A convex region in Euclidean space is one satisfying the condition that the closed line segment between any two points in the region is also in the region. For example, a sphere is convex. A convex polyhedron is one which is in some sense defined by a finite amount of data. What sense? There are several ways to make this precise, but the basic characterization is that the region is the intersection of a finite number of half-spaces $f_i \geq 0$, where the $f_i$ are affine functions. The simplest example, and the model in some sense for all others, is the coordinate octant $x_i \geq 0$. Another simple example is the simplex $x_i \geq 0$, $\sum x_i \leq 1$.

There is another, more geometric, way to specify a convex polyhedron. If $\Sigma$ is any set of points in $\mathbb{R}^n$, its convex hull is the set of all points which can be represented as a finite sum $\sum c_i P_i$ with $c_i \geq 0$, $\sum c_i = 1$, and all $P_i$ in $\Sigma$. This is clearly convex. For example, the line segment between two points $P$ and $Q$ is the convex hull of those two points. It turns out that the convex hull of any finite set of points and rays is a polyhedron. Conversely, any convex polyhedron is the convex hull of a finite set of points and rays.

Thus we arrive immediately at basic computational problems concerned with convex polyhedra—given a polyhedron specified in one of these ways, to find a specification in the other.

But there are several others. A face of a polyhedron is any affine polyhedron on its boundary.

- to tell whether the region is bounded;
- if it is bounded, to find its extremal vertices, and if not, its extremal edges;
- to find a minimal set of affine inequalities defining it;
- find the maximum value of a linear function on the polyhedron;
- describe the face where a linear function takes its maximum value;
- find the dimension of the polyhedron (taken to be $-1$ if the region is empty, which it may very well be);
- describe its facial structure completely;
- describe the dual cone of linear functions bounded on the polyhedron.

One way in which these problems arise is in interpreting geometry in high dimensions, which is a problem both practical and theoretical. There are now many algorithms known to answer such questions, but for most practical and theoretical purposes the one used most is the simplex method. The simplex method is known to require time exponential in the number of inequalities for certain somewhat artificial problems. But on the average it works very well.

There is another problem I’d like to add to the list, that of finding for a point of space the nearest point of the polyhedron. I know some simple algorithms in special cases, but in general I can only suggest the following: scan all inequalities, throwing away those satisfied by the point, and calculating perpendicular projections onto the other supports. Then throw away the projections that do not lie strictly inside the given face. This is not very efficient, but is there a better way?

Contents
I. Basic procedures
   1. Beginning
   2. An example
   3. Vertex bases
   4. Pivoting
   5. Tracking coordinate changes
   6. Finding a vertex basis to start

II. Applications
   7. Maximization
   8. Finding all vertices of a polygon
   9. Finding the dimension of a convex region
  10. Efficiency in computation

My main reference has been Vašek Chvátal, *Linear programming*, W. H. Freeman, 1983

Part I. Basic procedures

1. Beginning

Suppose given a finite set of affine functions $f_i$, and that we are concerned with the region $f_i \geq 0$. Suppose there to be in all $m$ functions of $n$ variables in this system. The first step in almost all computations is to find an affine basis of the set—a minimal subset of the $f_i$ in terms of which all the others may be affinely expressed. For example, suppose that the system is

$$
\begin{align*}
x - 1 &\geq 0 \\
y - 1 &\geq 0 \\
-x - y + 3 &\geq 0
\end{align*}
$$

The first two suggest an affine basis since $-x - y + 3 = -(x - 1) - (y - 1) + 1$. Thus it is useful to change coordinates from $(x, y)$ to $(x - 1, y - 1)$. In general, there might be several choices. In finding an affine basis, we are determining what you might call the essential dimension of our system. For example, a single inequality $f \geq c$ in two dimensions, with $f$ linear, has essential dimension one, since the region is invariant under translations along the line $f = 0$.

Once an affine basis is found, we replace the given set of inequalities by those in which all are expressed in terms of the given basis. The above system thus becomes

$$
\begin{align*}
X &\geq 0 \\
Y &\geq 0 \\
-X - Y + 1 &\geq 0
\end{align*}
$$

To start finding an affine basis, let $\mathbf{F}$ be the matrix whose rows are the linear components $\mathbf{f}_i$ of the $f_i$. Column reduce $\mathbf{F}$ to Gauss-Jordan form, multiplying on the right by elementary column operation matrices. At the same time keep track of the operations performed by applying the same operations to an $m \times m$ identity matrix. Doing this, we are replacing the original coordinates by new ones, and keeping track of the coordinate change. At the end we’ll have a matrix $\mathbf{E}$ in column echelon form. If $c$ is the (column) rank of the matrix, then for $1 \leq j \leq c$ we are given a row $r_i$ corresponding to the $j$-th
basis function \( f_r \). In row \( r_j \) there is a 1 in column \( j \) and 0 in all other columns. All entries in columns beyond \( r \) vanish identically. All entries \( e_{i,j} \) with \( i < r_j \) are 0. Here is a sample:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
\ast & \ast & 0 & 0 \\
0 & 0 & 1 & 0 \\
\ast & \ast & \ast & 0 \\
\ast & \ast & \ast & 0 \\
\ast & \ast & \ast & 0 \\
\end{bmatrix}
\]

The rows in which these special 1’s appear are the rows corresponding to the subset of \( f_i \) we are looking for. The coordinates of the linear components of the others with respect to this basis are laid out in the other rows. We now have to go from linear components back to express things in terms of the corresponding affine bases—if \( \mathbf{f} = \sum a_i f_i \) then

\[
\mathbf{f} + \mathbf{c} = \sum a_i (f_i + c_i) + c = \sum a_i f_i + \left( c - \left( \sum a_i c_i \right) \right).
\]

So we restore the constants to the system matrix.

We’ll also have at the end of this calculation an \( m \times m \) matrix \( R \) such that \( \mathbf{F} R = \mathbf{E} \). In fact, the process can be seen as one involving the original matrix \( \mathbf{F} \) and two matrices \( R \) and \( \mathbf{E} \) that get updated. At all times \( \mathbf{F} R = \mathbf{E} \). At the start \( \mathbf{E} = \mathbf{F} \) and \( R = I \). We might or might not need the final matrix \( R \) later, depending on what we want to do ultimately.

One thing we recover from this process is the degree of degeneracy of the inequalities, because at the end we can find trivially a basis for the linear subspace of translations leaving the polyhedron invariant. This degeneracy will not usually play a role once everything is expressed in terms of an affine basis, and we are working automatically on the quotient of our original space of \( n \) variables by the translations leaving our region invariant, which has dimension \( r \).

It is probably best to calculate \( R^{-1} \) as we go along. This way we can easily transform any function in either coordinate system to its expression in the other. In carrying out the column reduction, it is simplest but not most efficient to apply the inverse operations as row operations. If we multiply the current \( \mathbf{F} \) on the right by \( u \), we multiply the current \( R \) on the left by \( u^{-1} \).

From now on, we’ll assume that we have the system expressed in terms of an affine basis. This means that the region is described by coordinate inequalities \( x_i \geq 0 \) together with more inequalities expressed by affine functions expressed in terms of the \( x_i \).

2. An example

Suppose given the system of inequalities

\[
\begin{align*}
x + 2y & \geq 1 \\
2x + y & \geq 1 \\
x + y & \geq 1 \\
x - y & \geq -1 \\
3x + 2y & \leq 6.
\end{align*}
\]

We have the following picture, where the shading along a line \( f = 0 \) indicates the side where \( f \geq 0 \). The convex region defined by the inequalities is outlined in heavy lines.
The simplex method

\[
\begin{align*}
3x + 2y &= 6 \\
2x + y &= 1 \\
x + 2y &= 1 \quad \text{and expressed in this basis the system of inequalities is that corresponding to the matrix}
\end{align*}
\]

The matrix of linear coefficients is

\[
\begin{bmatrix}
1 & 2 \\
2 & 1 \\
1 & 1 \\
1 & -1 \\
-3 & -2
\end{bmatrix}
\]

which reduces like this:

\[
\begin{bmatrix}
1 & 0 \\
2 & -3 \\
1 & -1 \\
1 & -3 \\
-3 & 4
\end{bmatrix},
\begin{bmatrix}
1 & 0 \\
2 & 1 \\
1 & 1/3 \\
1 & 1 \\
-3 & -4/3
\end{bmatrix},
\begin{bmatrix}
1 & 0 \\
0 & 1 \\
1/3 & 1/3 & -1/3 \\
-1 & 1 & 1 \\
-1/3 & -4/3 & 13/3
\end{bmatrix}
\]

with

\[
R = \begin{bmatrix}
-1/3 & 2/3 \\
2/3 & -1/3
\end{bmatrix}
\]

The affine basis chosen is thus the pair of functions

\[
x + 2y - 1 \\
2x + y - 1
\]

and expressed in this basis the system of inequalities is that corresponding to the matrix

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
1/3 & 1/3 & -1/3 \\
-1 & 1 & 1 \\
-1/3 & -4/3 & 13/3
\end{bmatrix}
\]

The origin of the coordinate system with this basis is the point \((1/3, 1/3)\) in the original coordinates, the intersection of the two lines

\[
x + 2y - 1 = 0 \\
2x + y - 1 = 0
\]

It is not a vertex of the region. This agrees with the sign of the constant in the third row above:

\[
\begin{bmatrix}
1/3 & 1/3 & -1/3
\end{bmatrix}
\]

because this means that the affine function originally \(x + y - 1\) is now \((x + y - 1)/3\), and is negative when evaluated at the new origin.
3. Vertex bases

A convex polyhedron can be specified in many ways. One of the two most important is by means of affine inequalities \( f_i \geq 0 \), and the other is that it is the convex hull of its vertices. Each of these is required in certain circumstances, so it is important to be able to go back and forth between the two. I am interested here in finding the vertices given the inequalities. In fact, finding the extremal (irredundant) inequalities given the vertices is just this same problem in the dual space.

The way the simplex method approaches this problem is to first locate one vertex and then to traverse edges of the polyhedron to locate all the others. The principal structure in the simplex method is what I call a \textbf{vertex basis}, which is a subset \( B \) of the \( f_i \) with these properties: (1) it is an affine basis of the system; (2) the origin in the corresponding coordinate system is in the region, which means that the constants \( c_i \) are non-negative. This means that the origin is a vertex of the system, but not necessarily that the basis is \textit{tight} in the sense that its coordinate axes go along true edges of the region. The simplest situation is where the region is locally \textbf{simplicial}—given locally at the origin exactly by inequalities \( f \geq 0 \) for \( f \) in \( B \). In other words, where the polyhedron is locally a simplicial cone. This is the generic case, in the sense that perturbing a given system almost always turns any given polyhedron into one all of whose vertices are simplicial. A vertex not of this form is called \textbf{singular}. Keep in mind that it is the system that is singular, not necessarily the polyhedron itself. In two dimensions, for example, all vertices are geometrically simplicial, but systems of inequalities may nonetheless be singular, as the example given above is at the vertex \((0, 1)\).

In that example, the system is singular at \((0, 1)\) but not at the other three vertices of the region. The singularity at \((0, 1)\) arises because the three bounding lines

\[
\begin{align*}
2x + y - 1 &= 0 \\
x + y - 1 &= 0 \\
x - y + 1 &= 0
\end{align*}
\]

all pass through that point. Any two of the corresponding affine functions will form a vertex basis. Only the last two lines actually bound the region.

\[
\begin{array}{c}
\text{O} \\
\end{array}
\]

Being a vertex basis is a local property—it depends purely on the geometry of the region \( f_i \geq 0 \) in a neighbourhood of the origin.

We’ll see later how to find a vertex basis, given an affine basis. Almost all computations involving a convex polyhedron start by finding a vertex basis, and then moving from one vertex basis to another at a neighbouring vertex.
4. Pivoting

The basic step in using the simplex method is moving from one vertex basis to another—and, hopefully, from one vertex to another—along a single edge of the polygon. (Or rather, to be precise, along what you might call a virtual edge.) Geometrically we are moving from one vertex to a neighbouring one, except that in the case of a singular vertex where more than the generic number of faces meet the move may be a move along an edge of length 0. In all cases, this basic step is called a pivot.

The explicit problem encountered in pivoting is this: we are given a vertex basis and a variable $x_c$ in the basis. The $x_c$-axis represents an edge (or possibly a virtual edge, as I’ll explain in a moment) of the polygon. We want to locate the vertex of the polygon next along that edge, and replace the old vertex basis by one adapted to that vertex. The variable $x_c$ is part of the original basis. The new basis will be obtained by replacing that variable by one of the affine functions $f_r$ in the system of inequalities defining the polygon. In applications, the exiting variable $x_c$ is chosen according to various criteria, as we shall see in the next section. Sometimes there are several choices for $f_r$ as well.

In the example we have already looked at, we could start with the vertex basis $x - y + 1, -3x - 2y + 6$ at the vertex $(4/5, 9/5)$ and move along the edge $x - y + 1 = 0$, with $x_c = -3x - 2y + 6$. We can only go as far as the vertex $(0, 1)$, because beyond there both of the affine functions $2x + y - 1$ and $x + y - 1$ become negative. We can choose either one of them to replace $-3x - 2y + 6$ in the vertex basis.

When a pivot is made, the current affine basis is the given vertex basis, and in the course of the pivot it is changed to the new vertex basis. This involves expressing all the $f_i$ in terms of the new basis.

How does it work in general? With a vertex basis, any of the $f_i$ may be expressed as

$$f_i = a_i + \sum_j a_{i,j} x_j$$

with the $a_i \geq 0$ since the origin is assumed, by definition of a vertex basis, to lie in the region $f_i \geq 0$. We propose to move out along the directed edge where all basic variables but one of them, say $x_c$, are equal to 0, and in the direction where $x_c$ takes non-negative values. This amounts to moving along an edge of the polyhedron. We shall move along this edge until we come to the vertex at its other end, if there is one, or take into account the fact that this edge extends to infinity. So we must find the maximum value that $x_c$ can take along the line it is moving on. Along this edge the other $x_j$ vanish, and we have each

$$f_i = a_i + a_{i,c} x_c$$

There are two cases to deal with:
(a) If \( a_{i,c} \geq 0 \) there is no restriction on how large \( x_c \) can be;
(b) if \( a_{i,c} < 0 \) then we can only go as far as \( x_c = -a_i/a_{i,c} \).

Therefore if all the \( a_{i,c} \) are non-negative then the edge goes off to infinity, but otherwise the value of \( x_c \) is bounded by all of the \( -a_r/a_{r,c} \). In that case, we choose \( r \) such that \( a_{r,c} < 0 \) and set \( x_c \) equal to it. We also change bases—replacing \( x_c \) by \( f_r \). The minimum value might in fact be 0, in which case we change vertex bases (i.e. virtual vertices) without changing the geometric vertex. This sort of virtual move will happen only if the vertex is singular, which means as I have said that more than the generic number of faces meet in that vertex, and the maximal value of \( x_c \) will be 0.

Thus our choice of \( r \) is made so that
\[
\frac{a_i}{a_{i,c}} - \frac{a_r}{a_{r,c}} \geq 0
\]
for all \( i \), since if \( a_{i,c} \geq 0 \) there is no condition (\( a_i, a_{i,c} \) and \( -a_r/a_{r,c} \) are all non-negative) and if \( a_{i,c} < 0 \) then by choice of \( r \)
\[
\frac{a_i}{a_{i,c}} \geq -\frac{a_r}{a_{r,c}}.
\]

When we change basis, we have to change all occurrences of \( x_c \) (which becomes just a function \( f_c \) in our system) by substitution. We have
\[
x_c = \frac{1}{a_{r,c}} \left( f_r - a_r - \sum_{j \neq c} a_{r,j} x_j \right)
\]
so if
\[
f = A + \sum_j A_j x_j
\]
we rewrite it as
\[
f = (A - p a_r) + \sum_{j \neq c} (A_j - p a_{r,j}) x_j + p f_r
\]
where \( p = A_c / a_{r,c} \) (\( p \) is called the pivot value). By choice of \( r \), \( A - p a_r \geq 0 \) if \( f \) is one of the functions in the original set of inequalities, so the new vertex is again in the region we are looking at, and we do indeed have a new vertex basis.

Sometimes, as in the example, there will be several possible choices of \( f_r \), if the target vertex isn’t simplicial. Indeed, in practice when pivoting is done in an application there may be several possible edges to move out along—i.e. which variable \( x_c \) to exit from the basis. The most important thing in all cases is to prevent looping.

As I shall explain later, I choose the variables \( x_c \), and, in case of tie, the new \( f_r \), according to various criteria, principally in order to avoid going around in (virtual) circles.

5. Tracking coordinate changes

As we move among vertices, changing from one vertex basis to another, we’ll want to keep track of where we are in the original coordinate system. We can recover the location of a vertex from the basis, since it amounts to solving a system of linear equations. But this is a lot of unnecessary work. We can instead maintain some data as we go along that make this a much simpler task. What data do we have to maintain in order to locate easily the current vertex? A slightly better question: What data do we have to maintain in order to change back and forth between the current basis and the original one?

Keeping track of location comes in two halves: changing back and forth between the original coordinate system and the one determined by the affine basis we first calculated; changing back and forth between that first affine basis and the current vertex basis.
**Phase I.** An example should suffice. Suppose the matrix we get by column reduction is this:

$$E = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
* & * & 0 & 0 \\
0 & 0 & 1 & 0 \\
* & * & * & 0 \\
* & * & * & 0 \\
\end{bmatrix}.$$ 

This means that $FR = E$, where $R$ is a $4 \times 4$ matrix. But since column and row operations commute, this also means that $F^*R = \begin{bmatrix} f_1 \\ f_2 \\ f_4 \end{bmatrix} R = \begin{bmatrix} 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix} = I_{3,4}.$

Suppose now that we are given the values of $y_i = f_i$ for the basis functions $f_i$ at a point $P = (x)$, and wish to find $x$. We have

$$F^*x + c = y, \quad F^*x = -c$$

where

$$c = \begin{bmatrix} c_1 \\ c_2 \\ c_4 \end{bmatrix}$$

that is to say, $c$ represents the constants in the affine functions $f_i$, and $x$ has $r$ rows. Solving $F^*x = y - c$ amounts to solving

$$I_{3,4} R^{-1} x = y - c$$

$$R^{-1} x = \begin{bmatrix} y_1 - c_1 \\ y_2 - c_2 \\ y_3 - c_3 \\ * \end{bmatrix}$$

$$x = R \begin{bmatrix} y_1 - c_1 \\ y_2 - c_2 \\ y_3 - c_3 \\ * \end{bmatrix},$$

since it is simple to solve the system with coefficient matrix $I_{3,4}$. In this case, the ‘origin’ of the new system is a line whose parametrization is given by this formula when $y = 0$.

**Phase II.** As we change from one vertex basis to another, we just have to update the affine transformation from the current basis to the original one, substituting $f_i$ for $x_c$. This amounts to an elementary row operation on the special matrix describing this transform.

6. Finding a vertex basis to start

What remains is the problem of finding an initial vertex basis of a system $f_i \geq 0$ (or finding out that the system is inconsistent and that no vertex exists). This will turn out to be a special case of the original problem that motivated the simplex method, that of finding the maximum value of a linear function on a polyhedron, given an initial vertex basis. I postpone to the second part the explanation of how this works. At any rate, the solution of the problem here is to use pivoting applied to a simpler problem in space of one dimension higher obtained by homogenizing the original one—a problem for which an initial vertex basis is obvious.

Homogenize the affine functions with an added variable $x_0$ as the coefficient of the constant terms. Write out the matrix of the $f_i$ as rows. Add $x_0 \geq 0$ and $x_0 \leq 1$ to the list of inequalities. The origin is a vertex
of the region involved, and the $x_i$ form a vertex basis. Here is what happens for the simple case where
the original system is

$$
\begin{align*}
    x_1 & \geq 0 \\
    x_1 & \geq 1 \\
    x_1 & \leq 2 .
\end{align*}
$$

The new homogeneous system is

$$
\begin{align*}
    x_1 & \geq 0 \\
    -x_0 + x_1 & \geq 0 \\
    -2x_0 + x_1 & \leq 0 ,
\end{align*}
$$

and the picture becomes

In this example, the final move will be along one of the thick edges to one of the circles as vertex.

In general, the idea is now to maximize $x_0$. Actually, we just need to find some edge, any true edge, leading away from the origin. If there is none, the original region is empty. Otherwise the maximum value of $x_0$ is 1, and we’ll recover a vertex basis of the original problem by throwing away $-x_0 + 1$ from the basis we have at the end and setting $x_0 = 1$.

This part will be a bit unusual from the standpoint of linear programming, since all moves except the final one will be virtual. What we are doing, roughly, is moving around in the ‘web’ of frames defined by the homogenized functions $f_i$.

### Part II. Applications

#### 7. Maximization

The basic problem in linear programming is to maximize an affine function $z = c + \sum c_i x_i$ subject to the conditions $f_i = a_i + \sum a_{i,j} x_j \geq 0$. Here, as we change vertices we keep track of the expression for $z$ in the current coordinate system. The basic idea is to choose the edge to follow by requiring that $z$ increase on this edge, until $z$ is maximized and no choice of edge is possible. But if the move is virtual then something has to be done to avoid looping. One valid process to follow is to choose the exiting and entering variables to have the largest possible index. It is a theorem of R. Bland, explained in Chvátal’s book on pages 37–38, that this will always work.

This works because at any vertex that is the intersection of $n$ hyperplanes, either one edge goes down, or it is part of the bottom of the polyhedron.
8. Finding all vertices of a polygon

If we simply want to locate all possible vertices of a convex region, we start by finding a vertex basis for the region. (How to do this is explained in the next section.) Then we locate all vertices by locating all vertex bases, collecting them into equivalence classes. Note that each vertex will correspond to a subset of the original set of functions, so our search is finite. (Indeed, a good way to specify a vertex, at least when using exact arithmetic, is to tag a vertex by the functions vanishing there.) On the other hand, it may happen that we have a singular vertex where several different sets of basic variables give rise to the same geometrical vertex. If we are doing floating point arithmetic, because of possible floating point round-off we should ignore this possibility, and proceed as if it doesn’t occur. It shouldn’t cause trouble—a single point might decompose into several, but they will all be close together and effectively indistinguishable.

Of course the region may be unbounded, in which case we should find the infinite edges of the polyhedron, too.

Locate all edges and vertices by the simplex method, indexing the vertices by the set of basis variables, edges by the coordinate of the edge. Store vertices and edges as we find them. Chvátal recommends using a queue for this, but I don’t see why a stack wouldn’t work as well. Vertices are pushed onto the stack as they are met. They are distinguished by the subset of basic functions, so two vertices with the same geometric vertex might be distinguished in this scheme. Most serious is the fact that each vertex comes equipped with its tableau, which must be transformed in going from one vertex to a neighbour through a pivot. This makes the algorithm here of order $n^2$ for $n$ inequalities.

9. Finding the dimension of a convex region

The dimension of a polyhedron makes sense in computation only when exact arithmetic is used, because the smallest of perturbations can change a set of dimension zero to an open one.

Start with trying to find a vertex basis. If there is none, the dimension is $-1$. Throw away all but the $f_i$ vanishing at the origin. Look for an edge out of the origin; if none, the dimension is 0. Otherwise, throw away all those containing the variable $x_{c_j}$ in effect looking now at the tangent cone of that edge. Recursion: the dimension is one more than the dimension of that tangent cone. In fact, the whole process amounts to finding an edge and replacing the original region by the tangent cone along the edge.

10. Efficiency in computation

We don’t want to copy a lot of data around. Still, at the beginning put the columns into arrays, but later on the rows.