v for the other two boundary lines. We have just shown that a slack variable *is* the distance from a point to the boundary line, providing we respect our conventions above.

Our little derivation required that (a,b) be a unit normal vector, so we needed $a^2 + b^2 = 1$. Were this not true, we would conclude that s is *proportional* to the distance shown, and that is just fine.

Note 3: Here are some facts about linear mappings (taken from another document). We consider only the case of a linear mapping from $E^2 \rightarrow E^3$ which has the vector form $\mathbf{u} = M\mathbf{x} + \mathbf{c}$.

A. Show that the entire x,y E^2 plane maps into a plane in E^3

First, can we find a fixed vector **n** in E^3 and a constant D such that $\mathbf{n} \bullet \mathbf{u} = D$ for all **u** generated by our mapping? If we can, then the plane of E^2 maps into a plane in E^3 described by the equation $\mathbf{n} \bullet \mathbf{u} = D$. This is a plane that has normal vector **n**, and which is displaced distance D from the origin.

From our mapping equation, $\mathbf{n} \bullet \mathbf{u} = \mathbf{n} \bullet (M\mathbf{x}) + \mathbf{n} \bullet \mathbf{c}$. Now consider:

$$\mathbf{n} \bullet (M\mathbf{x}) = n_1 (-au*x - bu*y) + n_2 (-av*x - bv*y) + n_3 (-aw*x - bw*y)$$

Then

$$- \mathbf{n} \bullet (M\mathbf{x}) = [\, n_1*au + n_2*av + n_3*aw \,]\, x + [\, n_1*bu + n_2*bv + n_3*bw \,]\, y$$

$$= \mathbf{n} \bullet \mathbf{a}\ x + \mathbf{n} \bullet \mathbf{b}\ y \qquad \mathbf{a} = (au, av, aw) \text{ and similar for b}$$

Now as long as vectors **a** and **b** do not align, we can select **n** to be **a x b** and for that **n**, we will have $\mathbf{n} \bullet \mathbf{a} = 0$ and $\mathbf{n} \bullet \mathbf{b} = 0$ and therefore $\mathbf{n} \bullet (M\mathbf{x}) = 0$ for all **x**. For this $\mathbf{n} = \mathbf{a} \times \mathbf{b}$, we then have $\mathbf{n} \bullet \mathbf{u} = \mathbf{n} \bullet \mathbf{c} = (\mathbf{axb}) \bullet \mathbf{c} \equiv D$. That is, D is a constant independent of x. This shows that the plane E^2 maps into a plane in E^3.

B. Show that points/lines in E^2 map into points/lines in E^3

The fact that points map into points is pretty clear. We put an E^2 point **x** into $\mathbf{u} = M\mathbf{x} + \mathbf{c}$ and out pops the point **u** in E^3. This mapping is not 1-to-1 because many points in E^3 map back to the same **x** in E^2. This is clear from the fact that the above equation set is 3 equations in 2 unknowns x and y. The mapping cannot possibly be 1-to-1 because E^3 is a bigger space than E^2. We will show right away that the entire E^2 maps into a certain plane in E^3 so the range of the mapping is that plane. Since this plane is less than the entire space E^3, the mapping is "into".

Now consider an arbitrary line in E^2 written as $\mathbf{x} = \alpha \mathbf{x_1} + \mathbf{e}$ where $\mathbf{x_1}$ defines an arbitrary vector in some direction, and α then maps out the line as α takes all real values. The line passes through arbitrary point **e** when $\alpha = 0$. Here is what happens to that line as it goes through the mapping to the E^3 space:

$$\mathbf{u} = M\mathbf{x} + \mathbf{c} = M(\alpha \mathbf{x_1} + \mathbf{e}) + \mathbf{c} = \alpha\, [\, M\mathbf{x_1}] + [\mathbf{c} + M\mathbf{e}]$$

$$= \alpha \mathbf{u_1} + \mathbf{E} \qquad \text{where } \mathbf{u_1} = M\mathbf{x_1} \text{ and } \mathbf{E} = \mathbf{c} + M\mathbf{e}$$

The result is some straight line in E^3 that passes through a point **E** and has direction $\mathbf{u_1}$.

C. Show that the mapping does *not* preserve angles or line segment lengths, so shape is not preserved

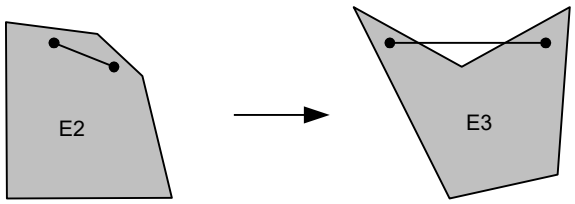
As an example, let $\mathbf{x} = (1,0)$. In this case we find $\mathbf{u}_x = M\mathbf{x}+\mathbf{c} = -\mathbf{a}+\mathbf{c}$. On the other had, it we start with x = (0,1), we get $\mathbf{u}_y = -\mathbf{b}+\mathbf{c}$. It is clear that our two starting vectors in E^2 are perpendicular, but in E^3 the dot product is going to be some strange number $\mathbf{u}_x \bullet \mathbf{u}_y = (-\mathbf{a}+\mathbf{c}) \bullet (-\mathbf{b}+\mathbf{c})$. Since a,b and c are arbitrary, this result is not going to be 0 in general, so the 90 degree angle is not preserved.

As for length of a line segment, consider again $\mathbf{x} = \alpha \mathbf{x}_1$ where now $\mathbf{x}_1 = (1,0)$. Let α range from 0 to 1, so we have a segment of unit length. In E^3 this becomes $\mathbf{u} = \alpha M\mathbf{x}_1 + \mathbf{c} = -\alpha \mathbf{a} + \mathbf{c}$. The endpoints of this line segment are at $\mathbf{u}_1 = \mathbf{c}$ and $\mathbf{u}_2 = -\mathbf{a} + \mathbf{c}$. Since **a** and **c** can be anything, the distance between these two points is generally not going to be 1.

So we have shown by counterexamples that, in general, angle and segment length is not preserved by our mapping. This means that if we map a polygon, we will end up with a polygon lying on some plane in E^3 but its shape will be distorted from that of the original polygon.

D. Show that convexity of the polygon is preserved by the mapping.

Here we are worried that maybe our mapped polygon in E^3 might look as shown on the right,



To disprove this possibility, we consider a hypothetical line segment that maps as shown. If the polygon is convex in E^3 then we should be able to find a segment that maps as shown. However, every point in the feasible region of E^2 must map into the feasible region of E^3. After all, we *obtain* the feasible region in E^3 from our mapping by letting the (x,y) coordinates run over all points in the feasible region in E^2. Since the E^2 polygon is convex, any line segment you draw in it must be fully contained in the feasible region. Thus, the reverse mapping of our hypothetical segment in E^3 must be entirely contained in E^2. But then all forward mapped points on this segment must lie inside the feasible region in E3. The above situation is therefore a contradiction and cannot happen.