1 Curves and Surfaces

$\mathbb{R}$ will stand for the field of real numbers. $\mathbb{R}^2$ and $\mathbb{R}^3$ will stand for the real euclidean plane and the real euclidean space, as we know it.

1.1 Lines and Planes

A line in space may be specified by a point $p \in \mathbb{R}^3$ and a vector v again, in $\mathbb{R}^3$, which will represent the 'direction' of the line. A general point on the line is given by $p(t) = p + t \cdot v$. If $p = (x_0, y_0, z_0)$ and $v = (x_1, y_1, z_1)$, then

$$p(t) = (x_0 + tx_1, y_0 + ty_1, z_0 + tz_1)$$

Thus, we have a map $f : \mathbb{R} \to \mathbb{R}^3$ given by $t \to p + tv$ such that the image of f is the required line. Such a representation is called the **parametric** representation. One sees at once, that there are many alternatives for chosing p, v and the 'parameter' t, which yield the same line as the image. For example: $g(t) = (p + v) + (1 - t^3)v$ is again a parametrization of the same line.

The second possible representation of a line is the **implicit** form, such as the zeros of $x + 2y + 3z = 4; 5x - 4y + 7z = 3$. These two equations define a line. Again, there are many possibilities: $x + 2y + 3z = 4; 6x - 2y + 10z = 7$ define the same line.

The plane may also be represented either parametrically, or in the implicit form. For a fixed point $p \in P$ and directions d_1 and d_2, we have the general point $s(u, v) = p + ud_1 + vd_2$ on the plane. Note that we may interpret s as a map $s : \mathbb{R}^2 \to \mathbb{R}^3$ with $(u, v) \to s(u, v) = (x(u, v), y(u, v), z(u, v))$. The simplest implicit definition of a plane is by one equation such as $4x - 5y - 6z = 7$. There are, of course, alternate implicit definitions, e.g., $(5x - 5y - 6z - 7)^2 = 0$.

1.2 Geometry of Curves and Surfaces

Having described planes and lines, we move to general curves and surfaces in $\mathbb{R}^3$. A parametric curve is a map $f : \mathbb{R} \to \mathbb{R}^3$. This is composed of three functions $(x(t), y(t), z(t))$ of the single parameter t, which denote the x, y and the z co-ordinate of the point on the curve as a function of t. An implicit representation of a curve is given by the common zeroes of two equations $f_1(X, Y, Z) = 0; f_2(X, Y, Z) = 0$.

As an example, we see that the plane circle is given by the implicit equation $X^2 + Y^2 - 1 = 0$. One parametrization of the circle is $c : \mathbb{R} \to \mathbb{R}^2$, given by $c = (x(t), y(t))$ where $x(t) = \frac{2t}{1+t^2}$ and $y(t) = \frac{1-t^2}{1+t^2}$. On may check that:

$$x^2(t) + y^2(t) = \frac{(2t)^2 + (1 - t^2)^2}{(1 + t^2)^2} = 1$$

Thus the general point $(x(t), y(t))$ does indeed lie on the circle.

Parametrized surfaces are given as maps $s : \mathbb{R}^2 \to \mathbb{R}^3$. This is also given by the three coordinate functions $(x(u, v), y(u, v), z(u, v))$ of two parameters u, v. On the other hand, implicitly defined surfaces may be given by a single equation $f(X, Y, Z) = 0$.

As an example, the equation $X^2 + Y^2 - 1 = 0$ defines a cylinder in $\mathbb{R}^3$ with the z-axis as the axis of symmetry. The parametrization of the cylinder is:

$$\left(\frac{2u}{1 + u^2}, \frac{1 - u^2}{1 + u^2}, v\right)$$

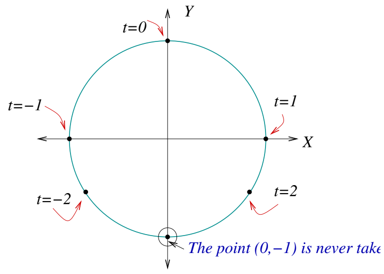

Figure 1: Missing Points.

Observe that as u varies, a circle is traced at the 'height' $z = v$. Another example is the sphere given implicitly as $X^2 + Y^2 + Z^2 - 1 = 0$, and given parametrically by:

$$(\frac{2v}{1+v^2}\frac{2u}{1+u^2}, \frac{2v}{1+v^2}\frac{1-u^2}{1+u^2}, \frac{1-v^2}{1+v^2})$$

One may check that:

$$x^2(u,v) + y^2(u,v) + z^2(u,v) = (\frac{2v}{1+v^2})^2 + z^2(u,v) = 1$$

We make three remarks which we will clarify later. Firstly, not all implicitly defines curves have a parametrization; in fact, most dont. One example is the the **elliptic** curve $Y^2 = X(X^2 - 1)$. However, every parametrically defined curve has an implict form, and similarly for parametrically defined surfaces. This is proved by the method of resultants, a technique for eliminating variables.

Secondly, even when a parametrization exists, it may not cover the whole curve. Eor example, for the parametrization $c(t)$ of the circle above, we may tabulate points on the circle for various values of t as below:

t	-2	-1	0	1	2
$p(t)$	$(-\frac{4}{5}, -\frac{3}{5})$	$(-1, 0)$	$(0, 1)$	$(1, 0)$	$(\frac{4}{5}, -\frac{3}{5})$

We see that as t ranges over $\mathbb{R}$, the point $(0, -1)$ is never taken, and therefore is missing from the image of the parametrization $c(t)$. This is to be expected since the line is 'open' and the circle is a 'closed' curve. Similarly, for the cylinder, the points $(0, -1, z)$ is never taken for any z: this simply follows from the circle case above. For the sphere, we see that the point $(0, 0, 1)$ is never taken by the parametrization.

The third remark is that different approaches to the representation of curves and surfaces serve different needs. For example, when in implicit form, it is difficult to produce points on the curve/surface, for that means solving equations in one/two variables. On the other hand, the parametrized representation makes it very convenient to produce points. On the other hand, given a candidate point p, it is easy to check if p lies on the curve; one just have to check if p satisfies the defining equations. Note that this question is hard to resolve for a parametrized curve.

2 Faces and Edges

Imagine, for the moment that we wish to represent parametrically, the whole circle. Since we have seen as above, that the natrural parametrization fails; in fact no such parametrization exists, we

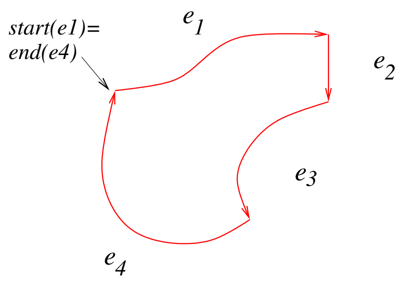

Figure 2: A Loop.

must reconcile with this fact, and at the same time, do get every point on the circle. One option is to break the circle into two parts, and represent each part with a parametrization. Thus, for example, the map c, restricted to $[-1, 1]$ gives the 'top' half of the circle. The bottom half may be represented by the parametrization $d(t) = (\frac{2t}{t^2+1}, \frac{t^2-1}{t^2+1})$. Note that $d(t)$ was obtained by replacing t by $\frac{1}{t}$ in $c(t)$. Thus t goes from 1 to -1, $d(t)$ will trace the bottom half of the circle. Thus we may represent the circle in terms of two parametrizations $c, d : \mathbb{R} \to \mathbb{R}^2$, however, the domains of c, d are restricted to the intervals $[-1, +1]$.

2.1 Edges and Loops

Let us now define the entity **edge** as a pair (I, f), where f is a function $f : \mathbb{R} \to \mathbb{R}^n$, and $I \subset \mathbb{R}$ is an **oriented interval** $[a, b]$. The edge may be identified as **starting** from the vertex $f(a)$ and ending at the vertex $f(b)$ while tracing out $f(t)$ for every point t between a and b. Note that $([a, b], f)$ and $([b, a], f)$, while geometrically being the same edge, are oriented inopposing directions. Thus the orientation gives a direction to the edge, and this frequently very useful. The vertices $f(a)$ and $f(b)$ will be denoted as $start(e)$ and $end(e)$. Typical kernels require the edge to not self-intersect and that $start(e) \neq end(e)$.

We thus see that our circle may be represented as two edges $e_1 = ([-1, 1], c(t))$ and $e_2 = ([1, -1], d(t))$. Also note that the orientation of e_2 as going from 1 to -1 is convenient, as it 'follows' the orientation of e_1.

The arrangement above is a special case of a **loop** which is defined as a collection of edges $(e_1, e_2, \ldots, e_k)$ where each e_i is an edge, and $start(e_i) = end(e_{i-1})$ with $start(e_1) = end(e_k)$. As before, there are other geometric requirements for a loop which are implemented differently for different kernels. One typical requirement is that no two edges e_i and e_j intersect other than the case when $i = j \pm 1$, then too only at the appropriate end-points.

2.2 Domains and Faces

A loop in the plane $\mathbb{R}^2$ always splits $\mathbb{R}^2$ into two parts, the inner (bounded) part and the outer (unbounded) part. The orientation of the loop may be used select one of these two sub-parts. A loop is said to be **anti-clock-wise** if there is a point in the bounded part such that the loops winds around this point in an anti-clock-wise manner. Otherwise, the loop is called **clockwise**. The loop in Figure 2 above is clock-wise.

Our next entity is the **domain** D which is a subset of $\mathbb{R}^2$. The simplest domain $D = domain(O)$ is defined by a single loop O in the plane. If O is clockwise, then D is taken to be the unbounded

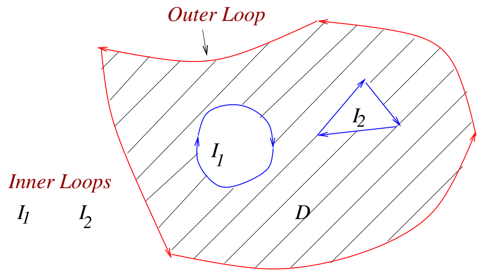

Figure 3: A Domain.

part outside (and including) the loop. If O is anti-clock-wise then D will denote the part inside (and including) the loop. The general domain is the intersection of simple domains

$$D = domain(O_1) \cap domain(O_2) \cap \ldots \cap domain(O_k)$$

One may check that it suffices to require that $O_1, \ldots, O_k$ do not intersect, and that atmost one, say O_1 is oriented anti-clockwise, and others must be oriented clockwise. If D is bounded, then there is indeed the O_1 which is anti-clockwise. This loop is called the **outer-loop**, and the others are called **inner loops**. Since, we will usually be interested in bounded domain, we will assume that D is given by a list $(L_1, \ldots, L_k)$ of mutually non-intersecting loops such that L_1 is the outer-loop and all other loops are inner.

A **face** may now be defined a tuple $F = (D, f)$ where $f : \mathbb{R}^2 \to \mathbb{R}^3$ is a function, and D is a domain. The points of the face are assume to come by restricting f to D. Thus, for all $(u, v) \in D$, $f(u, v) \in F$. Various kernels require many other properties such as (i) F should not self-intersect, (ii) f should be smooth, and so on.

We must add that the entities edge and face are **geometric** entities, since they must be accompanied by functions which actually do the parametrization. In contrast, the entities such as loop and domain are **topological** or **combinatorial** since they are defined by a collection of **geometric** entities in special configuration. Note that every edge e which occurs in one of the loops O defining domain D, are plane edges, i.e., in $\mathbb{R}^2$. Thus $e = (I, g)$ where I is an oriented interval and $g : \mathbb{R} \to \mathbb{R}^2$ is the parametrization. Thus the composition $g \circ f$ is a map from $\mathbb{R}$ to $\mathbb{R}^3$. This may be used to define the **co-edge** e' as the composition $e' = (I, g \circ f)$. This e' is a space edge and defines a part of the boundary of the face F. The space $\mathbb{R}^3$ is called the **object space**, while $\mathbb{R}^2$, when it is used to define the domain, is called the **parametrization space**.

Thus the data for a face includes (i) the domain, and the parametrizing function. The definition for the domain will include its loops, each loop, its edges and co-edges and so on. Note that each co-edge has an 'owner', viz., the face from which its definition came. The general solid is represented as a collection of faces with certain co-edges identified. Thus, when face F_1 meets face F_2 along an edge e, then the corresponding co-edges e_1 (coming from F_1) and e_2 (from F_2) are identified as equal.

3 Polynomials and Resultants

$\mathbb{C}$ will stand for the field of complex numbers. Let t be a 'variable' and let $\mathbb{R}[t]$ (or $\mathbb{C}[t]$) denote the collection of all polynomials in the variable t with real (or complex) coefficients. The typical

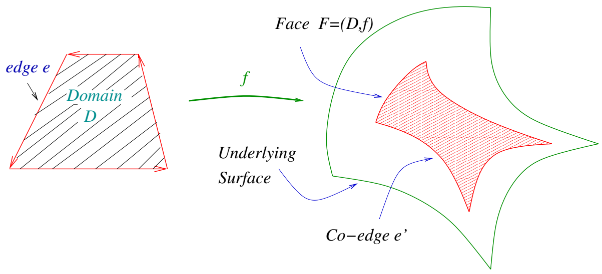

Figure 4: A Face.

polynomial $p(t)$ is represented as

$$p(t) = a_d t^d + a_{d-1} t^{d-1} + \ldots + a_1 t^1 + a_0 t^0$$

Let $1 \leq m \leq d$ be the largest number such that $a_m \neq 0$. The integer m is called the degree of the polynomial and is denoted by $deg(p)$. Two polynomials $p = \sum_i a_i t^i$ and $q = \sum_i b_i t^i$ may be added: if $r = \sum_i c_i t^i = p + q$, then $c_i = a_i + b_i$ for all i. Polynomials may be multiplied: if $s = \sum_i d_i t^i$ and $s = pq$, then $s_i = \sum_{j=0}^{k} a_j b_{i-j}$. Thus for example, $d_0 = a_0 b_0$, $d_1 = a_0 b_1 + a_1 b_0$, and so on. Note that $deg(pq) = deg(p) + deg(q)$, while $deg(p+q) \leq \max(deg(p), deg(q))$.

Let V_d denote the space of all polynomials p such that $deg(p) \leq d$. The space V_d is a vector space under polynomial addition. The dimension of V_d is $d+1$ and $\{1, t, t^2, \ldots, t^d\}$ is an obvious basis of V_d

We say that p **divides** q if there is a polynomial r such that $q = pr$. We say that $\alpha \in \mathbb{C}$ (or $\mathbb{R}$) is a **root** of $p(t) = \sum_i a_i t^i$ if $p(\alpha) = \sum_i a_i \alpha^i = 0$.

Theorem 3.1 The division algorithm: *Given polynomials a, b, there are unique polynomials q, r such that $a = bq + r$, such that either $r = 0$ or $deg(r) < deg(b)$.*

Proof: The existence of q, r is the so-called long division algorithm. The uniqueness is easy too: if $a = bq + r = bq' + r$, then $b(q - q') = (r' - r)$, whence, either q-q'=0, whence $r' - r = 0$ as well, or we have $deg(b(q - q')) \geq deg(b)$, while $deg(r' - r) < deg(b)$, a contradiction. $\square$

Corollary 3.2 *If α is a root of polynomial p, then $(t - \alpha)$ divides p.*

The following is a rewording of the famous *fundamental theorem of algebra*:

Theorem 3.3 *Every polynomial $p = a_d t^d + \ldots + a_0$ of degree d with real/complex coefficients may be factorized over complex numbers as $p(t) = a_d \prod_{i=1}^{k} (t - \alpha_i)^{m_i}$, where the α_i's are all distinct. This factorization of p is unique, and the collection $\{\alpha_1, \ldots, \alpha_k\}$ is the set of all roots of p. Furthermore, $m_1 + \ldots + m_k = d$, and the number m_i is called the multiplicity of the root α_i. If p has real coefficients and α is a root of p with multiplicity m, then so is the complex conjugate $\overline{\alpha}$, and with the same multiplicity.*

We now come to an important result. Let $p = a_m t^m + \ldots + a_0$ and $q = b_n t^n + \ldots + b_0$ be two polynomials of degree m and n respectively. We ask if p and q have a common root. The naive of doing this would be to solve p and q and then compare the roots of the two polynomials. However, other than numerical solution, there is no known procedure to write down the roots of a polynomials, given by its coefficients. It turns out that to check if p and q have a common root, it does not quite require us to find the roots. This is by the famous **gcd** algorithm for polynomials. Thus the $gcd(p,q)$ is 1 (or equivalently, a non-zero constant) iff p and q do not have a common root. Since the gcd is computable using the long-division process, it may be directly used to implicitize parametric definitions of curves and surfaces. Consider, for example, the circle parametrized by $x = \frac{2t}{1+t^2}$ and $y = \frac{1-t^2}{1+t^2}$, or equivalently we have the polynomials $p(x,y,t)$ and $q(x,y,t)$ below:

$$\begin{array}{rcl} xt^2 - 2t + x & = & 0 \\ (y+1)t^2 + (y-1) & = & 0 \end{array}$$

Clearly, those points (x_0, y_0) lie on the circle for which there is a t_0 which is a common solution to both the equations. In other words, the condition that $p_0(t) = p(x_0, y_0, t)$ and $q_0(t) = q(x_0, y_0, t)$ have a common solution is that their $gcd(p_0(t), q_0(t))$ should have the zero constant term. The gcd can be computed by the long-division algorithm: dividing p_0 by q_0, we get the remainder as:

$$-2t + \frac{2x_0}{y_0 + 1}$$

Dividing q_0 by the remainder, we get:

$$\frac{x_0^2 + y_0^2 - 1}{y + 1}$$

Thus, the gcd has zero constant term tells us that the numerator must be zero, which gives us the implicit equation of the circle, and the denominator $y+1$ gives us the exceptional point $(0,-1)$, with $y + 1 = 0$.

In general, this process can be used to eliminate any variable from two equations. A more explicit formulation is that of the **resultant**. We have:

Proposition 3.4 *If the degrees of* p *and* q *are* m *and* n *respectively, then* $gcd(p,q) \neq 1$ *if and only if there are polynomials* f *and* g *such that* $pf = qg$ *and* $deg(pf) < m + n$.

Proof: Suppose that $(t-\alpha)$ was a common root. Then $p = (t-\alpha)p'$ and $q = (t-\alpha)q'$. Thus with $f = q'$ and $g = q'$, we have $pq' = qp' = (t-\alpha)p'q'$. Also note that $deg(pq') = m + n - 1 < m + n$.

Conversely, if there were f, g as above, and assuming that p and q have no roots in common, for every root α of p with multiplicity m, we see that $(t-\alpha)^m$ divides pf, the LHS, and therefore must divide the RHS, qg. Since α is not a root of q, we must have that $(t-\alpha)^m$ must divide g. In other words, $deg(g) \geq m$ and $deg(qg) \geq m + n$, a contradiction. □

We know show that the condition of proposition 3.4 is easily checkable. We construct the polynomials $p^i = t^i p$, for $i = 0, 1, \ldots, n-1$, and $q^j = t^j q$, where $j = 0, 1, \ldots, m-1$. Let $P = \{p^0, \ldots, p^{n-1}\}$ and $Q = \{q^0, \ldots, q^{m-1}\}$ and note that every polynomial of $P \cup Q$ is of degree atmost $m + n - 1$. Thus the set $P \cup Q$ is potentially a collection of $m + n$ polynomials in the vector space V_{m+n-1}, the space of polynomials of degree atmost $m + n - 1$. We now have the following proposition:

Proposition 3.5 *The polynomials* p *and* q *have a common root if and only if the polynomials in* $P \cup Q$ *are linearly dependent.*

Proof: Let $\alpha_0, \ldots, \alpha_{n-1}$ and $\beta_0, \ldots, \beta_{m-1}$ be numbers, not all zero, such that $\sum_i \alpha_i p^i + \sum_j \beta_j q^j = 0$ (inside V_{m+n-1}). We thus see that:

$$(\alpha_0 t^0 + \ldots + \alpha_{n-1} t^{n-1})p = (-\beta_0 t^0 + \ldots + \beta_{m-1} t^{m-1})q$$

In other words, with $\sum_i \alpha_i t^i$ as f and $\sum_i (-\beta_i) t^i$ as g, we have $pf = qg$ is a polynomial of degree not exceeding than $m+n-1$. Conversely, if we have f and g as above, then they can be unfolded to construct α's and β's, a linear dependence between the polynomials in $P \cup Q$. $\square$.

For the polynomials p and q as above, we define the resultant matrix $Res_t(p,q)$ (with the t to denote the variable) as the $(m+n) \times (m+n)$ matrix below:

$$Res_t(p,q) = \begin{bmatrix} a_m & a_{m-1} & \ldots & a_0 & 0 & 0 & \ldots & 0 \\ 0 & a_m & a_{m-1} & \ldots & a_0 & 0 & \ldots & 0 \\ 0 & & \vdots & \ldots & & \vdots & & \ldots \\ 0 & \ldots & 0 & 0 & a_m & a_{m-1} & \ldots & a_0 \\ b_n & b_{n-1} & \ldots & b_0 & 0 & 0 & \ldots & 0 \\ 0 & b_n & b_{n-1} & \ldots & b_0 & 0 & \ldots & 0 \\ 0 & & \vdots & \ldots & & \vdots & & \ldots \\ 0 & \ldots & 0 & 0 & b_n & b_{n-1} & \ldots & b_0 \end{bmatrix}$$

Proposition 3.6 *The polynomials p and q have a common root iff $det(Res_t(p,q)) = 0$, or in other words, if the resultant matrix is singular.*

Proof: Recall that a basis of V_{m+n-1} are the polynomials $\{t^0, t^1, \ldots, t^{m+n-1}\}$. Whence, any polynomial in V_{m+n-1} may be expressed as a linear combination in this basis. In particular, we see that ordering the elements of $P \cup Q$ as $\{p^{n-1}, \ldots, p^1, p^0, q^{m-1}, \ldots, q^1, q^0\}$, we see that:

$$\begin{bmatrix} p^{n-1} \\ \vdots \\ p^1 \\ p^0 \\ q^{m-1} \\ \vdots \\ q^1 \\ 1^0 \end{bmatrix} = \begin{bmatrix} a_m & a_{m-1} & \ldots & a_0 & 0 & 0 & \ldots & 0 \\ 0 & a_m & a_{m-1} & \ldots & a_0 & 0 & \ldots & 0 \\ 0 & & \vdots & \ldots & & \vdots & & \ldots \\ 0 & \ldots & 0 & 0 & a_m & a_{m-1} & \ldots & a_0 \\ b_n & b_{n-1} & \ldots & b_0 & 0 & 0 & \ldots & 0 \\ 0 & b_n & b_{n-1} & \ldots & b_0 & 0 & \ldots & 0 \\ 0 & & \vdots & \ldots & & \vdots & & \ldots \\ 0 & \ldots & 0 & 0 & b_n & b_{n-1} & \ldots & b_0 \end{bmatrix} \begin{bmatrix} t^{m+n-1} \\ t^{m+n-2} \\ \vdots \\ t^1 \\ t^0 \end{bmatrix}$$

Thus, $P \cup Q$ are linearly dependent iff $Res_t(p,q)$ is singular. $\square$

We will illustrate a few application of the above proposition. Consider the case when $p = t^2 - 4t + 3$ and $q = -t^2 - 1$. To check if they share a root, we form the resultant matrix:

$$R = \begin{bmatrix} 1 & -4 & 3 & 0 \\ 0 & 1 & -4 & 3 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

We see that $det(R) = 0$ and that p, q above indeed share the root $t = 1$.

Another interesting case is to check if the general polynomial $f(t) = at^2 + bt + c$ has a double root. This is equivalent to checking if f and f', the derivative of f w.r.t. t have a common root. Clearly $f' = 2at + b$, and thus the resultant is:

$$D = \begin{bmatrix} a & b & c \\ 2a & b & 0 \\ 0 & 2a & b \end{bmatrix}$$

One may check that $det(D) = -a(b^2 - 4ac)$. Since $a \neq 0$, we see that f hasa double root if the discriminant $b^2 - 4ac = 0$. The general **discriminant** of a polynomial f is defined to be $Res_t(f, f')$.

We now apply this technique to convert parametric represnetations of a curve to implicit ones. We do this by an example, the general case will be left to the reader. Recall that $c(t) = (\frac{2t}{1+t^2}, \frac{1-t^2}{1+t^2})$ was the parametrization of the circle. Thus, for an (x, y), it would lie on the circle iff the equations $x = \frac{2t}{1+t^2}$ and $y = \frac{1-t^2}{1+t^2}$ have a common root. Clearing denominators, we have:

$$\begin{array}{rcl} xt^2 - 2t + x & = & 0 \\ (1+y)t^2 + y - 1 & = & 0 \end{array}$$

Forming the resultant of the above equations, we have:

$$S = \begin{bmatrix} x & -2 & x & 0 \\ 0 & x & -2 & x \\ 1+y & 0 & y-1 & 0 \\ 0 & 1+y & 0 & y-1 \end{bmatrix}$$

Expanding $det(S)$ we get $4(x^2 + y^2 - 1)$, which is the required implicit form for the circle.

4 Polynomial Bases

Polynomials will serve as the central object in which functions will be specified. The main advantage being their familiarity, ease in representation and implementation of operations, and finally, the strength of various theorems such as the Weierstrass Approximation theorem.

Our first study will be that of polynomials in one variable, i.e., the space $\mathbb{R}[t]$. Recall that V_d is the $d+1$-dimensional subspace of polynomials of degree d or less. The standard basis for V_d is obviously, $T = \{1, t, t^2, \ldots, t^d\}$. Different bases of V_d have different practical uses. For example, if we were given $f : \mathbb{R} \to \mathbb{R}$, and the values $f(0), f'(0), \ldots, f^d(0)$. For this data, a good approximation to f would be

$$f(t) \approx T(f) = \sum_{i=0}^{d} \frac{f^i(0)}{i!} t^k$$

Thus we see that the basis T is suited for the purpose of approximating f by its derivatives at 0. Of course, if $t = 1$ were to be special, then the basis $T_1 = \{1, t-1, (t-1)^2, \ldots, (t-1)^d\}$ would do the job.

The general interpolation problem is given by (i) a procedure to conduct $d+1$ independent observations of the function f, (ii) the construction of a basis $P = \{p_0, \ldots, p_d\}$ of V_d, which is specially suited to the nature of the observations, and (iii) the observations $o_0(f), \ldots, o_d(f)$ and the interpolant $P(f) = \sum_{i=0}^{d} o_i(f) p_i$. Note that the basis T depends only on (i) and not on the function f to be approximated/interpolated. Also note that the basis T above fits the above paradigm.

Let $\Lambda = \{\lambda_0, \ldots, \lambda_d\}$ be a fixed collection of distinct real/complex numbers. Another important basis of V_d is the **Lagrange** basis, $L(\lambda_0, \ldots, \lambda_d)$. Let (i) above correspond to the observation of f at the point λ_i, and thus $o_i(f) = f(\lambda_i)$. The lagrange basis $L = \{L_0, \ldots, L_d\}$ is such that $L(f)(t) = \sum_{i=0}^{d} f(\lambda_i) L_i(t)$ is a degree-d polynomial such that $L(f)(\lambda_i) = f(\lambda_i)$. Just this requirment forces us to have $L_i(\lambda_j) = \delta_{ij}$ if $i \neq j$ and 1 otherwise. One may solve for L_i as follows: let $L_i = \sum_j a_{ij} t^j$. Thus $\delta_{ij} = L_i(\lambda_j)$ requires $\delta_{ij} = \sum_j a_{ij} \lambda_i^j$. This is easily expressed in matrix form as:

$$\begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ & \vdots & \vdots & \\ 0 & \ldots & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & \ldots & a_{0d} \\ a_{10} & a_{11} & \ldots & a_{1d} \\ & \vdots & \vdots & \\ a_{d0} & a_{d1} & \ldots & a_{dd} \end{bmatrix} \begin{bmatrix} \lambda_0^0 & \lambda_1^0 & \ldots & \lambda_d^0 \\ \lambda_0^1 & \lambda_1^1 & \ldots & \lambda_d^1 \\ & \vdots & \vdots & \\ \lambda_0^d & \lambda_1^d & \ldots & \lambda_d^d \end{bmatrix}$$

Thus, we see that the lagrange basis may be expressed in terms of the standard basis by inverting the **Van der Monde** matrix $\Lambda = (\lambda_j^i)$.

Both, the standard basis and the lagrange basis are special cases of the Hermite basis: We are given distinct points $\alpha_0, \ldots, \alpha_k$, and positive integers $m_0, \ldots, m_k$ such that $\sum_i m_i = d + 1$. The observations are the first m_i-th derivatives of f at the point α_i. Thus, given $f(\alpha_0), f'(\alpha_0)$ and so on upto $f^{m_0-1}(\alpha_0)$, and so on for other α's, there is a unique polynomial $H(f)(t)$ matching this data. The basis for this is the **Hermite** basis for this data. Note that when $\alpha_0 = 0$, and $m_1 = d+1$, then we get the standard basis, while on the other end, if we have $k = d$ and $m_i = 1$ for all i, then we have the lagrange basis.

An even more general paradigm is the following: Let $o_0, \ldots, o_n$ be *linear observations* on the space V_n. In other words, $o_i : V_n \to \mathbb{R}$ are functions such that $o_i(\alpha p + q) = \alpha o_i(p) + o_i(q)$, where $\alpha \in \mathbb{R}/\mathbb{C}$, and $p, q \in V_n$. Examples of such observations are of course, $p \to p(0)$ or $p \to p'(0)$. Other interesting examples are $p \to \int_0^1 p(t)dt$, or $\int_a^b t \cdot p(t)dt$. Note that all these 'integral' observations are linear.

We would like to find polynomials $p_0, \ldots, p_n$ such that $o_i(p_j) = 0$ is $i \neq j$ and 1 when $i = j$, or simply $o_i(p_j) = \delta_{ij}$, the kronecker delta. We choose a standard basis, say $Q = \{q_0, \ldots, q_n\}$, and express $p_i = \sum_j a_{ij} q_j$, or in matrix notation $P(t) = AQ(t)$. Applying the observation o_i, we have:

$$\begin{bmatrix} o_i(p_0) \\ \vdots \\ o_i(p_n) \end{bmatrix} = \begin{bmatrix} \delta_{i,0} \\ \vdots \\ \delta_{i,n} \end{bmatrix} = \begin{bmatrix} a_{00} & \ldots & a_{0n} \\ \vdots & & \vdots \\ a_{n0} & \ldots & a_{nn} \end{bmatrix} \begin{bmatrix} o_i(q_0) \\ \vdots \\ o_i(q_n) \end{bmatrix}$$

Combining this for all i, we write $\Lambda = (o_j(q_i))$ we have:

$$I = A\Lambda$$

Thus, there is such a basis iff Λ is invertible.

Coming back, let us assume that we have a function $f : [0, 1] \to \mathbb{R}$. Our first task is to approximate f in the prescribed interval within a given $\epsilon > 0$. We have the famous:

Theorem 4.1 The Weierstrass Approximation Theorem: *Given any continuous function $f : [0, 1] \to \mathbb{R}$, and an $\epsilon > 0$, we have a polynomial p such that for all $t \in [0, 1]$, we have $|f(t) - p(t)| < \epsilon$.*

One may represent this pictorially in the figure below: The function f appears in bold. The ϵ-envelope of the function f is also shown and the graph of the desired approximating polynomial p.

The basic question is about how to implement the WAT. Surely, the first attempt would be to use the lagrange basis. In other words, let us select $\Lambda_n = \{\frac{0}{n}, \frac{1,n}{,} \ldots, \frac{n-1}{n}, \frac{n}{n}\}$, the $n + 1$ points equally spaced in the interval $[0, 1]$. Given the function f, let $y_i = f(\frac{i}{n})$, and form the lagrange interpolant $L_n(f) = \sum_i y_i L_i(t)$. We note that, while $L_n(f)$ matches f exactly on the points Λ_n, it strays very far from f in the interim. One suggestion then would be to increase n, but we see that the situation actually gets worse. As n increases, the n-th interpolant may actually worsen, and stay out of the ϵ-envelope for most points of the interval.

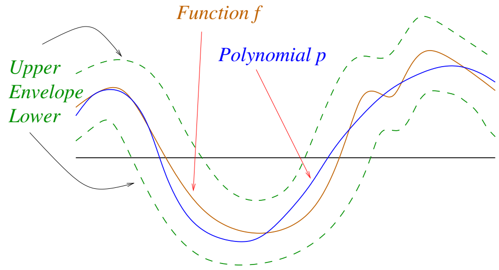

Figure 5: The Weierstrass Approximation Theorem (WAT).

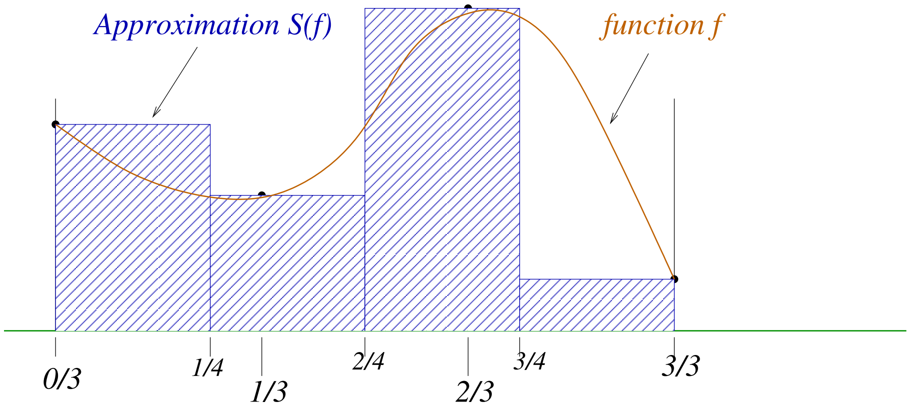

Figure 6: The Sohoni Approximation Theorem (SAT)!

Here is a simple basis, which though not polynomial, atleast approximates continuous. For a chosen n, let $p_i = \frac{i}{n}$, and let I_i be the interval $[\frac{i}{n+1}, \frac{i+1}{n+1}]$. Let χ_i be the function which is 0 outside I_i and 1 inside. In other words, $\chi_i(t) = 1$ if $t \in I_i$. Consider $s = \{\chi_0, \ldots, \chi_n\}$ as a collection of $n+1$ functions on the interval $[0,1]$. For a given function $f : [0,1] \to \mathbb{R}$, we observe the function f at the points $f(p_i)$ and for the function $S^n(f) = \sum_{i=0}^{n} f(p_i)\chi_i$. The case $n = 3$ is indicated in the Figure 6. Note that $p_i \in I_i$ and thus $S^n(f)$ agrees with f at some point in every interval I_i.

We have the famous **Sohoni Approximation Theorem**:

Theorem 4.2 *For any continuous $f : [0,1] \to \mathbb{R}$, and an $\epsilon > 0$, there is an N such that for all $n > N$, $S^n(f)$ approximates f to within ϵ. In other words, $|S^n(f)(t) - f(t)| < \epsilon$ for all $t \in [0,1]$.*

The proof of this is straight-forward and is omitted here.

Of course, this basis is of very limited use since it is not polynomial, not continuous and so on. But the point was that each χ_i had support only within I_i and thus a scalar multiple of χ_i could be chosen to approximate f in I_i. The larger the n, the finer is the approximation. Thus, if we could find **polynomials** with similar properties, we would have an implementable version of WAT. This need is satisfied precisely by the **Bernstein basis**.

Let $B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$, regarded as a function on the interval $[0,1]$. Let $\mathcal{B}^n = \{B_0^n, \ldots, B_n^n\}$: this collection is called the **Bernstein** basis. Note that each $B_i^n(t)$ is a polynomial of degree atmost n. We now state some properties of the collection $\mathcal{B}^n$.

Lemma 4.3 *1. $\mathcal{B}^n$ is a basis of V_n.*

2. $B_i^n(t) \geq 0$ for all $t \in [0,1]$. Furthermore, $\int_0^1 B_i^n(t)dt = \frac{1}{n+1}$.

3. $\frac{dB_i^n(t)}{dt} = n(B_{i-1}^{n-1}(t) - B_i^{n-1}(t))$.

4. The maximum value of $B_i^n(t)$ in the interval $[0,1]$ occurs at the point $\frac{i}{n}$.

Proof:

1. The first part is easy: If we express $B_i^n(t)$ in terms of the basis $\{1, t, t^2, \ldots, t^n\}$ we see that

$$B_i^n(t) = \binom{n}{i} t^i + \textit{higher terms}$$

   Thus the matrix expressing $\mathcal{B}^n$ in terms of the standard basis is upper triangular with non-zero entries on the diagonal, and therefore is invertible. This proves that $\mathcal{B}^n$ is a basis.

2. Clearly, for $t \in [0,1]$, both t and $(1-t)$ are non-negative. The second part follows a simple induction:

$$\begin{aligned}
\int_0^1 B_i^n(t)dt &= \int_0^1 \binom{n}{i} t^i (1-t)^{n-i} dt \\
&= \binom{n}{i} \frac{t^{i+1}}{i+1}(1-t)^{n-i} + \binom{n}{i} \frac{n-i}{i+1} \int_0^1 t^{i+1}(1-t)^{n-i-1} dt \\
&= \frac{1}{n+1} B_{i+1}^{n+1}(t)|_0^1 + \int_0^1 B_{i+1}^n(t)dt
\end{aligned}$$

   This is easily settled to get the result.

3. This is easily shown, and we move to (4). Solving for $\frac{dB_i^n(t)}{dt} = 0$, gives us

$$\begin{aligned}
0 &= B_{i-1}^{n-1}(t) - B_i^{n-1}(t) \\
&= \frac{1-t}{n-i} - \frac{t}{i} \\
&= i - nt
\end{aligned}$$

   The result follows. $\square$

We thus see that the bernstein basis has properties similar to the basis S^n. In fact, we have the famous theorem of Bernstein: For a fixed continuous $f : [0,1] \to \mathbb{R}$, and an n, let $B^n(f)(t) = \sum_{i=0}^n f(\frac{i}{n}) B_i^n(t)$. Note that $B^n(f)(t)$ is a polynomial of degree n and is obtained as a linear combination of the bernstein polynomials. Furthermore, the coefficient of $B_i^n(t)$ is precisely the observation $f(\frac{i}{n})$, as in the SAT. We have:

Theorem 4.4 *For any continuous $f : [0,1] \to \mathbb{R}$, and an $\epsilon > 0$, there is an N such that for all $n > N$, $B^n(f)$ approximates f to within ϵ. In other words, $|B^n(f)(t) - f(t)| < \epsilon$ for all $t \in [0,1]$.*

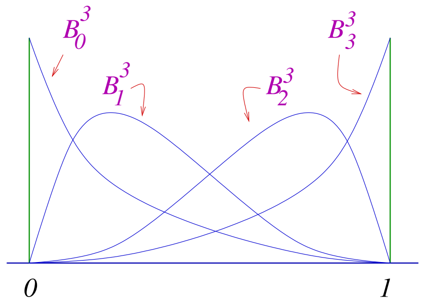

Figure 7: The Bernstein Polynomials for $n = 3$

The proof of this theorem is rather intricate, and we shall skip it. However, we note that

$$B^n(f)(0) = \sum_{i=0}^{n} f(\frac{i}{n}) B_i^n(0)$$

Since $B_i^n(0) = 0$ unless $i = 0$, and then it is 1, we have that $B^n(f)(0) = f(0)$. Similarly, we have $B^n(f)(1) = f(1)$, and thus the approximation $B^n(f)$ **interpolates** f at 0 and 1. However, this **not** the case at the intermediate points; in this it differs from the lagrange basis.

Interestingly, we also note that

$$(B^n(f))'(0) = \sum_i f(\frac{i}{n}) \cdot n \cdot (B_{i-1}^{n-1}(0) - B_i^{n-1}(0))$$

The only terms which matter are $i = 0$ and $i = 1$, and we have:

$$(B^n(f))'(0) = n(f(\frac{1}{n}) - f(0)) = \frac{f(1/n) - f(0)}{1/n}$$

5 Bezier Curves

We now move to the problem of constructing edges in $\mathbb{R}^2$ or $\mathbb{R}^3$. Bezier was a frenchman working in a car design shop, and was one of the first persons to use the Bernstein basis for edge and face design. the basic object in the design and analysis of bezier curves is the control-point sequence, which is also known as the **control polygon**.

Suppose that we have a curve C, as shown in figure 8, which is available to us as drawn by an artist. Our task is to parametrize this curve C. The first step is to fix the start vertex and the end-vertex. Having done this, let us assume that we desire to parametrize it with the interval $[0, 1]$. Next, we *imagine* a cyclist travelling along the curve, starting at $t = 0$ from the start vertex, and arriving at the end vertex at time $t = 1$. As the cyclist moves, we construct the functions $x(t), y(t) : [0, 1] \to \mathbb{R}$, which read out the x and the y-coordinate of the cyclist with time. Having plotted these associated functions $x(t), y(t)$, we may now use the bernstein basis for their approximation.

For example, we choose an n ($n = 3$ in figure 8), and proceed to construct the vectors $p_i = [x(i/n), y(i/n)]$ for $i = 0, \ldots, n$. The bezier curve $P^n(C)(t)$ is defined as $\sum_{i=0}^{n} p_i B_i^n(t)$. Note that $P^n(C)$ evaluates to a linear combination of the vectors $p_0, \ldots, p_n$, whose coefficients chane with t.

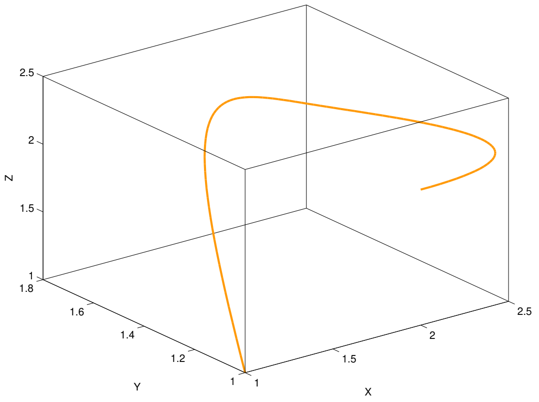

Figure 8: A curve C and its approximate parametrization.

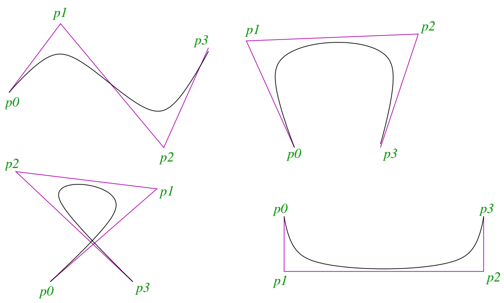

Figure 9: Examples of Control Polygons and their curves, for $n = 3$

Indeed, the function $P^n(C)$ is obtained by the *simulteneous* approximation of $x(t)$ and $y(t)$ by the bernstein recipe above.

It is possible that $P^n(C)$ does fails to be close enough to C and n need to be hiked up. See, in the figure, that $P^6(C)$ approximates C better than $P^3(C)$. In any case, note that since $B^n(x)(0) = x(0)$ and $B^n(y)(0) = y(0)$, we have $P^n(C)(0) = C(0)$, the start vertex, and similarly at $t = 1$.

The vector coefficients, $p_0, \ldots, p_n$, in that order is called the **control** polygon. The polygon determines the approximating curve P^n completely. While we have been chosing the control points to lie on the curve to be approximated, that is not really required. Frequently, the control points may be chosen outside the curve, and yet $P^n(t)$ hugs C better.

Examples of a few control polygons and the approximating curves are shown in figure 9.

Thus, we have arrived at the bezier representation of a curve, which is parametrized by the degree n, and a sequence $P = [p_0, \ldots, p_n]$ called the control polygon. The curve may be equivalently stored as the tuple (n, P), which is a fairly compact representation of the parametrization. The evaluate of $C(t)$ may be done by computing the coefficients $B_i^n(t)$ and forming the linear combination $C(t) = \sum_i p_i B_i^n(t)$. We have also seen that the above curve interpolates the first and the last control point. The derivative $C'(t)$ is also easily caluclated:

$$\begin{aligned} C'(t) &= \sum_i p_i \frac{dB_i^n(t)}{dt} \\ &= \sum_i p_i \cdot n \cdot (B_{i-1}^{n-1}(t) - B_i^{n-1}(t)) \\ &= \sum_{i=0}^{n-1} n \cdot (p_{i+1} - p_i) B_i^{n-1}(t) \end{aligned}$$

Thus, $C'(t)$ can be easily written in Bezier form as $(n-1, P')$ where, $P' = [p_1 - p_0, p_2 - p_1, \ldots, p_n - p_{n-1}]$. In general, the k-th derivative is also easily expressed as

$$C^k(t) = \frac{n!}{(n-k)!} \sum_{i=0}^{n-k} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} p_{i+j} B_i^{n-k}(t) \tag{1}$$

We also note here that geometric transformations on the curve are easily implemented by effecting the same transformation on the control points. Let $P = [p_0, \ldots, p_n]$ be the $3 \times (n+1)$ matrix representing the control points of a space curve. The general point is given by the equation:

$$C(t) = [p_0, \ldots p_n] \begin{bmatrix} B_0^n(t) \\ \vdots \\ B_n^n(t) \end{bmatrix}$$

we abbreviate this to $C(t) = PB(t)$. Thus is A is a linear transformation of the curve, say $C(t) \rightarrow AC(t)$, where A is a 3×3-matrix, then $AC(t) = APB(t)$. Thus, the operator A operates on P from the left and $B(t)$ from the right, and they commute. Thus whether we evaluate the curve first and then operate on it, or operate on the control points and then evaluate the curve, we get the same answer. Thus, to store the transformed curve, we may as well store just the transformed control points. Another example of this is **translation**. say, we wish to translate the curve by a fixed vector $v \in \mathbb{R}^3$. In other words, let $D(t) = C(t) + v$. We put $Q = [p_0 + v, \ldots, p_n + v]$, and observe that

$$QB(t) = PB(t) + [v, v, \ldots, v]B(t)$$

Now, note that

$$\sum_i B_i^n(t) = \sum_i \binom{n}{i} t^i (1-t)^{n-i} = (1 - t + t)^n = 1$$

Thus,

$$\sum_i v \cdot B_i^n(t) = v \cdot \sum_i \binom{n}{i} t^i = v$$

Thus $QB(t) = C(t) + v$, and $D(t)$ is thus represented merely by translating the control points by v.

6 Degree Elevation and Subdivision

Frequently, it is required to represent a given parametric curve $C = (n, P)$ by a higher degree curve. This need arises when the current degree of representation is deemed inadequate. Since an elevation

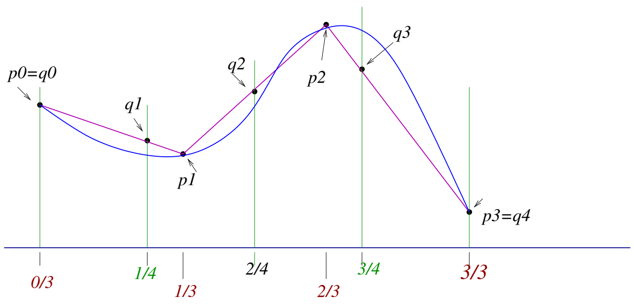

Figure 10: Degree elevation from $n = 3$ to $n = 4$.

of the degree by one, gives an additional control point, it may be desirable to elevate the degree of C. In other words, we would like to compute the control polygon $Q = [q_0, \ldots, q_{n+1}]$ so that $C = [n, P] = [n+1, Q]$.

For the standard basis, viz., $T^n = \{1, t, \ldots, t^n\}$, we see that $T^n \subseteq T^{n+1}$, and thus a polynomial expressed in t^n is automatically expressed in the higher degree basis T^{n+1}. In other words, in the standard basis $Q = [p_0, \ldots, p_n, 0]$. This is not the case for the bernstein basis, since $B_i^n \notin \mathcal{B}^{n+1}$. However, we see that:

$$\begin{aligned}
B_i^n(t) &= (1-t+t) \cdot B_i^n(t) \\
&= ((1-t)+t)\binom{n}{i} t^i (1-t)^{n-i} \\
&= \binom{n}{i} t^i (1-t)^{n-i+1} + \binom{n}{i} t^{i+1} (1-t)^{n-i} \\
&= \frac{n-i+1}{n+1} B_i^{n+1}(t) + \frac{i+1}{n+1} B_{i+1}^{n+1}(t)
\end{aligned}$$

Thus we have that

$$\begin{aligned}
\sum_{i=0}^{n} p_i B_i^n &= \sum_{i=0}^{n} p_i \left(\frac{n-i+1}{n+1} B_i^{n+1}(t) + \frac{i+1}{n+1} B_{i+1}^{n+1}(t)\right) \\
&= \sum_{i=0}^{n+1} \left(\frac{i}{n+1} p_{i-1} + \frac{n-i+1}{n+1} p_{i+1}\right) B_i^{n+1}(t)
\end{aligned}$$

Thus we see that $q_i = \frac{i}{n+1} p_{i-1} + \frac{n-i+1}{n+1} p_{i+1}$. This is pictorially seen much better in figure 10. The new control ponts are obtained by **interpolating the control polygon** at the points $\{\frac{0}{n+1}, \frac{1}{n+1}, \ldots, \frac{n+1}{n+1}\}$, in that order.

At this point, we turn to the implementation of the evaluation of a point on a typical bezier curve for a given parameter value, say t. The naive algorithm would compute each $B_i^n(t)$, and then form the linear combination $P(t) = \sum_i B_i^n(t) p_i$. A clever way of organising the computation is given by the de-Casteljeu algorithm. This scheme, besides being efficient and numerically stable, is also an important geometric formalism. The scheme is best illustrated by the following Figure 11. The scheme creates the variables `p[i,j]`, with $i \leq j$. The variables `p[i,i]` are instantiated to the control point p_i. The main iteration is the following:

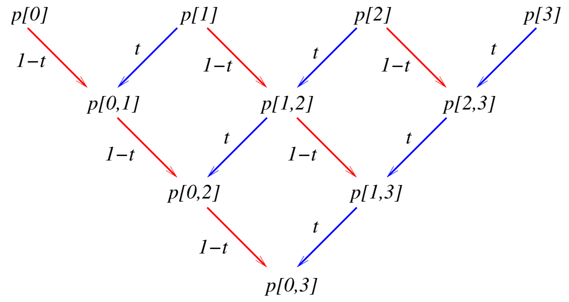

Figure 11: The de Casteljeu scheme for $n = 3$

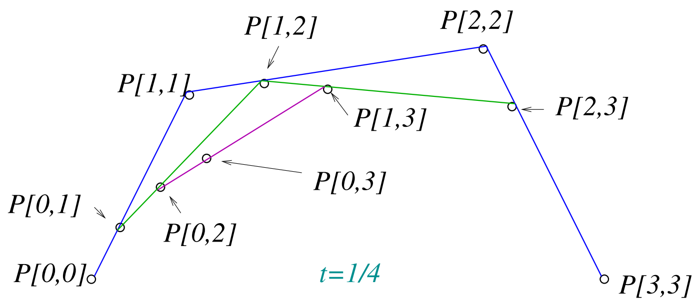

Figure 12: The geometry of the de Casteljeu scheme for $n = 3$

```
P[i,j+1]=(1-t)P[i,j]+tP[i+1,j+1].
```

It is easy to see that the iteration actually works, and this is best seen from the schema in Figure 11. In the figure, the arrows have been marked with the multipliers t or $(1-t)$. Note that every right arrow has a $(1-t)$ and every left arrow, a t. The coefficient of `p[i,i]` in `p[0,n]` is parametrized by the number of paths from `p[i,i]` to `p[0,n]`. Since, any such path will have exactly i left-arrows and $(n-i)$ right-arrows, we see that there are exactly $\binom{n}{i}$ such paths, and each has a multiplier $t^i(1-t)^{n-i}$. Thus `p[i,i]` appears with the coefficient $B_i^n(t)$, as required.

The geometric significance of the de Casteljeu algorithm is also nice, and is illustrated in Figure 12. In that figure, we show the the points `p[i,j]` for $n = 3$. Every succesive point is obtained as a **convex** combination of points in the previous layer. The final point `p[0,3]` is taken to lie on the curve. This also proves that the final point is a convex combination of convex combinations etc., of the control points, and thus sits inside the convex hull of the control polygon.

Another point which we note now, and we will use later is that, is clear from the schema of the de Casteljeu algorithm. For a fixed t, the points `p[0,j]` may be expressed as

$$p[0,j] = \sum_{i=0}^{j} p_i B_i^j(t)$$

The second operation which we wish to analyse is called **subdivision**. Let $e = ([0,1], f)$ be an edge, with $f : \mathbb{R} \to \mathbb{R}^3$. For a $0 < c < 1$, we wish to construct the edge e' which corresponds to the segment $[0,c]$ of e, *however*, we wish to re-parametrize it as from $[0,1]$. In other words, we wish to compute g such that $g(t) = f(ct)$ for $t \in [0,1]$.

Again, in the standard basis, this is easy: if $f(t) = \sum_i p_i t^i$, then $f(ct) = \sum_i (c^i p_i) t^i$. Thus, with $Q = [c^0 p_0, \ldots, c^n p_n]$, we have that Q parametrizes the 'subdivided' curve g in the standard basis.

We compute this g when f is given as a bezier curve, $(n, P = [p_0, \ldots, p_n])$. In other words, let $f(t) = \sum_i B_i^n(t) p_i$, we find $Q = [q_0, \ldots, q_n]$ such that $\sum_i q_i B_i^n(t) = f(ct)$.

The basic calculation is to express $B_i^n(ct)$ in terms of $\{B_j^n(t) | j = 0, \ldots, n\}$ and we begin in earnest:

$$\begin{aligned}
\binom{n}{i}(ct)^i(1-ct)^{n-i} &= \binom{n}{i} c^i t^i \{(1-c)t + (1-t)\}^{n-i} \\
&= \sum_{k=0}^{n-i} \binom{n}{i}\binom{i}{k} c^i (1-c)^k t^{i+k} (1-t)^{n-i-k} \\
&= \sum_{k=0}^{n-i} \binom{n}{i+k} t^{i+k}(1-t)^{n-i-k} \binom{i+k}{i} c^i (1-c)^k \\
&= \sum_{k=0}^{n-i} B_{i+k}^n(t) B_i^{i+k}(c)
\end{aligned}$$

With this small calculation behind us, we apply the usual trick of interchanging summations, and see that:

$$\begin{aligned}
\sum_i p_i B_i^n(ct) = &= \sum_i p_i \sum_{k=0}^{n-i} B_{i+k}^n(t) B_i^{i+k}(c) \\
&= \sum_i p_i \sum_{i \le j \le n} B_j^n(t) B_i^j(c) \\
&= \sum_{j=0}^{n} B_j^n(t) \sum_{i=0}^{j} p_i B_i^j(c)
\end{aligned}$$

Now, we recall that $\sum_{i=0}^{j} p_i B_i^j(c)$ is precisely `p[0,j]`, a quantity which is computed during the computation of the point $f(c) = P(c)$. Thus, for the curve $g(t) = f(ct)$, the control point sequence is $[p[0,0](c), p[0,1](c), \ldots, p[0,n](c)]$. This is pictorailly illustrated in Figure 13 below. Another important point is that while the control point sequence $Q_L = [p[0,0](c), p[0,1](c), \ldots, p[0,n](c)]$ parametrizes the curve C_L for the interval $[0,c]$, the sequence from the right, i.e., $Q_R = [P[0,n](c), P[1,n](c), \ldots, P[n,n](c)]$ parametrizes the curve C_R in the interval $[c,1]$. Since both C_L and C_R are part of the same curve C, it must be that all derivatives of C_L at it end-point equal those of C_R at its start point. Thus in the de Casteljeu triangle, if we start with a sequence of control points and fix a c, then the the left hand side of the triangle and the right hand side of the triangle correspond to control points for curves all of whose derivatives match.

In other words, we have that $C_L^k(1) = C_R^k(0)$ for all k, whence, following Eqn. (1), we have the relation:

$$\sum_{j=0}^{k}(-1)^{k-j}\binom{k}{j} p[n-k+j, n](c) = \sum_{j=0}^{k}(-1)^{k-j}\binom{k}{j} p[0,j](c) \qquad (2)$$

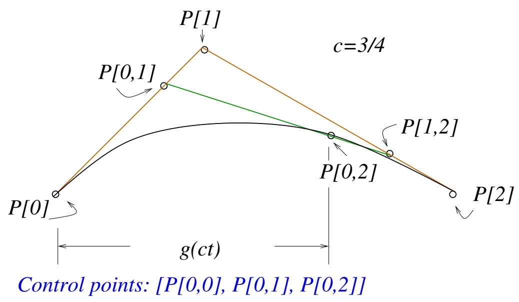

Figure 13: The Subdivision Scheme

7 B-Splines

In this section we will develop the theory of B-Splines, or simply splines. These structures are generalizations of bernstein/bezier systems, and overcome some of the drawbacks of the bezier scheme. To name a few, if we wish to design a long curve with many contours, this would require many control points, thus raising the degree of the bezier curve. The rise in the degree affects the evaluation time (by the de Casteljeu algorithm) quadratically, which is expensive. Another irritation is that local modifications of a few control points changes the curve everywhare. This is because every bernstein polynomial is non-zero on the open interval $(0, 1)$, An option is to split the curve into many parts and design them separately. However, in this case, the derivatives have to be matched at the end-points and this has to be done rather delicately.

7.1 B-Spline basics.

Splines are a device to the above systematically. Essentially, a spline is a continuous piece-wise polynomial function with clearly specified cross-derivative matching. Let us fix the degree to say, n. Let us have reals numbers $a_1 < a_2 < \ldots < a_k$ and numbers $m_2, \ldots, m_{k-1}$ with each $m_i \leq n$. For a polynomial p, we refer to its j-th derivative as p^j.

Definition 7.1 *A spline is a is a collection $(p_1, \ldots, p_{k-1})$ of polynomials of degree bounded by n such that for all $i = 2, \ldots, k-1$, we have $p_{i-1}(a_i) = p_i(a_i)$, $p^1_{i-1}(a_i) = p^1_i(a_i)$, and so on till $p^{n-m_i}_{i-1}(a_i) = p^{n-m_i}_i(a_i)$. the condition may be succintly expressed as:*

$$p^j_{i-1}(a_i) = p_i(a_i) \;\; for\; i = 2, \ldots, k-1 \;\; and\;\; j = 0, 1, \ldots, n - m_i$$

In other words, we may define a function $f : [a_1, a_k] \to \mathbb{R}$ such that $f(t) = p_i(t)$ when $t \in [a_i, a_{i+1}]$. The continuity requirements above ensure that f is continuous, and differentiable $(n-m_i)$-times at a_i.

The data $(a_1 < \ldots < a_k)$, and $m_1, \ldots, m_{k-1}$ is usually specified as a **knot vector** which we describe now. For a fixed degree n, a knot vector is a sequence of reals $\overline{\alpha} = (\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_m)$ with the following properties: (i) $\alpha_1 = \alpha_2 = \ldots = \alpha_{n-1}$, (ii) $\alpha_{m-n+1} = \alpha_{m-n+2} = \ldots = \alpha_m$, and (iii), $\alpha_i < \alpha_{i+n-1}$ for $i = 1, \ldots, m-n$, or in other words, other than the n duplications in the beginning and the end, there is no sequence of $n+1$ duplicates. the representation of the data $(a_1 < a_2 \ldots < a_k)$ and $m_1, \ldots, m_{k-1}$ is the knot vector:

$$(a_1, \ldots, a_1, a_2, \ldots, a_{k-1}, a_{k-1}, \ldots, a_k, \ldots, a_k)$$

Here a_1 and a_k occur n times, while every other a_i occurs m_{i-1} times. This may also be written as $a_1^n a_2^{m_2} \ldots a_{k-1}^{m_{k-1}} a_k^n$. Note that $\sum_{i=2}^{k-1} m_i + 2n = m$, the length of the knot vector.

Given a knot vector $\overline{\alpha}$, we define the space $V(\overline{\alpha})$ as the space of all spline function for the know vector $\overline{\alpha}$. It is clear that this a vector space of functions, since if $f_1, f_2 \in V(\overline{\alpha})$, then $f_1 + f_2$ will have the same continuity properties that f_1 and f_2 share, and thus will belong to $V(\overline{\alpha})$.

Let us compute the dimension of this space. Clearly, since every element $f \in V(\overline{\alpha})$ is fiven by a sequence of $k-1$ polynomials, each of degree n, the maximum dimension of this vector space is $(k-1)(n+1)$. However, the continuity equations are many, indeed, for each a_i, $i = 2, \ldots, k-1$, there are $n - m_i + 1$ of them. Thus we have the net dimension as:

$$(k-1)(n+1) - \sum_{i=2}^{k-1}(n - m_i + 1) = 1 + n + \sum_{i=2}^{k-1} m_i$$

In terms of m, we have $dim(V(\overline{\alpha})) = m - n + 1$.

As an example, we see that the knot vector $\overline{\beta} = 0^n 1^n$ (of length $2n$) corresponds to the familiar space V_n of polynomials of degree n on $[0,1]$. There are no side conditions, since $k = 2$. Thus $dim(V(\overline{\beta})) = m - n + 1 = n + 1$.

Just as with the bernstein polynomials, the numbers $\{0/n, 1/n, \ldots, n/n\}$ were special, for they corresponded to points in $[0,1]$ where observations $\{p_0, \ldots, p_n\}$ could be made. The number of these observations equalled the dimension $n+1$ of V_n. Furthermore, the bernstein basis B_i^n was such that these observations served as the coefficients of the approximating polynomial $B^n(P)$. A similar situation prevails for an arbitrary knot vector. The special set of points $\Xi = \{\xi_1, \ldots, \xi_{m-n+1}\}$ are called the **Greville abscissa**, and there are exactly $m - n + 1$ of them. There are the **spline** basis functions $N_i(\overline{\alpha})(t)$ for the space $V(\overline{\alpha})$.

The ξ_i are easily defined:

$$\xi_i = \frac{\alpha_i + \alpha_2 + \ldots + \alpha_{i+n-1}}{n}$$

Thus the greville abscissa are "moving averages" of n consecutive knots. Note that since $\alpha_1 = \alpha_2 = \ldots = \alpha_n$, and similarly for the last knot, we have $\xi_1 = \alpha_1$, and $\xi_{m-n+1} = \alpha_{m-n+1}$. Also observe that for $\overline{\beta} = 0^n 1^n$, the moving averages are $(0 + \ldots 0 + 1 + \ldots + 1)/n$, and thus $\Xi = \{0/n, \ldots, n/n\}$. as in the bezier case, there is a sequence of observations $P = (p_1, \ldots, p_{m-n+1})$ called the **control polygon** which form the schema for any element of $V(\overline{\alpha})$. See figure 14 for an example. The actual spline function given by the coefficients $P = [p_1, \ldots, p_{m_n+1}]$ is given by:

$$N(P) = \sum_{i=0}^{m-n+1} p_i N_i(\overline{\alpha})(t)$$

The basis functions $N_i(\overline{\alpha})$ are almost never used, but an evaluation strategy for any B-spline function is given as an algorithm, very similar to de Casteljeu. Given the control polygon P, and the parameter value $t \in [\alpha_1, \alpha_m]$, there is a 'triangular' scheme to evaluate $P(t)$, and we shall elaborate that in the next subsection. That scheme depends on the basic subroutine of **knot insertion**, which we now describe.

Given two knot vectors $\overline{\alpha}$ and $\overline{\alpha}'$, we say that $\overline{\alpha}'$ is obtained by inserting β into $\overline{\alpha}$, if the set $\{\alpha_1, \ldots, \alpha_m\} \cup \{\beta\}$ arranged in ascending order is the knot vector $\overline{\alpha}'$. In other words, there is an r such that:

$$\overline{\alpha}' = \alpha_1, \ldots, \alpha_{r-1}, \beta, \alpha_r, \ldots, \alpha_m$$

such that $\beta < \alpha_r$. let Ξ' be the greville absiccas of $\overline{\alpha}'$. Note that for $i = 1, \ldots, r-n$, we have that $\xi_i' = \xi_i$, and that for $j = r+1, \ldots, m-n+2$, we have $\xi_j' = \xi_{j-1}$. Furthermore, we see that for

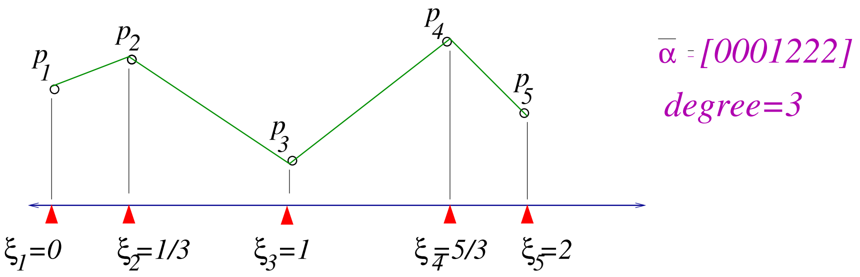

Figure 14: A degree 3 B-spline control polygon and Greville abscissa.

$j = r-n+1, \ldots, r$, we have:

$$\xi_{j-1} = \frac{\alpha_{j-1} + \alpha_j + \ldots + \alpha_{j+n-2}}{n} \leq= \frac{\alpha_j + \ldots + \alpha_{r-1} + \beta + \alpha_r + \ldots + \alpha_{j+n-2}}{n} = \xi'_j$$

$$\xi'_j = \frac{\alpha_j + \ldots + \alpha_{r-1} + \beta + \alpha_r + \ldots + \alpha_{j+n-2}}{n} \leq \frac{\alpha_j + \alpha_{j+1} + \ldots + \alpha_{j+n-1}}{n} = \xi_j$$

We further have:

$$\xi'_j = \frac{\alpha_{j+n-1} - \beta}{\alpha_{j+n-1} - \alpha_{j-1}} \xi_{j-1} + \frac{\beta - \alpha_{j-1}}{\alpha_{j+n-1} - \alpha_{j-1}} \xi_j \tag{3}$$

Thus the new greville abscissa ξ'_j are such that for all j. we have (i) $\xi_{j-1} \leq \xi'_j \leq \xi_j$, and thus (ii) ξ'_j is uniquely expressible as a **convex combination** of $\xi_{j-1} + \xi_j$ (i.e., as a non-negative linear combination with sum 1).

Clearly, $V(\overline{\alpha}) \subseteq V(\overline{\alpha}')$ since splines in $V(\overline{\alpha}')$ have the same continuity properties as in $V(\overline{\alpha})$ with the exception that at β, they may be **less** continuous than in $V(\overline{\alpha})$. Let $N_1(t), \ldots, N_{m-n+1}(t)$ be a basis for $V(\overline{\alpha})$ and let $M_1(t), \ldots, M_{m-n+2}(t)$ be one for $V(\overline{\alpha}')$. Since $V(\overline{\alpha}) \subseteq V(\overline{\alpha}')$, we must have that for any control polygon $P = [p_1, \ldots, p_{m+n-1}]$, there must be a control polygon $Q = [q_0, \ldots, q_{m-n+2}]$ such that:

$$\sum_{i=1}^{m-n+1} p_i N_i(t) = \sum_{i=1}^{m-n+2} q_i M_i(t)$$

The obvious question is: how is Q to be computed from P. Note that we still havent defined $N_i(t)$ or $M_j(t)$. However, these functions are such that Eq. (3) serves as the answer. In other words:

$$\begin{aligned} q_j &= p_j \ \textit{for}\ j = 1, \ldots, r-n \\ q_j &= \frac{\alpha_{j+n-1} - \beta}{\alpha_{j+n-1} - \alpha_{j-1}} p_{j-1} + \frac{\beta - \alpha_{j-1}}{\alpha_{j+n-1} - \alpha_{j-1}} p_j \\ q_j &= p_{j-1} \ \textit{for}\ j = r+1, \ldots, m-n+2 \end{aligned} \tag{4}$$

This is illustrated twice in Figure 15, once by inserting $\beta = 1$ in the example of Figure 14, and again by inserting 0.5 in $[0,0,0,1,1,2,2,2]$ to get the final knot vector $[0,0,0,0.5,1,1,2,2,2]$.

An important result is what we call the *sub-sequence theorem*. Let $\overline{\alpha}$ be a knot vector, and let r and s be such that $a = \alpha_{r+i}$ for $i = 0, \ldots, n-1$ and $b = \alpha_{s-i}$, for $i = 0, \ldots, n-1$. Thus the knots

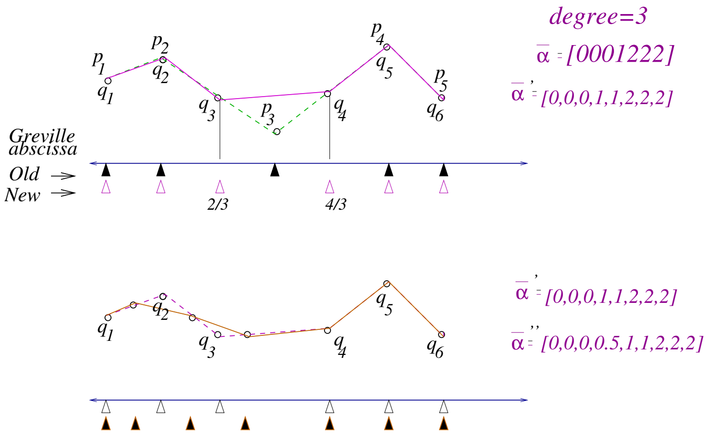

Figure 15: Two examples of knot insertion.

a and b appear n times in the sequence $\overline{\alpha}$. Let $\overline{\gamma} = (\alpha_r, \alpha_{r+1}, \ldots, \alpha_s)$ be the knot vector formed by the subsequencing it from $\overline{\alpha}$. Note that since a and b occur with multiplicity n in $\overline{\alpha}$, we have that $\xi_r = a$ and $\xi_{s-n+1} = b$. We set $P_\gamma = [p_r, p_{r+1}, \ldots, p_{s-n+1}]$, and note that if the length of $\overline{\gamma}$ is $m' = s - r + 1$, then the entries in P_γ are $s - n + 1 - r + 1 = m' - n + 1$. Thus $(\overline{\gamma}, P_\gamma)$ specify a knot vector and a corresponding control polygon. We say that $(\overline{\gamma}, P_\gamma)$ is a **sub-sequence** of $(\overline{\alpha}, P)$. Thus we have two splines $(\overline{\alpha}, P)$ and $(\overline{\gamma}, P_\gamma)$. The following theorem is about their equivalence.

Theorem 7.2 *Let $(\overline{\alpha}, P)$ be a spline and $(\overline{\gamma}, P_\gamma)$ a sub-sequence, as above. Let $a < \beta < b$ be a knot which is inserted in both $(\overline{\alpha}, P)$ and $(\overline{\gamma}, P_\gamma)$ to get the splines $(\overline{\alpha}', Q)$ and $(\overline{\gamma}', Q_\gamma)$. Then $(\overline{\gamma}', Q_\gamma)$ is a sub-sequence of $(\overline{\alpha}', Q)$ beginning at the index r and ending at the index $s+1$ in $\overline{\alpha}'$.*

Proof: Let $\Xi(\overline{\alpha})$ and $\Xi(\overline{\gamma})$ be the greville abscissas for the two knot vectors. First note that if we arrange $\Xi(\overline{\alpha})$ and $\Xi(\overline{\gamma})$ in increasing order, then we see that $\Xi(\overline{\alpha})$ actually appears as a subsequence in $\Xi(\overline{\alpha})$ staring at the index r and ending at the index $s - n + 1$. This property continues to hold for $\Xi(\overline{\alpha}')$ and $\Xi(\overline{\gamma}')$, after the insertion, except with the indices r and $s - n + 2$. Since a, b appear n times in the knot vector $\overline{\alpha}$, we have

$$\begin{aligned} \xi(\overline{\alpha})_r = \xi(\overline{\alpha}')_r = \xi(\overline{\gamma})_1 = \xi(\overline{\gamma}')_1 &= a \\ \xi(\overline{\alpha})_{s-n+1} = \xi(\overline{\alpha}')_{s-n+2} = \xi(\overline{\gamma})_{s-r-n+1} = \xi(\overline{\gamma}')_{s-r-n+2} &= b \end{aligned}$$

Thus the intervening elements of $\Xi(\overline{\gamma}')$ are the *same* linear combinations of $\Xi(\overline{\gamma})$ as those for $\Xi(\overline{\alpha}')$ and $\Xi(\overline{\alpha})$. Now, since the control points are given by the same linear combinations as the greville abscissas, the theorem is proved. $\Box$

7.2 Commutation of Knot Insertion.

Let $\overline{\alpha}$ be a knot vector, and β_1 and β_2 be two real numbers. Let $\overline{\alpha}_e$ be the result of adding β_e to $\overline{\alpha}$. Further, let $\overline{\alpha}_{ed}$ be obtained by adding β_d to $\overline{\alpha}_e$. As vector spaces, we have the following inclusions:

$$\theta_e : V(\overline{\alpha}) \to V(\overline{\alpha}_e) \text{ and } \theta_{ed} : V(\overline{\alpha})_e \to V(\overline{\alpha}_{ed}$$

Clearly, for a spline $f \in V(\overline{\alpha})$, we have:

$$\theta_{ed}(\theta_e(f)) = \theta_{de}(\theta_d(f))$$

However, we must verify that the rule in Eqn (3) also shows the same property. In other words, let $\{M_i^e(t)|i = 1, \ldots, m-n+2\}$ and $\{M_i^{ed}(t)|i = 1, 2, \ldots, m-n+3\}$ be the standard basis for $V(\overline{\alpha}_e)$ and $V(\overline{\alpha}_{ed})$. Further, if $P = [p_1, \ldots, p_{m-n+1}]$ is a control polygon for a function in $V(\overline{\alpha})$, then there are control polygons $Q^e = [q_1^e, \ldots, q_{m-n+2}^e]$ and $Q^{ed} = [q_1^{ed}, q_2^{ed}, \ldots, q_{m-n+3}^{ed}]$. All these control polygons are derived From P via Eqn. (3). Now, it had better be that $Q^{ed} = Q^{de}$, and thus the final coefficients are indeed independent of the order in which the knots β_1 and β_2 were inserted.

We will now verify the above. Let β_1 and β_2 be such that $\alpha_{r-1} \leq \beta_1 < \alpha_r$ and $\alpha_{s-1} \leq \beta_2 < \alpha_s$. here is the rough scheme of the proof: We will first insert β_1 and then β_2, and form the new knot vector $\overline{\alpha}^{12}$. We note that, during the insertion of β_1, we have $\xi_{i-1} \leq \xi_i^1 \leq \xi_i^1$, and thus ξ_i^1 is uniquely expressible as a convex combination of ξ_{i-1} and ξ_i. Next, we see that $\xi_{j-1}^1 \leq \xi_j^{12} \leq \xi_j^1$, and thus ξ_j^{12} is uniquely expressible as a convex combination of ξ_{j-1}^1 and ξ_j^1. Since these in turn, are convex combinations of ξ_{j-2}, ξ_{j-1} and ξ_{j-1}, ξ_j respectively, we see that ξ_j^{12} is a convex combination of ξ_{j-2}, ξ_{j-1} and ξ_j. In a similar way, via the ξ^2's, we have an expression of ξ_j^{21} in terms of ξ^{j-2}, ξ_{j-1} and ξ_j. Also note that, as real numbers $\xi_j^{21} = \xi_j^{12}$, though of course, there may be two distinct ways of expressing the same real number as a convex combination of *three* reals ξ_{j-2}, ξ_{j-1} and ξ_j.

Next, we note that in the final sequence

$$\alpha_1, \ldots, \alpha_{r-1}, \beta_1, \alpha_r, \ldots, \alpha_{s-1}, \beta_2, \alpha_s \ldots, \alpha_m$$

for $j+n-2 \leq s-1$ we see that the insertion of β_2 is immaterial and $\xi_j^{12} = \xi_j^1$, while $\xi_j^2 = \xi_j$. Thus $\xi_j^{12} = \xi_j^{21}$ is the same linear combination of only two ξ's, namely, ξ_{j-1} and ξ_j. Similarly, for $j \geq r+1$, we again see that the insertion of β_1 is irrelevant and $\xi_j^{12} = \xi_j^{21}$ is expressed as the same linear combination of ξ_{j-1} and ξ_{j-2}; this is because the insertion of β_1 though immaterial to the expressions, causes the indices in ξ^{12} to increase by 1.

Thus the real matter is when the ξ_j^{12} straddles both β_1 and β_2. Eqn. (3) tells us:

$$\xi_j^1 = \frac{\alpha_{j+n-1} - \beta_1}{\alpha_{j+n-1} - \alpha_{j-1}} \xi_{j-1} + \frac{\beta_1 - \alpha_{j-1}}{\alpha_{j+n-1} - \alpha_{j-1}} \xi_j \tag{5}$$

Just to keep the notation separate, we call the sequence after inserting β_1 as , $[\delta_1, \ldots, \delta_{m+1}]$. In this notation then,

$$\xi_j^{12} = \frac{\delta_{j+n-1} - \beta_2}{\delta_{j+n-1} - \delta_{j-1}} \xi_{j-1}^1 + \frac{\beta_2 - \delta_{j-1}}{\delta_{j+n-1} - \delta_{j-1}} \xi_j^1 \tag{6}$$

Noting that δ_{j+n-1} occurs after β_1, we have that $\delta_{j+n-1} = \alpha_{j+n-2}$. On the other hand, $\delta_{j-1} = \alpha_{j-1}$. Thus, we have:

$$\xi_j^{12} = \frac{\alpha_{j+n-2} - \beta_2}{\alpha_{j+n-2} - \alpha_{j-1}} \xi_{j-1}^1 + \frac{\beta_2 - \alpha_{j-1}}{\alpha_{j+n-2} - \alpha_{j-1}} \xi_j^1 \tag{7}$$

Paying special attention to the coefficient of of ξ_j we see that:

$$\xi_j^{12} = \frac{(\beta_2 - \alpha_{j-1})(\beta_1 - \alpha_{j-1})}{(\alpha_{j+n-2} - \alpha_{j-1})(\alpha_{j+n-1} - \alpha_{j-1})} \xi_j + \textit{ other terms} \tag{8}$$

We note that this term is *symmetric* with respect to β_1 and β_2, and thus ξ_j^{12} and ξ_j^{21} must have the same coefficient for ξ_j. We can also check this for ξ_{j-2}, which we see as:

$$\xi_j^{12} = \frac{(\alpha_{j+n-2}-\beta_2)(\alpha_{j+n-2}-\beta_1)}{(\alpha_{j+n-2}-\alpha_{j-1})(\alpha_{j+n-2}-\alpha_{j-2})}\xi_{j-2} + \ other\ terms \tag{9}$$

Thus, ξ_j^{12} and ξ_j^{21} have the same coefficient for ξ_j and ξ_{j-2}. Since both are convex combinations of the same three quantities, the third coefficient must also be equal. Thus we have the theorem:

Theorem 7.3 *Let $\overline{\alpha}$ be a knot vector and P be a control polygon for this knot vector. Let β_1 and β_2 be two reals and $\overline{\alpha}^{12}$ be obtained by inserting β_1 first and then β_2 into $\overline{\alpha}$, and let the final control polygon be Q^{12}. Let Q^{21} be that obtained by reversing the order of insertion. Then $Q^{12} = Q^{21}$.*

7.3 The Evaluation and Properties of Splines.

We are now ready to actually evaluate the spline function. This definition is via a modified de Casteljeu procedure, and depends crucially on Theorem 7.3.

So, let $\overline{\alpha}$ be a knot vector, and let $[p_1, \ldots, p_{m-n+1}]$ be the control polygon of a spline of degree n. We wish to evaluate $p(t)$. Here is the procedure:

`Procedure evaluate` $P(t)$ `with` $t \in [\alpha_1, \alpha_m]$.

1. `Let` k `be the number of times` t `appears in` $\overline{\alpha}$. `Insert` t `into` $\overline{\alpha}$ `exactly` $n-k$ `times.`
2. `Let` $P' = [p'_1, \ldots, p'_{m-k+1}]$ `be the new control points. Note that` t `is now a greville abscissa, i.e., there is an` i `such that` $\xi_i = t$. `Output` $p(t) = p'_i$.

It is not even clear that $P(t)$ so defined is continuous, let alone a piece-wise polynomial function of t. To warm up, the case with $n = 3$ and knot-vector 000111, in other words, the Bernstein system, is shown in Figure 16. Every row lists the knot vector and the greville abscissa. The straight arraows indicate that the greville abscissa comes down as before. the slanted arrows indicate that the greville abscissa is obtained as a linear combination. The coefficients of a combination are marked on the arrows. Thus we see that:

$$\frac{1+2t}{3} = \frac{1+t}{3} \cdot (1-\mathtt{t}) + \frac{2+t}{3} \cdot \mathtt{t}$$

Let us suppose that r is the largest i such that $\alpha_i < t$ and s is the smallest j such that $t < \alpha_j$. Let $a = \alpha_r$ and $b = \alpha_s$ and note that $a < t < b$. Also note that there are no *other* knots between a and b except possibly t intself and with multiplicity k as in Step 1 above. Let the multiplicity of a be m_1 and that of b be m_2 in $\overline{\alpha}$. Now here is the crux: Let I_t, I_a and I_b be the opration of inserting t exactly $(n-k)$-times, or a exactly $(n-m_1)$-times or b exactly $(n-m_2)$-times, respectively. Note that by Theorem 7.3, these operations commute, and thus, if we were to insert a, b after inserting t or before that, we would get the same control polygon, say $P'' = [p''_1, \ldots, p''_N]$. Consider the case of inserting a, b after inserting t. Thus, let $\overline{\alpha}_t = Insert(\overline{\alpha}, t^{n-k})$ and $\overline{\alpha}_{tab} = Insert(\overline{\alpha}_t, a^{n-m_1}b^{n-m_2})$. Since the greville abscissa t has already been achieved in inserting t, i.e., $t \in \Xi(\overline{\alpha}_t)$, we have $t \in \Xi(\overline{\alpha}_{tab})$, and the evaluation of the control point at this abscissa is unchanged. In other words, the control point assigned to the greville abscissa remains unschanged during the process of inserting a, b. On the other hand, inserting a, b into $\overline{\alpha}$ first, gives us the knot vector $\overline{\alpha}_{ab}$ and finally inserting t gives us $\overline{\alpha}_{abt} = \overline{\alpha}_{tab}$.

Now note that $\gamma = a^n t^k b^n$ is a sub-sequence of $\overline{\alpha}_{ab}$, and thus by Theorem 7.2, inserting t into $\overline{\alpha}_{ab}$ is equivalent to inserting t into γ. In effect, we may analyse the knot-vector $a^n t^k b^n$ for all properties

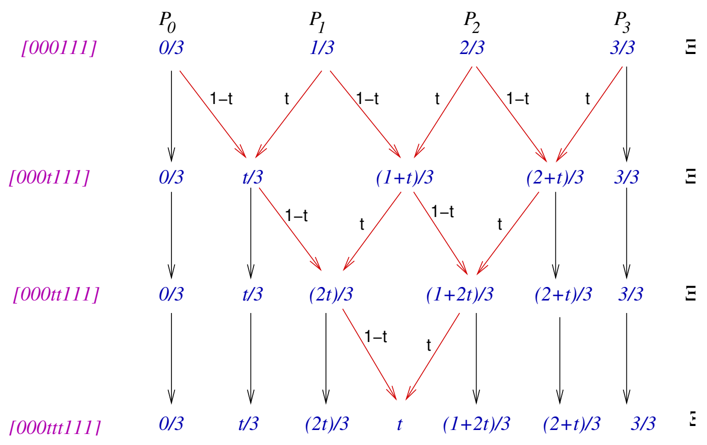

Figure 16: Bernstein as B-Spline.

of the function $P(t)$. Further note that if the multiplicity of t was zero, then the sequence is precisely $a^n b^n$. But this is nothing but a scaled and translated Bernstein system. Indeed, if ξ_i^k is the i-th greville abscissa for the knot-vector $a^b t^r b^n$, then we have:

$$\xi_i^r = \begin{cases} \xi_i^{r-1} & \text{for } i = 1, \ldots, r-1 \\ \frac{b-t}{b-a}\xi_{i-1}^{r-1} + \frac{t-a}{b-a}\xi_i^{r-1} & \text{for } i = r, \ldots, n+1 \\ \xi_{i-1}^{r-1} & \text{for } i = n+2, \ldots, n+r-1 \end{cases}$$

Thus the final ξ_{n+1}^n would be:

$$\xi_{n+1}^n = \sum_{i=0}^{n} \binom{n}{i} \frac{(t-a)^i (b-t)^{n-i}}{(b-a)^n} \xi_i^0$$

Since $P(t)$ is related to the control points with the same coefficients, we have:

$$P(t) = \sum_{i=0}^{n} p_i \binom{n}{i} \frac{(t-a)^i (b-t)^{n-i}}{(b-a)^n}$$

We call $\binom{n}{i}\frac{(t-a)^i(b-t)^{n-i}}{(b-a)^n}$ as $B_i^n(a,b,t)$. We thus see that, between any two control points, $P(t)$ is indeed a polynomial of the correct degree. Our next task is to show that across two spans, these polynomials meet with the correct continuity. As before, we just need to consider the *local* case, i.e., the knot vector $a^n c^k b^n$, and the continuity at c. There are $n+k-1$ control points $p_1, \ldots, p_{n+k-1}$. The evaluation of $P(t)$ at c requires us to insert c exactly $(n-k)$-times. The recurrence relation is as before, except that the base case is with $r = k$ and the iterations begin with $r = k+1$:

$$\xi_i^r = \begin{cases} \xi_i^{r-1} & \text{for } i = 1, \ldots, r-1 \\ \frac{b-t}{b-a}\xi_{i-1}^{r-1} + \frac{t-a}{b-a}\xi_i^{r-1} & \text{for } i = r, \ldots, n+1 \\ \xi_{i-1}^{r-1} & \text{for } i = n+2, \ldots, n+r-1 \end{cases}$$

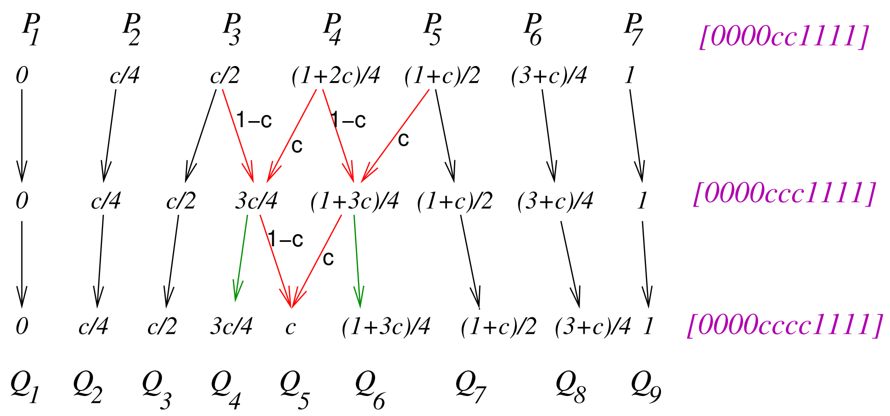

Figure 17: An example of meeting polynomials.

We thus see that at the end of inserting c exactly $(n-k)$-times, we have the knot vector $a^n c^n b^n$, and control points $q_1, \ldots, q_{2n+1}$. Clearly, $Q_L = (q_1, \ldots, q_{n+1})$ and the knot vector $a^n c^n$ define a polynomial $p_L(t)$, on the left interval $[a, c]$, and $Q_R = (q_{n+1}, \ldots, q_{2n+1})$ along with the knot vector $c^n b^n$, defines a polynomial $p_R(t)$ on the right interval $[c, b]$. the question is to determine how p_L meets p_R at $t = c$.

Towards this, as in the $a^n b^n$ case, we see that Q is determined by P, the initial control points rather explicitly.

$$q_i = \begin{cases} p_i & \text{for } i = 1, \ldots, k+1 \\ \sum_{j=0}^{i-k-1} B_j^{i-k-1}(a,b,c) p_{k+1+j} & \text{for } i = k+2, \ldots, n+1 \\ \sum_{j=0}^{2n+1-k-i} B_j^{2n+1-k-i}(a,b,c) p_{i-n+k+j} & \text{for } i = n+2, \ldots, 2n-k \\ p_{i+n-k} & \text{for } i = 2n-k+1, \ldots, 2n+1 \end{cases}$$

The case with $a = 0, b = 1$ and $n = 4$ with $k = 2$ is shown in the Figure 17. In the figure, we see that the control points $[q_3, q_4, q_5]$ and $[q_5, q_6, q_7]$ are obtained by the **subdivision** of the curve given by $[p_3, p_4, p_5]$ at c. This is true in general:

Proposition 7.4 *The control points $Q'_L = [q_{k+1}, \ldots, q_{n+1}]$ and $Q'_R = [q_{n+1}, \ldots, q_{2n-k+1}]$ are obtained by the subdivision at c of the curve of degree $n-k$ given by the control points $P' = [p'_0, \ldots, p'_{n-k}]$ where $p'_i = p_{k+1+i}$. In other words, $q_{k+1+i} = P'[0, i](c)$, and $q_{n+1+i} = P'[i, n-k](c)$.*

This ensures that the control points Q'_L and Q'_R satisfy Equations (2). Since Q'_L is the ending suffix of Q_L and Q'_R is the starting prefix of Q_R, *and* they satisfy the above equations, we see that $p_L(t)$ and $p_R(t)$ must meet upto $n-k$ derivatives at c. Thus, we have the main theorem of this section:

Theorem 7.5 *The function $P(t)$ defined above lies in $V(\overline{\alpha})$.*

At this point, we should also mention *locality* as an important property of B-spline curves. If $\overline{\alpha} = (\alpha_1, \ldots, \alpha_m)$ is a knot vector, and $P = (p_1, \ldots, p_{m-n+1})$ is the control polygon, then the

evaluation of the curve at $\alpha_i \leq t\alpha_{i+1}$ depends only on the control points in the 'neighborhood'. To see this, note that after t has been inserted n times, it will be expressed as a convex combination of a few greville abscissas of the original knot vector $\overline{\alpha}$. These greville abscissas are precisely those which 'span' t. In other words, these are ξ_j, where $j = i-n+1, \ldots, i$. Thus, in the evaluation of the curve at t, the only control points which matter are $p_{i-n+1}, \ldots, p_i$. This has a very useful consequence: modifying a control point changes the curve only locally.
Summary: A knot vector is given by a sequence $\overline{\alpha} = a_1^n, a_2^{m_2}, \ldots, a_{k-1}^{m_{k-1}}, a_k^n$. Such a knot vector defines the space $V(\overline{\alpha})$, of all spline functions $f : [a_1, \ldots, a_k] \rightarrow \mathbb{R}^3$ of piece-wise polynomials $r_1, \ldots, r_{k-1}$ of degree not exceeding n. Further, for $j = 0, \ldots, n-m_i$ we have the derivatives $r_{i-1}^j(a_i) = r_i^j(a_i)$. The dimension of $V(\overline{\alpha})$ is $m-n+1$, where m is the length of the knot vector. Each spline is parametrized by $m-n+1$ control points $P = [p_1, \ldots, p_{m-n+1}]$, such that $f(a_1) = p_1$, $f(a_k) = p_{m-n+1}$. The derivatives $f'(a_1)$ and $f'(a_k)$ are multiples of $p_2 - p_1$ and $p_{m-n+1} - p_{m-n}$, respectively. The evaluation of the spline function f at an argument proceeds by the knot-insertion algorithm, and takes $O(n^2)$ time. we also note that a spline $f \in V(\overline{\alpha})$ is continuous and differentiable at least k times, where k is the minium of $n - m_i$ for $i = 2, \ldots, k-1$. This number k is called the continuity of the knot vector $\overline{\alpha}$.

We thus see that an edge may be represented by the data $([a,b], f)$ where $[a,b]$ is an oriented interval, and f is a spline. Thus f may be in turn, represented by the data $(\overline{\alpha}, P)$, the knot-vector and the control polygon.

8 Surface Parametrization.

In this section, we shall develop the theory of polynomial parametrization of surfaces. Let I stand for the interval $[0,1]$ and I^2 for $[0,1] \times [0,1]$. The space I^2 will serve as the standard parametrization space for surfaces. Thus, our general surface S will be the image of a map $f : I^2 \rightarrow \mathbb{R}^3$. A sample map f is shown in Figure 18. A standard notion on surfaces are **isoparametric curves**. For example, fixing $u = u_0 \in R$, we get a map $f_{u_0} : I \rightarrow \mathbb{R}^3$, with $f_{u_0}(v) = f(u_0, v)$. This traces out a curve $C(u_0, *)$ on the surface S. One may similarly fix a $v_0 \in \mathbb{R}$ and define the v_0-isoparametric curve $C(*, v_0)$. Note that $C(u_0, *)$ and $C(*, v_0)$ always intersect on S and this point is $f(u_0, v_0)$.

Let u, v be two variables. A monomial in u, v is the term $u^i v^j$. A polynomial $p(u,v)$ in two variables is a finite sum of monomials, and thus $p = \sum_{i,j} a_{ij} u^i v^j$. The u-degree of p is the largest integer r such that $a_{rj} \neq 0$ for some j. The v-degree of p is similarly defined. The degree of p is the maximum of the u-degree and the v-degree.

To begin, let $V_{m,n}(u,v)$ or simply $V_{m,n}$ stand for the space of all polynomials $p(u,v)$ of u-degree not exceeding m and v-degree not exceeding n. The dimension of $V_{m,n}$ is obviously $(m+1)(n+1)$, and the standard basis for $V_{m,n}$ is $T = \{u^i v^j | 0 \leq i \leq m,\ 0 \leq j \leq n\}$.

Proposition 8.1 *Let $B = \{B_i^m(u) \cdot B_j^n(v) | 0 \leq i \leq m,\ 0 \leq j \leq n\}$ be the collection of all Bernstein polynomials in u of degree m and those in v of degree n. Then B is a basis of $V_{m,n}$.*

Proof: Suppose not, i.e., there are (a_{ij}) not all zero, such that $\sum_{i,j} a_{ij} B_i^m(u) B_j^n(v) = 0$, or in other words a linear combination of the product bernsteins equals the zero polynomial. Collecting terms, we have

$$\sum_{j=0}^{n} p_j(u) B_j^m(v) = 0$$

where $p_j = \sum_{i=0}^{m} a_{ij} B_i^m(u)$. For any fixed parameter u_0, we have $\sum_j p_j(u_0) B_j^m(v) = 0$. Now since $\{B_j^n(v) | j = 0, \ldots, n\}$ form a basis of $V_n(v)$, we must have that $p_j(u_0) = 0$ for all $u_0 \in \mathbb{R}$. Thus, we

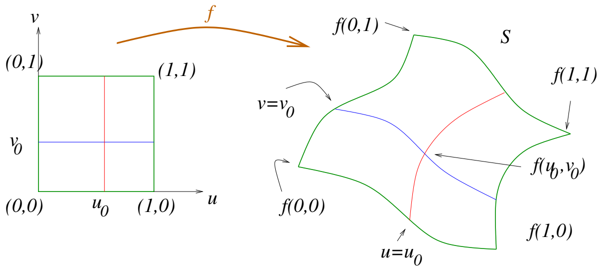

Figure 18: A surface S.

see that p_j must be the zero polynomial, whence $\sum_{i=0}^{m} a_{ij} B_i^m(u)$ is the zero polynomial. But this contradicts the linear dependence of the bernstein basis in u. $\Box$

Note that the same proof shows that if $X = \{p_0(u), \ldots, p_m(u)\}$ were a basis for $V_m(u)$ and $Y = \{q_0(v), \ldots, q_n(v)\}$ were a basis for $V_n(v)$, then $X \otimes Y = \{p_i(u)q_j(v) | 0 \leq i \leq m,\ 0 \leq j \leq n\}$ is a basis for $V_{m,n}$.

The **tensor product** Bezier surface of degrees (m, n) is given by a $(m + 1) \times (n + 1)$ matrix P of control points. The map $f : I^2 \to \mathbb{R}^3$ given by this data is

$$f(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} P[i, j] B_i^m(u) B_j^n(v)$$

We now examine some of the properties of the tensor-product surfaces. Firstly, at $u = 0$, we have $B_i^m(0) = 0$ for $i > 0$, and thus $C(0, *) = \sum_{j=0}^{n} P[0, j] B_j^n(v)$. Thus this isoparametric curve is a bezier curve with control points given by the first (or rather 0-th) row of the matrix P. Similarly, $C(1, *)$, $C(*, 0)$ and $C(*, 1)$ are all bezier, and their control points are easily worked out to be the last row, first column, and last column, respectively, of P.

The general isoparametric curve is only a bit more interesting: For a fixed $u_0 \in \mathbb{R}$, we have

$$f(u_0, v) = \sum_{j=0}^{n} (\sum_{i=0}^{n} P[i, j] B_i^m(u_0)) B_j^n(v)$$

In other words, the curve is a bezier curve in v of degree n, with the $(n + 1)$ control points $(q_0(u_0), \ldots, q_n(u_0))$, with q_j itself following a bezier curve in u of degree m with control points $(P[0, j], \ldots, P[m, j])$. Thus, we may say that the tensor-product surface is obtained as a family of bezier curves whose control points move together along 'rails' defined by bezier curves themselves. The evaluation procedure is easy too: for the point $f(u_0, v_0)$, we first get the control points q_j above for $C(u_0, *)$. These are obtained by the evaluations $n+1$ points on $n+1$ bezier curves corresponding to the columns of P. Once the q_j's have been obtained, we use them as control points to obtain $f(u_0, v_0)$. Thus the net time complexity is $O((n + 1)m^2 + n^2)$.

Given the map $f : I^2 \to \mathbb{R}^3$, there are the associated differentials $f_u = \frac{\partial f}{\partial u}$ and $f_v = \frac{\partial f}{\partial v}$. Note that both of these are again maps $f_u, f_v : I^2 \to \mathbb{R}^3$. We easily see that:

$$\frac{\partial f}{\partial u}(u, v) = \sum_{j=0}^{n} B_j^n(v) \frac{\partial}{\partial u} (\sum_{i=0}^{m} P[i, j] B_i^m(u))$$

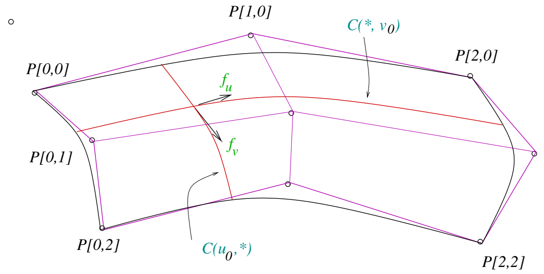

Figure 19: Tensor-product Bezier surface.

Thus, defining the $m \times (n+1)$ matrix Q, with $Q[i,j] = m(P[i+1,j] - P[i,j])$, we see that f_u is also a tensor product surface of degree $(m-1,n)$ in u,v, and given by the control points Q. A similar expression holds for f_v. See Figure 19.

Finally, let S and S' are two tensor-product bezier surfaces given by control matrices P and P'. Furthermore, let the u-degree of both P and P' be m, then $P[i,n] = P'[i,1] = Q[i]$ ensures that the two surfaces meet at the boundary given by the bezier curve Q of degree m.

Next, we tackle tensor-product B-spline surfaces. Let $\overline{\alpha}$ and $\overline{\beta}$ be two knot vectors of length m and m', both for the same degree n. The tensor product B-spline surface S of type $(\overline{\alpha}, \overline{\beta})$ and of degree n, is specified by the $(m-n+1) \times (m'-n+1)$-matrix P. For a (u,v), $f(u,v)$ is given by evaluating $q_i(u)$ on the curve C_i given by the control points $P[*,i]$. The curve C_i is a B-spline curve with knot vector $\overline{\alpha}$. Once $[q_1(u), \ldots, q_{m'-n+1}(u)]$ have been obtained, they are treated as the control points for a curve with knot vector $\overline{\beta}$ to obtain $f(u,v)$. It is a simple matter to show that we could have constructed the 'row' curves and then the column curve and gotten the same answer.

We also note that for a tensor-product B-spline surface, the continuity is the minimum of the continuities of $\overline{\alpha}$ and $\overline{\beta}$.

An Example: We consider here a small design problem, that of designing a soap tablet. Typically, the required soap tablet may be specified by its three sections, along the 3 coordinate planes (see Figure 20). We assume, for simplicity, that the soap tablet is symmetrical about each of these planes. In other words, it suffices to design the tablet in the positive 'orthant' and makes 8 copies of it by suitable reflections. Pasted together, these 8 copies will give us the complete soap tablet. We design the positive orthant as a single surface which matches the prescribed curves at the boundary.

For simplicity, we choose a $4,4$ tensor product bezier surface. The first objective is to parametrize the boundary curves C_1, C_2 and C_3 as shown in the figure. We assume that the tablet is 72mm long, 40mm broad and 24mm thick. This gives us the vertices as `[0,36,0]`, `[20,0,0]` and `[0,0,12]`. The curve C_1 may now be designed as starting from `[0,0,12]` and ending at `[0,36,0]`. Since C_1 must meet smoothly its reflection along the X-Z plane, we see that the tangent vector of C_1 at `[0,0,12]` must be along the Y-direction. This forces the next control point of C_1 to differ from `[0,0,12]` in only the Y-coordinate. We have chosen it to be `[0,27,12]`. The other control point, similarly, must differ from `[0,36,0]` in only the Z-direction. Proceeding similarly, we may design all the curves (see Figure 21).

The next task is to assign these curves as boundary curves of a tensor-product surface. Thus we have a matrix $P[i,j]$ with $0 \le i,j \le 3$, in which we fill in the appropriate control points. The first complication arises here, since a surface patch has 4 boundary curves, while we have only three. To circumvent this, we assign C_4 to be the *trivial* curve at the vertex `[0,0,12]`. We may thus fill in

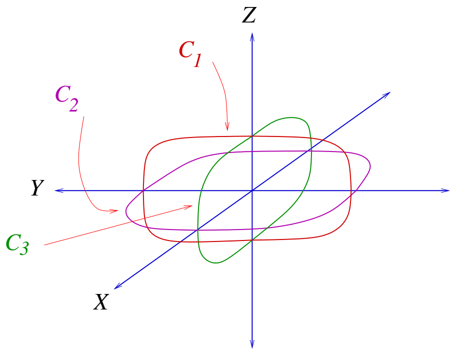

Figure 20: The specification of the soap tablet.

the boundary of our desired patch as follows:

$P =$

[0, 0, 12]	[0, 0, 12]	[0, 0, 12]	[0, 0, 12]
[0, 27, 12]			[15, 0, 12]
[0, 36, 9]			[20, 0, 9]
[0, 36, 0]	[10, 36, 0]	[20, 24, 0]	[20, 0, 0]

This leaves us with the four interior points to play with. However, we see that the bottom left and right corners, viz. $P[2,1]$ and $P[2,2]$ are not really in our hands: Both $P[2,1]$ and $P[2,2]$ must lie vertically above $P[3,1]$ and $P[3,2]$ respectively. This is because we would like the surface to meet the X-Y plane vertically so that upon reflection it meets its partner smoothly. Similary, we have that $P[2,1]$ may differ from $P[2,0]$ only in the X-direction, while $P[2,2]$ may differ from $P[2,3]$ only in the Y-direction. This fixes $P[2,1]$ and $P[2,2]$ completely. In fact, such considerations leave only two parameters, one in $P[1,1]$ and the other in $P[1,2]$. The complete matrix is given below and illustrated in Figure 22:

$P =$

[0, 0, 12]	[0, 0, 12]	[0, 0, 12]	[0, 0, 12]
[0, 27, 12]	[x, 27, 12]	[15, y, 12]	[15, 0, 12]
[0, 36, 9]	[10, 36, 9]	[20, 24, 9]	[20, 0, 9]
[0, 36, 0]	[10, 36, 0]	[20, 24, 0]	[20, 0, 0]

9 The solid model

Now that we have the parametrization of surfaces and curves well-understood, we set about describing the *boundary- representation* of a solid which is now an industry standard. The boundary representation or *Brep* of a solid stores the *boundary* between the solid and the outside space, and hence the name. The boundary is typically a collection of `faces`, each of which has a different parametrization. Different faces meet on `edges` and different edges meet on `vertices`.

An example solid is shown in Fig 23. On the left, we see a solid block with a four visible faces and a few faces not visible in this particular view. Note that F_1 has five edges while F_2 has only three. Also note that F_1 may be *only a part* of a tensor-product spline surface S_1, while F_2 that of S_2.

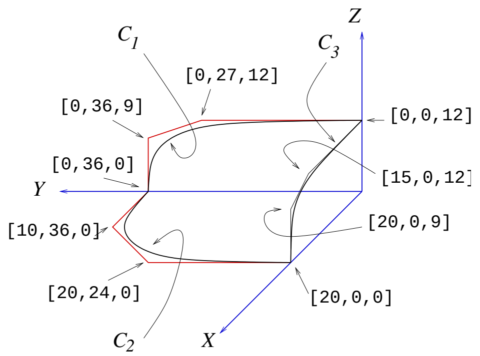

Figure 21: The specification of C_1, C_2 and C_3.

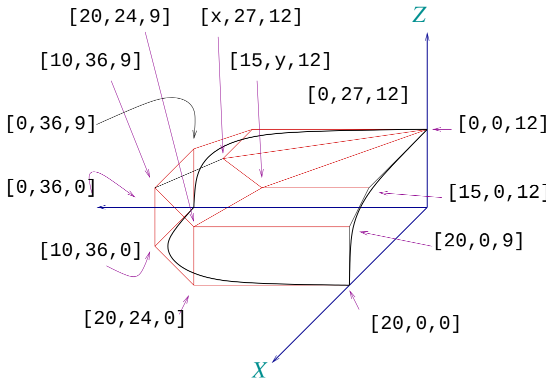

Figure 22: The complete specification.

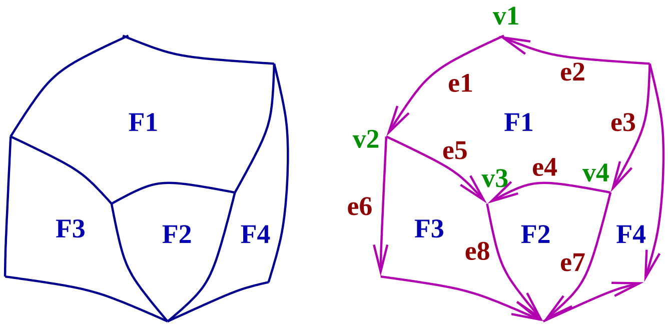

Figure 23: A typical solid

The representation of this solid (i) first labels each face, (ii) next, orients (i.e., assigns a direction) and labels each edge, and (iii) finally labels each vertex. This is shown in the RHS of the figure above. The orientation of an edge may be arbitrary.

Next, the data for the solid is stored in four lists, viz., `Vertex-List, Edge-List, Face-List` and Co-edge-List. We now explain this in detail.

The `Vertex-List` stores a list of vertices v with the following fields:

$v.label$	the name of the vertex
$v.geometry$	the coordinates as a 3-dimensional vector

Thus, for example, $v.label = v_1$ and $v.geometry = [-1, 0, 0]$ stores the vertex v_1 of our block. Next, the `Edge-List` stores edges e with fields *label, start, end, geometry*. An example:

$e.label$	e_1
$e.start$	v_1
$e.end$	v_2
$e.geometry$	C_1

The *start* and *end* of an edge is determined by the orientation of the edge. Here C_1 is a structure which stores a spline curve, i.e., its knot vector and the control points. Clearly, the first control point of C_1 must be $v_1.geometry$ and the last must be $v_2.geometry$. Next, we have the `Face-List`, where each face f has the structure below:

$f.label$	f_1
$f.geometry$	S_1
$f.sign$	± 1

Here, $f.geometry$ stores a B-spline surface S_1 which is a structure storing the knot-vectors and control-points. The field $f.sign$ stores the *outward-normal* to the space as a multiple of $\frac{\partial S}{\partial u} \times \frac{\partial S}{\partial v}$. Thus a sign of $+1$ implies that the normal is aligned along the cross-product. Also note that the surface S_1 may be *much larger* than F_1 and that F_1 is only a small part of S_1. The exact part of S_1 to be taken as the definition of F_1 comes from the `Co-edge-List`. Note that S_1 establishes a map

$$S_1 : [u_{min}, u_{max}] \times [v_{min}, v_{max}] \to \mathbb{R}^3$$

Thus, the part of the surface which corresponds to the face F_1 may be obtained by restricting the domain of the function S_1 to a domain D_1. See Figure 24.

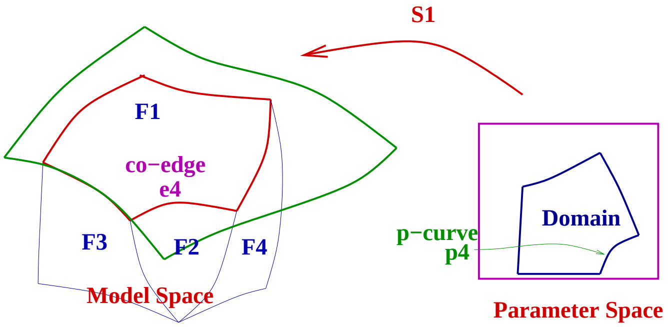

Figure 24: The co-edge

On the left, we see the UV-box of the surface S_1 and its image on the right, which subsumes F_1. Each edge bounding the face F_1 will have a corresponding *pre-image* in the UV-box of S_1. This curve in the parameter space is called the *p-curve* of the correponding edge. For example, for the bounding edge e_4 of F_1, we have the *pcurve* p_4. This data is stored in the `Co-edge-List` as follows. For every edge-face pair (e, f), where e is a bounding edge of the face f, we have an entry in the `Co-Edge-List`. For example:

f.label	f_1
e.label	e_4
pcurve.geometry	p_4
ef.sign	-1

The geometry of the *pcurve* is stored as a B-spline curve p_4. Thus, it has a knot-vector and control points. It is desirable that the knot-vector is the same as that of the curve geometry, in other words, that of e_4, and that the parametrization matches that of the edge. Note that these control points *lie in the UV-box of the face F_1*. The $ef.sign$ stores the orientation of the edge e_4 with respect to the surface normal of F_1. In this case, with the face-normal pointing outside, we see that e_4 is oriented in the clockwise sense with respect to the normal and hence carries the sign -1.

Also note that there is another co-edge for e_4 from the face F_2 for which we will store another *pcurve*, say p'_4 which lies in the UV-box of F_2. Furthermore, the $ef.sign$ for this co-edge will be $+1$. Further note that, as far as the geometry is concerned,

$$S_1(p_4(t)) = S_2(p'_4(t)) = e_4(t)$$

Thus, the geometry of the edge e_4 is available from *three* places, (i) from $e_4.geometry$, (ii) from $S_1.geometry$ and p_4 and (iii) from $S_2.goemetry$ and p'_4.

The above four lists suffice to define the complete structure of a solid. Note that there are several dependencies between the lists and these must be maintained, upto tolerance. Typical kernels will store a modification of the above lists and may buffer certain information such as loops and so on. For solids with cavities, there will be several connected boundaries and each boundary is then stored similarly, with a marked outer boundary. Every operation of the solid which the kernel offers must be accompanied by a procedure which update this representation.

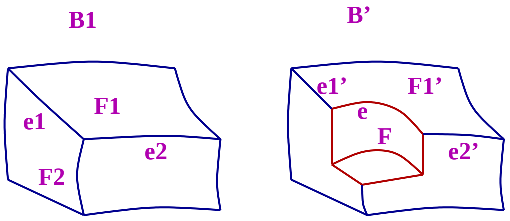

Figure 25: A boolean operation.

10 Kernel Operations

In this rather large section we will outline some of the operations that are offered by kernel and an outline of their implementations. The conceptually simplest operation would be the so-called **booleans**, i.e., solid-soild union, intersection and difference. See Fig. 25 for an example, where a cylinder is subtracted from a roughly cuboidal body B_1. Notice that the face F_1 of the cuboid gets **trimmed** further by the cylindrical face F which now appears on the new body. This calls for an **intersection edge** e formed by the intersection of the face F_1 and the cylindrical face F. Furthermore, the edges e_1 and e_2 further get trimmed by the face F. A similar set of operations define the other faces and edges of the new body B'.

Thus, to generate the representation of the new body B' substantial work is required. This may be divided into two parts:

- **Topological**: This relates to getting the new faces, edges and vertices of the new body and computing their adjacencies. In other words, this is the computation of all fields of the four lists but the **geometry**.
- **Geometric**: This fills up the actual gometries of the new and modified entities. Thus vertex, edge and pcurve geometries need to be computed.

The computation, though categorized separately, is actually done simultaneously, wherein the topological information of the participating bodies drives the construction and computations of the new body. For example, that the edge e_1 is cut by the face F itself indicates that F_2 must also intersect F.

This section concentrates on the geometric part of the computation. The boolean operation poses us two problems:

- **Edge-Face intersection** or more simply, curve-surface intersection.
- **Face-Face intersection** or simply surface-surface intersection.

The simplification of a edge-face operation to its underlying geometry is a useful conceptual simplification akin from moveing from the case of *constrained optimization* to the *unconstrained* case.

Example 10.1 *Let us look at even simpler operations such as a* curve-plane *intersection. We are given a curve $C(t) = (x(t), y(t), z(t))$ and a plane $ax + by + cz + d = 0$. The point of intersection then is given by:*

$$p(t) \equiv a \cdot x(t) + b \cdot y(t) + c \cdot z(t) + d = 0$$

Thus, we must solve for t in the equation above.

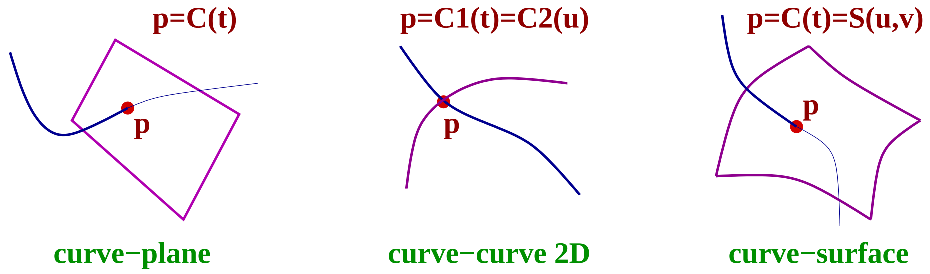

Figure 26: Intersections

Example 10.2 *Another simple example is the curve-curve intersection in 2-dimensional space. Thus, we are given $C_1(t) = (x_1(t), y_1(t))$ and $C_2(u) = (x_2(u), y_2(u))$. The intersection point p will obviously satisfy $p = C_1(t_0) = C_2(u_0)$ for some t_0 and u_0. Thus we have:*

$$\begin{aligned} x_1(t) - x_2(u) &= 0 \\ y_1(t) - y_2(u) &= 0 \end{aligned}$$

In other words, we have a 2-variable, 2-equation system

$$\begin{aligned} f_1(t, u) &= 0 \\ f_2(t, u) &= 0 \end{aligned}$$

Example 10.3 *We can pose the curve-surface intersection as well. If $C(t) = (x(t), y(t), z(t))$ and $S(u, v) = (x'(u, v), y'(u, v), z'(u, v))$, we have:*

$$\begin{aligned} x(t) - x'(u, v) &= 0 \\ y(t) - y'(u, v) &= 0 \\ z(t) - z'(u, v) &= 0 \end{aligned}$$

Thus, we have a 3-variable, 3-equation system to be solved.

The three examples above are explained in the figure Fig. 26. We see that all the above geometric operations yield a n-variable, n-equation system:

$$\begin{aligned} f_1(x_1, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, \ldots, x_n) &= 0 \end{aligned}$$

10.1 The Newton-Raphson Solver

Let us consider the case when $n = 1$, or in other words, there is a single function $f(x)$ to be solved. Here is a naive line of attack called the **bisection method**.

```
Input  xlo, xhi such that f(xlo)*f(xhi)<0
xmid=(xlo+xhi)/2;
while |f(xmid)|>epsilon
  if f(xmid)*f(xlo)<0
```

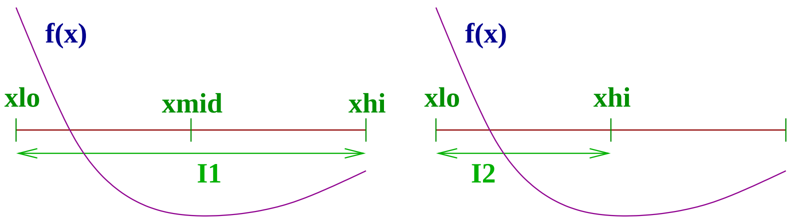

Figure 27: The Bisection Method

```
5        xhi=xmid;
6      else
7        xlo=xmid;
8      end;
9    endwhile;
10   return;
```

The condition on `line 1` ensures that the signs of the function f at x_{lo} and x_{hi} are opposite, and thus there is a root in the interval $[x_{lo}, x_{hi}]$. The `while` loop locates the mid-point x_{mid} and selects the new interval to be the left-half or the right-half of the original interval which maintains the negative-sign invariant. This is illustrated in Fig. 27. The interval I_1 was the initial guess for the location of the root. After checking on the sign of $f(x_{mid})$, the new interval becomes I_2, which is half the length of I_1. Thus after n interations, $|I_n| = 2^{-n}|I_1|$ and thus, if $\epsilon = 2^{-k}$, then we should expect $O(k)$ iterations. We call this method *linear* since the precision obtained is linear in the number of iterations. Also note that the procedure needs just to evaluate the function f and assumes its continuity. We summarize this as follows:

Name	Time	Assumptions	Initializations
Bisection	Linear	continuity of f	x_{lo} and x_{hi}

We now outline the **Newton-Raphson** procedure:

```
1    Input x, a guess for the zero of f
2    while |f(x)|>epsilon
3       m=f'(x);
4       xnew=x-f(x)/m;
5       x=xnew;
6    endwhile;
7    return;
```

We see that having evaluated the slope of the function f, we compute x_{new} in `line 4`. This value of x_{new} is precisely the intersection of the tangent at $(x, f(x))$ and the X-axis. The point x_{new} is taken as the new guess for the root and the loop continues.

The exact analysis of the method is outside the scope of this course, however we state the results. If α is the root of the function then the convergence is *quadratic* provided $f'(\alpha) \neq 0$. In other words, if k iterations are performed then the precision achieved is $O(k^2)$, i.e., $|x_k - \alpha| = C \cdot 2^{-k^2}$. However, the method is fairly unstable if the initial guess is not within a small interval of the actual root α. Bad cases are shown in Fig. 29 and a summary appears below. Typically, in practical situations,

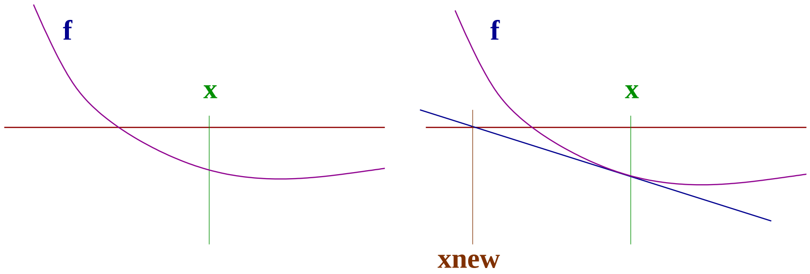

Figure 28: The Newton-Raphson Method

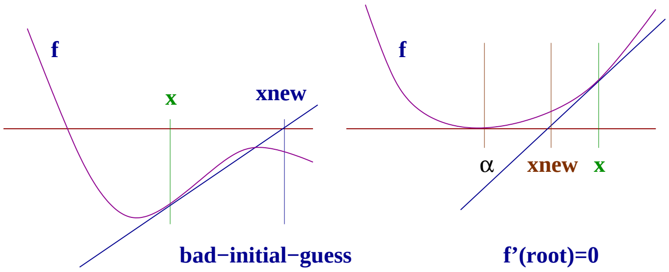

Figure 29: The bad cases

the first few iterations may be of bisection type to locate the root and then precision is obtained by subsequent Newton-Raphson iterations.

Name	Time	Assumptions	Initializations
Newton-Raphson	Quadratic	differentiability of f	initial guess *near* root

Also note that, if perchance our initial guess were the root α then the next iteration would return α. In other words, the root is the *fixed point* of the iteration scheme. Such methods are called **fixed point methods** and have seen considerable mathematical analysis.

We now consider the general n-variable, n-equation case. It is the the Newton-Raphson scheme which is easily extended to the general case. Observe that the process of obtaining the iterate x^{i+1} from x^i is explained as follows. Having computed $f(x^i)$ and $f'(x^i)$, we may compute $g^i(x)$, the i-th (linear) approximant to $f(x)$ as:

$$g^i(x) = f(x^i) + (x - x^i)f'(x^i)$$

Whence, note that $x^{i+1} = x_i - f(x^i)/f'(x^i)$ is merely the root of $g^i(x)$.

In the general situation, when we have

$$\begin{aligned} f_1(x_1, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, \ldots, x_n) &= 0 \end{aligned}$$

and the i-th iterate $y = (y_1, \ldots, y_n)$, we build the n linear approximations $g_1^y(x), \ldots, g_n^y(x)$ as follows:

$$\begin{bmatrix} g_1^y(x) \\ \vdots \\ g_n^y(x) \end{bmatrix} = \begin{bmatrix} f_1(y) \\ \vdots \\ f_n(y) \end{bmatrix} + \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(y) & \ldots & \frac{\partial f_1}{\partial x_n}(y) \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1}(y) & \ldots & \frac{\partial f_n}{\partial x_n}(y) \end{bmatrix} \begin{bmatrix} (x_1 - y_1) \\ \vdots \\ (x_n - y_n) \end{bmatrix}$$

Writing the $n \times n$-matrix above as the **Jacobian** $J(f, y)$, we may write the above in short as:

$$g^y(x) = f(y) + J(f, y) * (x - y)$$

The next iterate $z = (z_1, \ldots, z_n)$ is computed by assigning $g_i^y(z) = 0$ for all i. In other words,

$$z = y - J(f, y)^{-1} f(y)$$

This, then is the general form of the Newton-Raphson method. Note that the iteration depends on the non-singularity of the Jacobian. In practice, even if the Jacobian is non-singular, the convergence depends on how far away it is from being singular. Also note that, as opposed to the bisection method, the Newton-Raphson requires the functions to be differentiable.

Example 10.4 *Let $f_1(u, v) = u^2 + u + v$ and $f_2(u, v) = uv + u - v$. We wish to solve for $f_1(u, v) = f_2(u, v) = 0$ with an initial guess $y = (1, 1)$. We see that*

$$J(f, y) = \begin{bmatrix} 2u + 1 & 1 \\ v + 1 & u - 1 \end{bmatrix} (1, 1) = \begin{bmatrix} 3 & 1 \\ 2 & 0 \end{bmatrix} \quad J(f, y)^{-1} = \begin{bmatrix} 0 & 1/2 \\ 1 & -3/2 \end{bmatrix} \quad f(y) = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

Thus, the new iterate is:

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 & 1/2 \\ 1 & -3/2 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1/2 \\ -1/2 \end{bmatrix}$$

Notice that the solution is $(0, 0)$ and that the new iterate is much better than the old guess.

We see that the NR-framework may be easily deployed to solve the curve-curve intersection or the curve-surface intersection problem as seen above. All that is required are the evaluations of the associated functions and their derivatives. These functions are usually derived from the curve/surface evaluation functions in a systematic way and thus they and their derivatives are available for evaluation. The edge-face intersection algorithm has at its core the curve-surface intersection. Once the (u_0, v_0) and t_0 have been obtained, it must be verified that all these points lie on the face and edge respectively. The edge case is easy: one merely check that $a \leq t_0 \leq b$, where $[a, b]$ is the domain of definition of the edge. The surface case is harder. We must ensure that the point of intersection lies on the face, see Fig. 30. On the LHS we see the two possibilities. This corresponds to detecting if (u_0, v_0) lies in the domain of definition of the face. One way of doing this is by selecting a (u_1, v_1) known to lie on the face. Then, we compute the number of intersections of the straight-line connecting (u_0, v_0) to (u_1, v_1). This number is odd iff the point is outside.

The formulation of the problem as a NR-problem is the key, however there are many distinct possibilities. As an example, consider the curve-curve intersection in the plane, which we have posed as a 2-variable NR system. Another alternative is as follows:

```
1.   Input curves C1,C2 and initial guesses t1,t2, toggle=1;
2.   while norm(C1(t1)-C2(t2))>epsilon
```

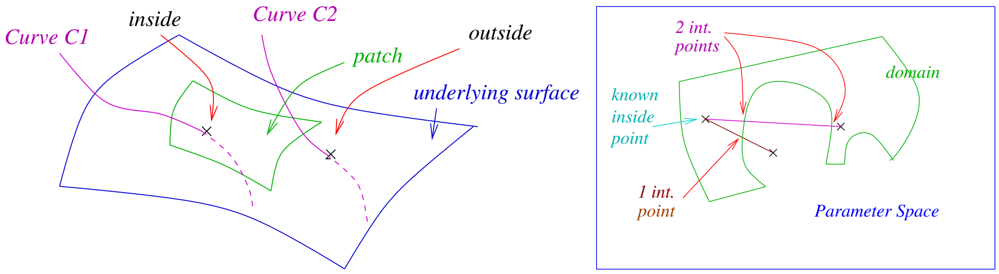

Figure 30: The edge-face intersection

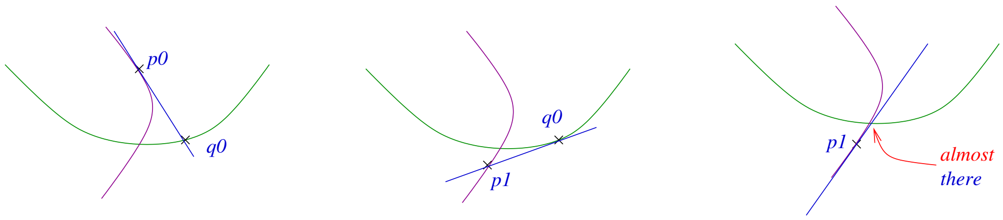

Figure 31: The mixed approach

```
  if toggle==1
    L=TangentLine(C1,t1) % tangent to C1 at t1
    t2=Intersection(L,C2) % use t2 as initial guess
    toggle=2;
  else
    L=TangentLine(C2,t2) % tangent to C2 at t2
    t1=Intersection(L,C1) % use t1 as initial guess
    toggle=1;
  end;
end;
```

In other words, For an initial guess p_0 on C_1, we compute q_0 on C_2 by intersecting the tangent at p_0 to C_1 with C_2. Next, a tangent at q_0 to C_2 is used to compute p_1, and so on. This is illustrated in Fig. 31. As we have seen, the curve-plane solution involves a 1-variable, 1-equation system. The general curve-curve system has 2-variables. Thus, we have an alternate NR formulation using two 1-dimensional NR systems.

10.2 The surface-surface intersection

We have thus seen that the n-variable, n-equation system is eminently solvable, provided the functions are well-conditioned and a good initital guess is available. This usually solves the curve-curve intersection or the curve-surface intersection problem as seen above. Let us consider next, the surface-surface intersection problem which will typically generate a curve. If we were to formulate

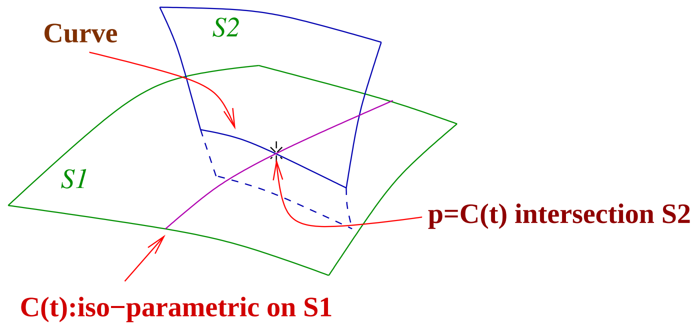

Figure 32: The surface-surface intersection

this in the above format, we see that if $S_1(u_1, v_1)$ and $S_2(u_2, v_2)$ were the surfaces, then we have:

$$\begin{aligned} x_1(u_1, v_1) - x_2(u_2, v_2) &= 0 \\ y_1(u_1, v_1) - y_2(u_2, v_2) &= 0 \\ z_1(u_1, v_1) - x_2(u_2, v_2) &= 0 \end{aligned}$$

In other words, we have 3 equations in 4 variables, viz., u_1, v_1, u_2 and v_2. This is clearly not in the Newton-Raphson (NR) framework. However, if we were to fix one of the variables, say $u_1 = c$, then we do get back into the NR framework. However, putting $u_1 = c$ is akin to picking the c-isoparameter curve on S_1 for the first parameter (u_1) and intersecting that with S_2. In other words, this is just the curve-surface intersection albeit with the curve being an iso-parameter curve. We may thus define the intersection curve $C(t)$ as the point of intersection of the iso-par curve $S_1(t, v_1)$ with $S_2(u_2, v_2)$.

This defines not only the intersection curve but also its *parametrization*. While, the definition of $C(t)$ above is functionally convenient, there are two problems with it. Firstly, it is not clear how is one to store the curve as a B-spline. This is usually done by selecting a *suitable* sequence $t_1 < t_2 < \ldots < t_N$ and computing $p_i = C(t_i)$. The sequence (p_i) is then used as the control polygon of a curve $C'(t)$ with an appropriate *long* knot-vector. While the points p_i do lie on the intersection curve $C(t)$, the constructed curve $C'(t)$ may stray from the ideal $C(t)$. Typically, this is controlled by choosing t_i appropriately close and dense. Also note that, the process of solving for p_i, the intermdeiate data is $(t_i, (v_1)_i)$ and $((u_2)_i, (v_2)_i)$, the surface parameters of p_i on either surface. This data must be used to construct the *pcurves* for the intersection edge. See Fig. 33.

The second issue is more subtle. While, to begin with, u_1 may be a good parameter along which to compute the intersection curve. As the curve proceeds, u_1 may fail to do the job. See Fig. 33. In the good case, the u_1-iso-par lines to indeed meet the intersection curve uniformly and serve to parametrize the intersection curve. On the right hand, we see a not-so-rare situation where the u_1-iso-par lines do not meet the intersection curve at only one point. Even worse, as t increases from t_1 to t_3, we will see that the NR system will become more and more ill-conditioned and finally be unsolvable. One way around this is to shift the *constant* parameter from u_1 to v_1 or on the other surface once u_1 is found to misbehave. All this makes the actual program rather bulky. We will later see a more abstract method of parametrizing the intersection curve.

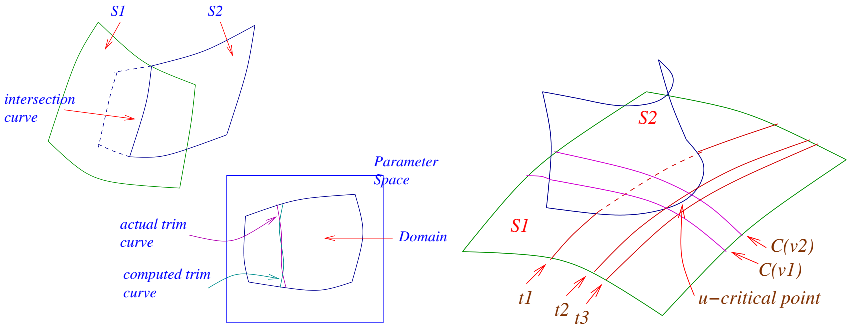

Figure 33: The trim curves and critical points

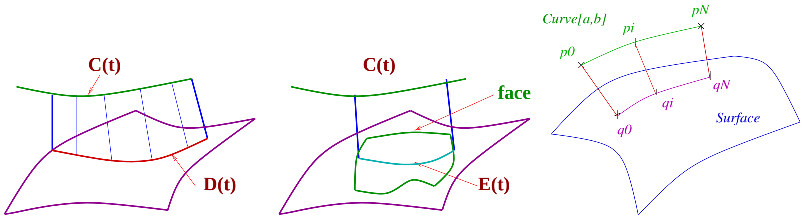

Figure 34: The curve-surface projection

10.3 The projection construction

We outline, next, another typical kernel operations, viz., *projection*. Let S be a surface and $p \in \mathbb{R}^3$, a fixed point. It is desired to project the point p on the surface S to obtain the point $q \in S$. This q is thus a local minima of the distance function $d(s) = \|p - s\|$ for $s \in S$. Assuming that $s = S(u, v)$, we see that

$$f(u, v) = d^2(u, v) = (x(u, v) - p_x)^2 + (y(u, v) - p_y)^2 + (z(u, v) - p_z)^2$$

Thus, if $q = S(u_0, v_0)$, then we have:

$$\frac{\partial f}{\partial u}(u_0, v_0) = 0 \quad \frac{\partial f}{\partial v}(u_0, v_0) = 0$$

Thus (u_0, v_0) are obtained as the solution of a 2-variable, 2-equation system.

The next projection operator is the curve-surface projection, viz., a curve $C(t)$ to be projected on the surface S. If $D(t)$ is the projected curve, then we define

$$D(t) = project(C(t), S)$$

In other words, every point on $C(t)$ is projected to obtain the point $D(t)$. Of course, if the domain of $C(t)$ is $t \in [a, b]$, the domain of $D(t)$ may be a smaller $[a_1, b_1]$, see Fig. 34. Further, the face may be smaller than the surface, and thus $D(t)$ may have to be trimmed further to $[a_2, b_2]$. In general, in this paper, we skip the associated and additional topology based operation and stress on the base construction.

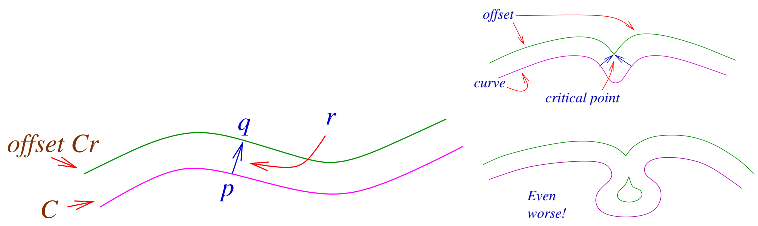

Figure 35: The curve offset

Since, once again, a non-zero dimensional entity (a curve) is to be created, we follow the usual **sampling approach**. An appropriate sequence $a_1 = t_0 < t_2 < \ldots < t_N = b_1$ is selected. Next, corresponding to each t_i, we construct p_i, u_i, v_i. The p_i's are used to construct the curve, while the (u_i, v_i)'s the corresponding *pcurves*.

10.4 Some more constructions

The above discussion paves the way for discussing other operations and constructions supported by typical kernels. We will only state the mathematical definitions and the parametrizations of the constructed curves and surfaces. The precise construction may follow the *sampling technique* above.

Curve Offsets: This is a method of constructing new curves from old. So let $C(t)$ be a curve in a plane and $N(t)$ be a continuous unit normal within the plane to $C(t)$. Note that $N(t)$ is easily constructed from $C(t)$ and its derivative. Given $C(t)$ and r, the offset curve is obtained by moving each point p on C by a distance r in the unit normal direction to obtain the point q. The collection of such points is called the offset-curve C_r. Clearly

$$C_r(t) = C(t) + r \cdot N(t)$$

Even when $C(t)$ is a B-spline, C_r need not be, and must be constructed by using the sampling technique. Note that frequently the r is too large and bad situations arise. This is illustrated on the right in Fig. 35, where the offset curve defined above will self-intersect. This must then be detected and cleaned up.

Extrudes: This is a method of constructing new surfaces from given curves. Say, $C(t)$ (for $t \in [a, b]$) is a curve in the plane with unit normal w. The extrude operation takes as input this curve C and a signed distance d. The extrude surface S is obtained by moving the curve C in the direction w by distance d. See Fig. 36 for an illustration. The parametrization of the surface S is easily given. If $t \in [a, b]$ and $u \in [0, d]$, then

$$S(t, u) = C(t) + v \cdot w$$

It is clear that this is indeed the required surface. Further note that if C is a B-spline curve with control points $P = (p_1, \ldots, p_n)$, then $S(t, u)$ is also a B-spline tensor-product surface with the degree/knot vector in t the same as in C. The degree in u is obviously 1 and thus S may be represented by by $2 \times n$-matrix Q of control points:

$$Q = \begin{bmatrix} p_1 & p_2 & \ldots & p_n \\ p_1 + d \cdot w & p_2 + d \cdot w & \ldots & p_n + d \cdot w \end{bmatrix}$$

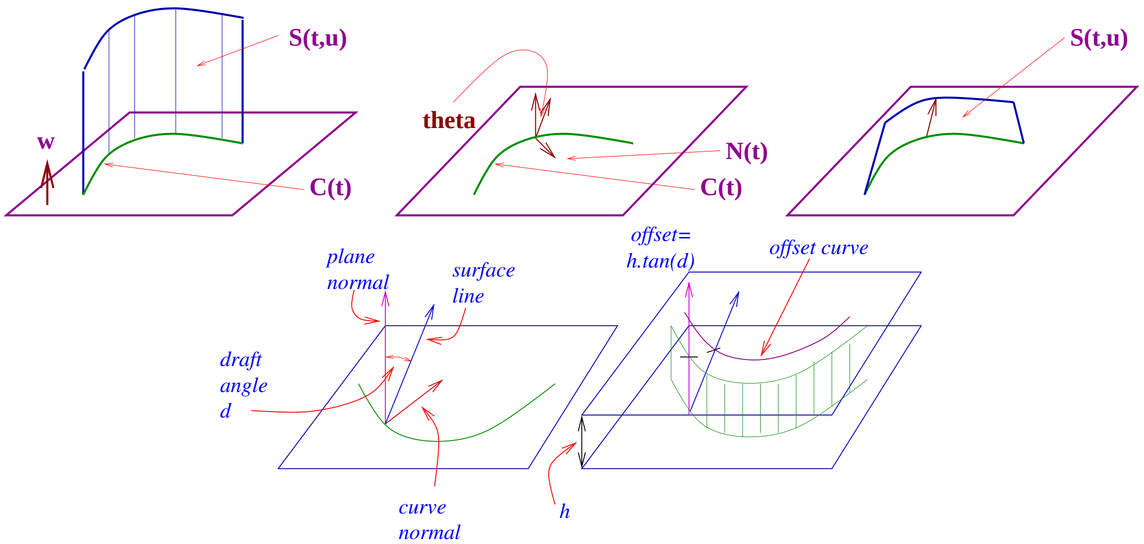

Figure 36: The extruded surface

This settles the simple extrude. A more complicated extrude is the *drafted* extrude. Here, the inputs are

- the curve $C(t)$ in a plane with unit-normal w.
- A distance d.
- An angle θ, and
- A unit normal $N(t)$ to the curve within the plane.

It is intended that as the curve moves along w, it *moves inwards* with an angle θ. Thus, while the walls of the surface in the normal extrude are "vertical" to the plane, here they make an angle θ. The equation of the surface is:

$$S(t,u) = C(t) + \cos\theta \cdot w + \sin \cdot \theta N(t)$$

The surface $S(t,u)$ is thus easily defined. However, in this case, even when $C(t)$ is a B-spline, the surface usually is not expressible as a B-spline, and we must use the sampling approach.

Offset Surface: The input to this construction is a surface $S(u,v)$ with unit surface normal $N(u,v)$, and a distance d. The offset surface $SO(u,v)$ is defined as:

$$SO(u,v) = S(u,v) + d \cdot N(u,v)$$

Thus, the offset surface is obtained by moving each point of $S(u,v)$ d times the unit normal $N(u,v)$ at that point. The offset surface is also the locus of the center of a sphere of radius d rolling on the surface S. Even when $S(u,v)$ is a B-spline surface, $SO(u,v)$ is usually not and a sampling must be used to construct an approximation to the offset surface. The usual problems are highlighted on the right of Fig. 37.

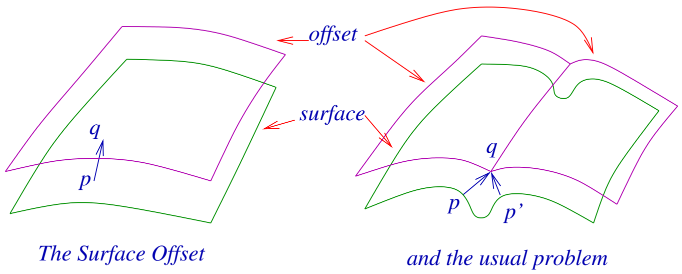

Figure 37: The offset surface

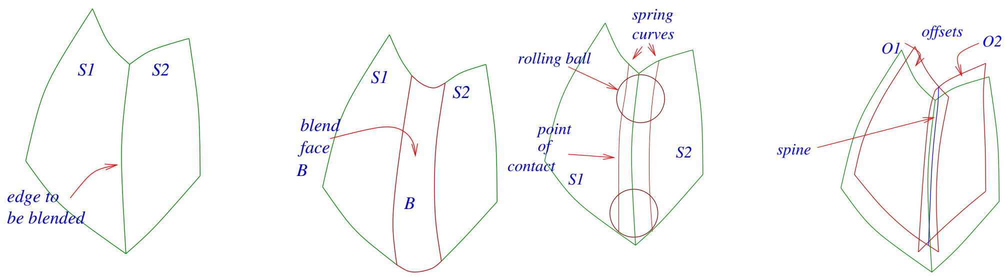

Figure 38: The blend surface

The constant-radius blend: This is a rather delicate contruction involving two surfaces S_1 and S_2 to create a *transition* blend surface. See Fig. 38. We see there that two surfaces S_1 and S_2 meet along a sharp edge. It is desired to remove this sharp edge. This is done by rolling a ball of radius r such that it is in contact with both surfaces. The envelope of this rolling ball creates a surface. Parts of S_1 and S_2 and the edge in between are removed and this envelope inserted. Thus a smooth transition surface B between S_1 and S_2 is created.

Important computational structures related to the blend construction are listed below (also see the RHS of Fig. 38).

- **The Spine** is the locus of the center of the rolling ball which is in contact with both surfaces. Observe that the spine is constructed by the intersection of the offset surfaces O_i obtained by offsetting S_i by r.

- **The Spring Curves** are the locus of the point of contact on either surface. It is along the spring curves that the surfaces S_i have to be trimmed and the blend surface inserted.

The computation of the spine as a parametrized curve $Sp(t)$ is done first. Now, from the mathematics, its clear that either spring curve $C_i(t)$ is merely the projection of the spine $Sp(t)$ on the surface S_i. Thus, for a parameter value t, we have $p = Sp(t)$ and $q_i = Proj(p, S_i)$ lying on the spring curve $C_i(t)$. By a further stretch of imagination, the great circle R with center p and joining q_1 with q_2 is on the blend surface B. Thus the blend surface is obtained by the collection of great circles $R(t)$ for each spine point $Sp(t)$. This is shown in Fig. 39. Each $R(t)$ is an arc of a circle of

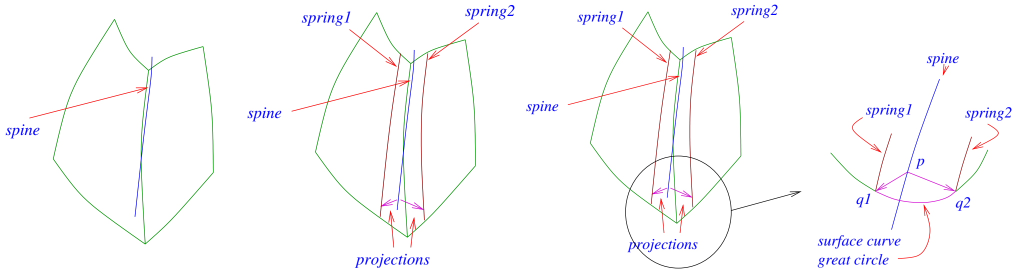

Figure 39: The spring curves and the cross curve

radius r and thus may be parametrized by $0 \leq \theta \leq \alpha(t)$, where $\alpha(t)$ will certainly depend on t. We thus have the parametrization of the blend surface $B(t, \theta)$.

The construction of the blend surface thus uses offsets, intersections and projections in a novel way to create a very useful surface. There are, in actual kernels, several variations of this theme and general blends are still an active area of research.

10.5 The abstract constructor

From the examples that we have seen earlier, we see that there are four main issues to be handled, when a kernel operation is to be defined. These are:

- **The mathematical definition** states in a precise manner the definition of the curve/surface or other entities to be created or modified. This is usually at the geometry level with topological modifications to be subsumed in a different stage.
- **A formulation for iterative solutions** stipulates the the problem as an n-variable, n-equation system which is to be solved by NR methods or their variations.
- **An analysis of convergence** of the methods proposed.
- **Parametrization** of all the structures proposed.

We disregard the **Topological machinery** to migrate the kernel operation from the geometric curve/surface entities to the edge/face topological entities.

To illustrate this, we use the contsruction of the projection of a curve $C(t)$ on a surface $S(u, v)$ to obtain the curve $D(t)$. The mathematical definition of the structure $D(t)$ was obtained in three steps.

- Choose a sequence (t_i) in the domain $[a, b]$ of $C(t)$. This choice may be done on the basis of the expected tolerance and the convolutedness of the surface and curves.
- Create a sequence (q_i) and surface parameters (u_i, t_i) obtained by projecting $p_i = C(t_i)$.
- String up the (q_i, u_i, v_i) into geometric and parametric curves by using them as control points. The resulting structure would be $D(t)$.

Thus $D(t)$ would actually be a B-spline and its parametrization could be quite different from $C(t)$, and further, it would match the exact projected curves only upto a tolerance.

This settled the points of *mathematical definition* and *parametrization* above. The need for iterative structures was in the process of computing the projections q_i from $C(t_i)$. Once these were computed, there was no need for any iterative structures at all. The convergence of the initial iterators is discussed later, in a more general framework.

There is, however, another approach possible. This is called the *abstract* or *procedural* approach and is gaining some currency in modern kernels.

It relies on the fact that, as far as the kernel operations are concerned, a curve is merely a collection of programs which allow their callers to (i) access key attributes of teh curve, and (ii) evaluate a curve and its derivatives at a parameter value which is passed by the caller. Thus in other words, a curve class may have the following functions:

- `mycurve.domain`, accesses the domain of definition of the curve
- `mycurve.dimension` is the dimension of the model space
- `mycurve.eval(t0,d)` returns the d-th derivative of $mycurve(t)$ evaluated at t_0.

Perhaps, these are the only APIs which suffice to define a curve. If `mycurve` is a B-spline, then perhaps additional functions make sense, such as `mycurve.degree` and `mycurve.knotv`. Most applications use only the basic curve APIs above and the derived functions.

In view of this, it makes sense to define a curve construction only in terms of the availability of the three basic curve APIs mentioned above. This is exactly the *abstract* or *procedural approach*.

For the specific problem of constructing the *exact* projection curve $PC(t)$, we define the auxiallary functions $u(t), v(t)$ and define them as the solutions to:

$$\begin{array}{c}\langle C(t) - S(u(t), v(t)), \frac{\partial S}{\partial u}(u(t), v(t))\rangle = 0 \\ \langle C(t) - S(u(t), v(t)), \frac{\partial S}{\partial v}(u(t), v(t))\rangle = 0\end{array}$$

Note that this is the same formulation as an NR as before, except that the "coefficients" of the problem are now parametrized by t. Thus, for any specific $t = t_0$, we merely evaluate the point $C(t_0)$ and then solve the NR problem to obtain $u(t_0), v(t_0)$.

This is akin to the following example: If $A(t)$ and $b(t)$ are $n \times n$ and $n \times 1$ matrices, then we define $x(t)$ as the solution to:

$$A(t)x(t) = b(t)$$

Coming back, we define $PC(t) = S(u(t), v(t))$. Clearly this is exactly what we wanted. Thus, the pseudo code for $PC(t)$ is as follows:

```
0.  function evaluate PC
1.  Input t, initial guesses u0,v0
2.  Evaluate p=C(t);
3.  compute (U,V)=Project(S,p,u0,v0); % u0,v0 to serve as NR init
4.  PC=S(U,V);
5.  return;
```

Let us look at the various attributes of the curve $PC(t)$. We see that `PC.dimension=3`, `PC.domain=[a,b]` where $[a, b]$ is the appropriate sub-domain of C and has to be computed by an honest calculation. The above procedure does indeed almost supply a procedure to evaluate the projection curve. There are two weaknesses: (i) how do we compute the initial inputs u_0, v_0 to seed the NR process, and (ii) a curve requires not only it but its derivatives to be evaluated. How do we do that.

Let us tackle the first issue. We observe that the process of computing the approximation curve in the earlier approach serves to supply the initial u, v-values. In other words, we construct and store a B-spline *Approx-PC(t)*, which returns the approximate values to $u(t), v(t)$. We modify the pseudo-code as:

```
function evaluate PC
Input t, Approx_PC
Evaluate p=Approx_PC(t);
compute (U,V)=Project(S,p,u0,v0); % u0,v0 to serve as NR init
PC=S(U,V);
return;
```

The second issue is more mathematical. We look at the defining equations of $u(t), v(t)$:

$$\begin{aligned}\langle C(t) - S(u(t), v(t)), \tfrac{\partial S}{\partial u}(u(t), v(t))\rangle &= 0\\ \langle C(t) - S(u(t), v(t)), \tfrac{\partial S}{\partial v}(u(t), v(t))\rangle &= 0\end{aligned}$$

Let the primed variables denote the derivatives w.r.t. t and S_u denote $\frac{\partial S}{\partial u}$, and so on. Note that

$$\frac{dS(u(t), v(t))}{dt} = \frac{\partial S(u(t), v(t))}{\partial u}\frac{du}{dt} + \frac{\partial S(u(t), v(t))}{\partial v}\frac{dv}{dt}$$

Thus, differentiating the defining equation with respect to t, and suppressing in the notation that everything depends on t, we have:

$$\begin{aligned}\langle C' - u' \cdot S_u(u, v) - v' \cdot S_v(u, v), S_u(u, v)\rangle + \langle C - S(u, v), u' \cdot S_{uu}(u, v) + v' \cdot S_{uv}(u, v)\rangle &= 0\\ \langle C' - u' \cdot S_u(u, v) - v' \cdot S_v(u, v), S_v(u, v)\rangle + \langle C - S(u, v), u' \cdot S_{vu}(u, v) + v' \cdot S_{vv}(u, v)\rangle &= 0\end{aligned}$$

For a specific t_0, we have (by our earlier procedure) $u(t_0), v(t_0)$. Thus the only unknowns in the equations above are u', v', i.e., $du/dt(t_0)$ and $dv/dt(t_0)$. Note that $C'(t_0), S_u(u(t_0), v(t_0))$ and other similar terms are all evaluations of the derivatives of the curve C and surface S at specific u, v and thus result in 3-dimensional vectors. Thus the above system results in a 2×2 linear system in u', v'. After obtaining these values, we use $PC(t) = S(u(t), v(t))$ to obtain:

$$PC' = u' \cdot S_u + v' \cdot S_v$$

Thus the tangent to the abstract curve $PC(t)$ is also "abtractly computable". The reader should be convinced that higher derivatives are similarly computable.

Thus my abstract projection curve is a 2-tuple of

- An approxmate projection curve to seed the solvers.
- Procedures to compute $PC(t)$ and its derivatives for any given value of t. These procedures may call similar evaluators for the input curve $C(t)$ and surface $S(u, v)$.

This finishes the description of the abstract projection curve. Let us pause to understand what has been achieved and at what costs and benefits. Firstly, in comparison to the earlier elementary approach,

- It bypasses the selection of the sequence (t_i) and its inherent difficulty.
- It matches the exact projection curve and thus more accurate.
- It comes with a natural parametrization.

In practice, the NR-solvers are rather fast and the computational overhead in not maintaining the projection curve as a B-spline are usually small. Furthermore, the conceptual framework generally allows a cascade (such as in blends) of kernel-operations without the cost of accuracy. The method has a host a problems, but usually these are present in the elementary approach as well. One example is when the "natural" parametrization of the projection curve leads to large accelerations and deccelerartions (i.e., large values of $PC''(t)$). In this case, the elementary construction too will have the same problem.

In summary, the abstract framework is much closer to the mathematical definition of a curve or surface and generally leads to an easier programmability.

10.6 Two numerical examples

Finally, we consolidate the material above with two numerical examples around the curve projection problem. Besides illustrating some numerical aspects of the problem, they also motivate the notion of curve and surface curvatures.

The projection of a curve on a cylinder. Consider the curve and the cylinder given below:

$$C(t) = (t, t, 2) \quad S(u, v) = (\sin v, u, \cos v)$$

Thus $C(t)$ is a straight-line on the $z = 2$ plane, while $S(u, v)$ is a cylinder with the Y-axis as its axis and radius 1.

We see that $p = C(0) = (0, 0, 2)$ and that $q = S(0, 0) = (0, 0, 1)$ and thus $p - q = (0, 0, 1)$. We see that:

$$\begin{array}{rcl} S_u = \frac{\partial S}{\partial u}(0,0) & = & (0,1,0) \\ S_v = \frac{\partial S}{\partial v}(0,0) & = & (\cos v, 0, -\sin v) \quad = \quad (1,0,0) \end{array}$$

Since $\langle S_u, p-q\rangle = \langle S_v, p-q\rangle = 0$, we see that the projection of the point p is indeed q. Next, let us abstract definition of the projection curve as $PC(t)$. We will compute $PC'(0)$, the tangent to the projected curve at 0. For this, we will need:

$$\begin{array}{rcl} S_{uu}(0,0) & = & (0,0,0) \\ S_{uv}(0,0) & = & (0,0,0) \\ S_{vv}(0,0) & = & (-\sin v, 0, -\cos v) \quad = \quad (0,0,-1) \\ C'(0) & = & (1,1,0) \end{array}$$

The equations to compute u', v' may now be computed:

$$\begin{array}{rcl} \langle (1,1,0) - u'\cdot(0,1,0) - v'\cdot(1,0,0), (0,1,0)\rangle + \langle (0,0,1), u'\cdot(0,0,0) + v'\cdot(0,0,0)\rangle & = & 0 \\ \langle (1,1,0) - u'\cdot(0,1,0) - v'\cdot(1,0,0), (1,0,0)\rangle + \langle (0,0,1), u'\cdot(0,0,0) + v'\cdot(0,0,-1)\rangle & = & 0 \end{array}$$

This gives us:

$$\begin{array}{rcl} 1 - u' & = & 0 \\ 1 - 2v' & = & 0 \end{array}$$

Or in other words, $u' = 1$ and $v' = 0.5$ and thus

$$PC'(0) = u' \cdot S_u + v' \cdot S_v = (0.5, 1, 0)$$

Thus, we see that the projection curve moves in the direction $(0.5, 1, 0)$ at $t = 0$. Now consider the half-cylinder and half-plane scenario as shown in Fig. 40 and note that the composite projection curve is *not differentiable* at $t = 0$ since on the plane part $PC'(0) = (1, 1, 0)$ which is **not** equal to

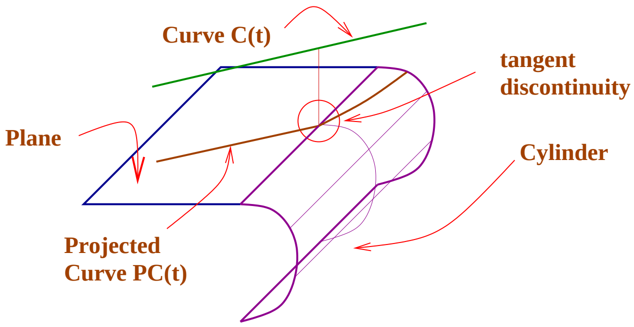

Figure 40: The projection curve on a junction

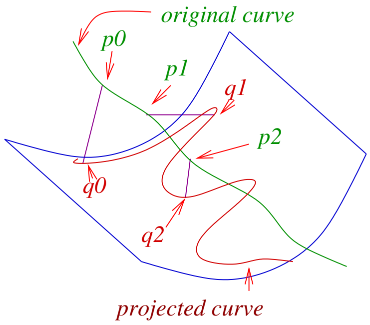

Figure 41: The projection curve: variations

the tangent on the cylindrical part. We will see that this is because, though there is only geometric C^1-continuity between the two surfaces, and not C^2. This is better explained using the notion of surface curvatures, a notion we will introduce later.

The 2D-projection: Let us understand the *sensitivity* of the projection operation. Consider and almost cylinder and a curve which passes close to the axis of the cylinder as in Fig. 41. It seems intuitive that the variations in the curve get exaggerated while projection. We will explain this fact in the simpler 2-dimensional situation.

Consider a a point $p = (0, h)$ which projects on to the point $q = (0, 0)$ on a curve $y = a(x)$ (see the left of Fig. 42). Suppose that

$$a(x) = \frac{a_2}{2!}x^2 + \frac{a_3}{3!}x^3 + \ldots$$

Clearly $a_0, a_1 = 0$ since the curve passes through $(0, 0)$ and has slope 0 at that point.

Our question is that if we move the point p to the point $p' = (\epsilon, h)$, for small ϵ, where does the projection point q' go? If $q' = (x, a(x))$, then we have the equation:

$$\frac{h - a(x)}{\epsilon - x} \cdot a'(x) = -1$$

Which merely says that the product of the slopes of the line $p'q'$ and the tangent at q' is -1. In

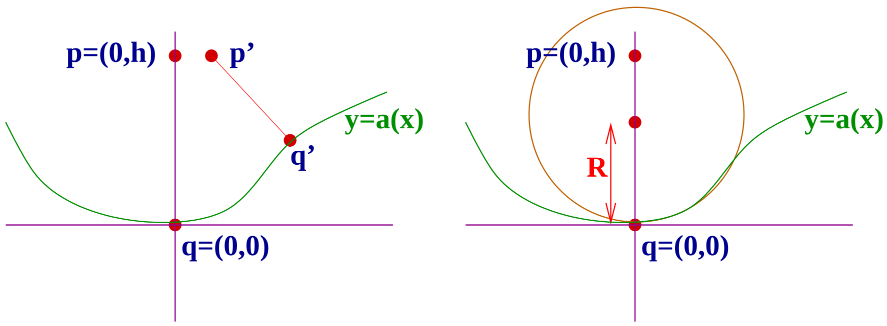

Figure 42: The 2D projection

other words:

$$(a(x) - h) \cdot a'(x) + x = \epsilon$$

This expands to:

$$(1 - a_2 h)x + \ldots = \epsilon$$

Thus

$$x = \frac{1}{1 - a_2 h}\epsilon + \ldots \text{ higher terms}$$

It is thus clear that the instability of the projection will depend on the term $1 - a_2 h$. If this term is close to 0 then higher will be the instability. We will next interpret the term a_2. For this, we will make a small calculation. Consider the circle of radius R with center $(0, R)$. Note that this is tangent to the curve $y = a(x)$ and thus will have the form

$$y = \frac{b_2}{2!}x^2 + \frac{b_3}{3!}x^3 + \ldots$$

Let us evaluate the term b_2. We see that

$$\begin{aligned} y &= R - \sqrt{R^2 - x^2} \\ &= R - [R(1 - \tfrac{x^2}{2R^2} + \ldots)] \\ &= \tfrac{x^2}{2R} + \ldots \end{aligned}$$

and thus $b_2 = 1/R$. Whence we see that $r = 1/a_2$ is the radius of best-fit circular approximation to the function $y = a(x)$. Thus, the term

$$1 - a_2 h = 1 - \frac{h}{r}$$

Hence, closeness of p to the center of the best-fit circle leads to the instability of the projection.

The quantity $r = 1/a_2$ is called the **radius of curvature** of the curve at that point, and is an invariant upto translation and rotations. We will examine later, how to compute the curvature for a arbitrary point on a general curve.

11 Tangent Space

In this section, we will begin with the *local* analysis of curves and surfaces. The first notion will be of the **tangent space** which we is associated with most points on a curve/surface. Next we analyse the effect of continuous and differentiable maps on the tangent spaces.

For a point $p \in \mathbb{R}^n$ and $\epsilon > 0$, the set $B_\epsilon(p)$ will denote the open ball of radius ϵ around p. Let

$$\mathcal{I}_n = \{(i_1, \ldots, i_n) | i_j \geq 0\}$$

and for $I \in \mathcal{I}$, let x^I be $x_1^{i_1} \ldots x_n^{i_n}$. A function $f : \mathbb{R}^n \to \mathbb{R}$ is called analytic at p if there is an $\epsilon > 0$ such that for all $x \in B_\epsilon(p)$, and constants $a_I \in \mathbb{R}$ for every $I \in \mathcal{I}_n$ such that

$$f(x) = \sum_{I \in \mathcal{I}_n} a_I (x - p)^I$$

Thus, $f(x)$ has a Taylor expansion around x. It is a classical result that an analytic function is infinitely differentiable.

Definition 11.1 *Let $S \subseteq \mathbb{R}^n$ and $p \in S$. We say that S is a d-dimensional* implicitly defined manifold *at p if there is an $\epsilon > 0$ and functions $f_1, \ldots, f_{n-d} : \mathbb{R}^n \to \mathbb{R}$, analytic at p such that*

1. *$S' = B_\epsilon(p) \cap S$ is given by*

$$S' = \{x \in B_\epsilon(p) | f_i(x) = 0 \; for \; 1 \leq i \leq n - d\}$$

   Thus, S' is locally defined as the zeros of $n - d$ analytic functions.

2. *The $(n - d) \times n$-matrix*

$$J(f, p) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(p) & \cdots & \frac{\partial f_1}{\partial x_n}(p) \\ \vdots & & \vdots \\ \frac{\partial f_{n-d}}{\partial x_1}(p) & \cdots & \frac{\partial f_{n-d}}{\partial x_n}(p) \end{bmatrix}$$

   is of rank $n - d$.

For a function $f : \mathbb{R}^n \to \mathbb{R}$, the vector $\nabla f(p)$ (pronounced as 'grad'f), is the vector

$$\nabla f(p) = [\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}](p)$$

Thus, for a sequence $(f_1, \ldots, f_{n-d})$, we have:

$$J(f, p) = \begin{bmatrix} \nabla f_1(p) \\ \vdots \\ \nabla f_{n-d}(p) \end{bmatrix}$$

Definition 11.2 *Let $S \subseteq \mathbb{R}^n$ and $p \in S$. We say that S is a d-dimensional* parametrized manifold *at p if there is an $\epsilon > 0$ and a $\delta > 0$ and functions $g_i : \mathbb{R}^d \to \mathbb{R}$, for $i = 1, \ldots, n$ all analytic at $\overline{0} = (0, 0, \ldots, 0) \in \mathbb{R}^d$ such that $(g_1(\overline{0}), \ldots, g_n(\overline{0})) = p$ and*

1. *For every point x in $S' = B_\epsilon(p) \cap S$, there is a unique $t = (t_1, \ldots, t_d) \in B_\delta(\overline{0})$ such that*

$$(x_1, \ldots, x_n) = (g_1(t), \ldots, g_n(t))$$

   Thus S' is parametrized by n analytic functions on a neighborhood $B_\delta(\overline{0}) \subseteq \mathbb{R}^d$.

2. *The $d \times n$-matrix*

$$K(f,p) = \begin{bmatrix} \frac{\partial g_1}{\partial t_1}(\overline{0}) & \ldots & \frac{\partial g_n}{\partial t_1}(\overline{0}) \\ \vdots & & \vdots \\ \frac{\partial g_1}{\partial t_d}(\overline{0}) & \ldots & \frac{\partial g_n}{\partial t_d}(\overline{0}) \end{bmatrix}$$

   is of rank d.

Here is the important theorem relating the two definitions which we state without proof.

Theorem 11.3
- *If $S \subseteq \mathbb{R}^n$ and if $p \in S$ is an implicitly defined manifold point then it is also parametrized manifold point. In other words, for the analytic functions $f_1, \ldots, f_{n-d}$, as above, there are $g_1, \ldots, g_n$ such that the image of g consides with the zeros of f.*
- *If $p \in S$ is a parametrized manifold point then it is also an implicitly defined manifold point. In other words, for $g_1, \ldots, g_n$ as above, there are $f_1, \ldots, f_{n-d}$ so that the image of g is precisely the zeros of f.*

Example 11.4 *Let $S = \{(x,y) | x^2 + y^2 - 1 = 0\}$ be the unit circle in $\mathbb{R}^2$. We see that S is a 1-dimensional implicitly defined manifold at $p = (1,0)$. Consider the function $f = x^2 + y^2 - 1$, we have*

$$J(f,p) = \nabla f(p) = \begin{bmatrix} 2x & 2y \end{bmatrix}(p) = [2, 0]$$

which is of rank 1. For the point p, consider

$$g = (g_1, g_2) = \left(\frac{1-t^2}{1+t^2}, \frac{2t}{1+t^2}\right)$$

We see that $g(0) = p$ and that

$$K(g,p) = \begin{bmatrix} \frac{-4t}{(1+t^2)^2}(0) & \frac{2(1-t^2)}{(1+t^2)^2}(0) \end{bmatrix} = [0, 2]$$

which is of rank 1. Thus S is both implicit and parametrized at p. In fact, every point $q \in S$ is a manifold point. Note that for $q = (-1, 0)$, the above g's do not work and some other g's have to be constructed. The f above does work at q as well. Also note that

$$J(f,p)K(g,p)^T = 0$$

Example 11.5 *For the set $S = \mathbb{R}^n \subseteq \mathbb{R}^n$, every point p is an n-manifold point. There are $0 = n - n$ defining equations and n parametrizing functions, viz., $g_i = t_i$.*

Proposition 11.6 *If $p \in S$ is a manifold point with implicit data $(f_i)_{i=1}^{n-d}$ and parametric data $(g_j)_{j=1}^n$, then*

$$J(f,p)K(g,p)^T = 0$$

Proof: Let us the i-th row of $J(f,p)$ and the j-th row of $K(g,p)$ and show that:

$$\sum_k \frac{\partial f_i}{\partial x_k}(p)\frac{\partial g_k}{\partial t_j}(\overline{0}) = 0$$

Since the f's and the g's define the same neighborhood of p, we have:

$$f_i(g_1(t_1,\ldots,t_d), g_2(t_1,\ldots,t_d),\ldots,g_n(t_1,\ldots,t_d)) = 0$$

Treat the LHS as a function of $t_1,\ldots,t_d$ and differentiate it with respect to t_j at the point $\overline{0}$. Observing that $g(\overline{0}) = p$ and applying the chain rule gives us the result. □

Proposition 11.7 *Let $p \in S \subseteq \mathbb{R}^n$ be a manifold point via the implict sequence (f_i) and the parametric sequence (g_j). The* **tangent space** *TS_p is defined as:*

A. All row vectors $v \in \mathbb{R}^n$ such that $J(f,p)\cdot v^T = 0$.

B. All row vectors $v \in \mathbb{R}^n$ such that v is in the row-space of $K(g,p)$.

Then, (i) the two definitions A,B above are equivalent, and (ii) they are independent of the choice of the sequences (f) and (g). Thus, TS_p is a d-dimensional subspace of $\mathbb{R}^n$.

Proof: The part (i) above is a direct consequence of Prop. 11.6 above. After all any v satisfying A must be in the null-space of $J(f,p)$. Since $rank(J(f,p) = n-d$, this null-space must be d-dimensional. On the other hand $K(g,p)$ is d-dimensional and thus must span this null-space, whence B follows. The converse is similar. Next, if (g') was yet another parametrizing family, then we would still have

$$J(f,p)K(g',p)^T = 0$$

Thus both $K(g,p)$ and $K(g',p)$ would span the same null-space of $J(f,p)$. This proves invariance under change of parametrization. The other case is similar. □

Thus, the rows of $K(g,p)$ span the tangent space TS_p, while the rows of $J(f,p)$ span the space of *normals* to the tangent space.

Thus, we have looked at sets $S \subseteq \mathbb{R}^n$ and their algebraic definitions, either as parametrized sets or implicitly defined sets. However, for most purposes, this is unsatisfactory, since most objects are defined by a *single* parametrization or implicitization. We have the following definition:

Definition 11.8

- *Let $g : \mathbb{R}^d \to \mathbb{R}^n$ be an analytic function such that $g(0) = p$. We say that p is (parametrically) non-singular if $rank(K(g,p)) = d$.*
- *Let $f_1,\ldots,f_{n-d} : \mathbb{R}^n \to \mathbb{R}$ be analytic functions which vanish at $p \in \mathbb{R}^n$. We say that p is (implicitly) non-singular if $J(f,p)$ has rank $n-d$.*

Let us now look at the standard parametrized curves and surfaces. Let $g : I \to \mathbb{R}^3$ be a parametrized curve C. Thus $g = (g_1(t), g_2(t), g_3(t))$ are the three parametrizing functions. Let $g(t_0) = p$ be a particular point on the curve C. We would like to argue that a point $p = g(t_0)$ is a 1-manifold point on C. We see that p is **parametrically non-singular** if $\frac{dg}{dt}|_{t_0} \neq 0$. This ensures that $K(g,p)$ is of rank 1. If p is non-singular, then $v = \frac{df}{dt}|_{t_0}$ is a vector in $\mathbb{R}^3$. The space TC_p, the tangent space at p to the curve C, is the linear subspace of $\mathbb{R}^3$ generated by v.

Next, let $g : I^2 \to \mathbb{R}^3$ parametrize a surface S. Let $p = g(u_0, v_0)$ be a fixed point on S. We see that p is **parametrically non-singular** if the vectors $v_1 = \frac{\partial g}{\partial u}(u_0,v_0)$ and $v_2 = \frac{\partial g}{\partial v}(u_0,v_0)$ are non-zero and linearly independent. This again ensures that $K(g,p)$ is of rank 2. The space generated by v_1 and v_2 is the tangent space TS_p at p to the surface S.

Example 11.9 *Consider the circle in* $\mathbb{R}^2$ *given by* $g(t) = (\frac{2t}{1+t^2}, \frac{1-t^2}{1+t^2})$. *We see that* $df/dt = (\frac{2(1-t^2)}{(1+t^2)^2}, \frac{-4t}{(1+t^2)^2})$. *We see, for example, that* $g(1) = (1,0) = p$, *and* $dg/dt(1) = (0,-1)$. *We also see that for for all* $t \in \mathbb{R}$, $dg/dt(t) \neq 0$, *and thus every point on the circle (minus the special point) is non-singular. The tangent space* TC_p *is generated by* $(0,-1)$ *in* $\mathbb{R}^2$. *The implicit definition is given by* $f(x,y) = x^2 + y^2 - 1 = 0$. *For this,* $J(f,p) = \nabla f(p) = (2,0)$ *and thus is the normal to the curve at* p.

Usually, non-singular points of curves or surfaces are actually manifold points. Exceptions arise when the parametrization is such that the curve/surface *crosses itself.* Thus, it may be that $p = g(t_1) = g(t_2)$, whence there is no ϵ-ball around p so that $C \cap B_\epsilon(p)$ equals the image of g for an interval around t_1.

Example 11.10 *Let* $g(t) = (\sin 2t, \cos t)$ *be a curve in the plane. We see that*

$$\begin{array}{rcl} g(\pi/2) = (\sin\pi, \cos\pi/2) & = & (0,0) \\ g(3\pi/2) = (\sin 3\pi, \cos 3\pi/2) & = & (0,0) \end{array}$$

Furthermore $g'(t) = (2\cos 2t, -\sin t)$, *and thus:*

$$\begin{array}{rcl} g'(\pi/2) = (2\cos\pi, -\sin\pi/2) & = & (-2,-1) \\ g'(3\pi/2) = (2\cos 3\pi, -\sin 3\pi/2) & = & (-2,1) \end{array}$$

Thus, the curve intersects iteself at $p = (0,0)$ *with two different tangents at* p. *Such a point is called a* **double point** *of the curve (See Fig. 43). The implicit definition of the curve is easily obtained:*

$$x^2 = \sin^2 2t = 4\sin^2 t \cos^2 t = 4(1-y^2)y^2$$

Thus the curve is defined by:

$$f(x,y) = x^2 - 4y^2 + 4y^4 = 0$$

We see that:

$$J(f,p) = \nabla f(p) = [2x, -8y + 16y^3](p) = [0,0]$$

Thus p *is an implicitly singular point.*

Example 11.11 *Let us look at* $g(t) = (t^2, t^3)$ *and* $f(x,y) = x^3 - y^2 = 0$. *The plot of this function is the* **cusp** *in Fig. 43. For the point* $p = g(0) = (0,0)$, *we see that* $K(g,p) = [0,0]$ *and thus is a singular point for the parametrization above. On the other hand, we see that*

$$K(f,p) = [3x^2, 2y](p) = (0,0)$$

Thus p *is singular for the implicit definition as well. Intuitively, this is because there is a unique tangent at* p *but it is a* double root.

The above examples illustrate the following:

- Some parametrically singular points may actually be non-manifold points. This happens because of self-intersections.
- Singularity in the implicit form may also be defined suitably and this is usually a more reliable notion.

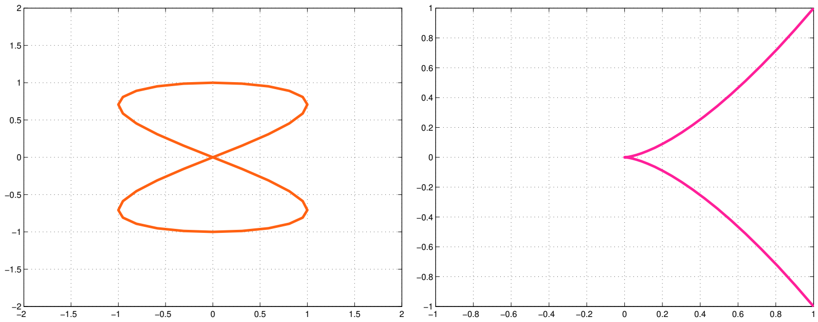

Figure 43: The double-point and the cusp.

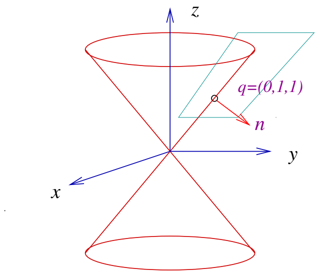

Figure 44: The Cone.

Example 11.12 *Consider the cone X given by*

$$f(u,v) = (\frac{2uv}{1+u^2}, \frac{(1-u^2)v}{1+u^2}, v)$$

We consider the point $f(0,0) = (0,0,0) = p$. We see that:

$$\begin{aligned} f_u(u,v) &= (\frac{(2(1-u^2)v}{(1+u^2)^2}, \frac{-4uv}{(1+u^2)^2}, v) \\ f_v(u,v) &= (\frac{2u}{1+u^2}, \frac{(1-u^2)}{1+u^2}, 1) \end{aligned}$$

We thus see that at $(0,0)$, we have $f_u(0,0) = 0$ and thus the apex of the cone is a singular point. On the other hand, $(u,v) = (0,1)$, we get $f_u = (2,0,0)$ and $f_v = (0,1,1)$ and thus $q = (0,1,1) = f(0,1)$ is a non-singular point. The tangent space TX_q is given by the span of $(2,0,0)$ and $(0,1,1)$ in $\mathbb{R}^3$. See Figure 44. The same cone above is implicitly defined by the equation $X^2 + Y^2 - Z^2 = 0$. We see that $J(f,*) = [2X, 2Y, -2Z]$. Thus, we see that $p = (0,0,0)$ is a singular point. On the other hand, for $q = (0,1,1)$, we see that $\nabla f(p) = [0,1,-1]^T$. We will denote this vector by $\overline{n}$. Note that $[2,0,0].\overline{n} = [0,1,1].\overline{n} = 0$ and thus here again, the tangent space definitions match.

Example 11.13 *Let S be a tensor-product bezier surface of degree 2 in both u and v. Thus $f(u,v) = \sum_{i=0}^{2}\sum_{j=0}^{2} P[i,j]B_i^2(u)B_j^2(v)$. we see that $f_u(0,0) = 2(P[1,0] - P[0,0])$ and $f_v(0,0) = 2(P[0,1] -$*

$P[0,0]$). Thus the parameter point $(0,0)$ is non-singular iff $P[1,0]-P[0,0]$ and $P[0,1]-P[0,0]$ are non-zero and non-colinnear. In other words, $P[0,0], P[0,1]$ and $P[1,0]$ must be non-co-planar.

For a general (u_0, v_0), we see that $f_u(u_0, v_0)$ and $f_v(u_0, v_0)$ are obtained as the tangent vectors to the iso-parametric curves $C(*, v_0)$ and $C(u_0, *)$. See Figure 19.

Theorem 11.14 *(i) Let S be a surface and $p=(x_0, y_0, z_0)$ be a non-singular point on S. Further, let TS_p be the tangent space and $\overline{n}$ a normal to it. If $\overline{n}$ is such that $[0,0,1].\overline{n} \neq 0$, then there is an open subset $Q \subseteq \mathbb{R}^2$ containing (x_0, y_0) and a function $f: O \rightarrow \mathbb{R}$, such that for all $(x,y) \in O$, the point $(x, y, f(x,y))$ lies on the surface S.*
(ii) Let C be a curve and $p=(x_0, y_0, z_0)$ be a non-singular point on C. Let $v \in TC_p$ be such that $v.[1,0,0]^T \neq 0$, then there is an open interval $O \subseteq \mathbb{R}$ containing x_0 and two functions $f, g: O \rightarrow \mathbb{R}$ such that for all $x \in O$, the point $(x, f(x), g(x)$ lies on the curve.

Now that we have seen some examples of tangent space computations, let us see how tangents spaces of different manifolds relate to each other. The simplest result is the following:

Proposition 11.15 *Let C be a curve on a surface S, and p be a non-singular point on C. Then $TC_p \subseteq TS_p$.*

Proof: Let $f(x,y,z)=0$ be a local definition of the surface and $g(t)=(x(t), y(t), z(t))$ be a local parametrization of the curve. Since $C \subseteq S$, we see that:

$$f(x(t), y(t), z(t))=0$$

Upon differentiating this, we get:

$$J(f,p)K(g,p)=0$$

But since the row-span of $K(g,p)$ is TC_p, and that it annihilates $J(f,p)$, we see that $TC_p \subseteq TS_p$. $\square$

Our next objective is to get an alternative understanding of the individual vectors $v \in TS_p$ for an $S \subseteq \mathbb{R}^n$. Let ϵ be a quantity so that $\epsilon^2=0$. Let $S \subseteq \mathbb{R}^n$ be a d-manifold at $p \in S$, and is, for convenience, implicitly defined by $f_1, \ldots, f_{n-d}$. Let $v \in \mathbb{R}^n$ and ϵ an abstract quantity which is so small that $\epsilon^2=0$. We say that $p+\epsilon v$ satisfies f, if on the substitution $p+\epsilon v$, and the assumption that $\epsilon^2=0$, the function does evaluate to zero. As an example, lets look at $(0,1,1)+\epsilon(2,0,0)=(2\epsilon,1,1)$, and the function $X^2+Y^2-Z^2=0$, for the cone. On substitution, we get $\epsilon^2+1-1=0$, and thus the above 'point' does satisfy f.

We define the **infinitesimal directions** at p as all those v such that $p+\epsilon v$ belong to S, or in other words, satisfy $f_1, \ldots, f_{n-d}$.

Proposition 11.16 *For the above data, the set of infinitesimal directions is precisely the set TS_p.*

Proof: We see that for the point $p+\epsilon v$, we have:

$$f_i(p+\epsilon v)=f(p)+\sum_j \epsilon v_j \frac{\partial f_i}{\partial j}(p)+ \text{ higher terms involving } \epsilon^2$$

Since $f(p)=0$ and so is $\epsilon^2=0$, we have:

$$f_i(p+\epsilon v)=\sum_j \epsilon v_j \frac{\partial f_i}{\partial j}(p)=\epsilon J(f_i,p)\cdot v^T$$

Thus $p + \epsilon v$ satisfies f_i iff $J(f_i, p) \cdot v^T = 0$. Thus the set of v so that $p + \epsilon v$ satisfy all the f_i's is precisely the set of v such that $J(f, p) \cdot v^T = 0$. But this precisely defines TS_p. $\Box$

This makes it clear that our mathematical construction of the tangent space indeed matches our intuition.

Next, let $R \subseteq \mathbb{R}^m$ be a r-manifold at $p \in R$ and $S \subseteq \mathbb{R}^n$ be an s-manifold at $q \in S$. Let us assume that $x_1, \ldots, x_m$ index $\mathbb{R}^m$ and $y_1, \ldots, y_n$ index $\mathbb{R}^n$ above. We would like to construct smooth maps $h : R \to S$ and analyse the effect of h on TR_p. If $h(p) = q$, then we claim that nearby points to p go to nearby points to q, or in other words, h sets up a map $h^*_p : TR_p \to TS_q$.

Firstly, how is one to construct a map $h : R \to S$. Perhaps, the simplest way is to define functions $h_1, \ldots, h_n : \mathbb{R}^m \to \mathbb{R}$, and then for any point $x \in R$, define

$$h(x) = (h_1(x), \ldots, h_n(x))$$

Let us assume that such a sequence of functions have been found such that (i) $h(x) \in S$ whenever $x \in R$ and is near p, and (ii) $h(p) = q$. Further assume that $g_1, \ldots, g_m : \mathbb{R}^r \to \mathbb{R}$ locally parametrize R at p, and that $w_1, \ldots, w_{n-s}$ locally define (implicitly) S at p. Thus, we have chosen a parametric definition of R and an implicit definition of S.

Proposition 11.17 *Let $h : R \to S$ be a smooth map such that $h(p) = q$. Then there is a* **natural** *linear map $h^*_p : TR_p \to TS_q$ on the tangent spaces at p and q. Furthermore, if we have the diagram of manifolds:*

$$R \xrightarrow{h} S \xrightarrow{k} T$$

such that $h(p) = q$ and $h(q) = t$, then we have:

$$k^*_q \circ h^*_p = (k \circ h)^*_p$$

Proof: Let $v \in TR_p$, i.e., v is in the row-span of $K(g, p)$. For a point $p + \epsilon v \in R$, we have:

$$\begin{bmatrix} h_1(p+\epsilon v) \\ \vdots \\ h_n(p+\epsilon v) \end{bmatrix} = \begin{bmatrix} h_1(p) \\ \vdots \\ h_n(p) \end{bmatrix} + \epsilon \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_m} \\ \vdots & & \vdots \\ \frac{\partial h_n}{\partial x_1} & \cdots & \frac{\partial h_n}{\partial x_m} \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_m \end{bmatrix}$$

Thus, we see that:

$$h(p + \epsilon v) = q + \epsilon J(h, p) \cdot v^T$$

Since $J(h, p)$ is a real $n \times m$-matrix, we are lead to believe that

$$h^*_p(v) = J(h, p) \cdot v^T$$

is the required map $h^*_p : TR_p \to TS_q$ between the tangent spaces. We next verify that $J(h, p) \cdot v^T$ does indeed belong to TS_q. For that we use the fact that w's define S near q and that for $i = 1, \ldots, n-s$ and all $x \in R$ near p and thus for every $w = w_i$ for any i, we have:

$$w(h_1(x), h_2(x), \ldots, h_n(x)) = 0$$

Since $x = (g_1(t_1, \ldots, t_r), \ldots, g_m(t_1, \ldots, t_r))$, we may differentiate this w.r.t, say t_ℓ and see that:

$$\sum_\ell \sum_k \frac{\partial w}{\partial y_k}(q) \frac{\partial h_k}{\partial x_j}(p) \frac{\partial g_j}{\partial t_\ell} = 0$$

Collecting all i and j, k, ℓ, this is easily abbrieviated as:

$$J(w,q) \cdot J(h,p) \cdot K(g,p)^T = 0$$

Now, we see that $h_p^*(v) = J(h,p) \cdot v^T$ and that since $v \in TR_r$, we have $v = [\alpha_1, \ldots, \alpha_r]K(g,p)$. Whence :

$$J(w,q) \cdot h_p^*(v) = J(w,q) \cdot J(h,p) \cdot K(g,p)^T \cdot \alpha^T = 0$$

Thus $h_p^*(v) \in TS_q$.

The second assertion is clear from the *naturality* of the jacobian map, i.e., the chain rule. If $k : S \to T$ is another map then

$$J(h \circ k, p) = J(h,p) \cdot J(k,q)$$

This finishes the proof. $\square$

There are two unsatisfactory aspects to the above proposition:

- The analysis is based on a map $h : \mathbb{R}^m \to \mathbb{R}^n$, i.e., the *ambient* spaces of the manifold in question. Thus, not only have we implemented a map from R to S but from a whole neighborhood of the point $p \in \mathbb{R}^m$. There are situations where the map $h : R \to S$ is really fundamental to R, though thankfully, it can be extended to a neighborhood of R by general theory, which we skip here.
- Futhermore, the map h_p^* is really implemented by an $n \times n$-matrix. The dimensions of the two tangent spaces TR_p and TS_q are only r and s respectively. Thus, ideally, we should choose a basis $b_1, \ldots, b_r$ of TR_p and $c_1, \ldots, c_s$ of TS_q and express h_p^* as an $r \times s$-matrix for the above chosen basis. The naturality of this map then is clearer than in the earlier map, where ambient dimensions seem to play a role. This sharper naturality is indeed true, i.e., the map $h_p^* : TR_p \to TS_q$ does not depend on its behaviour on the ambient space, but that proof is outside the scope of the discussion here.

Example 11.18 *Let us consider $R \subseteq \mathbb{R}^2$ as given by $f(x_1, x_2) = x_1^2 + x_2^2 - 2 = 0$ and $S \subseteq \mathbb{R}^2$ as given by $w(y_1, y_2) = 4y_1^2 + 9y_2^2 - 72 = 0$. We select $p = (1,1) \in R$ and define $h_1 = 3x_1$ and $h_2 = 2x_2$. We check that*

$$w(h_1, h_2) = 36 \cdot (x_1^2 + x_2^2 - 2)$$

Thus, if p lies on R then $w(h_1, h_2) = 0$ and thus h does indeed take R to S. We see that $J(f,p) = [2,2]$ and thus $[1,-1] \in TR_p$. Next

$$J(h,p) = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \text{ and thus } h_p^*([1,-1]) = [3,-2]$$

Next, we see that $h(p) = [3,2]$ and that

$$J(w,q) = [8y_1, 18y_2] = [24, 36] \text{ and thus } J(w,q) * (h_p^*)^T = 0$$

whence, $[3,-2] \in TS_q$.

The above h is not the only possible map. Consider

$$h_1' = 3x_1 \quad h_2' = 2 \cdot (\sqrt{2 - x_1^2})$$

and see that:

$$w(h_1', h_2') = 36x_1^2 + 36(2 - x_1^2) - 72 = 0$$

Thus, w is identically *zero. In other words, h' is a map from* the whole of $\mathbb{R}^2$ *to S. All the same, note that for any $r \in R$, we have $h(r) = h'(r)$. Next, we see that:*

$$J(h',p) = \begin{bmatrix} 3 & 0 \\ -2x_1(2-x_1^2)^{-1/2} & 0 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ -2 & 0 \end{bmatrix} \text{ and thus } h'^*_p([1,-1]) = [3,-2]$$

Thus, curiously enough, for the vector $[1,-1] \in TR_p$, we see that:

$$h'^*_p([1,-1]) = h^*_p([1,-1]) = [3,-2] \in TS_q$$

This is what we meant by the independence of h^*_p from the behaviour of h on the ambient space.

Example 11.19 *Consider a parametrized surface S given by the map:*

$$S : \mathbb{R}^2 \to \mathbb{R}^3$$

This is implemented by three functions $x(u,v), y(u,v)$ and $z(u,v)$. For any point $p = (u,v) \in \mathbb{R}^2$ the tangent space $T\mathbb{R}^2_p$ is precisely $\mathbb{R}^2$. For a nearby point $p + \epsilon w = (u + \epsilon u', v + \epsilon v')$ we see that

$$S(p+\epsilon w) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix} + \epsilon u' \begin{bmatrix} \frac{\partial x}{\partial u}(u,v) \\ \frac{\partial y}{\partial u}(u,v) \\ \frac{\partial z}{\partial u}(u,v) \end{bmatrix} + \epsilon v' \begin{bmatrix} \frac{\partial x}{\partial v}(u,v) \\ \frac{\partial y}{\partial v}(u,v) \\ \frac{\partial z}{\partial v}(u,v) \end{bmatrix}$$

This may be abbrieviated as:

$$S(p+\epsilon w) = S(u,v) + \epsilon(u' S_u(u,v) + v' S_v(u.v))$$

If the surface is non-singular at $q = S(u,v)$ then the vectors S_u and S_v span the tangent space at TS_q. The map $S^*_p : T\mathbb{R}^2_p \to TS_q$ is given by:

$$S^*_p([1,0]) = S_u \quad S^*_p([0,1]) = S_v$$

The tangent map for the parametrized curve is similar.

Example 11.20 *Consider the cylinder $R \in \mathbb{R}^3$ given by $x^2 + y^2 - 1 = 0$ and the cone $S \subseteq \mathbb{R}^3$ given by $x^2 + y^2 - z^2 = 0$. We consider the "pinching map" $h : \mathbb{R}^3 \to \mathbb{R}^3$ given by:*

$$h(x,y,z) = (g_1(X,Y,Z) = (xz, yz, z)$$

We see that

$$X^2 + Y^2 - Z^2 = z^2 \cdot (x^2 + y^2 - 1)$$

Thus, whenever $x^2 + y^2 - 1 = 0$, the mapped point $h(x,y,z)$ satisfies the equation of the cone. Thus h restricts to a map $h : R \to S$. Let $p = (1,0,1)$ with $q = h(p) = (1,0,1)$. We see that the tangent space to the cylinder R at p is given by:

$$TR_p = \mathbb{R} \cdot (0,1,0) + \mathbb{R} \cdot (0,0,1) = \mathbb{R} \cdot b_1 + \mathbb{R} \cdot b_2$$

We see that

$$h^*_p(v) = \begin{bmatrix} \frac{\partial g_1}{\partial x} & \frac{\partial g_1}{\partial y} & \frac{\partial g_1}{\partial z} \\ \frac{\partial g_2}{\partial x} & \frac{\partial g_2}{\partial y} & \frac{\partial g_2}{\partial z} \\ \frac{\partial g_3}{\partial x} & \frac{\partial g_3}{\partial y} & \frac{\partial g_3}{\partial z} \end{bmatrix}_p \cdot v^T = \begin{bmatrix} z & 0 & x \\ 0 & z & y \\ 0 & 0 & 1 \end{bmatrix}_p \cdot v^T = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot v^T$$

We thus see that $h^*_p(b_1) = (0,1,0)$ and $h^*_p(b_2) = (1,0,1)$. Indeed, we see that $h^*_p(TR_p) \subseteq TS_q$ and in fact, equals TS_q. For $p' = (1,0,0)$, we see that b_1, b_2 still span $TR_{p'}$ and

$$h^*_{p'}(b_1) = (0,0,0) \text{ and } h^*_{p'}(b_2) = (1,0,1)$$

Example 11.21 *In this example, we construct the tangent map for a curve-to-surface map. Let $C(t) = (t, 0, d)$ and $S = s(u, v) = (u, v, uv)$. Thus, C is a curve and S a surface. We consider the map $h : C \to S$ where $h(p) = q$ is the projection of the curve-point p on the surface q. Thus, there is no explicit definition of h available. Let $p = (0, 0, d)$ and $q = (0, 0, 0)$. We see that*

$$TC_p = \frac{d}{dt}(t, 0, d) = (1, 0, 0)$$

The tangent space TS_q is:

$$TS_q = \mathbb{R} \cdot (s_u)_q + \mathbb{R} \cdot (s_v)_q = \mathbb{R} \cdot (1, 0, v)_q + \mathbb{R} \cdot (0, 1, u)_p = \mathbb{R} \cdot (1, 0, 0) + \mathbb{R} \cdot (0, 1, 0)$$

Thus, we see that $q - p = (0, 0, d)$ is perpendicular to TS_q and thus $h(p) = q$. For a general point $C(t)$, we see that the governing equations are:

$$\begin{array}{rcl} \langle C(t) - S(u, v), s_u \rangle & = & \langle (t - u, -v, d - uv), (1, 0, v) \rangle = t - u + dv - uv^2 = 0 \\ \langle C(t) - S(u, v), s_v \rangle & = & \langle (t - u, -v, d - uv), (0, 1, u) \rangle = -v + du - u^2 v = 0 \end{array}$$

Thus, this a system of two equations in three variables. Assuming t as the parametrizing variable, we see that u, v depend on t. Differentiating the two equations w.r.t t and at the point $t = 0$ (and thus $u, v = 0$), we get:

$$\left[\begin{array}{cc} -1 - v^2 & d - 2uv \\ d - 2uv & -1 - u^2 \end{array} \right]_p \left[\begin{array}{c} \frac{du}{dt} \\ \frac{dv}{dt} \end{array} \right]_p \left[\begin{array}{cc} -1 & d \\ d & -1 \end{array} \right]_p \left[\begin{array}{c} \frac{du}{dt} \\ \frac{dv}{dt} \end{array} \right]_p = \left[\begin{array}{c} 1 \\ 0 \end{array} \right]$$

Thus, we see that

$$\frac{du}{dt} = \frac{1}{d^2 - 1} \text{ and } \frac{dv}{dt} = \frac{d}{d^2 - 1}$$

We then see that

$$h_p^*(C'(t)) = h_p^*(1, 0, 0) = (s_u)_p \cdot \frac{du}{dt} + (s_v)_p \cdot \frac{dv}{dt} = (\frac{1}{d^2 - 1}, \frac{d}{d^2 - 1}, 0) \in TS_q$$

Notice that at $d = 1$, all hell breaks loose.

Example 11.22 *In this example, we examine the construction of the tangent map between a surface and its offset. Let*

$$s(u, v) = (x(u, v), y(u, v), z(u, v)) = (u, v, uv)$$

be the parametrized hyperboloid S. We construct the offset surface O parametrized by $o(u, v)$ as

$$o(u, v) = s(u, v) + r \cdot n(u, v)$$

where $n(u, v)$ is the unit normal at $s(u, v)$. Note that

$$s_u = (1, 0, v) \text{ and } s_v = (0, 1, u)$$

We compute this normal as the unit vector in the direction $s_u \times s_v$ and see it to be:

$$n(u, v) = (\frac{-v}{\sqrt{u^2 + v^2 + 1}}, \frac{-u}{\sqrt{u^2 + v^2 + 1}}, \frac{1}{\sqrt{u^2 + v^2 + 1}})$$

Note that we have differentiated between the surface (say S) and its parametrization (say $s(u, v)$) just so that we clearly understand the process.

Thus, we have the map $h : S \to O$ given as follows: if $p = s(u, v)$ then $h(p) = o(u, v)$. Note that the definition of h for a point p needs the (u, v) from which the point has been obtained. Indeed, we have the following diagram:

$$\begin{array}{ccc} \mathbb{R}^2 & \xrightarrow{s} & S \\ \downarrow_o & \swarrow_h & \\ O & & \end{array}$$

Thus the tangent map h^*_p must be computed as:

$$h^*_p = (s^{-1})^*_p \circ o^*_{uv}$$

Towards this, we calculate o^*_{uv}. For this, we need the partials n_u and n_v which we see are:

$$\begin{array}{rcl} n_u & = & (uv(u^2+v^2+1)^{-3/2}, (-v^2-1)(u^2+v^2+1)^{-3/2}, -u(u^2+v^2+1)^{-3/2}) \\ n_v & = & ((-u^2-1)(u^2+v^2+1)^{-3/2}, uv(u^2+v^2+1)^{-3/2}, -v(u^2+v^2+1)^{-3/2}) \end{array}$$

We see that

$$\begin{array}{rcl} n_u & = & (u^2+v^2+1)^{-3/2}[\quad uv \cdot s_u \quad -(v^2+1)\cdot s_v] \\ n_v & = & (u^2+v^2+1)^{-3/2}[\quad -(u^2+1)\cdot s_u \quad +uv \cdot s_v] \end{array}$$

Thus for a point $w = s(u, v)$, the tangent space TS_w is the span of s_u and s_v. The partials n_u, n_v also lie in the same 2-dimensional subspace of $\mathbb{R}^3$. Thus $TO_{h(w)}$ is the span of $s_u + r \cdot n_u$ and $s_v + r \cdot s_v$ and is thus the same as TS_w. This is no coincidence, since $\langle n, n\rangle = 1$ implies that $\langle n_u, n\rangle = \langle n_v, n\rangle = 0$.

We choose the point $(u, v) = (0, 0)$ and see that $n_u = [0, -1, 0]$ and $n_v = [-1, 0, 0]$. Thus

$$\begin{array}{rcl} s^*_{0,0}([0,1]) & = & [1,0,0] \\ s^*_{0,0}([1,0]) & = & [0,1,0] \\ o^*_{0,0}([0,1]) & = & [1,-r,0] \\ o^*_{0,0}([1,0]) & = & [-r,1,0] \end{array}$$

Whence, we have:

$$\begin{array}{rcl} h^*_p([1,0,0]) & = & [1,-r,0] \\ h^*_p([0,1,0]) & = & [-r,1,0] \end{array}$$

Since $TS_p = TO_{h(p)}$, we may choose a common basis for both, viz., $a_1 = [1, 0, 0]$ and $a_2 = [0, 1, 0]$. We see that

$$h^*_p(\alpha_1 a_1 + \alpha_2 a_2) = \begin{bmatrix} \alpha_1 & \alpha_2 \end{bmatrix} \begin{bmatrix} 1 & -r \\ -r & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

This implements the linear map $h^*_p : TS_p \to TO_{h(p)}$ as a matrix.

12 Curves: Orientation and Curvature

Let C be a curve in $\mathbb{R}^3$ given by a parametrization $(x(t), y(t), z(t))$. In this section we develop the concept of orientation, the Gauss map, and finally curvature.

The unit sphere $X^2 + Y^2 + Z^2 - 1 = 0$ will be denoted by S^2. The unit circle in $\mathbb{R}^2$ will be denoted by S^1.

An **orientation** of C is a map $o : C \to S^2$ so that:
(i) o is continuous and smooth.
(ii) for all non-singular $p \in C$, $o(p) \in TC_p$.

Thus, an orientation is a continuously varying unit velocity vector. Since TC_p is 1-dimensional, there are only two choices for $o(p)$ at each non-singular point. This generalizes further:

Proposition 12.1 *Let C be a connected curve so that every $p \in C$ is non-singular. Then there are only two orientations possible for C. If $f : I \to \mathbb{R}^3$ is a parametrization, then the two orientations are $o^+ = \frac{f_t}{||f_t||}$ and $o^- = -\frac{f_t}{||f_t||}$.*

Proof: Since C is non-singular, we see that f_t is a tangent vector which is never zero. Thus, both o^+ and o^- are indeed unit tangent vectors. Secondly, since f is smooth, so is f_t and $\frac{f_t}{||f_t||}$. Thus both o^+ and o^- are smooth varying unit tangent vectors, and therefore orientations. Next, let o be some other orientation. Since $o^+(p)$ is a basis for TC_p for all p, we see that there is an $r(p)$ such that $o(p) = r(p)o^+(p)$. Let $r : C \to \mathbb{R}$ be the map such that $o(p) = r(p)o^+(p)$. Since both o and o^+ are smooth, r must also be smooth. However, since $o(p)$ and $o^+(p)$ are unit vectors in a 1-dimensional space TC_p, it must be that $r(p) = \pm 1$. Since r is smooth and C is connected, r must be constant, and thus $o = o^+$ or o^-. $\square$

Example 12.2 *Consider the circle again, given by $(\frac{2t}{1+t^2}, \frac{1-t^2}{1+t^2})$. We see that $f_t = (\frac{2(1-t^2)}{(1+t^2)^2}, \frac{-4t}{(1+t^2)^2})$. Thus $o^+(t) = (\frac{1-t^2}{1+t^2}, \frac{-2t}{1+t^2})$. At $t = 1$, we may evaluate $o^+(t)$ to get $(0, -1)$. Whence $o^-(1) = (0, 1)$.*

We may attempt to construct orientations when $C \subset \mathbb{R}^2$ is implicitly defined as $f(X, Y) = 0$. Whence $\nabla f(p)$ is a smoothly varying **normal** to the curve. This may be adapted to get $n : C \to S^1$ by defining

$$n(p) = (\frac{f_x}{\sqrt{f_x^2 + f_y^2}}, \frac{f_y}{\sqrt{f_x^2 + f_y^2}})$$

The two orientations then are $R(\pi/2)n$ and $R(-\pi/2)n$, which are n rotated anti-clockwise by $\pi/2$ and $-\pi/2$ respectively. For example, we see that $X^2 + Y^2 - 1 = 0$ gives us $\nabla f = (2X, 2Y)$, and thus $n = (X/(X^2 + Y^2), Y/(X^2 + Y^2))$. The two orientations may be obtained from n.

We should also point out that having fixed $o(p)$, the orientation o, does not depend on the parametrization of the curve *at all*. In other words, if f and f' were two parametrizations such that $o^+(p) = (o')^+(p)$, then $o^+ = (o')^+$.

Let us fix an orientation o of a curve $C \subseteq \mathbb{R}^3$. thus we have a map $o : C \to S^2$. Let $o(p) = v$ be a unit tangent vector at p. The map o translates to a linear map from $o_p^* : TC_p \to TS_v^2$. Points on the curve, infinitesimally close to p will have unit velocities infinitesimally close to v. To evaluate the map o^*, say when C is parametrized by $(x(t), y(t), z(t))$, we have

$$o(t) = \frac{1}{\sqrt{(dx/dt)^2 + (dy/dt)^2 + (dz/dt)^2}}[dx/dt, dy/dt, dz/dt]$$

Let t_0 be such that $f(t_0) = p$. For a small motion ϵt, we see that $f(t_0 + \epsilon t) = f(p) + \epsilon t f_t(t_0)$. On the other hand, this small motion produces the change $o(t_0 + \epsilon t) = o(p) + \epsilon t o_t(t_0)$. Thus the element $f_t(t_0) \in TC_p$ produces the changes $o_t(t_0) \in TS_v^2$. This is the implementation of the map o^*.

The vector $\frac{o_t(t_0)}{||f_t(t_0)||}$ is the called the **curvature vector**, its magnitude as the **curvature**, and its reciprocal, the **radius of curvature** and is denoted by κ.

Example 12.3 *Let C be given by $f = (r \sin t, r \cos t)$. We see that $f_t = (r \cos t, -r \sin t)$, and $o(t) = f_t/(||f_t||) = (\cos t, -\sin t)$. Thus $o_t = do/dt = (-\sin t, -\cos t)$ and $||do/dt|| = 1$. The map o^* is given by $o^*(\alpha f_t) = \alpha o_t$ for all $\alpha \in \mathbb{R}$. While $||o_t|| = 1$, we see that $||f_t|| = r$ and thus the curvature at any point t is $1/r$ and the radius of curvature κ is r, as expected.*

Example 12.4 *Let C be the parabola $f = (t, t^2)$. We have $f_t = (1, 2t)$ and therefore $o(t) = \frac{1}{1+4t^2}(1, 2t)$. Thus*

$$o_t(t) = \frac{1}{(1+4t^2)^2}(-8t, 2+20t^2)$$

Thus the curvature is $||o_t||/||f_t||$, which evaluates to:

$$\frac{400t^4 + 144t^2 + 4}{(1+4t^2)^3}$$

This is not very enlightening, though we see that the curvature is an even function with maximum at $t = 0$, and monotonically reduces as $|t|$ increases.

Example 12.5 *Here is a more curious example, the right helix, given by $f = (\sin t, \cos t, t)$. We have $f_t = (\cos t, -\sin t, 1)$ and*

$$o(t) = \frac{1}{2}(\cos t, -\sin t, 1)$$

Whence $o_t = (-\sin t, -\cos t, 0)$ and $||o_t|| = 1$. Since $||f_t|| = 2$, we have the curvature as $1/2$ and $\kappa = 2$. One see that the left helix $(\sin t, \cos t, -t)$ will have the same curvature. The reader may wonder how is one to distinguish the two helices by a geometric invariant. One important invariant is the sign of the determinant of the vectors $df/dt, d^2/dt^2$ and d^3f/dt^3. In this case, we have

$$det \begin{bmatrix} \cos t & -\sin t & \pm 1 \\ -\sin t & -\cos t & 0 \\ -\cos t & \sin t & 0 \end{bmatrix}$$

Thus the two helices will have different signs.

Note that the vectors f_t and o_t are always perpendicular to each other. This is because, o_t resides in the space TS^2_v, where $v = o(p)$. And, on the sphere, the tangent space at the point v is always perpendicular to v. Thus $o_t(v)$ and $v = o(p)$ are perpendicular. On the other hand, $o(p) = f_t/||f_t||$, and thus is a scalar multiple of f_t. Thus o_t would be perpendicular to f_t.

The second comment is that curvature κ is an **affine invariant**. In other words, if the curve C is translated/rotated into C', whereby the point $p \in C$ goes to $p' \in C'$. we still have $\kappa(p) = \kappa(p')$. Further, and more importantly, the number κ *does not* depend on the parametrization f. This is because it is the multiplier in the map $o^* : TC_p \to TS^2_v$ and o does not depend on the parametrization. This is also directly argued as follows: Let $\alpha : \mathbb{R} \to \mathbb{R}$ be smooth and 1-1 and monotonic, and let $f : \mathbb{R} \to \mathbb{R}^3$ be a parametrized curve. Let $F(t) = f(\alpha(t))$ be a re-parametrization of the same curve. Let $o(t)$ and $O(t)$ be the corresponding orientations. We see that

$$O(t) = \frac{F'(t)}{||F'(t)||} = \frac{f'(\alpha(t))\alpha'(t)}{||f'(\alpha(t))|| \cdot ||\alpha'(t)||}$$

Since α is so special, we have $\alpha'(t) = |\alpha'(t)|$, and thus $O(t) = o(\alpha(t))$. Thus

$$K(t) = \frac{O'(t)}{||F'(t)||} = \frac{o'(\alpha(t))}{||f'(\alpha(t))||} = k(\alpha(t))$$

Thus, for a point t_0 with $F(t_0) = f(\alpha(t_0)) = f(t_1) = p$, we have $K(t_0) = k(t_1) = k(p)$.

By a similar token, $sign(det(df/dt, d^2f/dt^2, d^3f/dt^3))$ is also an invariant.

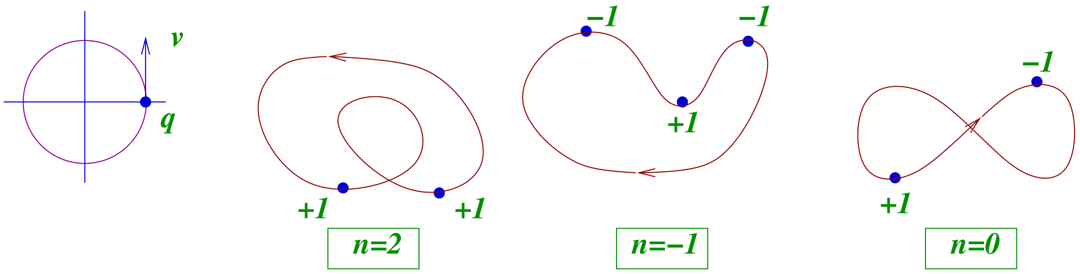

Figure 45: Winding Numbers.

13 2D curves and the winding number

We consider here the special case of closed curves in $\mathbb{R}^2$. With each closed curve C, we associate an integer $n(C)$, called the **winding number**. Let us fix an orientation $o : C \rightarrow S^1$, where $S^1 \subseteq \mathbb{R}^2$ is the unit circle. Fix a point $q \in S^1$ and locate all points $p_1, \ldots, p_k$ such that $o(p_i) = q$ for all $i = 1, \ldots, k$. Let v be a non-zero tangent vector at the point $q \in S^1$. Clearly $o'(t_i) = \alpha_i \cdot v$ for some $\alpha_i \in \mathbb{R}$. With each point p_i, we associate a sign s_i defined as $sign(\alpha_i)$. We assume that q is such that $\alpha_i \neq 0$ for all i. The winding number $n(C, q, v)$ is defined as:

$$n(C, q, v) = \sum_{i=1}^{k} s_i$$

We leave it to the reader to show that:

- $n(C, q, \beta v) = sign(\beta) \cdot n(C, q, v)$
- If q' is another point on S^1 and v' is a tangent at q' *similarly oriented as* v *at* q then $n(C, q, v) = n(C, q', v')$.
- If C^- is the oppositely oriented curve then $n(C^-, q, v) = -n(C, q, v)$.

Thus, having chosen an orientation of the unit circle S^1, say the anti-clockwise one, we may associate a number $n(C) = n(C, q, v)$ which we call as the **winding number**. See Fig. 45 for examples.

Another way of defining the winding number for a parametrized curve $C(t)$ is to define $\theta(t)$ as the angle formed by the tangent $C'(t)$ with the positive X-axis. Note that there is an ambiguity in $\theta(t)$ since $\theta(t) + 2\pi$ is another possibility. Even assuming that $0 \leq \theta(t) < 2\pi$ forces us into another problem. If we were to require that $\theta(t)$ to be continuous, we see that if we were to start at a point p on the curve and travel around C and come back to p again, we see that the desired continuity of θ will cause us to arrive at p with a θ-value of $\theta(p) + ndir\pi$. This n is precisely the winding number. Thus, for a closed curve $C : [0, 1] \rightarrow \mathbb{R}^2$, we may define the winding number as

$$n(C) = \int_C d\theta$$

To evaluate the integral, we use the parametrization and choose a sequence $0 = t_1 < \ldots < t_n = 1$ and see that:

$$\begin{array}{rcl} n(C) = \int_C d\theta & = & \lim_{n\rightarrow\infty} \sum_i (\theta(t_{i+1}) - \theta(t_i)) \\ & = & \int_0^1 \frac{d\theta}{dt} dt \end{array}$$

Let $C(t) = (x(t), y(t))$ and then locally, we have $\theta(t) = \tan^{-1}(\dot{y}/\dot{x})$. Thus, we have:

$$\dot{\theta} = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{\dot{x}^2 + \dot{y}^2}$$

We now state the important relationship between the curvature function $\kappa : C \to \mathbb{R}$ and the winding number.

Proposition 13.1 *The integral of curvature κ over the curve C equals the winding number. Thus, the local curvature data yields the global winding number $n(C)$.*

Proof: Given the curve $(x(t), y(t)$, we have the orientation:

$$o(t) = \frac{1}{\sqrt{\dot{x}^2 + \dot{y}^2}}(\dot{x}, \dot{y})$$

Thus, we see that

$$\dot{o} = \left(\frac{\sqrt{\dot{x}^2 + \dot{y}^2} \cdot \ddot{x} - \dot{x} \cdot (\dot{x}\ddot{x} + \dot{y}\ddot{y})(\dot{x}^2 + \dot{y}^2)^{-1/2}}{\dot{x}^2 + \dot{y}^2}, \text{ etc.} \right)$$

We may simplify this to get:

$$\dot{o} = \left(\frac{\dot{y}^2\ddot{x} - \dot{x}\dot{y}\ddot{y}}{(\dot{x}^2 + \dot{y}^2)^{3/2}}, \text{ etc.} \right)$$

Since $\dot{o}$ is perpendicular to $(\dot{x}, \dot{y})$ (see Fig. 46) and $\kappa(C(t))$ is the ratio of their norms, we construct the vector $(-\dot{y}, \dot{x})$ of the same norm as the velocity vector. This vector is alligned along $\dot{o}$ and thus the ratio of the X-coordinates gives us $\kappa(C(t))$. Thus we arrive at:

$$\kappa(C(t)) = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{(\dot{x}^2 + \dot{y}^2)^{3/2}}$$

Now, we are almost there. We wish to integrate the curvature function κ over the curve, or in other words, the integral

$$\int_C \kappa(p)dp$$

We use the parametrization and select points $0 = t_1 < \ldots < t_n = 1$ and express the integral as:

$$\begin{array}{rcl} \int_C \kappa(p)dp & = & \lim_{n\to\infty} \sum_i \kappa(C(t_i)) \cdot \|C(t_{i+1}) - C(t_i)\| \\ & = & \int_0^1 \kappa(C(t))\|C'(t)\|dt \\ & = & \int_0^1 \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{(\dot{x}^2+\dot{y}^2)^{3/2}} \sqrt{\dot{x}^2 + \dot{y}^2}dt \\ & = & \int_0^1 \dot{\theta}dt \\ & = & n(C) \end{array}$$

This finishes the proof. □

14 Surfaces: Orientation and Curvature

14.1 The Gauss Map and Examples

As before S^2 will denote the unit sphere. Let S be a surface given parametrically as a map $f : I^2 \to \mathbb{R}^3$. We assume that every point $(u, v) \in I$ is non-singular, and thus f_u, f_v span the tangent space

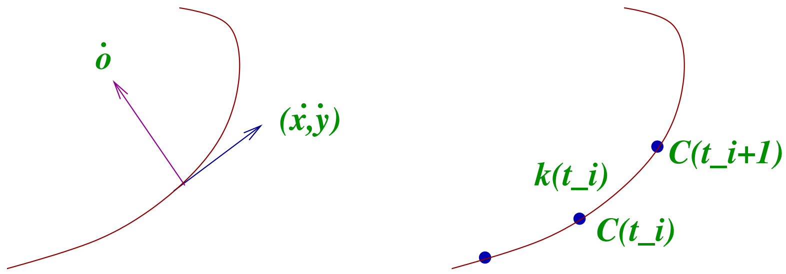

Figure 46: The Integration

TS_p at the point $p = f(u,v)$. An **orientation** of S is a smooth map $o : S \to S^2$ such that for any point $p \in S$, the unit vector $o(p)$ is perpendicular to TS_p. Traditionally, such a map is also called the **Gauss map**.

Proposition 14.1 *(i) If o is an orientation then so is o^-, where $o^-(p) = -o(p)$.*
(ii) If o' is another orientation on S then $o' = o$ or $o' = o^-$.
(iii) If S is non-singular, $o' = \frac{f_u \times f_v}{||f_u \times f_v||}$ is an orientation of S.

Proof: (i) is easy: if o is smooth then so is o^-. Certainly o^- is perpendicular to TS_p, for every point p. Next, if o' is any orientation, then $o'(p)$ is a unit vector perpendicular to TS_p, and thus must equal $o(p)$ or $o^-(p)$. Thus $o'(p)/o(p)$ is a function which takes the values ± 1. Since both $o(p)$ and $o'(p)$ are smooth, the value must be a constant, and hence $o' = o$ or $o' = o^-$. Finally, by our hypothesis, $f_u \times f_v$ is never zero, and is always perpendicular to TS_p. Clearly, f_u and f_v are smooth and therefore so is $f_u \times f_v$. Whence, o' is an orientation. $\Box$

For an implicitly defined surface S, with say, $f(X,Y,Z) = 0$, we define the orientation of S as $\frac{\nabla f}{||\nabla f||}$ evaluated at the point p. If S is non-singular then the orientation of an implicitly defines surface matches that if S were parametrically defined as well.

Example 14.2 *(i) The cylinder is given by $(\frac{2u}{1+u^2}, \frac{1-u^2}{1+u^2}, v)$, and implicitly as $g = X^2 + Y^2 - 1 = 0$. Let $p = (1,0,0)$ which corresponds to $u = 1, v = 0$. We see that $f_u = (\frac{2-2u^2}{(1+u^2)^2}, \frac{-4u}{(1+u^2)^2}, 0)$, and $f_v = (0,0,1)$. Thus at $(1,0)$, we see that $f_u = (0,-1,0)$ and $f_v = (0,0,1)$. Thus the orientation arising from $f_u \times f_v$ gives us $(-1,0,0)$. On the other hand, the implicit definition gives us $\nabla g = (2X, 2Y, 0)$ with the orientation $(\frac{X}{\sqrt{X^2+Y^2}}, \frac{Y}{\sqrt{X^2+Y^2}}, 0)$. This evaluated at $(1,0,0)$ gives us $(1,0,0)$. Thus the two orientations differ by a sign.*

14.2 The Bilinear map and Curvature

Our next objective is to fix a gauss map/orientation $o : M \to S^2$, and apply the theory that we have developed above. As pointed out above, if $o(p) = n \in S^2$, then o sets up a map $o_p^* : TM_p \to TS_n^2$. Now note that both TM_p and TS_n^2 are perpendicular to the same vector n. The first by virtue that n is a normal to TM_p, and the second because for a sphere, the tangent space at a point p is actually perpendicular to it.

Thus, the map o_p^* is actually a map on the same 2-dimensional subspace of $\mathbb{R}^3$.

Example 14.3 *Let S_r be the sphere $X^2+Y^2+(Z-r)^2-r^2=0$. The point $p=(0,0,0)$ lies on S_r. We see that $\nabla g=(2X,2Y,2(Z-r))$, and thus*

$$o=\frac{1}{r}(X,Y,Z-r)$$

The tangent plane at p is the X-Y-plane and $o(p)=[0,0,-1]$. We choose a basis $e_1=[1,0,0]$ and $e_2=[0,1,0]$, and discover that

$$o(p+\epsilon(\alpha e_1+\beta e_2))=[0,0,-1]+\frac{\epsilon}{r}(\alpha e_1+\beta e_2)$$

We thus see that

$$o_p^*\left(\begin{bmatrix} e_1 \\ e_2 \end{bmatrix}\right)=\begin{bmatrix} \frac{1}{r} & 0 \\ 0 & \frac{1}{r} \end{bmatrix}\begin{bmatrix} e_1 \\ e_2 \end{bmatrix}$$

Example 14.4 *Let M be the cone given by the equation $g\equiv X^2+Y^2-Z^2=0$, and let us pick $p=(1,0,1)$ as the non-singular point for analysis. The tangent space TM_p is given by all $w=(x',y',z')$ such that $g(p+\epsilon w)=0$ as well. Since $p+\epsilon w=[1+\epsilon x',\epsilon y',1+\epsilon z']$, we get:*

$$0=g(p+\epsilon w)=1+2\epsilon x'-1+2\epsilon z'$$

Or in other words, $TM_p=\{[\alpha,\beta,\alpha]|\alpha,\beta\in\mathbb{R}\}$ with a basis $b_1=[1,0,1]$ and $b_2=[0,1,0]$. An orientation is obtained by taking $o=\frac{\nabla g}{||\nabla g||}$. Since $\nabla g=[2X,2Y,-2Z]$, we have

$$o(X,Y,Z)=\frac{1}{(X^2+Y^2+Z^2)}[X,Y,-Z]$$

On substituting the point $[1,0,1]+\epsilon[\alpha,\beta,\alpha]$, we get:

$$o(p+\epsilon w)=\frac{1}{\sqrt{(1+2\epsilon\alpha+1+2\epsilon\alpha)}}([1,0,-1]+\epsilon[\alpha,\beta,-\alpha]$$

Upon simplifying this expression, and using that $\frac{1}{\sqrt{1+2\epsilon\alpha}}=1-\epsilon\alpha$, we get:

$$o(p+\epsilon w)=\frac{1}{\sqrt{2}}([1,0,-1]+\epsilon[0,\beta,0])$$

$$o_p^*\left(\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}\right)=\begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} \end{bmatrix}\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Example 14.5 *Let M be given by the equation $g\equiv Z-f(X,Y)=0$ where f is of the form $f(X,Y)=aX^2+2bXY+cY^2+\ldots$, and let $p=(0,0,0)$. We see that*

$$\nabla g=(2aX+2bY+\ldots,2cY+2bX+\ldots,1)$$

and thus $\nabla g(p)=[0,0,1]$ whence TM_p is the X-Y-plane. The points $p+\epsilon w$ with $w=(x',y',0)$ lie on M.

The orientation map is $o=\frac{\nabla g}{||\nabla g||}$ which gives us (for some constants A,B,C):

$$o(X,Y,Z)=\frac{1}{\sqrt{1+AX^2+CY^2+BXY+\ldots}}(2aX+2bY+\ldots,2bX+2cY+\ldots,1)$$

We see then for $w = (x', y', 0)$ that

$$o(p + \epsilon w) = (0, 0, 1) + \epsilon(2ax' + 2by', 2bx' + 2cy', 0)$$

Thus, we see that the vector w transforms under the linear transformation:

$$o_p^* \left(\begin{bmatrix} x' \\ y' \end{bmatrix} \right) = \begin{bmatrix} 2a & 2b \\ 2b & 2c \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix}$$

We have seen from the above examples that the map o_p^* is an easily computed map on the 2-dimensional spaces TM_p. Furthermore, the matrix of the map was symmetric when we chose a basis of orthogonal tangent vectors. This fact is true in general and is easily proved via Example 14.5. After all, for any surface M and a non-singular point $p \in M$, we may as well translate the coordinate system so that p is the origin. Clearly this translation is not going to the change the orientation map and therefore its derivative. Next, we may as well rotate the frame of reference so that the tangent plane at p is the X-Y-plane. This rotates the orientation by the fixed rotation, and therefore also its derivative. Thus, under the rotation, the matrix A representing the transformation o_p^* will undergo a similarity transform $UAU^{-1} = UAU^T$, where U is an appropriate orthogonal matrix.

In this 'standard' form, a surface is expressible as in Example 14.5. We see then that the matrix is symmetric for an orthogonal basis of TM_p. Now since the original matrix is related to this matrix by a similarity operation by an orthogonal matrix, that too must be symmetric for an orthogonal basis of TM_p.

We also note that the map o_p^* in the standard form, depends only on the leading term of $f(X, Y)$, which is quadratic. The standard form may be further simplified. We consider the rotation of the X-Y-plane by an angle θ, and see that under the transformation

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

the form $aX^2 + 2bXY + cY^2$ transforms to $Ax^2 + 2Bx + Cy^2$, with

$$B = (a - c)\sin 2\theta + b\cos 2\theta$$

There is always a θ so that $B = 0$; in fact, since $\tan 2\theta$ is $\pi/2$-periodic.

Thus after this transformation, the matrix o_p^* is diagonal.

In the general case, we have seen that the matrix is real and symmetric, and thus have real eigenvalues $\lambda_1(p), \lambda_1(p)$. These eigenvalues are called the **curvatures** at p. The reciprocals $\kappa_1(p), \kappa_2(p)$ are called the radii of curvatures. Note that for a sphere above, of radius r, the two radii were equal to r. The product $\lambda_1(p)\lambda_2(p)$ is called the **gaussian curvature** and equals the determinant of the matrix o_p^* and the quantity $\lambda_1(p) + \lambda_2(p)$ is called the **mean curvature** and equals the trace of the matrix o_p^*.

The eigenvectors of o_p^* are called the **principal directions** at p. Roughly speaking, an infinitesimal motion along the pricipal direction moves the normal in the same direction. Note that for a point p on a sphere, every direction is a principal direction, while on the cone, the direction of the straight line passing through the apex and p, and the circle passing through p are the two principal directions.

A curve C on a surface is aclled a **principal curve** if for every $p \in C$, the direction TC_p matches a principal direction on p. Great circles on spheres are principal curves. So are apical straight lines and circles on the cone. However note that the radius of the circle through p is **not** the radius of curvature at the point p. One should be careful about guessing that very nice curves on a surface are principal curves: here is an example:

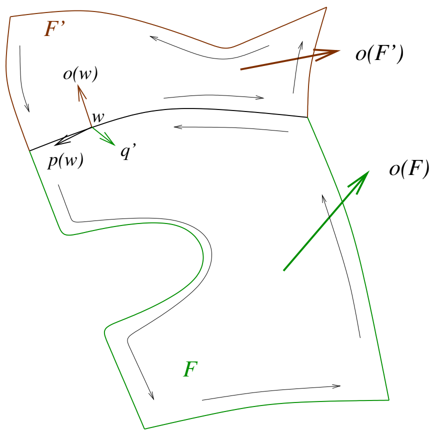

Figure 47: Associated Orientations.

Example 14.6 *Let S be defined by 'rotating' and infinite line around the X-axis. Thus for example, a parmetrization of the surface would be $(u, v\cos u, v\sin u)$. Implicitization gives us $g \equiv Z - Y\tan X = 0$. $\nabla g = (-Y\sec^2 X, -\tan X, 1)$ giving us $\nabla g(p) = [0,0,1]$ at $p = [0,0,0]$. Note that every point $[u,0,0]$ has a straight line lying on the surface and passing through it. Further more, the X-axis also lies on the surface, and thus, at $p = [0,0,0]$ there are two lines, the X-axis, and the Y-axis which lie on the surface. One is tempted to conclude that these are principal curves, which is incorrect.*

We see that

$$o = \frac{1}{\sqrt{1 + Y^2\sec^4 X + \tan^2 X}}(-Y\sec^2 X, -\tan X, 1)$$

Thus $o([0,0,0] + \epsilon[\alpha,\beta,0] = [0,0,1] - [\beta,\alpha,1]$, whence we have:

$$o_p^*\left(\begin{bmatrix} e_1 \\ e_2 \end{bmatrix}\right) = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}\begin{bmatrix} e_1 \\ e_2 \end{bmatrix}$$

We see thus that ± 1 are the eigenvalues, with eigenvectors $[1,-1,0]$ and $[1,1,0]$, which are **not** *along the X or the Y axis.*

14.3 The Boundary Orientation of Edges and Faces

A final point is about extending an orientation of a surface patch to its boundary. This is customarily required to check when two surface patches meet, their orientations are consistent. See Figure 47. This is done by (i) extending the orientations o, o' of each face F, F' to orientations of the boundary edge, say p, p', and (ii) ensuring that $p(w) = -p'(w)$ at some point w (and therefore at every point) on the edge E. The orientation p obtained from o is called the boundary orientation of o.

The basic idea is to orient the outside loop 'anti-clockwise', and the inner loops as 'clockwise' with respect to the orientation o. This is expressed uniformly as follows. For the point w on the edge E, let $o(w)$ be the surface normal. Let TE_w be the tangent space at w for the edge E and let e be a unit tangent vector at w to the edge E. The question is whether p assigns e or $-e$ at w.

Suppose that we find a tangent vector $q \in TF_w$ such that q points 'into' the surface patch. If such a q is obtained, then by subtracting a suitable multiple of e, we may compute q' which points 'into' the patch and is perpendicular to TE_w. Clearly, the orienation that we need for E is given at that point w by $q' \times o(w)$. Thus, the task reduces to computing a vector $q \in TF_w$ which points 'into' the patch.

This is done using the parametrization of the face F. Let $f : D \rightarrow \mathbb{R}^3$ be the parametrization, where $D \subseteq \mathbb{R}^2$ is the domain of definition. let $z \in D$ be such that $f(z) = w$. Let us fix beforehand a point $z_0 \in D$, and thus $f(z_0) \in F$. We draw a staright-line from z to z_0. Clearly, if this segment intersects the p-curves defining the boundary of D at an odd number of points (besides z), then the vector $\epsilon(z_0 - z)$ points outside the domain D. On the other hand if this segment intersected the boundary of D an even number of times, then we have that $\epsilon(z_0 - z)$ points into D. If $z' = \epsilon(z_0 - z) = \epsilon(\alpha u + \beta v)$, then $v' = \epsilon(\alpha f_u + \beta f_v)$ is a tangent vector at w. Furthermore v' points into F iff z' points into D. Thus either v' or $-v'$ is the required q above, depending on whether z' points into D or not.

15 Genus

This section is about a global invariant of a solid, called its **genus**. Roughly speaking, it is realted to the 'number of holes' in the solid. A formulation of genus that we will present here, proves the Euler formula for planar graphs, and generalizes it. However, the formulation is fairly intricate and uses some amount of linear algebra. We will first recapitulate some of this and build a few lemmas which will help us later.

To recall, a vector space over $\mathbb{R}$ is a set V with an operation $+$. The operation is associative and commutative, i.e., $v + (v' + v'') = (v + v') + v''$ and $v + v' = v' + v$, for all $v, v', v'' \in V$. There is a special element, the **zero vector** 0 such that $v + 0 = 0 + v = v$, and every v has an additive inverse, i.e., a v' such that $v + v' = 0$. This v' is frequently denoted as $-v$. Thus $(V, +)$ forms an **abelian** group. The operation $+$ is intimately inter-twined with the scalars $\mathbb{R}$. Indeed, there is a map $cdot : \mathbb{R} \times V \rightarrow V$. Given an $\alpha \in \mathbb{R}$ and $v \in V$, we denote $\cdot(\alpha, v)$ as $\alpha \cdot v$, or simply αv. We further have: (i) $(\alpha + \beta)v = \alpha v + \beta v$, (ii) $(\alpha\beta)v = \alpha(\beta v)$, and $\alpha(v + v') = \alpha v + \alpha v'$. A **subspace** $V' \subset V$ is a subset of V which is closed under vector addition and scalar multiplication. In other words, (i) $v' + v'' \in V'$, and $\alpha v' \in V'$, for all $v', v'' \in V'$ and $\alpha \in \mathbb{R}$. Thus V' is a vector space under the addition and scalar multiplication inherited from V.

A crucial notion in a vector space is that of **linear dependence**. A subset $S \subseteq V$ is called linearly dependent if there is a finite set of vectors $v_1, \ldots, v_k \in S$, and **non-zero** real numbers $\alpha_1, \ldots, \alpha_k$ such that $\sum_{i=1}^{k} \alpha_i v_i = 0$. If S is not linearly dependent, then it is called **linearly independent**. Let B be a maximal linearly independent set in V, i.e., B is linearly independent, but $B \cup \{v\}$ is dependent for all $v \in V$. Then B is called a basis of V. For a fixed basis $B = \{b_1, \ldots, b_n\}$, every vector v may be expressed as a *unique* linear combination: $v = \sum_i \alpha_i v_i$. If $V' \subseteq V$ is a subspace then every basis B' of V' can be extended to a basis B of V, and thus $dim(V') \leq dim(V)$. Note that is $dim(V') = dim(V)$, then $V' = V$.

If B is finite then V is called **finite-dimensional** and $|B|$ is called the dimension of V and is denoted as $dim(V)$. Of course, that $dim(V)$ is well-defined is tantamount to showing that all maximal linearly independent sets have the same cardinality.

Suppose that we have two vector spaces V with basis $B = \{v_1, \ldots, v_m\}$, and W with basis $C = \{w_1, \ldots, w_n\}$. Let $f : V \rightarrow W$ be a map. We say that f is **linear** if (i) $f(v + v') = f(v) + f(v')$ and (ii) $f(\alpha v) = \alpha f(v)$, for all $v, v' \in V$ and $\alpha \in \mathbb{R}$. Let f be linear, and let $ker(f) = \{v | v \in V, f(v) =\}$, and $Im(f) = \{w \in W | \exists v \in V, f(v) = w\}$. Thus $ker(f)$ is the collection of all vectors in v which are mapped to the zero vector in W, while $Im(f)$ is the usual image of the map f. When f is linear, both $ker(f)$ and $Im(f)$ are vector subspaces of V and W respectively. we have the following lemma:

Lemma 15.1 *Let $f : V \to W$ be a linear map, then*

$$dim(V) = dim(ker(f)) + dim(Im(f))$$

Proof: Let $dim(Im(f)) = r$ and $dim(ker(f)) = s$. Let $\{w_1, \ldots, w_r\}$ be a basis of $Im(f)$. Let $v_1, \ldots, v_r$ be arbitrary elements of V such that $f(v_o) = w_i$. Thus each v_i is a *pullback* of w_i. Next, let $\{v'_1, \ldots, v'_s\}$ be a basis of $ker(f)$. We claim that $B = \{v_1, \ldots, v_r, v'_1, \ldots, v'_s\}$ is a basis of V.

Firstly, suppose that:

$$\alpha_1 v_1 + \ldots + \alpha_r v_r + \beta_1 v'_1 + \ldots + \beta_s v'_s = 0$$

Applying f to this addition, and noting that $f(v'_j) = 0$, we get the following equation in W:

$$\alpha_1 w_1 + \ldots + \alpha_r w_r = 0$$

Since $w_1, \ldots, w_r$ are linearly independent, we conclude that $\alpha_i = 0$ for all i. Thus, we are reduced to:

$$\beta_1 v'_1 + \ldots + \beta_s v'_s = 0$$

But that is also impossible, since $v'_1, \ldots, v'_s$ were picked as independent elements of $ker(f)$. Thus, we see that B is a linearly independent set.

Next, we show that B spans V, i.e., every vector $v \in V$ is expressible as a linear combination of elements in B. To see this, examine $w = f(v)$. Since $w = f(v)$, it is in the image of f. Thus there are constants $\alpha_1, \ldots, \alpha_r$ such that $w = \sum_i \alpha_i w_i$. Consider $v' = v - \sum_i \alpha_i v_i$. We see that $f(v') = f(v) - \sum_i \alpha_i f(v_i) = w - \sum_i \alpha_i w_i = 0$. Thus $v' \in ker(f)$, and thus $v' = \sum_j \beta_j v'_j$, whence $v = \sum_i \alpha_i v_i + \sum_j \beta_j v'_j$, and we are done. $\Box$

The next concept we need is of **quotient spaces**. Let $W \subset V$ be a subspace. We define an equivalence relation $\sim$ on V. We say that $v \sim v'$ is $v - v' \in W$. Note that (i) $v - v = 0 \in W$, (ii) $v - v' \in W$ implies $v' - v \in W$, and (iii) $v - v' \in W$ and $v' - v'' \in W$ implies $v - v'' \in W$. Thus $\sim$ is indeed an equivalence relation. We denote the equivalence class of v as $[v]$. Next note that (i) $v \sim v'$ implies that $\alpha v \sim \alpha v'$, and (ii) $x \sim x'$ and $y \sim y'$, then $x + y \sim x' + y'$. Thus the equivalence classes $\{[v] | v \in V\}$ has the structure of a vector space. Define $[v] + [v']$ as $[v + v']$, and $\alpha[v]$ as $[\alpha v]$. This vector space is denoted as the **quotient** V/W. The zero-vector in V/W is $[0] = [w]$, for any $w \in W$. To understand linear dependence in V/W, we see that $[v_1], \ldots, [v_r]$ are linearly dependent if there exist $\alpha_1, \ldots, \alpha_r$, not all zero, such that $\sum_i \alpha_i v_i \in W$.

Lemma 15.2 *(*Baby Ratsnake Lemma*): The dimension of V/W is $dim(V) - dim(W)$.*

Proof: Let $[v_1], \ldots, [v_r]$ be a basis of V/W, and $w_1, \ldots, w_s$ be that of W. We claim that $B = \{v_1, \ldots, v_r, w_1, \ldots, w_s\}$ is a basis of V. First, we prove linear independence of B. Suppose $\sum_i \alpha_i v_i + \sum_j \beta_j w_j = 0$, then we see that $\sum_i \alpha_i v_i = -\sum_j \beta_j w_j \in W$. This is a relation on $[v_1], \ldots, [v_r]$, whence, by their linear independence, $\alpha_i = 0$ for all i. Finally, since w_j's are also independent, we have $\beta_j = 0$ for all j. Next, to see that B spans V, consider $[v]$, and continue along familiar lines. $\Box$

Proposition 15.3 *(The Ratsnake Lemma): Let X, Y and Z be vector spaces. Let us have the following diagram of linear maps:*

$$X \xrightarrow{f} Y \xrightarrow{g} Z$$

such that $g \circ f : X \to Z$ is the zero map. Further f is injective and g is surjective. Then $Im(f)$ is a subspace of $ker(g)$, and

$$dim(ker(g)/Im(f)) = -dim(X) + dim(Y) - dim(Z)$$

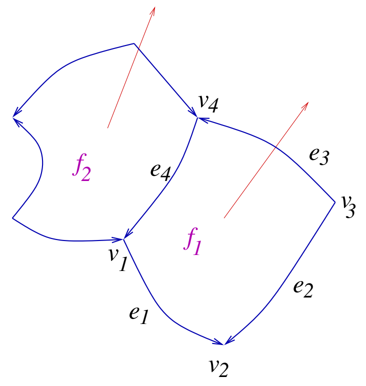

Figure 48: Face and Edge Orientations.

Proof: Note f is an injection implies that $ker(f) = 0$ and thus $dim(ker(f)) = 0$. Whence, by Lemma 15.1, we have that $dim(Im(f)) = dim(X)$. Next, since g is surjective, we have $dim(ker(g)) = dim(Y) - dim(Z)$. Finally, by Lemma 15.2, $dim(ker(g)/Im(f)) = dim(ker(g)) - dim(Im(f))$, which gives us the result. □

We are now ready to construct the vector spaces which will be used to define genus. Let X be a set of symbols $\{x_1, \ldots, x_r\}$. By $\mathbb{R}X$, we will denote the vector space of *formal linear combinations* of elements of X. In other words:

$$\mathbb{R}X = \{\alpha_1 x_1 + \ldots + \alpha_r x_r | \alpha_1, \ldots, \alpha_r \in \mathbb{R}\}$$

Let S b a solid with F as its faces, E as its edges, and V as its vertices. We assume that each face f has been supplied with an orientation $o(f)$ along the outside normal. Furthermore, each edge e is oriented in an arbitrary way, say $o(e)$. Note that each face $o(f)$ orientation defines an orientation of the edges on its boundary, when the boundary of the face is travelled in an anticlockwise sense. We call this orientation $o(f, e)$ and denote by $sign(f, e)$ the sign of $o(f, e)$ with respect to $o(e)$.

We assume that (i) Every face is simple with no inner loops, (ii) two faces intersect only along edges, (iii) there is only one shell, and finally (iv) every edge e belongs to exactly two faces f and f'. Furthermore, the signs $sign(f, e)$ and $sign(f', e)$ are opposite.

Let $F = \{f_1, \ldots, f_r\}$, $E = \{e_1, \ldots, e_m\}$ and $V = \{v_1, \ldots, v_n\}$. We construct the spaces $\mathbb{R}F$, $\mathbb{R}E$, and $\mathbb{R}V$. For a face f with boundary edges $e_1, \ldots, e_k$, we define the **boundary map**

$$\partial_2(f) = sign(f, e_1) \cdot e_1 + \ldots + sign(f, e_k) \cdot e_k$$

Note that the RHS is a formal linear combination of the edges on the boundary of f, and is thus an element of $\mathbb{R}E$. The map ∂_2 extends to a map $\partial_2 : \mathbb{R}F \to \mathbb{R}E$.

By the same token, we define the map $\partial_1(e) = v_1 - v_2$, where the orientation of e begins at v_2 and ends at v_1. This defines the **cycle** map $\partial_1 : \mathbb{R}E \to \mathbb{R}V$. See Figure 48: for the face f_1, we have:

$$\begin{aligned} \partial_2(f_1) = &\quad e_1 - e_2 + e_3 + e_4 \\ \partial_1(e_2) = &\quad v_2 - v_3 \end{aligned}$$

Let us now try and understand first, $ker(\partial_1)$, and then $Im(\partial_2)$. We say that $\alpha_1 \cdot e_1 + \ldots + \alpha_m \cdot e_m$ is a *cycle* if

$$\partial_1(\alpha_1 \cdot e_1 + \ldots + \alpha_m \cdot e_m) = 0$$

Note that if one were to walk along the edges of the solid, and reach ones starting point, then the edges on the path, listed with their signs (i.e., whether travelled along $o(e)$ or opposite it), will constitute a cycle. Thus $ker(\partial_1)$ is the space of all cycles on the solid, whatever that means. An element of the image of ∂_2 is called a *boundary*. If one were to enclose some area on the solid, then the edges on the boundary of this enclosure, listed with signs, would be a boundary. Clearly, when the area constitute a single face, this is so. On the other hand, when the enclosure constitute a bunch of faces, that the inter-face boundary is travelled in opposite signs, ensures that $\partial_2(f + f' + \ldots + f'')$ is the boundary of the enclosure.

Lemma 15.4 *We consider the diagram*

$$\mathbb{R}F \overset{\partial_2}{\rightarrow} \mathbb{R}E \overset{\partial_1}{\rightarrow} \mathbb{R}V$$

(i) The composition $\partial_1 \circ \partial_2$ *is the zero map.*
(ii) The kernel $ker(\partial_2)$ *is* 1*-dimensional and consists of the vector* $f_1 + \ldots + f_r$*, and its multiples.*
(iii) The image $Im(\partial_1)$ *is of dimension* $|V| - 1$*, and consists of all linear combination* $\sum_i \alpha_i v_i$ *such that* $\sum_i \alpha_i = 0$*.*

Proof: It suffices to show that $\partial_1 \circ \partial_1(f) = 0$ for all $f \in F$. And this is easy to check. Next, we see that in $\partial_2(f_1 + \ldots, f_k)$, every edge is covered *exactly* twice, and with opposite signs. Thus $\partial_2(\sum_j f_j) = 0$, and thus this element is indeed in $ker(\partial_2)$. Next, suppose that $\partial_2(\sum_j \beta_j f_j) = 0$, and $F' = \{f_j | \beta_j \neq 0\}$. We may assume that $F' \neq F$ since otherwise, we could use $\sum_j f_j$ to eliminate ne of the coefficients. Let e be an edge of the *boundary* of F'. Clearly, the coefficient of e in $\partial_2(\sum_j \beta_j f_j)$ cannot be zero! This contradicts that $\sum_j \beta_j f_j \in ker(\partial_2)$.

Finally, we examine $Im(\partial_1)$. Note that $\partial_1(e) = v - v'$ satisfies the relation that the sum of the vertex coefficients is zero. Thus $Im(\partial_1$ is certainly a subspace of the claimed space. Next, note that the space of all vertex combinations with coefficient sum zero, is of dimension $|V| - 1$. A basis for this are the $n - 1$ vectors $v_n - v_1, \ldots, v_2 - v_1$. Now, we prove that each of these vectors is in the image $Im(\partial_1)$. Since the surface of the solid consists of exactly one shell, there is a path π_k from v_1 to every vertex v_k. Suppose

$$\Pi_k = sign(\pi_k, e_1) \cdot e_1 + \ldots + sign(\pi_k, e_r) \cdot e_r$$

where $sign(\pi_k, e_j)$ is the sign of the orientation $o(e_j)$ with respect to the travel direction of π_k. We see easily that $\partial_1(\Pi_k) = v_k - v_1$. $\Box$

With this behind us, let $F' = F - \{f_1\}$, be the set of faces F *minus* a fixed face. Let $\mathbb{R}F'$ be the space of all formal linear combinations of faces in F'. Further, let $\mathbb{R}V'$ be the subspace of all elements $\sum_i \alpha_i v_i$, such that $\sum_i \alpha_i = 0$. We have the following proposition:

Proposition 15.5 *We consider the diagram*

$$\mathbb{R}F' \overset{\partial_2'}{\rightarrow} \mathbb{R}E \overset{\partial_1'}{\rightarrow} \mathbb{R}V'$$

(i) The composition $\partial_1' \circ \partial_2'$ *is the zero map.*
(ii) The map ∂_2' *is injective, i.e.,* $ker(\partial_2')$ *consists of just the zero vector* 0*. Further,* $Im(\partial_2') = Im(\partial_2)$*.*
(iii) The map ∂_1' *is surjective, i.e., the image* $Im(\partial_1')$ *equals* $\mathbb{R}V'$*. Further* $ker(\partial_1') = ker(\partial_1)$*.*

Proof: Part (i) follows from the lemma above. Next, (ii) also follows since the vector $f_1 + \ldots + f_r$ is not an element of $\mathbb{R}F'$. Further, note that any element of $\mathbb{R}F$ is obtained as a sum of an element of $\mathbb{R}F'$ and the vector $f_1 + \ldots + f_r$. Thus, the image of ∂_2 restricted to $\mathbb{R}F'$ must equal that of the image of $\mathbb{R}F$. Finally (iii) is just a restatement of (iii) in the previous lemma. $\Box$

We are now in a position to use the snake lemma:

Theorem 15.6 *For the boundary map $\partial_2 : \mathbb{R}F \to \mathbb{R}E$ and the cycle map $\partial_1 : \mathbb{R}E \to \mathbb{R}V$, we have*

$$dim(ker(\partial_1)/Im(\partial_2)) = |E| - |V| - |F| + 2$$

Proof: This follows easily from Proposition 15.5 and Proposition 15.3. $\Box$

16 The MATLAB spline toolbox

We explain the basic spline structure as supported by Matlab. The knot vector in Matlab is slightly different from our notation, viz., the Bezier knot of degree 3 for us is [000111] while for Matlab is [00001111]. Our knot [0001222] becomes [000012222]. In short, Matlab knots have an extra repetition at the start and the end. Thus to create a bezier curve of degree 3, we call:

```
 >>cu=KcreateCurve([0 0 0 0 1 1 1 1],[2 3 1 2])

cu =

      form: 'B-'
     knots: [0 0 0 0 1 1 1 1]
     order: 4
    number: 4
       dim: 1
     coefs: [2 3 1 2]
```

We see above the associated fields of the curve structure. The function `KcreateCurve` is given below:

```
  function sp=KcreateCurve(kn,coefs);

  [ig,nos]=size(kn);
  yes=1; first=kn(1);

  count=1;
  while yes==1
  count=count+1;
  if kn(count)~=first
    yes=0;
  end;
  end;

  order=count-1;
  [mm,nn]=size(coefs);

  sp.form='B-';
```

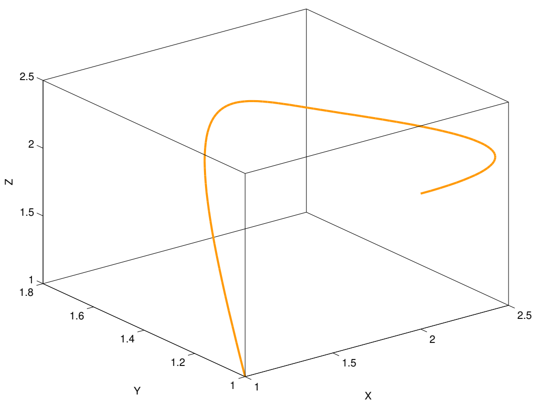

Figure 49: A spline curve in 3-space

```
sp.knots=kn;
sp.order=order;
[ig,m1]=size(kn);
sp.number=m1-order;
if sp.number~=nn
  error('incorrect number of coefficients');
end;

sp.dim=mm;
sp.coefs=coefs;

return;
```

The only thing of note is the assignment of `sp.dim`. This is the dimension of the model space. The coefficient matrix `coefs` consists of the right number of columns in the model space. Here is another example of a 3-dimensional curve.

```
cu=KcreateCurve([0 0 0 0 1 2 2 2 2],[2 3 1 2 1; 1 1 1 2 1; 2 2 3 2 1])

cu =

      form: 'B-'
     knots: [0 0 0 0 1 2 2 2 2]
     order: 4
    number: 5
       dim: 3
     coefs: [3x5 double]
```

We may plot this curve using `fnplt(cu)` to obtain Fig. 16

The function `fnder` computes the derivative of a given curve as a spline:

```
dcu=fnder(cu,1)

dcu =

      form: 'B-'
     knots: [0 0 0 1 2 2 2]
     coefs: [3x4 double]
    number: 4
     order: 3
       dim: 3

ddcu=fnder(cu,2)

ddcu =

      form: 'B-'
     knots: [0 0 1 2 2]
     coefs: [3x3 double]
    number: 3
     order: 2
       dim: 3
```

Notice how the knot-vector changes as we differentiate. The function `fnval` evaluates a spline at the specified parameter. For example:

```
p1=fnval(cu,0.5)

p1 =

    2.3438
    1.0312
    2.2500

tangent1=fnval(dcu,0.5)

tangent1 =

   -0.9375
    0.1875
    0.7500

q1=p1+tangent1

q1 =

    1.4062
    1.2188
    3.0000
```

We may plot `p1,q1` to obtain Fig. 16 Subdivision is done by `spbrk`. Again notice the knot-vector

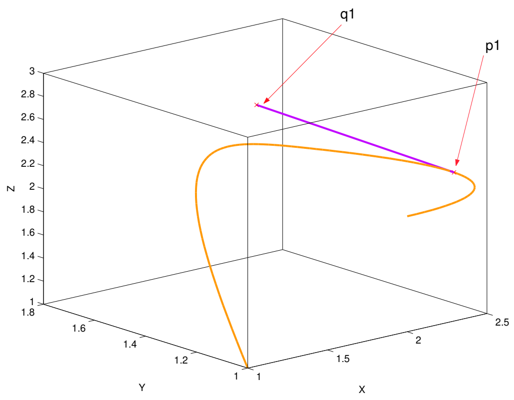

Figure 50: Tangents

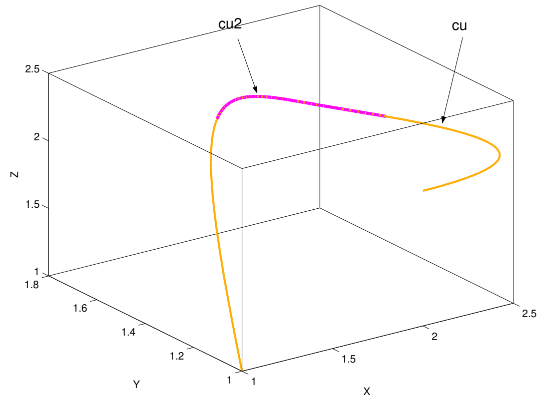

Figure 51: Subdivision

and the plot (Fig. 16).

```
cu2=spbrk(cu,[0.8 1.5])

cu2 =

      form: 'B-'
     knots: [0.8000 0.8000 0.8000 0.8000 1 1.5000 1.5000 1.5000 1.5000]
     coefs: [3x5 double]
    number: 5
     order: 4
       dim: 3
```

Surfaces are created and operated upon by similar procedures. Here is an example of surface creation.

```
cc

cc(:,:,1) =

    0.9553    0.8931    0.6967    0.4236    0.2675
   -0.2823   -0.4298   -0.6853   -0.8654   -0.9205
    0.0873    0.1330    0.2120    0.2677    0.2848


cc(:,:,2) =

    0.9553    0.8931    0.6967    0.4236    0.2675
   -0.2382   -0.3627   -0.5782   -0.7302   -0.7767
    0.1749    0.2663    0.4246    0.5361    0.5703


cc(:,:,3) =

    0.9553    0.8931    0.6967    0.4236    0.2675
   -0.1679   -0.2556   -0.4075   -0.5146   -0.5473
    0.2432    0.3703    0.5904    0.7455    0.7930


cc(:,:,4) =

    0.9553    0.8931    0.6967    0.4236    0.2675
   -0.0791   -0.1204   -0.1919   -0.2423   -0.2578
    0.2848    0.4335    0.6912    0.8728    0.9284

su3=KcreateSurf({[0 0 0 0 0.5 1 1 1 1] [0 0 0 0 1 1 1 1]},cc)

su3 =

      form: 'B-'
     knots: {[0 0 0 0 0.5000 1 1 1 1]  [0 0 0 0 1 1 1 1]}
     order: [4 4]
    number: [5 4]
       dim: 3
     coefs: [3x5x4 double]

fnplt(su3)
```

The matrix `cc` is $3 \times 5 \times 4$. See that `kn` is now a 2-tuple of knot vectors, the u-knot and the v-knot. Also note that `su3.order` and `su3.number` are 2-tuples corresponding to the u-knot and the v-knot. The coefficients have been copied from the input argument `cc`. The (i,j)-th control point is the vector $[cc(1,i,j), cc(2,i,j), cc(3,i,j)]$.

The plotted surface appears in Fig. 16. The control points happen to lie on a sphere for our example.

Other operations are illustrated in the following code:

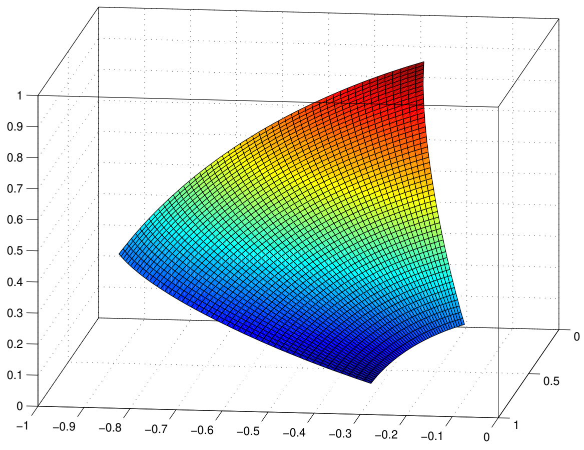

Figure 52: A spline surface

```
su4=spbrk(su3,{[0.1 0.5] [0.5 1]})

su4 =

      form: 'B-'
     knots: {1x2 cell}
     coefs: [3x4x4 double]
    number: [4 4]
     order: [4 4]
       dim: 3

fnval(su4,[0.2 0.6]')

ans =

    0.8645
   -0.2809
    0.3546
dsu=fnder(su3,[0 1])

dsu =

      form: 'B-'
     knots: {[0 0 0 0 0.5000 1 1 1 1]  [0 0 0 1 1 1]}
     coefs: [3x5x3 double]
    number: [5 3]
     order: [4 3]
       dim: 3
```

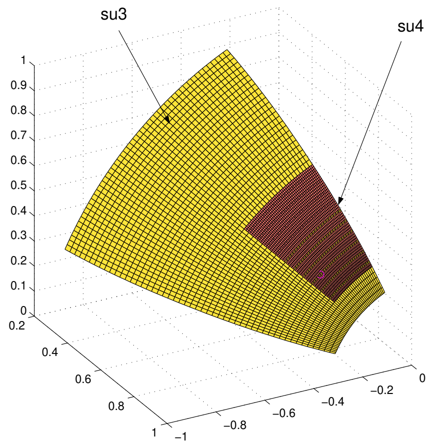

Figure 53: Subdivision