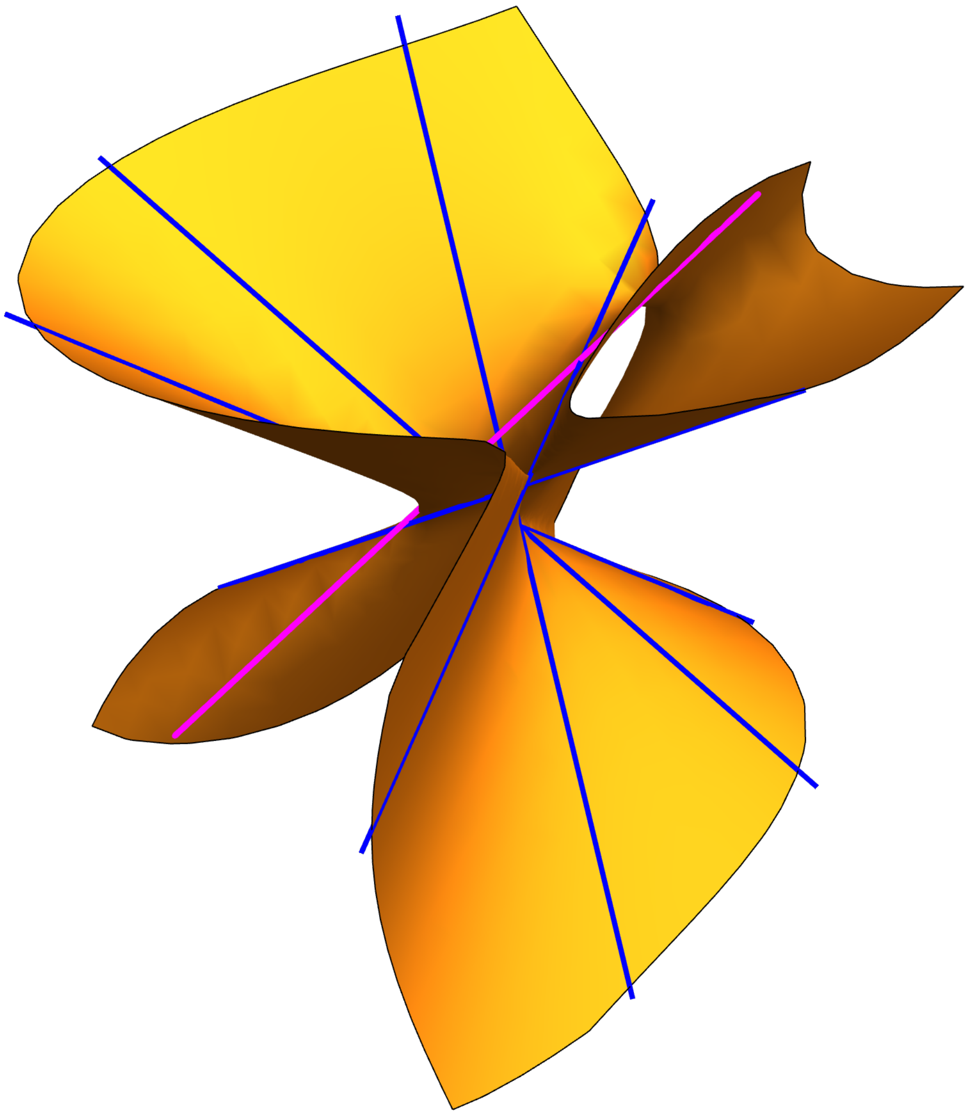
Surface Story

Part I

Barry H Dayton
barryhdayton.space



0. Introduction

Surfaces are much more complicated than curves. For example Riemann defined genus of an algebraic curve in the 1850's. But there was no genus of an algebraic surface until the 1950's. The interim was spent abstracting algebra and topology to build the tools for the general Riemann-Roch theorem. Unfortunately the new abstract formulation, while impressive mathematically, gave little insight into actual surfaces. Instead I will attempt to discuss surfaces not with abstractions but with Mathematica algorithms.

In this story I will restrict my attention to surfaces which are either naive algebraic surfaces or surfaces defined by a rational parameterization. In particular I will then be able to plot these surfaces, at least locally, using Mathematica's **`ContourPlot3D`** in the first case and **`ParametricPlot3D`** in the second. Again my intention is to be visual and numerical rather than mathematically exact.

This book is addressed to readers of my Plane curve book and my Space Curve Book. In particular one should be familiar with working with machine numbers in Mathematica. Other than that there will be no prerequisite. Many of the functions used in this book are already in the Global Functions notebook for my Space Curve book which already contains many of the Plane curve functions. There is an inclusive **`GlobalFunctionsNS.nb`** notebook for this book. Global functions specifically for surfaces may end in NS (naive surface) or RS (rational surface).

Belated Acknowledgement: This acknowledgment was left out of the Plane Curve book by the editors, possibly inadvertently. I did not think to include it in the Space Curve Book. But it is time to give a belated acknowledgement to Hugh J Hamilton who was my Professor in a year course in Advanced Calculus at Pomona College in the 1964-65 academic year. Hamilton appeared to be a very precise and thrifty little Scotsman who always wore a Scottish plaid tie and sports coat but he was active in the Southern California Socialist party. His lecture style was to copy the textbook to the blackboard. He especially emphasized the epsilons and deltas. But his unique characteristic was his tests, 2 or 3 problems asking for an essay style presentation of an application of the theory similar to my presentation of material in this book. Later he published the textbook *A primer of Complex Variables with an Introduction to Advanced Techniques* , Brooks Cole Publisher, a completely heuristic book with hardly an epsilon or delta. These books are motivated by his example.

Table of Contents

1. Introduction to Surfaces

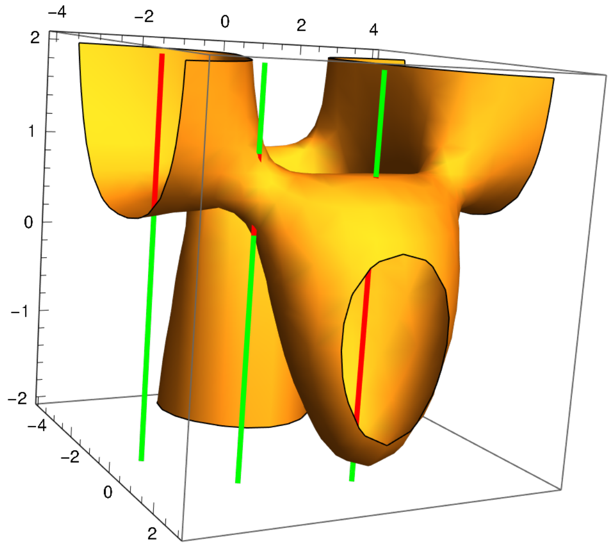
1.1 Introduction to Naive Surfaces

A naive surface is a surface in $\mathbb{R}^3$ which is the full zero set of a single polynomial equation f=f(x,y,z) in three variables subject to a few conditions to be discussed later. For example the polynomial might be

```
ts3 = 1.752 - 6.4 x - 11.464 x^2 + 0.64 x^3 + x ^ 4 + 1.536 y^2 +
   0.64 x y^2 + x^2 y^2 + 2.88 x ^ 2 z - 5.12 y ^ 2 z + 3.584 z^2 + 3.84 x z^2 + x^2 z^2;
```

Analogously to Gauss' principle in my Plane Curve Book this zero set divides the plane into two sets $f^+ = \{\{x,y,z\} \mid f(x, y, z) > 0\}$ and $f^- = \{\{x, y, z\} \mid f(x, y, z) < 0\}$ which have the zero set of f as the complete boundary. This allows us to recover this zero set, which we will often just call f, by looking for points where the value of f(x,y,z) on neighboring points changes from positive to negative or vice versa. In **Mathematica** this is obtained using the built-in function **ContourPlot3D.** For example we can visualise a small part of the **surface ts3** by

Out[◦]=



Plot 1.1a

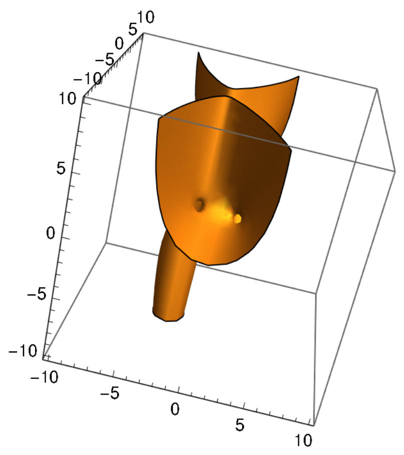
Note that in this book I will generally use the option **Mesh->None** because we will often be drawing curves on our surfaces. It is important to note that the boundary curves in this picture are simply the curves where this surface meets the bounding box, they are not intrinsic to this surface. Note the 3 vertical lines colored green where ts3 > 0 and red where ts3 < 0. What we notice is that they are red "inside" the surface and green "outside". This shows that the surface is two sided with an inside and outside. We talk about this more in a bit.

Note this plot changes as we change the bounding box or orientation. We can see more or less of the surface or more or less detail.

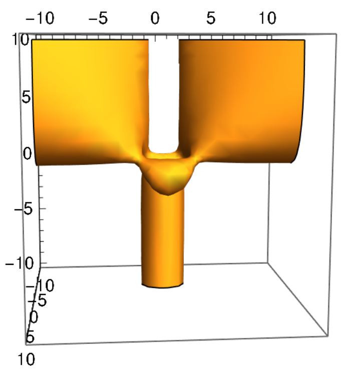
```
In[ ]:= Pl1b = ContourPlot3D[ts3 == 0, {x, -10, 10}, {y, -10, 10}, {z, -10, 10}, Mesh → None];
```

```
In[ ]:= {Pl1b, Pl1b}
```

Out[ ]= {



,

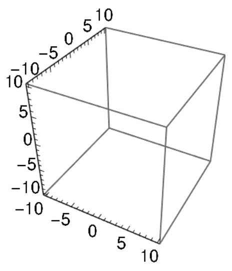


}

Some things can go wrong . The equation $x^2+y^2+z^2=0$ has only one solution, {0,0,0}. We call equations that do not give a 2-dimensional figure *degenerate.* Also note that the equation $ts3^2 = 0$ has the same solution set as ts3 = 0 but the contour plot

```
In[ ]:= ContourPlot3D[ts3^2 == 0, {x, -10, 10}, {y, -10, 10}, {z, -10, 10}, Mesh → None]
```

Out[ ]=



is empty. This is because there is no sign change from positive to negative. Remember that since the function **ContourPlot3D** is numerical, zero is not recognized as a number. So changes from positive to zero are not detected. So to get a correct picture we must use square free polynomials only. Fortunately we have a global function sqFreeMD, this will not only tell us if a polynomial is square free but if it is not it will return a square free polynomial with the same solution set. Fortunately this function does not require us to factor the polynomial so it works on numerical as well as integer polynomials.

Here is a more complicated problem that came up with a surface related to ts3, I call it ts2.

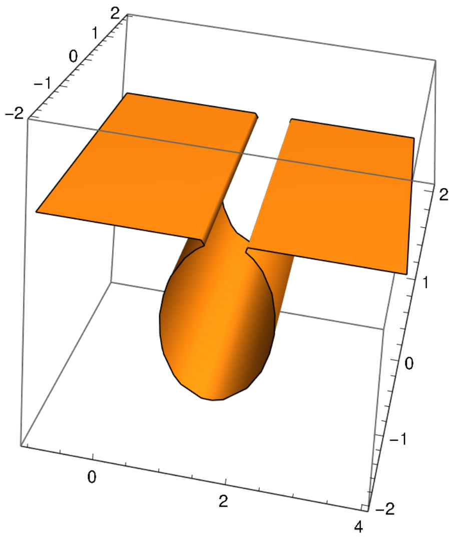
In[ ]:=
```
ts2 = N[Expand[(-1 + z) × (48 - 80 x + 25 x^2 + 16 z^2)]]
```

Out[ ]=
```
-48. + 80. x - 25. x^2 + 48. z - 80. x z + 25. x^2 z - 16. z^2 + 16. z^3
```

When we try to plot ts2 we get the following

In[ ]:=
```
ContourPlot3D [ts2 == 0, {x, -1, 4}, {y, -2, 2}, {z, -2, 2}, Mesh → None]
```

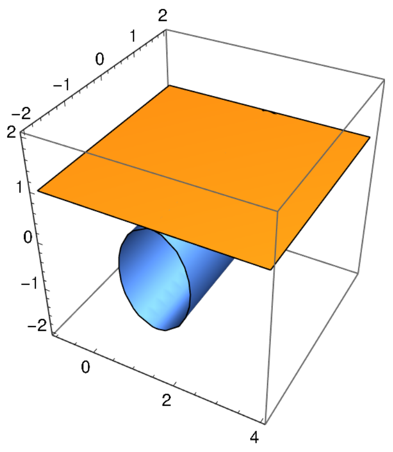
Out[ ]=



But this surface is the union of a plane and a cylinder so the plot should be

In[ ]:=
```
ContourPlot3D [{z - 1 == 0, 48 - 80 x + 25 x^2 + 16 z^2 == 0},
 {x, -1, 4}, {y, -2, 2}, {z, -2, 2}, Mesh → None]
```

Out[ ]=



The problem is that there is a line of intersection $\{y = 0,\ z = \frac{25}{16}\}$ of these two surfaces. Even though this line is not a factor of either component it is somehow counted twice in the contour plot of the product, which is square free.

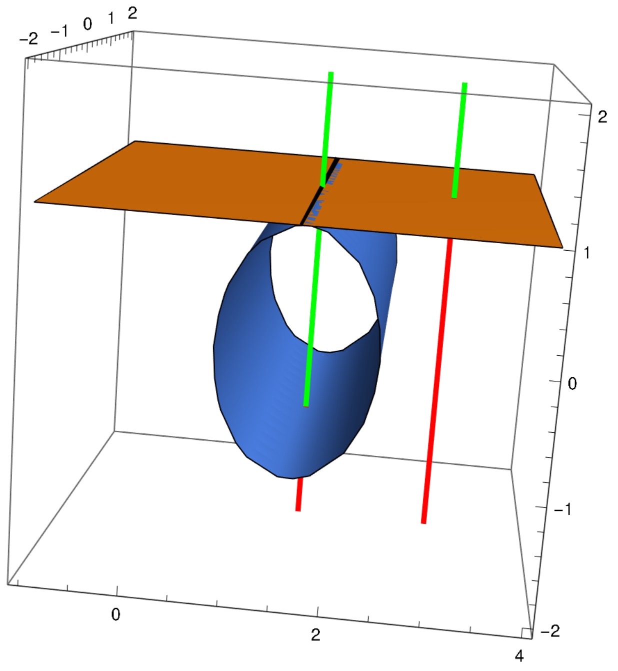
In[ ]:=
```
sqFreeMD[ts2, {x, y, z}, dTol]
```

```
» Square Free
```

Out[ ]= $-48. + 80.\,x - 25.\,x^2 + 48.\,z - 80.\,x\,z + 25.\,x^2\,z - 16.\,z^2 + 16.\,z^3$

Here is a picture .

In[ ]:=
```
Show[ContourPlot3D [{z - 1 == 0, 48 - 80 x + 25 x^2 + 16 z^2 == 0}, {x, -1, 4}, {y, -2, 2}, {z, -2, 2},
   Mesh → None], ParametricPlot3D [{1.5625 , t, 1}, {t, -2, 2}, PlotStyle → Black],
 ParametricPlot3D [{1.5625 , 0, t}, {t, 1, 2}, PlotStyle → Green],
 ParametricPlot3D [{1.5625 , 0, t}, {t, -1, 1}, PlotStyle → Green],
 ParametricPlot3D [{1.5625 , 0, t}, {t, -2, -1}, PlotStyle → Red],
 ParametricPlot3D [{3, 0, t}, {t, 1, 2}, PlotStyle → Green],
 ParametricPlot3D [{3, 0, t}, {t, -2, 1}, PlotStyle → Red]]
```

Out[ ]=


In some ways the original, wrong picture, did a better job of explaining the inside and outside of the surface!

1.1.2 Regular and Smooth Surfaces

Before stating our main theorem in this section we make a definition. A point p in a surface f is *regular* if the norm of the gradient is greater than zero. This is implemented, in the case of point p in ts2

```
In[ ]:= p = {25 / 16, 2, 1}
        ts2 /. Thread[{x, y, z} → p]
        grd = Grad[ts2, {x, y, z}] /. Thread[{x, y, z} → p]
```

Out[ ]= $\left\{\frac{25}{16}, 2, 1\right\}$

```
Out[ ]= 0.

Out[ ]= {0., 0, 0.0351563}
```

Here p, and ts2 are exact so the last component of the gradient is sufficiently large to be non-zero.

An important property of regular points is that we get a *tangent plane* and *normal line*.

```
In[ ]:= tangentPlaneNS[f_, p_, X_] := (Grad[f, X] /. Thread[{x, y, z} → p]).(X - p)
        normalLineNS[f_, p_, X_] := lineMD[p, Append[(Grad[f, X] /. Thread[{x, y, z} → p]), 0], X]
```

In the example above

```
In[ ]:= tpp = tangentPlaneNS[ts2, {25 / 16, 2, 1}, {x, y, z}]
        nlp = normalLineNS[ts2, {25 / 16, 2, 1}, {x, y, z}]

Out[ ]= 0. + 0.0351563 × (-1 + z)
```

Out[ ]= $\{-0.100593 - 0.759004\ x + 0.643268\ y + 9.28877 \times 10^{-17}\ z,$
$0.924931 - 0.309563\ x - 0.22062\ y - 1.97547 \times 10^{-17}\ z\}$

Of course this just says the tangent plane to the plane $z = 1$ at the regular point of ts2 is the plane $z - 1$. But this example exposes a problem because we want to consider the points where the cylinder meets the plane tangently as *singular.* Fortunately we did give a good discussion of multiplicity in my *Space Curve Book* section 2.3.3.1. In this example

```
In[ ]:= multiplicityMD[Prepend[nlp, ts2], {25 / 16, 2, 1}, {x, y, z}, 1*^-6]

Out[ ]= 2
```

Note that we can also get the multiplicity directly from NSolve.

```
In[ ]:= NSolveValues[Append[nlp, ts2], {x, y, z}, Reals]

Out[ ]= {{1.5625, 2., -0.998901}, {1.5625, 2., 0.998901}, {1.5625, 2., 1.}}
```

The last two zeros are numerically *p* so *p* is a double point.

So our normal line meets the surface in a double point, as can be easily seen from the plot above.

We thus define a surface to be *smooth* or *non-singular* at point *p* if both the gradient is non-zero and the multiplicity of the intersection of the normal line and surface is 1. A point where either the gradient is zero or the intersection of the normal line and surface has multiplicity 2 or greater with a loose tolerance is called *singular.*

It should be mentioned that [Abhyankar, p.205] mentions that, in our notation, the set of non-regular points must be algebraic, in our case a finite point set or a curve, as in ts2, but the set of singular points need not be algebraic.

Our main theorem, slightly modified from a standard theorem of differential geometry is

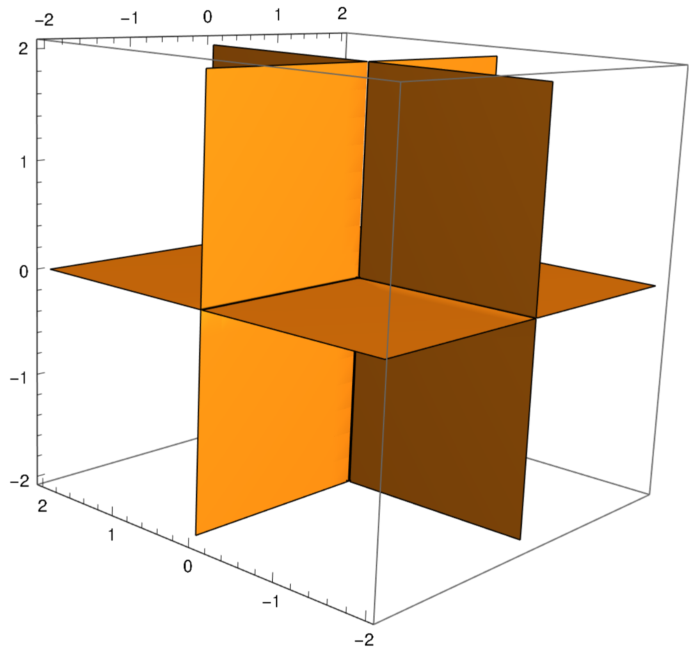
Jordan - Brouwer *Let f be a non-degenerate square free polynomial giving a smooth surface. Then the surface f is two sided, moreover for p in the surface there is a neighborhood of p which is topologically an open plane disk.*

What this means is that the points of a smooth naive surface define an *oriented manifold.* To see a definition and discussion this see a differential geometry text such as [Montiel, Ros].

We will only refer to smooth surfaces as having sides. As an example consider the surface $x\,y\,z = 0$

```
In[ ]:= ContourPlot3D [x y z == 0, {x, -2, 2}, {y, -2, 2}, {z, -2, 2}, Mesh → None , MaxRecursion → 4]
```

Out[ ]=



These planes actually break up space into 8 regions rather than 2, so sides are not actually a useful concept.

The Jordan - Brower theorem only refers to affine surfaces. The projective surface defined by $f = 0$, more specifically the projective closure of the affine $f = 0$, need not be two sided, orientable. A crite-rion for a smooth real projective surface to be two sided is that for every line which intersects this surface only transversely, that is does lie in this surface , must intersect the surface in an even number, counted by multiplicity, of projective points. In particular the plane $z = 0$ is one sided as a projective surface. For more information see section 1.9.

1.2 Introduction to Rational Parametric Surfaces

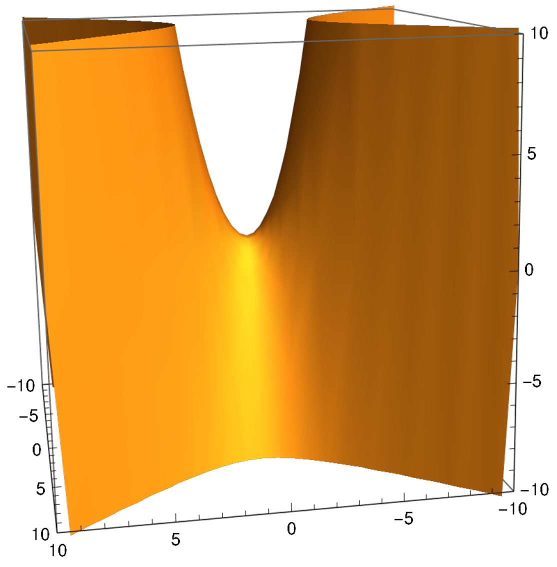
A second way to define a surface is to use a rational parametric function. A simple one is

```
In[ ]:= F1 = {s, t, s^2 - t^2}
```

Out[ ]= $\{s, t, s^2 - t^2\}$

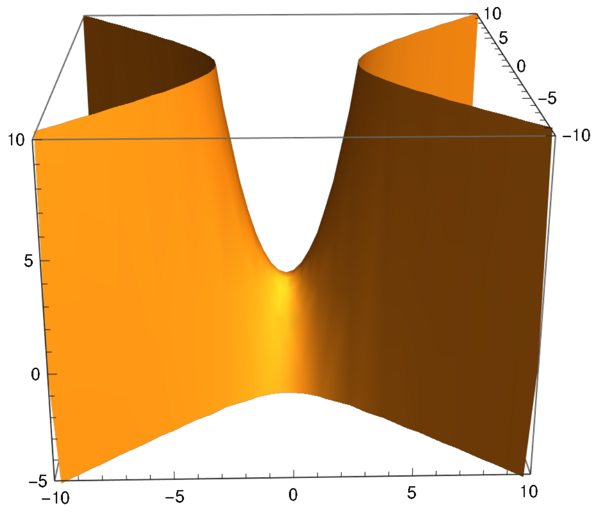
We can plot part of this surface using `ParametricPlot3D`.

```
In[ ]:= ParametricPlot3D[F1, {s, -10, 10}, {t, -10, 10}, PlotRange → 10, Mesh → None]
```

Out[ ]=


Unlike contour plots giving a plot range is optional, but in most cases a good idea to get a nice plot. Once could also do this to control each variable separately with

```
In[ ]:= ParametricPlot3D[F1, {s, -10, 10}, {t, -10, 10},
          PlotRange → {{-10, 10}, {-10, 10}, {-5, 10}}, Mesh → None]
```

Out[ ]=


As with contour plots I disable the Mesh because I will want to draw my own curves on this surface. One can also use the option `MaxRecursion` with parametric plots if the plot is complicated.

More generally a *rational parametric surface in* $\mathbb{R}^3$ is given by a function

$$F = \left\{\frac{f_1(s, t)}{f_4(s, t)}, \frac{f_2(s, t)}{f_4(s, t)}, \frac{f_3(s, t)}{f_4(s, t)}\right\}$$

where the f_i are polynomial functions of the two variables s, t.

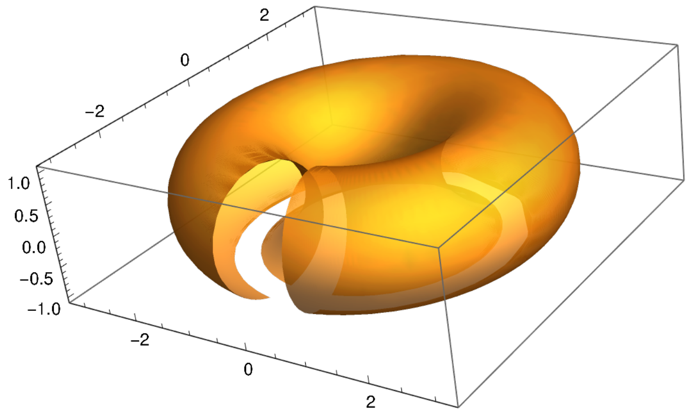
We generally like to have the common denominator f_4 but it is not absolutely required as it can be calculated, the important thing is that no denominator is the constant 0. We do not require the numerators and the denominator to have the same degree, the degree of the numerators may be less than, equal or greater than the degree of the numerator and different from each other. In the polynomial case of F1 above the denominators are all the constant 1 of degree 0. When the parameters $\{s, t\}$ make $f_4(s, t) = 0$ we say F is undefined or *infinite,* in Chapter 2, particularly, we will use the latter terminology. This zero set of the denominator may be a discrete point set or a curve. When working with rational parametric surfaces the default range of s, t is $-\infty < s, t < \infty$ in this chapter, however specific examples may have a smaller range.

Here is a non-trivial example of a rational parametric surface, the *torus*. Note in this case the definition does not give a common denominator but it is easily seen that a common denominator would be $(1+s^2)\times(1+t^2)$.

In[ ]:=
$$Ts = \left\{\frac{4 s (1+t+t^2)}{(1+s^2)\times(1+t^2)}, -\frac{2\times(-1+s^2-t+s^2 t-t^2+s^2 t^2)}{(1+s^2)\times(1+t^2)}, \frac{1-t^2}{1+t^2}\right\};$$

In plotting a rational surface we can not, in general, show the entire surface so we pick a large bounded range.

In[ ]:=
```
ParametricPlot3D [Ts, {s, -10, 10}, {t, -10, 10}, PlotRange → All,
  Mesh → None, MaxRecursion → 4, PlotStyle → Opacity[.8]]
```

Out[ ]=


We see this finite range gives a deformed rectangle curved in both dimensions. We can easily imagine that if we used the full range $-\infty < s, t < \infty$ we would get a torus. The opacity[.8] is to help visualize that there is a strip missing on the bottom, the
MaxRecursion→ 4 helps to smooth out the plot.

At a given point of a rational parameterization $\{s_0, t_0\}$ we can take the partial derivatives and evaluate to get vectors. For example with the torus Ts and point $p = \{2, 3\}$

```
In[ ]:= p = {2, 3};
        vs = D[Ts, s] /. Thread[{s, t} → p]
        vt = D[Ts, t] /. Thread[{s, t} → p]
        Tsp = Ts /. Thread[{s, t} → p]
```

Out[ ]= $\left\{-\frac{78}{125}, -\frac{104}{125}, 0\right\}$

Out[ ]= $\left\{-\frac{16}{125}, \frac{12}{125}, -\frac{3}{25}\right\}$

Out[ ]= $\left\{\frac{52}{25}, -\frac{39}{25}, -\frac{4}{5}\right\}$

The normal vector is is the cross product vs×vt

```
In[ ]:= nv = Cross[vs, vt]
```

Out[ ]= $\left\{\frac{312}{3125}, -\frac{234}{3125}, -\frac{104}{625}\right\}$

and the tangent plane is nv.(X-F(p))

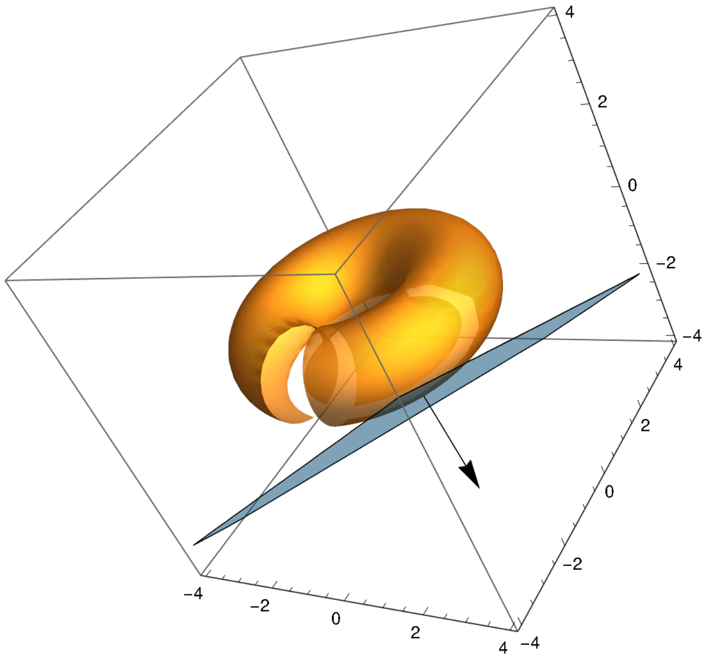
```
In[ ]:= tp = nv.({x, y, z} - Tsp)
```

Out[ ]= $\frac{312 \times \left(-\frac{52}{25} + x\right)}{3125} - \frac{234 \times \left(\frac{39}{25} + y\right)}{3125} - \frac{104}{625} \times \left(\frac{4}{5} + z\right)$

or, better

```
In[ ]:= tp = Expand[N[tp]]
Out[ ]= -0.4576 + 0.09984 x - 0.07488 y - 0.1664 z
```

In[•]:=
```
Show[ContourPlot3D[tp == 0, {x, -4, 4}, {y, -4, 4}, {z, -4, 4},
   Mesh → None, ContourStyle → Directive[Cyan, Opacity[.5]]],
 ParametricPlot3D[Ts, {s, -10, 10}, {t, -10, 10}, PlotRange → All,
   Mesh → None, MaxRecursion → 4, PlotStyle → Opacity[.8], PlotRange → All],
 Graphics3D[{Black, Arrow[{Tsp, Tsp + 10 nv}]}]]
```

Out[•]=


The general code is

In[•]:=
```
normalVectorRS[F_, st0_, st_] := Module[{pv, vs, vt},
   vs = D[F, st[[1]]] /. Thread[st → st0];
   vt = D[F, st[[2]]] /. Thread[st → st0];
   Cross[vs, vt]]

tangentPlaneRS[F_, st0_, st_, X_] := Module[{nv, p},
   p = F /. Thread[st → st0];
   nv = normalVectorRS[F, st0, st];
   N[Expand[nv.(X - p)]]]
```

For this example

In[•]:=
```
normalVectorRS[Ts, {2, 3}, {s, t}]
tangentPlaneRS[Ts, {2, 3}, {s, t}, {x, y, z}]
```

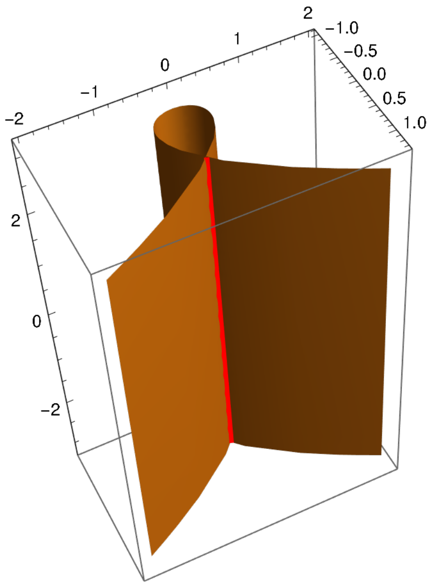
Out[•]= $\left\{\frac{312}{3125}, -\frac{234}{3125}, -\frac{104}{625}\right\}$

Out[•]=
```
-0.4576 + 0.09984 x - 0.07488 y - 0.1664 z
```

As with naive surfaces a rationally parameterized surface $F(s, t)$ is regular at {s0, t0} if there is a tangent plane at F(s0, t0). But as with naive algebraic surfaces regularity at {s0,t0} does not imply smoothness at F(s0,t0). But the situation is very different. For naive surfaces it is a local problem, for rationally parameterized surfaces it is a global problem. Here are two examples.

```
In[ ]:= node3D = {t ^ 2 - 1, t ^ 3 - t, s}
Out[ ]= {-1 + t^2, -t + t^3, s}
```

```
In[ ]:= Show[ParametricPlot3D [node3D , {s, -3, 3}, {t, -1.5 , 1.5}, Mesh → None],
         Graphics3D [{Red , Thickness [.01], Line[{{0, 0, -3}, {0, 0, 3}}]}]]
```

Out[ ]=


Note the line $x = y = 0$ appears to be a singular locus of this surface. But points on this line are of the form

```
In[ ]:= node3D /. Thread[{s, t} → {s, -1}]
        node3D /. Thread[{s, t} → {s, 1}]
Out[ ]= {0, 0, s}
Out[ ]= {0, 0, s}
```

However

```
In[ ]:= normalVectorRS [node3D , {s, -1}, {s, t}]
        normalVectorRS [node3D , {s, 1}, {s, t}]
Out[ ]= {-2, -2, 0}
Out[ ]= {-2, 2, 0}
```

are non - zero, so all of these points are regular in the the parameters. The problem is that different parameter values give the same points. While harder to deal with the problem is no worse than with ts2 so we have nothing to do.

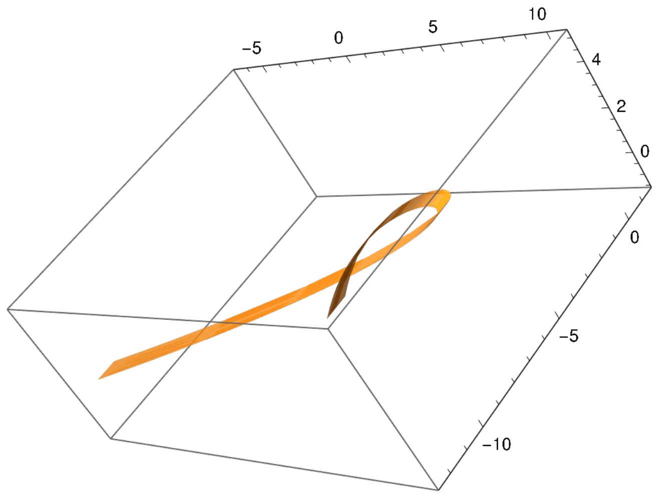
A second example is similar but causes an additional problem.

```
In[ ]:= ribbon = {t^3 + 2, s^2 - 3 t^2, t^2 + t - 2 + 1}
```

Out[ ]= $\{2 + t^3,\ s^2 - 3\,t^2,\ -1 + t + t^2\}$

```
In[ ]:= ParametricPlot3D[ribbon, {s, -1, 1}, {t, -2, 2}, Mesh → None, PlotStyle → Opacity[.8]]
```

Out[ ]=



Here the plot does not show a self intersection. However

```
normalVectorRS[ribbon, {s, b}, {s, t}]
```

Out[ ]= $\{2\,s + 4\,b\,s,\ 0,\ -6\,b^2\,s\}$

so when $s = 0$ this is not regular. When s, t are both non-zero then it is regular but note that ribbon3D(s,t) = ribbon3D(-s,t) so each point on the surface is double, that is, comes from two different parameter values so cannot be considered smooth.

This reminds one of Einstein's "spooky action at a distance". If we can only see a parameter space for the universe rather than the actual universe then an atom seemingly far away perhaps behaves the same as one nearby because in the universe it may actually be the same atom. A spooky alien transmission from a planet circling a distant star could just be Fox News.

This is not a pleasant thought. For the ribbon example we can fix this problem by insisting that $s > 0$. But this parametric surface has an edge, it does not go on infinitely in the negative s direction.

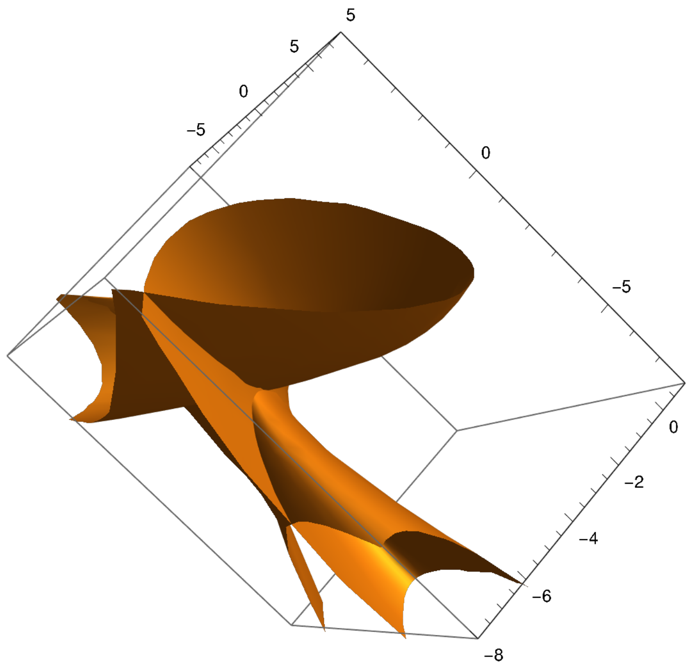
In the next section we will discover the real answer to this problem that we can not see the true nature of a point of the parametric surface just working locally, mainly that rational parametric surfaces, even the ribbon, are subsets of naive algebraic surfaces.

I leave you with a plot of a more complicated rational parametric example using only cubic functions. I will not try to analyze this here.

In[ ]:= **strange1** = $\{-3 - 3\,s^2 - 3\,s^3 + 3\,s\,t - s^2\,t + 2\,t^2 - 3\,s\,t^2 + 3\,t^3,$
$-2 - 3\,s^2 - s^3 + 2\,t - s^2\,t - t^2 + s\,t^2,\ s + 2\,s^2 + 3\,s^3 - 3\,t - s\,t - 2\,t^2 - 3\,s\,t^2 + 3\,t^3\};$

```
In[ ]:= ParametricPlot3D[strange1, {t, -5, 5}, {s, -5, 5},
         PlotRange → {{-8, 8}, {-8, 1}, {-8, 5}}, Mesh → None, MaxRecursion → 4]
```

Out[ ]=



1.3 Implicit Equation Theorem for Rational Parametric Surfaces

Here we give two proofs that every rational parametric surface is contained in a naive in a naive surface. The first is more theoretical, the second somewhat more practical.

1.3.1 Theoretical Method

A proof in the curve case appeared in my Mathematica Journal article [Dayton, *Degree vs Dimension of Rational Parametric Curves]*. Another discussion is in my **`Space Curve Book 3.1.4.`**
The proof for surface is slightly modified but the idea is the same: a rational parametric function can be viewed as Fractional Linear Transformation (FLT) from an appropriate generic curve.

We write our parametric surface in the standard form of §1.1 with a common denominator. Since we now have two parameters if m is the largest degree of a monomial there are binomial coefficient $\binom{m+2}{2}$ bivariate monomials of degree m or less. This number, the dimension of the space of generic curves of degree m, can become uncomfortably large. It turns out that it enough to just use the monomials actually used in the rational parametric function and monomials that divide these.

The method is thus to take this set of n monomials, calling them $X[1], X[2]\ldots, X[n]$. We take a set of relations between these variables and find a HBasis for this using, because it is faster in this case, a Groebner basis for this basis. We construct a (n+1)×4 matrix for our FLT matrix. Then an application of FLTMD will give an equation set defining the smallest algebraic surface in $\mathbb{R}^3$ containing the image surface of our FLT. Any equation of this set will contain our parameterized surface so we can just pick one. While this single equation, defining a naive surface, may not completely describe our surface which may be smaller it will serve to give us a Jordan-Brouwer theorem and this surface can find locally the local behaviour of this function at a smooth point.

We proceed with an example

In[ ]:= **hyperboloid** $= \left\{\frac{t - s\text{^}2\, t}{1 - s\text{^}2}, \frac{1 + s^2 - 2\, s\, t}{1 - s^2}, \frac{2\, s - t - s^2\, t}{1 - s^2}\right\};$

I collect the monomials used

In[ ]:= $V = \{s, t, s\, t, s^2, s^2\, t\};$

I now find a 8×4 matrix which produces this rational function via a transformation function, note the first 7 columns can be indexed by the monomials in V and the last column is the constant. The first 3 rows are from the numerator, the last from the denominator.

```
In[ ]:= A = {{0, 1, 0, 0, -1, 0}, {0, 0, -2, 1, 0, 1}, {2, -1, 0, 0, -1, 0}, {0, 0, 0, -1, 0, 1}};
        A // MatrixForm
Out[ ]//MatrixForm=
```

$$\begin{pmatrix} 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -2 & 1 & 0 & 1 \\ 2 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$

To check

```
In[ ]:= TransformationFunction [A][V]
```

Out[ ]= $\left\{\frac{t - s^2 t}{1 - s^2}, \frac{1 + s^2 - 2 s t}{1 - s^2}, \frac{2 s - t - s^2 t}{1 - s^2}\right\}$

Next I treat the monomials as variables

```
In[ ]:= Clear[Y]

In[ ]:= AY = Table[Y[i] → V⟦i⟧, {i, 5}]
```

Out[ ]= {Y[1] → s, Y[2] → t, Y[3] → s t, Y[4] → s^2, Y[5] → s^2 t}

Note that the Y[i] have only one bracket, thus these are independent variables rather than members of a list. However I don't want these to be independent so I give a set of relations on these Y[i]s.

```
In[ ]:= sys = {Y[3] - Y[1] × Y[2], Y[4] - Y[1] ^ 2, Y[5] - Y[2] × Y[4]};
```

To find a H - basis for this large exact system I use Groebner Bases.

```
In[ ]:= gBasis = GroebnerBasis [sys, Keys[AY], MonomialOrder → DegreeLexicographic ]
```

Out[ ]= {- $Y[3]^2$ + Y[2] × Y[5], Y[2] × Y[4] - Y[5], - Y[3] × Y[4] + Y[1] × Y[5],
  Y[1] × Y[3] - Y[5], Y[1] × Y[2] - Y[3], $Y[1]^2$ - Y[4], $Y[3]^2$ Y[4] - $Y[5]^2$}

Note

```
In[ ]:= Length[gBasis]
Out[ ]= 7
```

I now find my implicit equation by

```
In[ ]:= {time, eq} = Timing[FLTMD[gBasis, A, 4, Keys[AY], {x, y, z}, dTol]]
  » Initial Hilbert Function  {1, 4, 9, 16, 25}
  » Final Hilbert Function  {1, 4, 9, 16, 25}
```

Out[ ]= {2.68174, {1. - 1. x^2 - 1. y^2 + 1. z^2}}

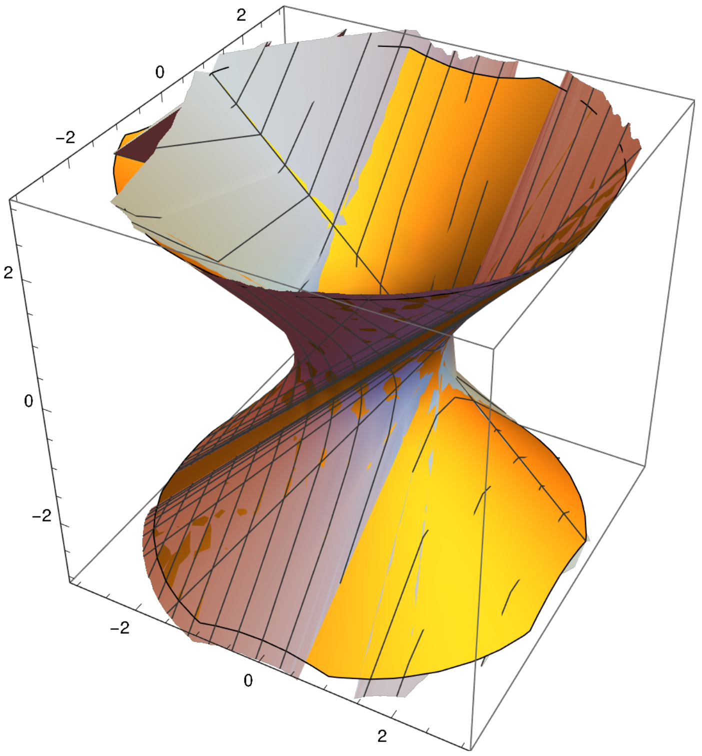
```
In[ ]:= qpEq = eq⟦1⟧
```

Out[ ]= 1. - 1. x^2 - 1. y^2 + 1. z^2

So I get my equation in under 3 seconds.

Finally I check by comparing plots . The second one has the mesh.

In[•]:=
```
Show[ContourPlot3D[qpEq == 0, {x, -3, 3}, {y, -3, 3}, {z, -3, 3}, Mesh → None],
 ParametricPlot3D[hyperboloid, {s, -20, 20}, {t, -3, 3}, PlotStyle → LightGray]]
```

Out[•]=


1.3.2 Direct Method

Although I was able compute the example above in around 3 seconds of computer time this is an eternity for Mathematica. With many more monomials this method is impractical. The following method may work better, but we must first consider polynomial parameterizations.

The function here is based on the Space Curve Book function p2aRawMD which in turn was based on the algorithm in Appendix 1.5 of the plane curve book. The reader who wants an explanation of how this works should look there. This routine expects exact or very accurate numerical coefficients. Here *F* is the polynomial parameterization, *d* is the maximal degree of a monomial in *F*, md is the maximum degree you are allowing an implicit equation, *T* are the variable in *F* and *X* are the variables in $\mathbb{R}^3$. Actually this works for parameterized surfaces in $\mathbb{R}^n$ for any *n* so *X* will be the variables there.

```
In[ ]:= par2affRS[F_, d_, md_, T_, X_] :=
         Module[{n, TB, ar, cr, SA, AK, mon, ncr, nak, NSA, medNSA, FA, SAA},
          n = Length[X];
          If[Length[F] ≠ n, Echo["Dimension mismatch F,X"]; Abort[]];
          TB = Expand[Table[mon /. Thread[X → F], {mon, mExpsMD[md, X]}]];
          cr = <|CoefficientRules[#, T]|> & /@ TB;
          ncr = Length[cr];
          AK = exps[2, md * d];
          nak = Length[AK];
          SAA = Reap[For[i = 1, i ≤ ncr, i++, For[j = 1, j ≤ nak, j++,
                If[KeyExistsQ[cr⟦i⟧, AK⟦j⟧], Sow[{i, j} → cr⟦i⟧[AK⟦j⟧]]]]]]⟦2, 1⟧;
          SA = Transpose[SparseArray[SAA]];
          NSA = NullSpace[SA];
          If[Length[NSA] == 0, Return["Fail, try higher md"],
           Echo[Length[NSA], "Number of equations"]];
          medNSA = Median[Abs[Flatten[NSA]]] + 1;
          N[NSA / medNSA].mExpsMD[md, X]
         ]
```

We demonstrate this on our ribbon example from the previous section.

```
In[ ]:= {time, ribboneqs} = Timing[par2affRS[ribbon, 3, 3, {s, t}, {x, y, z}]]
```

» Number of equations 1

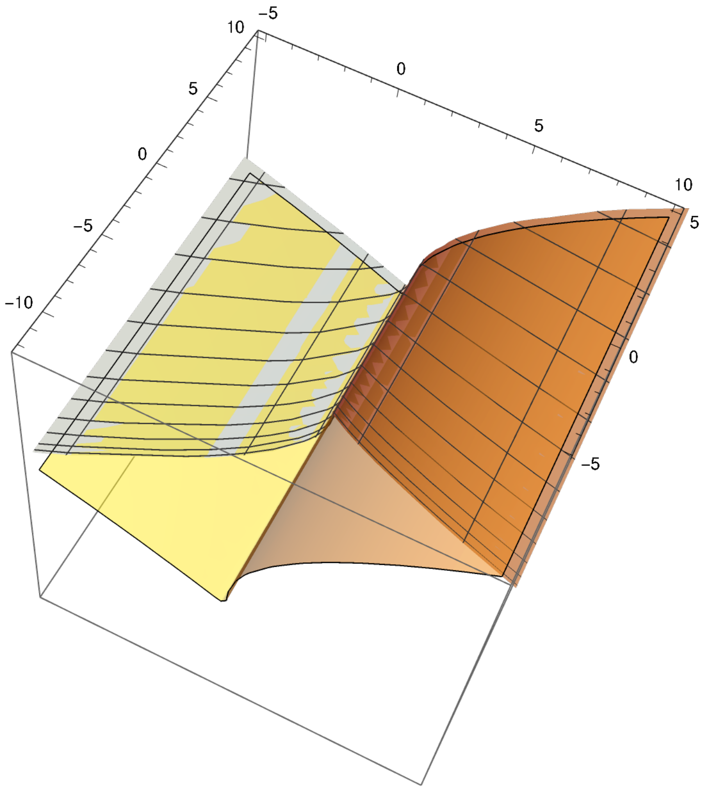
Out[ ]= $\{0.018665, \{5. - 1.\,x^2 + 9.\,z - 3.\,x\,z + 3.\,z^2 + 1.\,z^3\}\}$

```
In[ ]:= ribboneq = roundPolyMD[ribboneqs⟦1⟧, {x, y, z}, 1]
```

Out[ ]= $5 - x^2 + 9\,z - 3\,x\,z + 3\,z^2 + z^3$

```
In[ ]:= Show[ContourPlot3D[ribboneq == 0, {x, -5, 10},
          {y, -12, 10}, {z, -5, 5}, Mesh → None, ContourStyle → Opacity[.5]],
         ParametricPlot3D[ribbon, {s, .001, 6}, {t, -5, 5}, PlotStyle → LightGray]]
```

Out[ ]=



Again the parameterized image is given by the mesh. We note that there is a lower part of the plot of the implicit surface that is not covered by the parameterized surface which had a domain of $s > 0$. But even if we used parameter values of $s < 0$ we would not get more coverage. Thus the parameterization ribbon only parameterizes part of the implicit surface.

Here is a discouraging example. We try to implicitize a polynomial parameterized surface with coordinates of degree 3. We start with a random A:

```
In[ ]:= A = Append[RandomInteger[{-4, 4}, {3, 10}], {0, 0, 0, 0, 0, 0, 0, 0, 0, 1}]

Out[ ]= {{2, 3, -3, 1, -3, 3, -1, -2, 1, 0}, {1, -2, -4, -4, -3, -2, 2, 3, -4, 1},
         {1, -4, -4, -2, 0, 1, -1, -4, 1, -3}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 1}}

In[ ]:= Dimensions[A]

Out[ ]= {4, 10}

In[ ]:= Y = Drop[mExpsMD[3, {s, t}], 1]
```

Out[ ]= $\{s, t, s^2, s\,t, t^2, s^3, s^2\,t, s\,t^2, t^3\}$

```
In[ ]:= TransformationFunction [A][Y]
```

Out[ ]= $\{2 s - 3 s^2 + 3 s^3 + 3 t + s t - s^2 t - 3 t^2 - 2 s t^2 + t^3,$
$1 + s - 4 s^2 - 2 s^3 - 2 t - 4 s t + 2 s^2 t - 3 t^2 + 3 s t^2 - 4 t^3,$
$-3 + s - 4 s^2 + s^3 - 4 t - 2 s t - s^2 t - 4 s t^2 + t^3\}$

```
Eqns = par2affNS[F, 3, 3, {s, t}, {x, y, z}];
Out[ ]= Fail, try higher md

Eqns = par2affNS[F, 3, 5, {s, t}, {x, y, z}];
Out[ ]= Fail, try higher md

Eqns = par2affNS[F, 3, 8, {s, t}, {x, y, z}];
Out[ ]= Fail, try higher md

In[ ]:= Eqns = par2affNS[F, 3, 9, {s, t}, {x, y, z}];
  » Number of equations 1

In[ ]:= Length[Eqns[[1]]]
Out[ ]= 148
```

Our smallest implicit equation is of degree 9 with 148 terms! In fact this will almost always be the case but it shows that there is an implicit equation. Of course this gets much worse for higher degrees.

There is a trick we can use to handle a rational parameterization : see my Mathematica Journal article [*Degree vs Dimension of Rational Parametric Curves]*.

Take the original parameterization and strip all constants, also put the common denominator as a 4th component. Check to make sure components are independent in space of 2 variable polynomials, if not see my Mathematica Journal article for a reduction. Use pol2affNS to find an implicit polynomial system with variables {x,y,z,w}. If this is more than 2 or 3 equations reduce by hBasisMD. Now create a matrix by taking the first 4 rows of the 5×5 identity matrix. In the 5th column replace the constants that you stripped. Then apply FLTMD to the implicit polynomial system using this 4×5 matrix. You should get your implicit system of the rational parametric surface. We illustrate using the above

In[ ]:= hyperboloid = $\left\{\frac{t - s^2 t}{1 - s^2}, \frac{1 + s^2 - 2 s t}{1 - s^2}, \frac{2 s - t - s^2 t}{1 - s^2}\right\};$

Strip off the constants from each term in the numerator and denominator.

```
In[ ]:= strippedh = {t - s^2 t, s^2 - 2 s t, 2 s - t - s^2 t, -s^2};
```

Note we can recover **hyperboloid** from strippedh by an FLT: Let

```
In[ ]:= AH = {{1, 0, 0, 0, 0}, {0, 1, 0, 0, 1}, {0, 0, 1, 0, 0}, {0, 0, 0, 1, 1}};
        AH // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

```
In[ ]:= TransformationFunction [AH][strippedh]
```

Out[ ]=
$$\left\{\frac{t - s^2 t}{1 - s^2}, \frac{1 + s^2 - 2 s t}{1 - s^2}, \frac{2 s - t - s^2 t}{1 - s^2}\right\}$$

```
In[ ]:= raweq = pol2affNS[strippedh , 3, 3, {s, t}, {x, y, z, w}]
```

» Number of equations 8

Out[ ]=
$$\{0. + 2.\, w - 1.\, w^3 - 1.\, x^2 + 2.\, w x^2 - 2.\, y + 1.\, w y^2 - 2.\, w x z + 1.\, z^2,$$
$$0. + 1.\, w x + 1.\, x^3 + 1.\, x y + 1.\, w x y + 1.\, x y^2 - 1.\, w z - 1.\, w^2 z - 1.\, y z - 1.\, w y z - 1.\, x z^2,$$
$$0. + 2.\, w - 1.\, x^2 - 1.\, w x^2 - 2.\, y + 2.\, w y - 2.\, y^2 + 2.\, w x z + 1.\, z^2 - 1.\, w z^2,\ 0. - 1.\, w x - 1.\, x^3 -$$
$$1.\, x y - 1.\, w x y - 1.\, x y^2 - 1.\, w z + 1.\, x^2 z + 3.\, y z + 1.\, w y z + 1.\, y^2 z + 1.\, x z^2 - 1.\, z^3,$$
$$0. + 1.\, w - 2.\, x^2 - 1.\, w x^2 - 3.\, y - 2.\, w y - 1.\, w^2 y - 2.\, y^2 - 1.\, w y^2 - 1.\, x z + 1.\, w x z + 1.\, z^2,$$
$$0. - 1.\, w + 2.\, x^2 + 1.\, w x^2 + 3.\, y + 1.\, x^2 y + 4.\, y^2 + 1.\, w y^2 + 1.\, y^3 + 1.\, x z - 1.\, w x z - 1.\, z^2 - 1.\, y z^2,$$
$$0. - 2.\, w x - 1.\, w^2 x + 1.\, x^3 + 2.\, x y + 1.\, x y^2 - 1.\, x z^2,\ 0. - 2.\, w - 1.\, w^2 + 1.\, x^2 + 2.\, y + 1.\, y^2 - 1.\, z^2\}$$

We have lots of equations so we can reduce the system

```
In[ ]:= hbeq = hBasisMD[raweq , 3, {x, y, z, w}, 1.*^-10]
```

» Initial Hilbert Function {1, 4, 9, 13}

» Final Hilbert Function {1, 4, 9, 13}

Out[ ]=
$$\{2.\, w + 1.\, w^2 - 1.\, x^2 - 2.\, y - 1.\, y^2 + 1.\, z^2,$$
$$1.\, w - 2.\, x^2 - 1.\, w x^2 - 3.\, y - 1.\, x^2 y - 4.\, y^2 - 1.\, w y^2 - 1.\, y^3 - 1.\, x z + 1.\, w x z + 1.\, z^2 + 1.\, y z^2,$$
$$1.\, w x + 1.\, x^3 + 1.\, x y + 1.\, w x y + 1.\, x y^2 + 1.\, w z - 1.\, x^2 z - 3.\, y z - 1.\, w y z - 1.\, y^2 z - 1.\, x z^2 + 1.\, z^3,$$
$$-2.\, w + 1.\, x^2 + 1.\, w x^2 + 2.\, y - 2.\, w y + 2.\, y^2 - 2.\, w x z - 1.\, z^2 + 1.\, w z^2\}$$

Now produce the transformation matrix AH adding back the 1 in the second and 4 component.

```
In[ ]:= eq = FLTMD[raweq , AH, 3, {x, y, z, w}, {x, y, z}, dTol][[1]]
```

» Initial Hilbert Function {1, 4, 9, 16}

» Final Hilbert Function {1, 4, 9, 16}

Out[ ]= $1. - 1.\, x^2 - 1.\, y^2 + 1.\, z^2$

which is exactly what we got before!

The point of this section is not really about how to implicitize an actual example but just to emphasize the theorem that theoretically *every rational parametric surface is contained in a naive implicit surface.* Thus we can apply the Jordan-Brouwer Theorem of Section 1.1 about smooth points. But the plot at the end of Section 1.2 shows there can be many non-smooth points!

1.4 The Torus Story

This example has served as motivation for this book. Here I have a simpler, but more ad-hoc, method for implicitizing rational parametric functions. The theme of studying surfaces by curves on the surface will be a major technique in this book and has been a major tool also in classical algebraic geometry. Some of the surfaces mentioned in Section 1.1 are constructed here.

1.4.1 Preliminaries

Before getting into this I remind the reader that the first method in the previous section 1.3 is based on the method in section 3.1.4 of my *Space Curves Book* for finding implicit equations of rationally parameterized space curves. For degrees $d = 2,3,4$ and 5 one writes the curve in the form

$\texttt{TransformationFunction[A][\{t}^{d}\texttt{, t}^{d-1}\texttt{, \ldots, t\}]}$

or the equivalent

$\texttt{fltMD[\{t}^{d}\texttt{, t}^{d-1}\texttt{, \ldots, t\}, A]}$

for an appropriate $(d+1)\times(n+1)$ transformation matrix A. Here $n = 3$. Essentially we are viewing the parametric curve as an image of the rational normal curve of degree d. Then the implicit equation is given by

```
FLTMD[tBasisd, A, m, {x1, x2, … xd}, {x, y, z}, tol]
```

for appropriate m. Often $m = d$ but a possibly smaller m might work or a larger m may be needed. Naive space curves have 2 equations, rather than the one for surfaces, but often the correct system of equations for a rational parametric curve will not be naive and require more than 2 equations but for our use we may find 2 equations that serve our purposes.

One other important preliminary idea from Space Curves is that we can approximate ideals of algebraic spaces using Sylvester matrices. The rows of a Sylvester, or other, matrix can be viewed as the basis of a subspace of an appropriate n-space $\mathbb{R}^n$ where often n is large. To take the union of two algebraic spaces a row equivalent matrix to the Sylvester matrix of a union is the intersection of the Sylvester matrices of the parts. So one of the main tools I will use in this book is the following simple algorithm for the intersection of two vector subspaces of $\mathbb{R}^n$.

Note that in the Space Curve Book we adopt some of the language of Macaulay.
Matrices A, B are called (Macaulay) duals if

1. $A\,B$ is defined and $A\,B = 0$
2. If $v\,B = 0$ then v is in the row space of A
3. If $v\,B = 0$ then v is in the row space of A

That is, A, B are maximal satisfying $A\,B = 0$. It is sufficient that the columns of B form the nullspace of A or the rows of A form the column space of B. In my software if either **`A = localDualMatrix[B,tol]`** or **`B = dualMatrix[A, tol]`** then A,B are duals, in particular B is the dual of A and A is the local dual of B.

Here let V, W be matrices with the same number of columns whose row spaces are the two vector spaces Let dV, dW be the duals of V, W and dd the column join of dV, dW.

If v is in the intersection of the vector spaces then v.dV = v.dW = 0 so v.dd=0 and v is in the row space of the local dual of dd.

Conversely, if v is in the row space of the local dual of dd then v. dd = 0 meaning v.c = 0 for any column of dd. In particular v.dV = 0, v.dW =0 so v is in the row space of V and the row space of W, hence in the intersection.

Thus our algorithm is

```
In[ ]:= vectorSpaceIntersection [V_, W_, tol_] := Module[{dV, dW, dd},
          dV = dualMatrix [V, tol];
          dW = dualMatrix [W, tol];
          dd = Join[dV, dW, 2];
          localDualMatrix [dd, tol]]
```

This can be extended to 3 or more subspaces if useful , see GlobalFunctionsS.nb

To use this to find the union of two algebraic sets we take Sylvester matrices of the same appropriate order for the two sets. We then intersect the underlying row spaces to get a row matrix which we multiply by an mExpsMD list of monomials to convert back to equations. If necessary we find a smaller hBasis of this list. Examples are below.

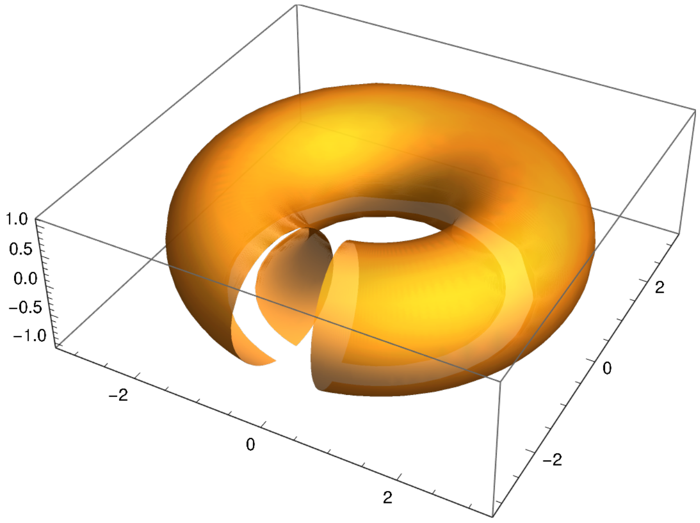
1.4 .2 The Torus

Here is our rationally parametrized surface .

In[ ]:=
$$T = \left\{\frac{4\,s\,(1+t+t^2)}{(1+s^2)\times(1+t^2)},\ -\frac{2\times(-1+s^2-t+s^2\,t-t^2+s^2\,t^2)}{(1+s^2)\times(1+t^2)},\ \frac{(1-t^2)\times(1+s\,\hat{}\,2)}{(1+t^2)\times(1+s\,\hat{}\,2)}\right\};$$

Plotting, using a finite range instead of the {-∞,∞} theoretical range, we get

In[ ]:=
```
PT := ParametricPlot3D [T, {t, -10, 10}, {s, -10, 10},
   PlotRange → All, Mesh → None, MaxRecursion → 4, PlotStyle → Opacity[.8]]
PT
```

Out[ ]=


This seems to be most of a torus.

Step 1

We can find curves on this surface by restricting to one variable by making the other a constant, in this case we will set *t* to 0 and then, for later consistency, set *s* to *t*.

In[ ]:=
```
ft0 = T /. {t → 0, s → t}
```

Out[ ]= $\left\{\frac{4\,t}{1+t^2},\ -\frac{2\times(-1+t^2)}{1+t^2},\ 1\right\}$

Since $1 = \frac{1+t\wedge 2}{1+t\wedge 2}$ we can use the transformation matrix

In[ ]:=
```
At0 = {{0, 4, 0}, {2, 0, -2}, {1, 0, 1}, {1, 0, 1}}
```

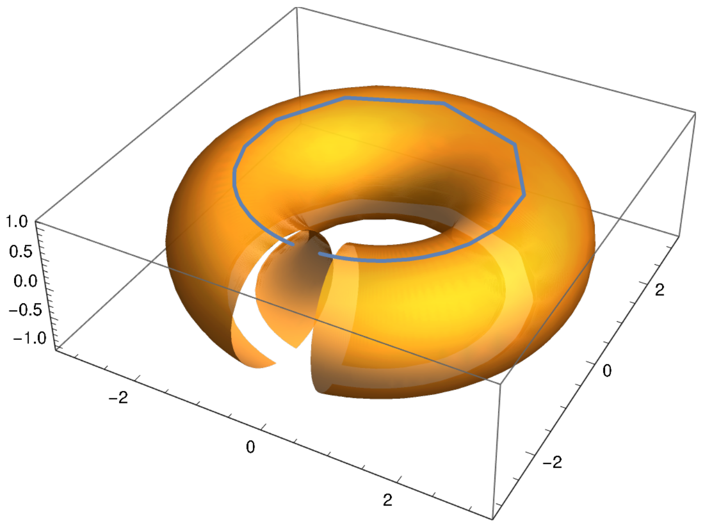
Out[ ]=
```
{{0, 4, 0}, {2, 0, -2}, {1, 0, 1}, {1, 0, 1}}
```

Checking

In[ ]:=
```
fltMD[{t^2, t}, At0]
```

Out[ ]= $\left\{\frac{4\,t}{1+t^2},\ \frac{-2+2\,t^2}{1+t^2},\ 1\right\}$

```
In[ ]:= Show[PT, ParametricPlot3D[ft0, {t, -20, 20}]]
```

Out[ ]=


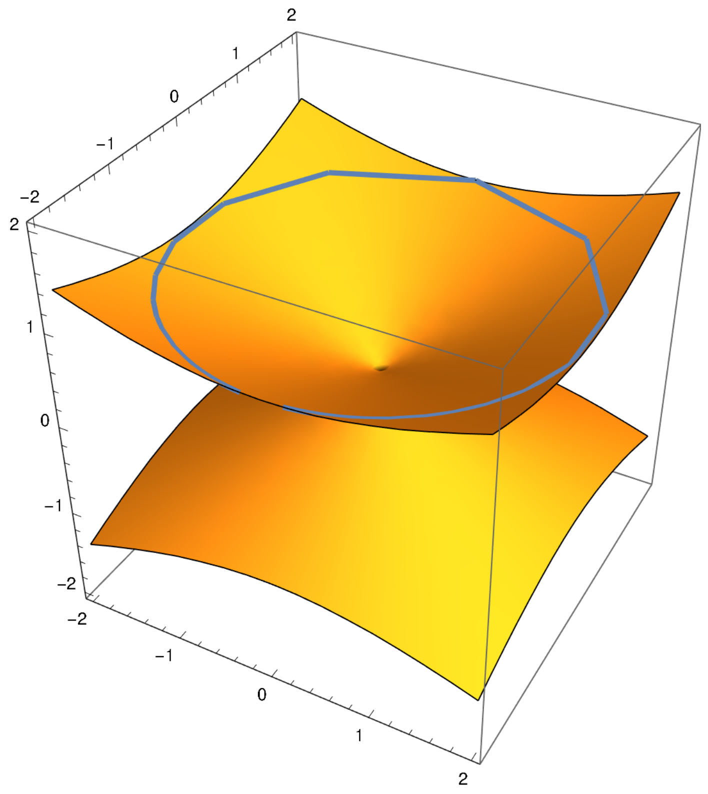
We find a basis by

```
In[ ]:= ideal1 = FLTMD[tBasis2, At0, 2, {x2, x1}, {x, y, z}, dTol]
```

```
Out[ ]= FLTMD[tBasis2, {{0, 4, 0}, {2, 0, -2}, {1, 0, 1}, {1, 0, 1}}, 2, {x2, x1}, {x, y, z}, dTol]
```

Using the second, more complicated basis element we see this curve generates the surface

In[∘]:=
```
Show[ContourPlot3D[ideal1[[-1]] == 0, {x, -2, 2}, {y, -2, 2}, {z, -2, 2}, Mesh → None],
 ParametricPlot3D[ft0, {t, -20, 20}, MaxRecursion → 4]]
```

Out[∘]=


Step 2

We then consider a curve on the torus by making *s* a constant, we already have variable *t*. Again, we are working ad-hoc so perhaps a bit of trial and error is necessary.

In[∘]:=
```
fs2 = T /. {s → 2}
```

Out[∘]= $\left\{\frac{8\times(1+t+t^2)}{5\times(1+t^2)}, -\frac{2\times(3+3t+3t^2)}{5\times(1+t^2)}, \frac{1-t^2}{1+t^2}\right\}$

A transformation matrix is

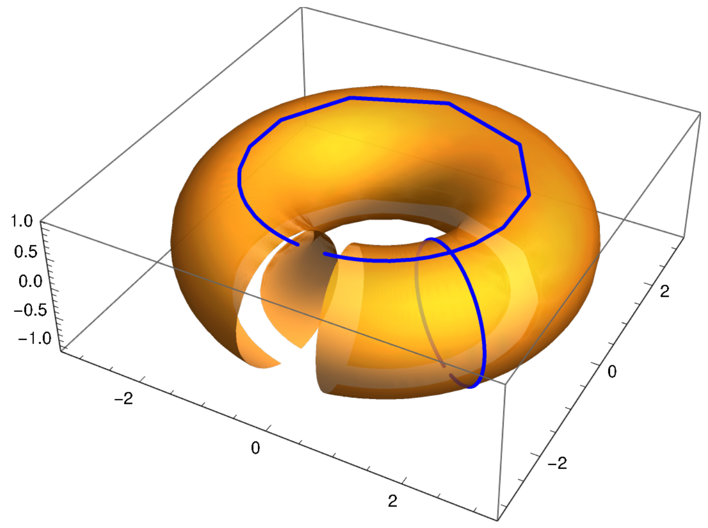
In[∘]:=
```
As2 = {{8, 8, 8}, {-6, -6, -6}, {-5, 0, 5}, {5, 0, 5}};
```

Checking

In[∘]:=
```
fltMD[{t^2, t}, As2]
```

Out[∘]= $\left\{\frac{8+8t+8t^2}{5+5t^2}, \frac{-6-6t-6t^2}{5+5t^2}, \frac{5-5t^2}{5+5t^2}\right\}$

```
In[ ]:= Show[PT, ParametricPlot3D [{fs2, ft0}, {t, -20, 20}, PlotStyle → Blue, MaxRecursion → 4]]
```

Out[ ]=


These curves are on the torus as the plot shows, but we want to see what sort of surface is determined by these curves alone. We now use ideas of 1.4.1.

```
In[ ]:= ideal2 = FLTMD[tBasis2, As2, 2, {x2, x1}, {x, y, z}, dTol]

Out[ ]= FLTMD[tBasis2, {{8, 8, 8}, {-6, -6, -6}, {-5, 0, 5}, {5, 0, 5}}, 2, {x2, x1}, {x, y, z}, dTol]
```

We use $m = 4$ because we think the torus will have an equation of degree 4.

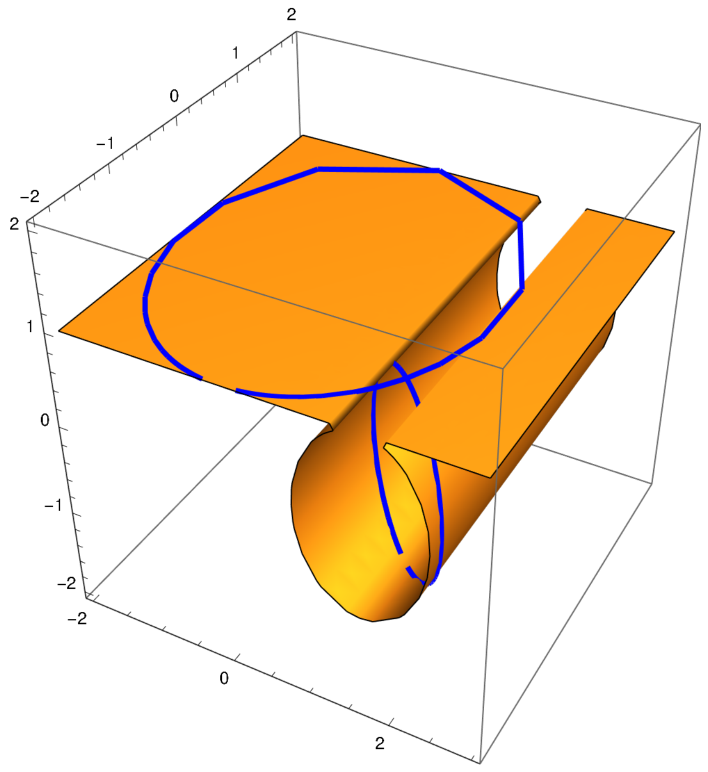
```
In[ ]:= syl1 = sylvesterMD[ideal1, 4, {x, y, z}];
        syl2 = sylvesterMD[ideal2, 4, {x, y, z}];

In[ ]:= intersec2 = vectorSpaceIntersection [syl1, syl2, 1.*^-10];
        Length[intersec2]

Out[ ]= 18
```

This says we will get a basis of 18 polynomials, which is too cumbersome. So we do

```
In[ ]:= basis2 = hBasisMD[intersec2.mExpsMD[4, {x, y, z}], 4, {x, y, z}, 1.*^-10]

Out[ ]= hBasisMD[intersec2.mExpsMD[4, {x, y, z}], 4, {x, y, z}, 1. × 10^-10]
```

to get a basis of 4 polynomials . Plotting the last one w2 have

In[ ]:=
```
Show[ContourPlot3D [basis2[[-1]] == 0, {x, -2, 3}, {y, -2, 2}, {z, -2, 2}, Mesh → None],
  ParametricPlot3D [{fs2, ft0}, {t, -20, 20}, PlotStyle → Blue, MaxRecursion → 4]]
```

Out[ ]=


This is just the surface ts2 of section 1.1. We saw that this is the union of a plane with an infinite cylinder and the intersection line was regular but not smooth so **ContourPlot3D** can not plot this correctly, but the upper circle is in ts2.

Step 3

We now add another vertical circle .

In[ ]:=
```
ftp5 = Expand[T /. {t → .5, s → t}]
```

Out[ ]=
$$\left\{\frac{5.6\,t}{1+t^2},\ \frac{2.8}{1+t^2}-\frac{2.8\,t^2}{1+t^2},\ 0.6\right\}$$

Putting the last component over the common denominator gives transformation matrix

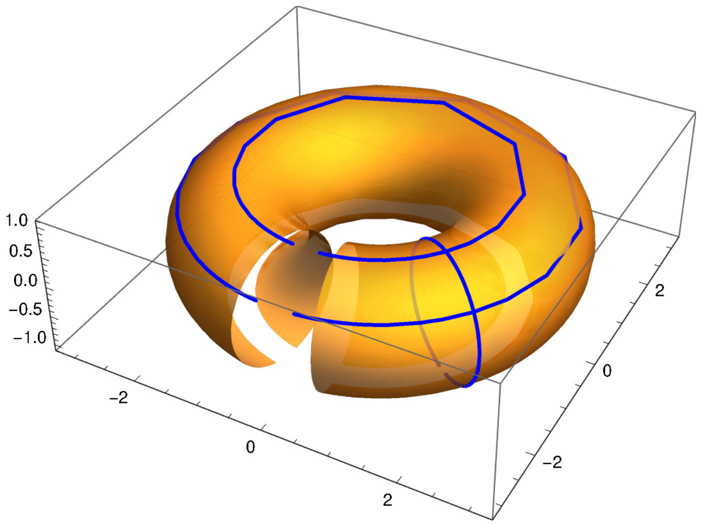
In[ ]:=
```
Atp5 = {{0, 5.6, 0}, {-2.8, 0, 2.8}, {.6, 0, .6}, {1, 0, 1}};
```

In[ ]:=
```
fltMD[{t ^ 2, t}, Atp5]
```

Out[ ]=
$$\left\{\frac{5.6\,t}{1.+1.\,t^2},\ \frac{2.8-2.8\,t^2}{1.+1.\,t^2},\ \frac{0.6+0.6\,t^2}{1.+1.\,t^2}\right\}$$

```
In[ ]:= Show[PT,
         ParametricPlot3D [{ftp5, fs2, ft0}, {t, -20, 20}, PlotStyle → Blue, MaxRecursion → 4]]
```

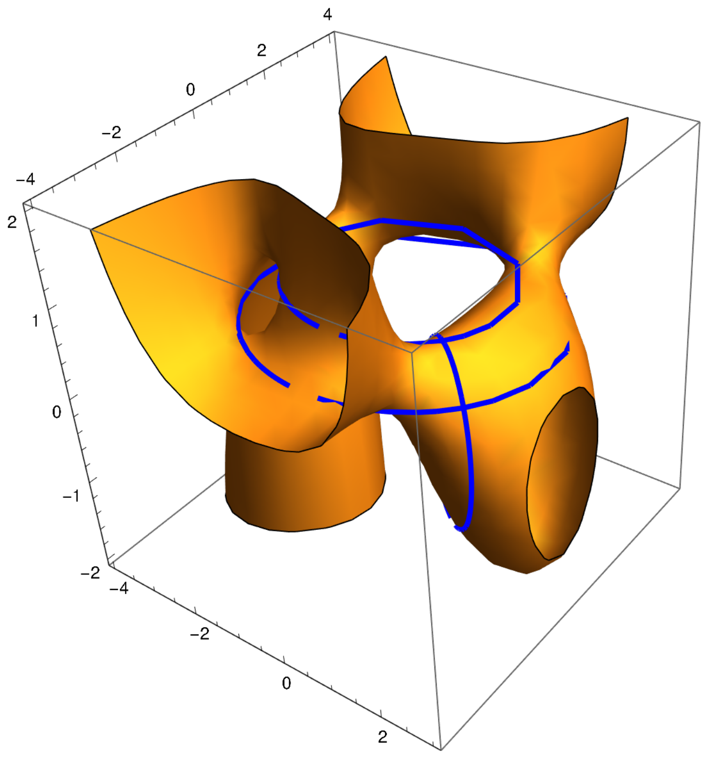
Out[ ]=



```
In[ ]:= ideal3 = FLTMD[tBasis2, Atp5, 2, {x2, x1}, {x, y, z}, dTol]

   » Initial Hilbert Function  {1, 3, 5}

   » Final Hilbert Function  {1, 3, 5}

Out[ ]= {1. - 1.66667 z, -0.0459184 x^2 - 0.0459184 y^2 + 1. z^2}

In[ ]:= syl3 = sylvesterMD[ideal3, 4, {x, y, z}];
        syl3b = sylvesterMD[basis2, 4, {x, y, z}];
        intersect3 = vectorSpaceIntersection [syl3, syl3b, 1.*^-10];
        Length[intersect3]

Out[ ]= 12

In[ ]:= basis3 = hBasisMD[intersect3.mExpsMD[4, {x, y, z}], 4, {x, y, z}, 1.*^-10]

   » Initial Hilbert Function  {1, 3, 6, 7, 6}

   » Final Hilbert Function  {1, 3, 6, 6}

Out[ ]= {-10.2 x + 0.75 x^3 - 13.6 y + 1. x^2 y + 0.75 x y^2 + 1. y^3 + 7.2 x z + 9.6 y z,
         0.45 x + 0.6 y - 1.2 x z - 1.6 y z + 0.75 x z^2 + 1. y z^2,
         -6. + 8.125 x - 0.625 x^3 - 0.625 x y^2 + 3. z - 5. x z + 1. x^2 z + 1. y^2 z - 2. z^2 - 0.625 x z^2 + 1. z^3,
         10.752 - 6.4 x - 11.464 x^2 + 0.64 x^3 + 1. x^4 + 1.536 y^2 + 0.64 x y^2 +
          1. x^2 y^2 + 2.88 x^2 z - 5.12 y^2 z + 3.584 z^2 + 3.84 x z^2 + 1. x^2 z^2}
```

In[ ]:=
```
Show[ContourPlot3D [basis3[[-1]] == 0, {x, -4, 3}, {y, -4, 4}, {z, -2, 2}, Mesh → None],
 ParametricPlot3D [{ftp5, fs2, ft0}, {t, -20, 20}, PlotStyle → Blue, MaxRecursion → 4]]
```

Out[ ]=



Step 4.

We find another vertical circle

In[ ]:=
```
fs4 = T /. {s → 4}
```

Out[ ]=
$$\left\{\frac{16\times(1+t+t^2)}{17\times(1+t^2)},\ -\frac{2\times(15+15\,t+15\,t^2)}{17\times(1+t^2)},\ \frac{1-t^2}{1+t^2}\right\}$$

In[ ]:=
```
As4 = {{16, 16, 16}, {-30, -30, -30}, {-17, 0, 17}, {17, 0, 17}};
```

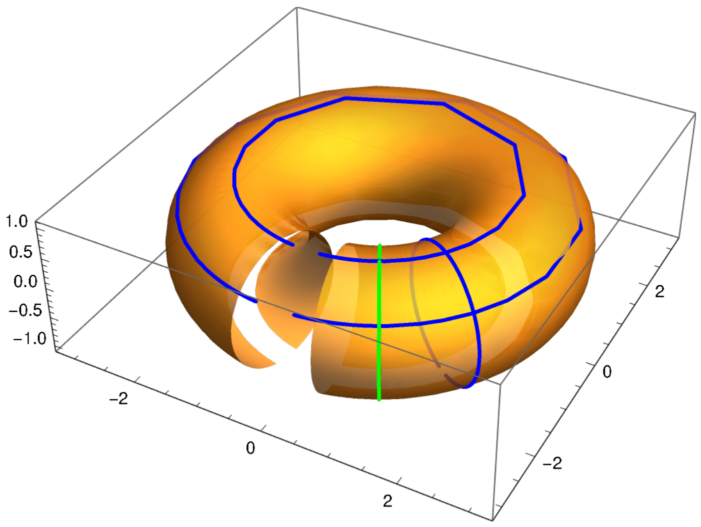
Checking :

In[ ]:=
```
fltMD[{t ^ 2, t}, As4]
```

Out[ ]=
$$\left\{\frac{16+16\,t+16\,t^2}{17+17\,t^2},\ \frac{-30-30\,t-30\,t^2}{17+17\,t^2},\ \frac{17-17\,t^2}{17+17\,t^2}\right\}$$

```
In[ ]:= Show[PT, ParametricPlot3D [{ftp5, fs2, ft0}, {t, -20, 20}, PlotStyle → Blue],
          ParametricPlot3D [fs4, {t, -20, 20}, PlotStyle → Green]]
```

Out[ ]=



Continuing as above

```
In[ ]:= ideal4 = FLTMD[tBasis2, As4, 2, {x2, x1}, {x, y, z}, dTol]

   » Initial Hilbert Function  {1, 3, 5}

   » Final Hilbert Function  {1, 3, 5}

Out[ ]= {1.875 x + 1. y, 1. - 2.83333 x + 1.50521 x^2 + 0.333333 z^2}
```

```
In[ ]:= syl4 = sylvesterMD[ideal4, 4, {x, y, z}];
        syl4b = sylvesterMD[basis3, 4, {x, y, z}];
        intersect4 = vectorSpaceIntersection [syl4, syl4b, 1.*^-10];
        Length[intersect4]

Out[ ]= 7
```

```
In[ ]:= basis4 = hBasisMD[intersect4.mExpsMD[4, {x, y, z}], 4, {x, y, z}, 1.*^-10]

   » Initial Hilbert Function  {1, 3, 6, 9, 9}

   » Final Hilbert Function  {1, 3, 6, 9, 9}
```

Out[ ]= $\{-6. + 4.33333\, x - 0.333333\, x^3 - 5.05556\, y + 0.388889\, x^2 y - 0.333333\, x y^2 + 0.388889\, y^3 + 3.\, z - 2.66667\, x z + 1.\, x^2 z + 3.11111\, y z + 1.\, y^2 z - 2.\, z^2 - 0.333333\, x z^2 + 0.388889\, y z^2 + 1.\, z^3,$

$-6.4 + 2.03175\, x - 2.92619\, x^2 - 0.203175\, x^3 + 0.154762\, x^4 - 2.37037\, y - 13.\, x y + 0.237037\, x^2 y + 1.\, x^3 y - 0.914286\, y^2 - 0.203175\, x y^2 + 0.154762\, x^2 y^2 + 0.237037\, y^3 + 1.\, x y^3 + 4.28571\, x^2 z + 8.\, x y z + 3.04762\, y^2 z - 2.13333\, z^2 - 1.21905\, x z^2 + 0.154762\, x^2 z^2 + 1.42222\, y z^2 + 1.\, x y z^2,$

$-19.125\, x^2 + 1.40625\, x^4 - 35.7\, x y + 2.625\, x^3 y - 13.6\, y^2 + 2.40625\, x^2 y^2 + 2.625\, x y^3 + 1.\, y^4 + 13.5\, x^2 z + 25.2\, x y z + 9.6\, y^2 z,$

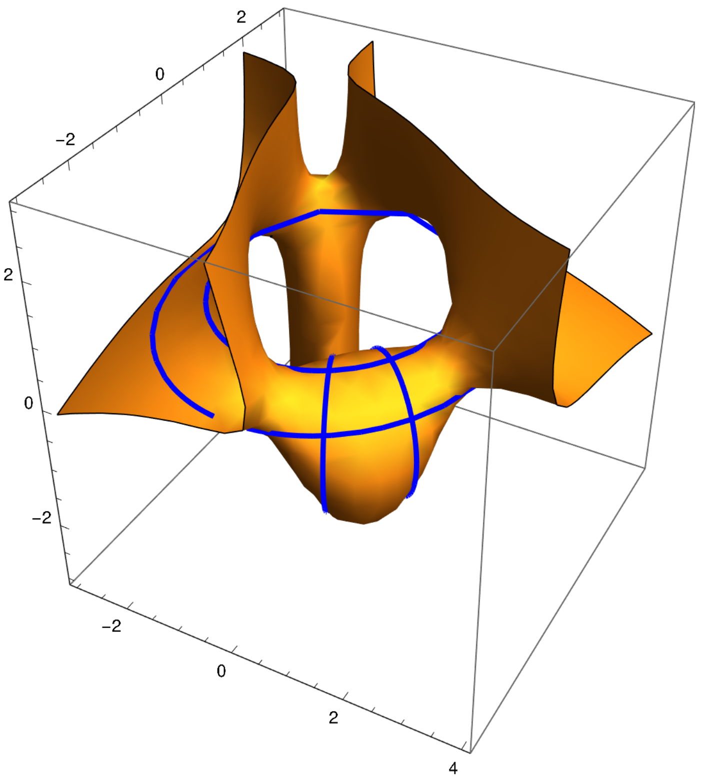
$16.8 - 5.33333\, x + 8.525\, x^2 + 0.533333\, x^3 - 0.40625\, x^4 + 6.22222\, y + 35.7\, x y - 0.622222\, x^2 y - 2.625\, x^3 y + 3.\, y^2 + 0.533333\, x y^2 - 0.40625\, x^2 y^2 - 0.622222\, y^3 - 2.625\, x y^3 - 13.5\, x^2 z - 25.2\, x y z - 9.6\, y^2 z + 5.6\, z^2 + 3.2\, x z^2 + 1.\, x^2 z^2 - 3.73333\, y z^2 + 1.\, y^2 z^2\}$

```
In[ ]:= Length[basis4]
Out[ ]= 4
```

As before the last equation gives an example of a surface of degree 4 containing these 4 curves .

```
In[ ]:= Show[ContourPlot3D [basis4⟦-1⟧ == 0, {x, -3, 4}, {y, -3, 3}, {z, -3, 3}, Mesh → None],
          ParametricPlot3D [{fs4, ftp5, fs2, ft0}, {t, -20, 20}, PlotStyle → Blue]]
```

Out[ ]=



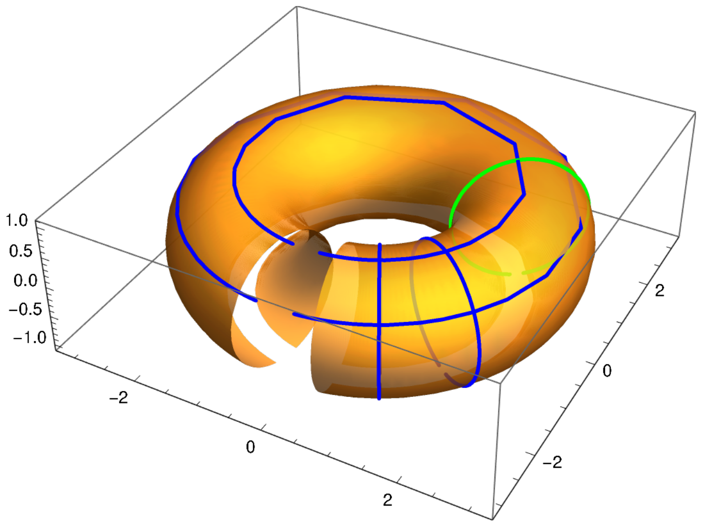
Step 5.

Now we add another vertical circle .

```
In[ ]:= fsp5 = T /. {s → .5}
```

Out[ ]= $\left\{\frac{1.6 \times (1 + t + t^2)}{1 + t^2}, -\frac{1.6 \times (-0.75 - 0.75\, t - 0.75\, t^2)}{1 + t^2}, \frac{1 - t^2}{1 + t^2}\right\}$

```
In[ ]:= Show[PT, ParametricPlot3D [{fs4, ftp5, fs2, ft0}, {t, -20, 20}, PlotStyle → Blue],
          ParametricPlot3D [fsp5, {t, -20, 20}, PlotStyle → Green]]
```

Out[ ]=


```
In[ ]:= Asp5 = {{1.6, 1.6, 1.6}, {1.2, 1.2, 1.2}, {-1, 0, 1}, {1, 0, 1}};
```

Checking

```
In[ ]:= fltMD[{t^2, t}, Asp5]
```

Out[ ]= $\left\{\frac{1.6+1.6\,t+1.6\,t^2}{1.+1.\,t^2}, \frac{1.2+1.2\,t+1.2\,t^2}{1.+1.\,t^2}, \frac{1.-1.\,t^2}{1.+1.\,t^2}\right\}$

```
In[ ]:= ideal5 = FLTMD[tBasis2, Asp5, 2, {x2, x1}, {x, y, z}, dTol]
   » Initial Hilbert Function  {1, 3, 5}
   » Final Hilbert Function  {1, 3, 5}
```

Out[ ]= $\{-0.75\,x+1.\,y,\ 1.-1.66667\,x+0.520833\,x^2+0.333333\,z^2\}$

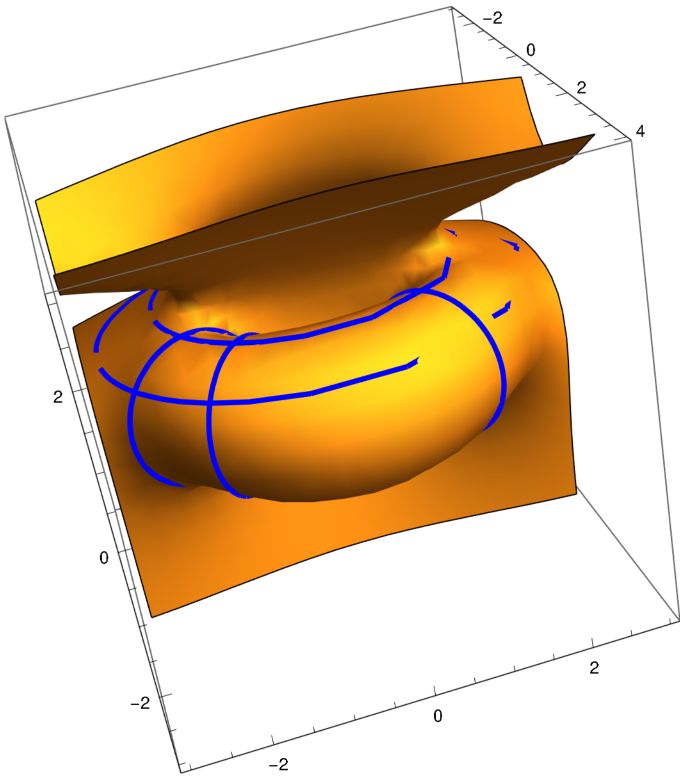
```
In[ ]:= syl5 = sylvesterMD[ideal5, 4, {x, y, z}];
        syl5b = sylvesterMD[basis4, 4, {x, y, z}];
        intersect5 = vectorSpaceIntersection [syl5, syl5b, 1.*^-10];
        Length[intersect5]

Out[ ]= 3

In[ ]:= basis5 = hBasisMD[intersect5.mExpsMD[4, {x, y, z}], 4, {x, y, z}, 1.*^-10]
```

```
» Initial Hilbert Function  {1, 3, 6, 10, 12}

» Final Hilbert Function  {1, 3, 6, 10, 12}
```

Out[ ]= $\{40.5\,x - 47.5313\,x^2 + 3.65625\,x^4 - 13.\,y^2 + 4.65625\,x^2 y^2 + 1.\,y^4 - 20.25\,x z + 29.25\,x^2 z - 6.75\,x^3 z + 8.\,y^2 z - 6.75\,x y^2 z + 13.5\,x z^2 + 3.65625\,x^2 z^2 + 1.\,y^2 z^2 - 6.75\,x z^3,$
$-11.25\,x + 15.2344\,x^2 - 1.17188\,x^4 - 6.\,y + 8.125\,x y - 0.625\,x^3 y - 1.17188\,x^2 y^2 - 0.625\,x y^3 + 5.625\,x z - 9.375\,x^2 z + 1.875\,x^3 z + 3.\,y z - 5.\,x y z + 1.\,x^2 y z + 1.875\,x y^2 z + 1.\,y^3 z - 3.75\,x z^2 - 1.17188\,x^2 z^2 - 2.\,y z^2 - 0.625\,x y z^2 + 1.875\,x z^3 + 1.\,y z^3,$
$9. - 81.\,x + 85.0625\,x^2 - 6.3125\,x^4 + 16.\,y^2 - 7.3125\,x^2 y^2 - 1.\,y^4 + 40.5\,x z - 58.5\,x^2 z + 13.5\,x^3 z - 16.\,y^2 z + 13.5\,x y^2 z + 6.\,z^2 - 27.\,x z^2 - 5.3125\,x^2 z^2 + 13.5\,x z^3 + 1.\,z^4\}$

```
In[ ]:= Show[ContourPlot3D [basis5[[-1]] == 0, {x, -3, 4}, {y, -3, 3}, {z, -3, 3}, Mesh → None],
         ParametricPlot3D [{fsp5, fs4, ftp5, fs2, ft0}, {t, -20, 20}, PlotStyle → Blue]]
```

Out[ ]=



We will name this surface ts5 for later use .

Step 6.

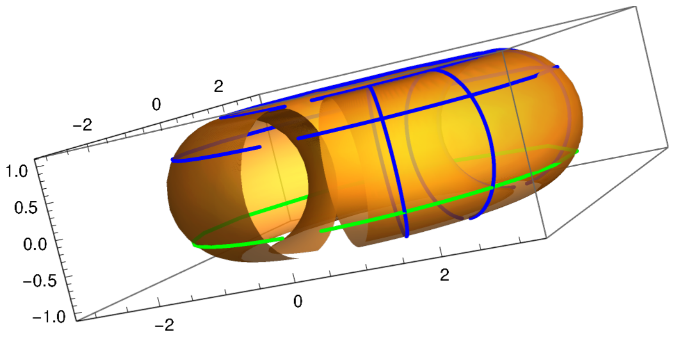
One more horizontal circle .

```
In[ ]:= ft2 = Expand[N[T /. {t → 2, s → t}]]
```

Out[ ]= $\left\{\frac{5.6\,t}{1.+t^2},\ \frac{2.8}{1.+t^2} - \frac{2.8\,t^2}{1.+t^2},\ -0.6\right\}$

```
In[ ]:= Show[PT, ParametricPlot3D [{fsp5, fs4, ftp5, fs2, ft0}, {t, -20, 20}, PlotStyle → Blue],
          ParametricPlot3D [ft2, {t, -20, 20}, PlotStyle → Green]]
```

Out[ ]=



```
At2 = {{0, 5.6, 0}, {-2.8, 0, 2.8}, {-.6, 0, -.6}, {1, 0, 1}};
fltMD[{t^2, t}, At2]
```

Out[ ]= $\left\{\frac{5.6\,t}{1.+1.\,t^2}, \frac{2.8-2.8\,t^2}{1.+1.\,t^2}, \frac{-0.6-0.6\,t^2}{1.+1.\,t^2}\right\}$

```
In[ ]:= ideal6 = FLTMD[tBasis2, At2, 2, {x2, x1}, {x, y, z}, dTol]

   » Initial Hilbert Function  {1, 3, 5}

   » Final Hilbert Function  {1, 3, 5}
```

Out[ ]= $\{1.+1.66667\,z, -0.0459184\,x^2-0.0459184\,y^2+1.\,z^2\}$

```
In[ ]:= syl6 = sylvesterMD[ideal6, 4, {x, y, z}];
        syl6b = sylvesterMD[basis5, 4, {x, y, z}];
        intersect6 = vectorSpaceIntersection [syl6, syl6b, dTol];
        Length[intersect6]

Out[ ]= 1
```

Since the length is 1 we do not need an hBasis calculation

```
In[ ]:= Teq = Chop[intersect6.mExpsMD[4, {x, y, z}], dTol]⟦1⟧
```

Out[ ]= $0.493939-0.548821\,x^2+0.0548821\,x^4-0.548821\,y^2+0.109764\,x^2\,y^2+0.0548821\,y^4+0.329293\,z^2+0.109764\,x^2\,z^2+0.109764\,y^2\,z^2+0.0548821\,z^4$

We check that this is a surface containing our original parameterization

```
In[ ]:= Chop[Simplify[Teq /. Thread[{x, y, z} → T]], 1.*^-10]

Out[ ]= 0
```

Simplifying a little more

```
In[ ]:= Teq = Expand[9 Teq / Teq⟦1⟧]
```

Out[ ]= $9.-10.\,x^2+1.\,x^4-10.\,y^2+2.\,x^2\,y^2+1.\,y^4+6.\,z^2+2.\,x^2\,z^2+2.\,y^2\,z^2+1.\,z^4$

```
In[ ]:= Teq = FromCoefficientRules [Normal[Round[<| CoefficientRules [Teq, {x, y, z}] |>]], {x, y, z}]
```

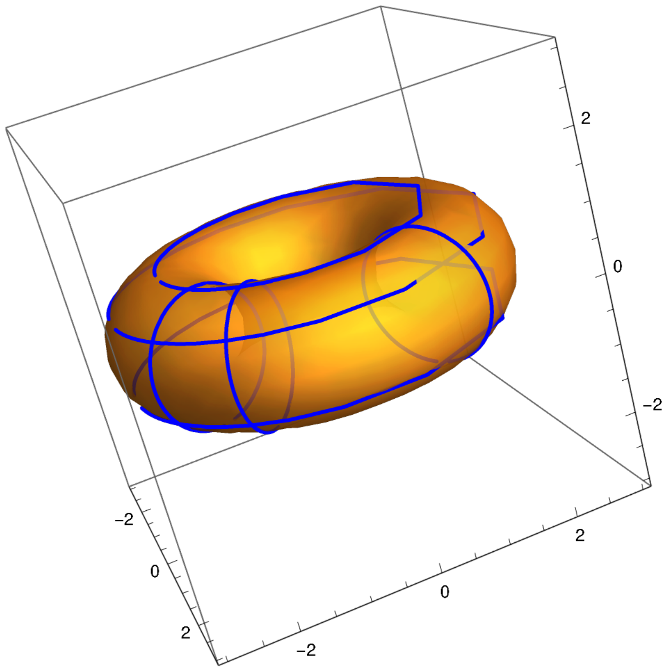
Out[ ]= $9 - 10 x^2 + x^4 - 10 y^2 + 2 x^2 y^2 + y^4 + 6 z^2 + 2 x^2 z^2 + 2 y^2 z^2 + z^4$

we actually get an integer coefficient surface.

```
In[ ]:= Simplify[Teq /. Thread[{x, y, z} → T]]
Out[ ]= 0
```

```
In[ ]:= Show[ContourPlot3D [Teq == 0, {x, -3, 3}, {y, -3, 3}, {z, -3, 3},
          Mesh → None, ContourStyle → Opacity[.8]], ParametricPlot3D [
          {ft2, fsp5, fs4, ftp5, fs2, ft0}, {t, -20, 20}, PlotStyle → Blue, MaxRecursion → 6]]
```

Out[ ]=



Thus we have implicitized our torus! In other words the torus Teq is the only surface of degree 4 contain ing these 6 curves.

1.5 Curves in surfaces

Our calculation shows that one can find out a lot about a curve by studying curves in the surface . This is a classical idea where these curves are called *divisors.* However rarely did one see an actual example. In our own, explicit, way we will find these curves a major technique for studying surfaces.

1.5.1 Curves in rational parametric surfaces.

We study these first since since they are somewhat easier. Since our parameter space is just a plane **every** plane curve lifts to a curve in the parameterized surface. If our parameterization is not one-to-one the curve may be collapsed, or if the parameterization has non-regular points new singularities may be added, so the curve may not look exactly like it looked in the plane. The method is easy, how-ever there are two cases.

We will use the torus in the previous section as we now know both a parametric and implicit equation.

In[ ]:=

$$\text{Tor} = \left\{\frac{4\,s\,(1+t+t^2)}{(1+s^2)\times(1+t^2)},\ -\frac{2\times(-1+s^2-t+s^2\,t-t^2+s^2\,t^2)}{(1+s^2)\times(1+t^2)},\ \frac{1-t^2}{1+t^2}\right\};$$

$$\text{TorEq} = 9-10\,x^2+x^4-10\,y^2+2\,x^2\,y^2+y^4+6\,z^2+2\,x^2\,z^2+2\,y^2\,z^2+z^4;$$

We can just substitute our plane parameterization for the parameters. Here is an example from my Plane Curve Book section 7.3. We change the parameter to *u* so it won't conflict with *s*, *t*.

In[ ]:=
```
F1 = {3 u - u ^ 2 + 1, -2 u + u ^ 2 - 2} / (1 + u + u ^ 2)
```

Out[ ]=

$$\left\{\frac{1+3\,u-u^2}{1+u+u^2},\ \frac{-2-2\,u+u^2}{1+u+u^2}\right\}$$

This is an ellipse .

In[ ]:=
```
A1 = {{-1, 3, 1}, {1, -2, -2}, {1, 1, 1}};
F1eq = FLTMD[tBasis2, A1, 2, {x2, x1}, {x, y}, dTol][[1]]
```

```
» Initial Hilbert Function  {1, 3, 5}

» Final Hilbert Function  {1, 3, 5}
```

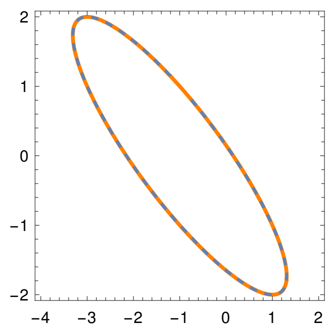
Out[ ]=

$$1.-6.\,x-3.\,x^2-6.\,y-6.\,x\,y-4.\,y^2$$

Note the error in the Plane Curve book!

In[ ]:=
```
Show[ContourPlot[F1eq == 0, {x, -4, 2},
   {y, -2, 2}, ContourStyle → Directive[Thick, Orange]],
  ParametricPlot[F1, {u, -20, 20}, PlotStyle → Dashed]]
```

Out[ ]=



We then get a space curve

In[ ]:=
```
TF1 = Simplify[Tor /. {s → F1⟦1⟧, t → F1⟦2⟧}]
```

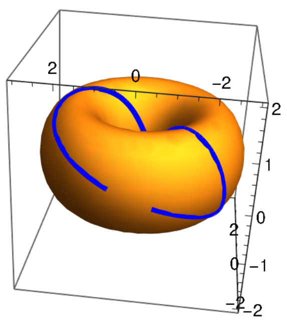
Out[ ]=

$$\left\{-\frac{6\times(-1-3u+u^2)\times(1+u+u^2)\times(1+2u-u^3+u^4)}{(1+4u+5u^2-2u^3+u^4)\times(5+10u+3u^2-2u^3+2u^4)},\ \frac{12u(-1-3u+5u^3-3u^5+2u^6)}{(1+4u+5u^2-2u^3+u^4)\times(5+10u+3u^2-2u^3+2u^4)},\ \frac{3\times(-1-2u+u^2+2u^3)}{5+10u+3u^2-2u^3+2u^4}\right\}$$

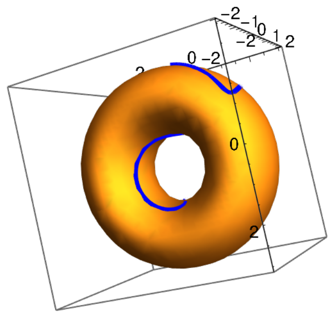
This is somewhat complicated and we end up with a curve of degree 8, the product of the degrees. This is why no-one attempts this by hand. Two views are given.

```
Show[ContourPlot3D[TorEq == 0, {x, -3, 3}, {y, -3, 3}, {z, -2, 2}, Mesh → None],
  ParametricPlot3D[TF1, {u, -20, 20}, PlotStyle → Blue]]
```

In[ ]:= {



,



}

1.5.2 Curves in Implicit Surface

Curves in implicit surfaces can easily be defined by intersecting with another implicit surface. In this case we get a naive space curve as defined in my *Space Curve Book.* Possibly this curve is empty. In other cases we have to use the techniques of that book to describe the curve. Typically the degree of this curve will be the product of the two degrees so can be large. As in the Torus example of Section 1.4 we often use a plane as our second surface to preserve the degree. For example, when our surface is quadratic using a second quadratic surface we already have a hard problem to describe the curve, this is the Quadratic Surface Intersection problem of Section 3.2 of the Space Curve Book.

Now we introduce the important Fermat surface

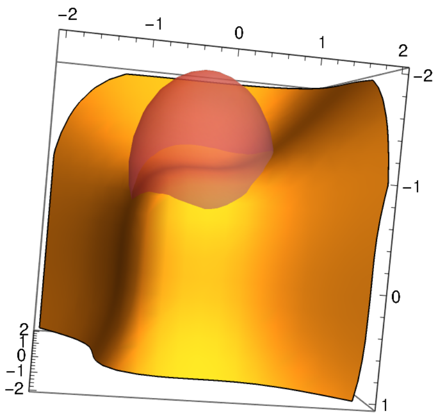
```
In[ ]:= fermat = x^3 + y^3 + z^3 + 1;
```

We will make a curve on fermat by intersecting with the sphere

```
In[ ]:= sph = (x + 1)^2 + y^2 + z^2 - 1;
```

```
In[ ]:= Show[ContourPlot3D[{fermat == 0}, {x, -2, 1}, {y, -2, 2}, {z, -2, 2}, Mesh → None],
         ContourPlot3D[sph == 0, {x, -2, 2}, {y, -2, 2}, {z, -1, 2},
          Mesh → None, ContourStyle → Directive[Opacity[.6], Pink]]]
```

Out[ ]=



To plot we need to find some points on the intersection curve

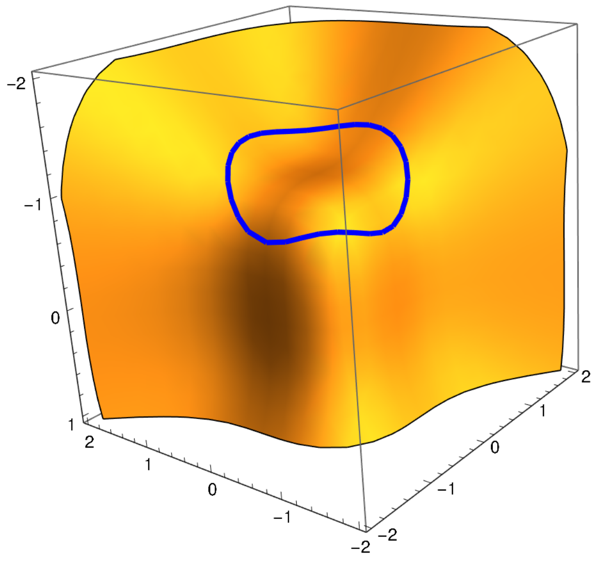
```
In[ ]:= cp = criticalPoints3D[{fermat, sph}, {x, y, z}]

Out[ ]= {{-1.24155, 0., 0.970389}, {-0.606468, -0.919311, 0.}, {-1.20364, 0.458357, 0.865123},
         {-0.713712, -0.75704, -0.587307}, {-1.24155, 0.970389, 0.}, {-0.606468, 0., -0.919311}}
```

We can now find points on the curve by

```
In[ ]:= P1 = pathFinder3D[{fermat, sph}, cp⟦1⟧, cp⟦6⟧, .2, {x, y, z}]

Out[ ]= {{-1.24155, 0., 0.970389},
         {-1.23247, 0.195949, 0.952659}, {-1.21237, 0.384334, 0.898436},
         {-1.19336, 0.557564, 0.8073}, {-1.18662, 0.707692, 0.681428},
         {-1.19629, 0.827291, 0.526364}, {-1.21646, 0.911499, 0.349736},
         {-1.23537, 0.958754, 0.159351}, {-1.24116, 0.969773, -0.0371887},
         {-1.22443, 0.946568, -0.231607}, {-1.17909, 0.892284, -0.414435},
         {-1.10329, 0.811157, -0.575635}, {-1.00059, 0.707695, -0.706518},
         {-0.880796, 0.584756, -0.802403}, {-0.760452, 0.441632, -0.864626},
         {-0.663004, 0.275756, -0.900218}, {-0.606468, 0., -0.919311}}
```

```
In[ ]:= P2 = pathFinder3D[{fermat, sph}, cp[[1]], cp[[6]], .2, {x, y, z}, dir → -1]
Out[ ]= {{-1.24155, 0., 0.970389},
  {-1.22964, -0.195509, 0.953437}, {-1.18999, -0.381293, 0.904722},
  {-1.11984, -0.547315, 0.828302}, {-1.02166, -0.684434, 0.728753},
  {-0.90404, -0.787017, 0.609422}, {-0.782202, -0.855196, 0.470323},
  {-0.67842, -0.895117, 0.30879}, {-0.617386, -0.915248, 0.126204},
  {-0.609214, -0.917986, -0.067741}, {-0.640588, -0.896043, -0.260633},
  {-0.685277, -0.839144, -0.443606}, {-0.716363, -0.741066, -0.608581},
  {-0.715878, -0.604332, -0.744351}, {-0.684168, -0.438983, -0.841157},
  {-0.639503, -0.255812, -0.896996}, {-0.606468, 0., -0.919311}}
```

```
In[ ]:= Show[ContourPlot3D[fermat == 0, {x, -2, 1}, {y, -2, 2}, {z, -2, 2}, Mesh → None],
  Graphics3D[{{Blue, Thick, Line[P1]}, {Blue, Thick, Line[P2]}}]]
Out[ ]=
```



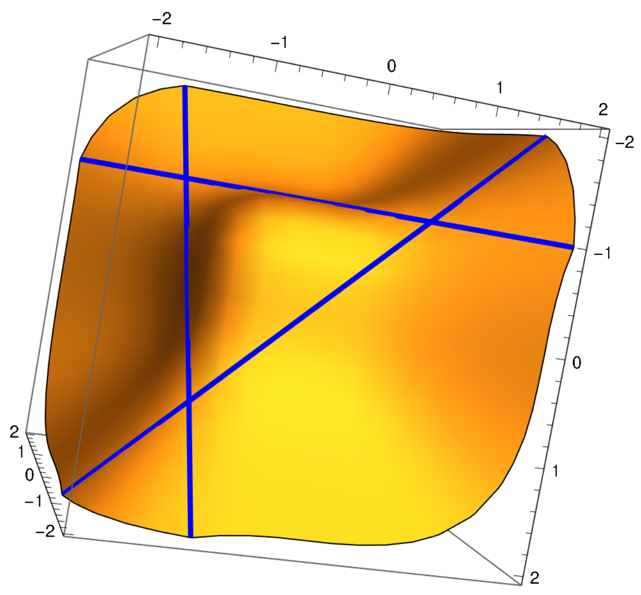
1.5.3 Implicit Surface and Parametric Curve

A third possibility is to use a parametric curve with the implicit surface. However this requires some cleverness as there is no general method for doing this. For example one may observe that the that the Fermat surface above contains the parametric lines $\{t, -t, -1\}$, $\{t, -1, t\}$ and $\{-1, t, -t\}$ in this surface. We will see later there are no other real lines in this surface.

```
In[ ]:= fermat /. Thread[{x, y, z} → {t, -t, -1}]
Out[ ]= 0
```

In[∘]:=
```
Show[ContourPlot3D [fermat == 0, {x, -2, 2}, {y, -2, 2}, {z, -2, 2}, Mesh → None],
 ParametricPlot3D [{{t, -t, -1}, {t, -1, -t}, {-1, t, -t}}, {t, -2, 2}, PlotStyle → Blue]]
```

Out[∘]=



1.5.4 Some Code

For the reader's convenience we give the code for the two routines we used in 1.5.2, the are, of course in *GlobalFunctionsS.nb*. But some readers of the Space Curve book may notice that pathFinder3D has changed, new options are allowed, in particular the option dir→-1 which allowed us to change directions.

In[∘]:=
```
criticalPoints3D [{f_, g_}, {x_, y_, z_}] := Module[{J, ob},
   ob = RandomReal [{.7, 1.3}, 3].{x^2, y^2, z^2};
   J = D[{f, g, ob}, {{x, y, z}}];
   {x, y, z} /. NSolve[{f, g, N[Det[J]]}, {x, y, z}, Reals]]
```

In[ ]:=
```
Options[pathFinder3D] = {maxit → 30, tol → 1.*^-8, dir → 1};
pathFinder3D[{f_, g_}, p_, q_, s_, {x_, y_, z_}, OptionsPattern[]] :=
  Module[{k, p0, p1, tv1, tv, L},
   p0 = p;
   L = Reap[Sow[p];
     k = 0;
     While[Norm[q - p0] > 2 s && k < OptionValue[maxit],
      tv1 = OptionValue[dir] *
        tangentVector3D[{f, g}, p0, {x, y, z}, tol → OptionValue[tol]];
      If[tv1.(q - p0) > 0, tv = tv1, tv = -tv1];
      p0 = closestPoint3D[{f, g}, p0 + s * tv, {x, y, z}];
      Sow[p0];
      k++];
     If[k ≥ OptionValue[maxit], Print["Warning, iteration limit reached"]];
     Sow[q]];
   L[[2, 1]];
```

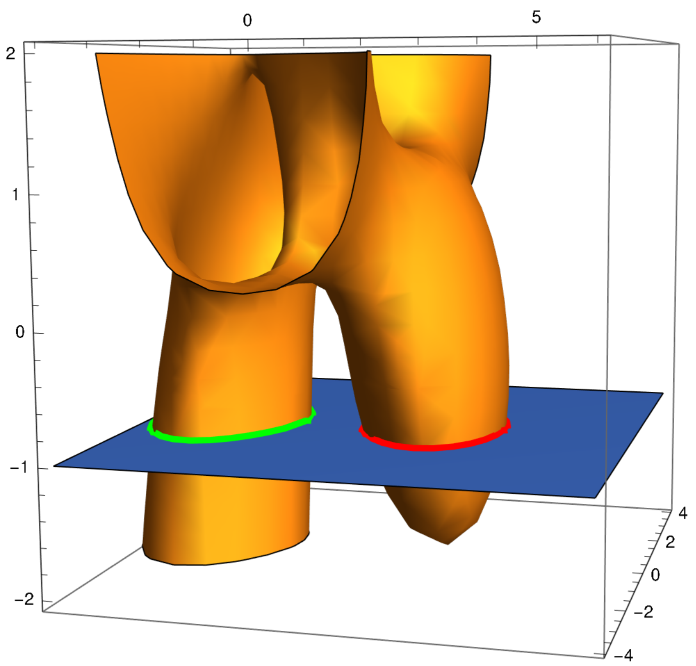
1.5.5 Ovals and Pseudo-Lines

In both my *Plane Curve Book* and *Space Curve Book* I discuss my *Fundamental Theorem* as well as ovals and pseudo-lines which make most sense for non-singular curves. The Euler graph of a curve may not be connected, in the graph theory sense. A connected component of the graph then refers to a closed topological subcurve of the curve which may or may not be an entire algebraic curve. This subcurve will be an *oval* if it meets the infinite plane in an even number of points, a *pseudo-line* if it meets the infinite plane in an odd number of points counting multiplicity in both cases. Actually any fixed plane of projective space can be used instead of the infinite plane, so any closed subcurve which misses some plane entirely is an oval, in particular bounded closed curves are ovals.

As an example the curve in the fermat surface of section 1.5.2 is an oval whereas the lines in 1.5.3 are, of course, pseudo-lines. Consider the surface from Section 1.1

In[ ]:=
```
ts3 = 1.752 - 6.4 x - 11.464 x² + 0.64 x³ + x^4 + 1.536 y² +
   0.64 x y² + x² y² + 2.88 x^2 z - 5.12 y^2 z + 3.584 z² + 3.84 x z² + x² z²;
```

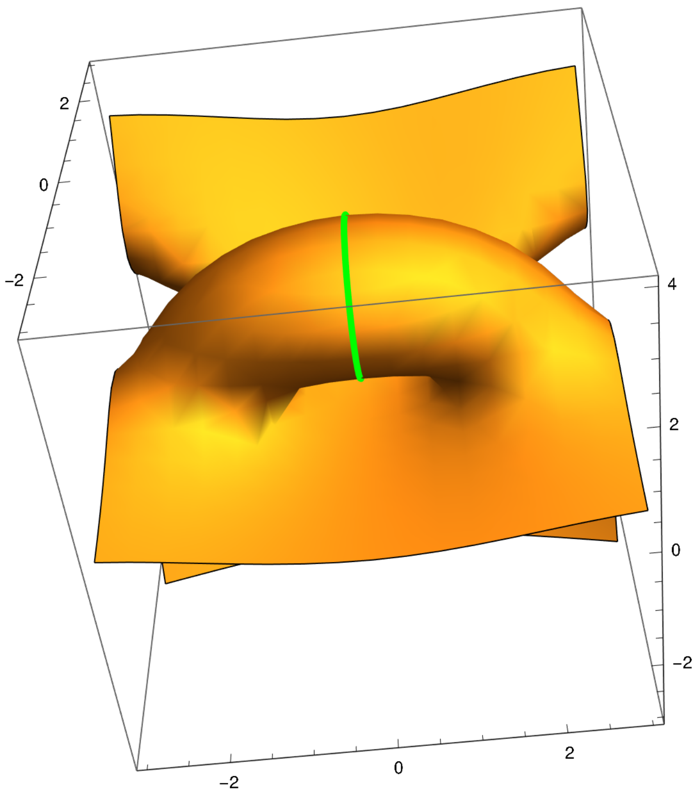
We intersect this with the plane $z = -1$ and get two ovals as shown in the plot in red and green. We suppress the work.



Neither oval is a curve alone, but the union is the naive space curve {ts3, $z + 1$}. We had to use path finding to draw these.

There is a new difference between these ovals. The red oval is *null homotopic* which means that if one thinks of this as a ring on a finger then it can be slipped off without hurting the surface. More precisely it can be moved continuously on the surface until it degenerates into a point at the bottom. The reader should note here that we are purposely being heuristic. On the other hand the green oval can not be obviously deformed to a point or “removed”. Another difference is that the red oval *separates* the surface into the part on that finger which is above the oval and the small part below the oval. Again it is not clear from this picture if the green oval does this, we will have to wait until later when we treat these surfaces as projective surfaces. The surface in Section 1.4 called **ts5** (in step 5) gives a better picture, work suppressed.

$$\mathtt{ts5} = 9 - 81\,x + \frac{1361\,x^2}{16} - \frac{101\,x^4}{16} + 16\,y^2 - \frac{117\,x^2\,y^2}{16} - y^4 + \frac{81\,x\,z}{2} - \frac{117\,x^2\,z}{2} + \frac{27\,x^3\,z}{2} - 16\,y^2\,z + \frac{27}{2}\,x\,y^2\,z + 6\,z^2 - 27\,x\,z^2 - \frac{85\,x^2\,z^2}{16} + \frac{27\,x\,z^3}{2} + z^4;$$



Here the green oval, a subcurve of the naive curve {ts5, y}, clearly does not separate this surface. Of course our 6 curves on the torus in Section 1.4 also do not separate the torus.

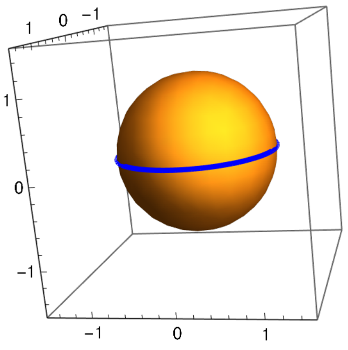
We also note from the torus example that the 3 horizontal curves each meet the three vertical curves in exactly one point. This is in stark difference where any algebraic curve meets an oval in an even number of points by multiplicity. That property of an oval was a crucial step in our proof of Harnak's Theorem, but it not true in the surface case. The other difference is that, in general, ovals do not have an inside and outside like plane ovals. Some, like the end of the finger of ts3 do, that is, the end part is topologically equivalent to a disk while the other part is not.

An example here is the sphere which, if anything, has two interiors when cut by the equator. The equator is clearly null-homotopic and can be deformed to either the north or south poles.

In[ ]:=
```
sphere = x^2 + y^2 + z^2 - 1;
```
$$\text{equator} = \left\{\frac{2\,t}{1+t\verb|^|2}, \frac{1-t\verb|^|2}{1+t\verb|^|2}, 0\right\};$$

In[ ]:=
```
Show[ContourPlot3D [sphere == 0, {x, -1.5, 1.5}, {y, -1.5, 1.5}, {z, -1.5, 1.5},
  Mesh → None], ParametricPlot3D [equator , {t, -20, 20}, PlotStyle → Blue]]
```

Out[ ]=



In summary, there are three kinds of closed curves/ subcurves. The pseudo-lines, the non-null-homo - topic ovals and the null-homotopic ovals. The first two do not separate a surface into connected components while he third does separate the surface. Often we will call a non-null-homotopic oval an *essential oval.*

1.6 Rational Points and Rational Surfaces

We have been discussing rationally parameterized surfaces, Section 2. Here we make the distinction between these and *Rational surfaces,* note the capital R. These are rationally parameterized surfaces with the additional property that the coefficients of all the polynomials in the numerators and denominators have rational, equivalently integer, coefficients. In previous sections most of my examples are of this type, but given my wide use of Mathematica machine numbers it would certainly be permissible to use a non-rational machine number as a coefficient.

An observation is that because a Rational parameterization has only rational coefficients then every rational value of the parameters gives a rational point, that is a point where all components are rational numbers. For example for the torus

In[ ]:=
$$\text{Tor} = \left\{\frac{4\,s\,(1+t+t^2)}{(1+s^2)\times(1+t^2)},\ -\frac{2\times(-1+s^2-t+s^2\,t-t^2+s^2\,t^2)}{(1+s^2)\times(1+t^2)},\ \frac{(1-t^2)\times(1+s^2)}{(1+t^2)\times(1+s^2)}\right\};$$

if one takes, say $t = \frac{13}{7}$, $s = \frac{21}{4}$ then

```
In[ ]:= p = Tor /. {s → 21 / 4, t → 13 / 7}
```

Out[ ]=
$$\left\{\frac{51\,912}{49\,813},\ -\frac{131\,325}{49\,813},\ -\frac{60}{109}\right\}$$

one gets this horrible denominator but none-the-less a rational number. We don’t notice since we work numerically and the point appears as

```
In[ ]:= N[p]
Out[ ]= {1.04214, -2.63636, -0.550459}
```

which looks like any other point. But we have illustrated the following fact:

```
The set of rational points in a rational surface is dense.
```

The precise meaning is that for any point on the rational surface and any $\epsilon > 0$ there is a rational point within euclidean distance ϵ of that point. This also works for a rational curve which implies, using the fact that the circle $x^2 + y^2 - 1$ is rational , that any right triangle is arbitrarily close to a right triangle with rational sides. If the early mathematicians knew this there would be no need for irrational numbers. But of course Euclid never thought about fractions like

```
In[ ]:= -p⟦2⟧
```

Out[ ]=
$$\frac{131\,325}{49\,813}$$

We may then ask the question about a general surface: are there many rational points? For curves with only rational coefficients Gerd Faltings proved in 1983 a 1922 conjecture of Louis Mordell that if the genus is 2 or greater there can only be finitely many rational points. It turns out that this is more complicated for surfaces. Here is one of many places where algebraic geometry meets number theory.

The Fermat surface used in the previous section is a good example. This is a surface that is known to not be rational. Yet we noticed that there are 3 rational lines, $\{t, -t, -1\}$, $\{t, -1, -t\}$, $\{-1, t, -t\}$. Thus plugging in any rational value for *t* gives a rational point of the surface. So there are infinitely many rational points in this surface. Are there others?

We can experiment with Mathematica . A *Diophantine problem* is to find integer solutions to a polyno-mial equation with integer coefficients. Mathematica has some good algorithms to find solutions to these problems. A general routine is the build in `FindInstance`. In this case we can use it as follows. We start with the equation of the Fermat surface

```
In[ ]:= fermat = x^3 + y^3 + z^3 + 1;
```

To get rational solutions we homogenize this by replacing 1 by a new variable *w* which we will use as a denominator.

```
In[ ]:= fermatH = x^3 + y^3 + z^3 + w^3;

In[ ]:= FindInstance[fermatH == 0, {x, y, z, w}, Integers]

Out[ ]= {{x → 0, y → 0, z → 0, w → 0}}
```

That was rather obvious, but doesn't actually give a rational solution, try again.

```
In[ ]:= FindInstance[fermatH == 0 && w ≠ 0, {x, y, z, w}, Integers]

Out[ ]= {{x → 1, y → -1, z → -1, w → 1}}
```

Still quite obvious but gives {1, −1, 1}, a point in one of our lines. Lets try for a point not on one of our lines.

```
In[ ]:= FindInstance[fermatH == 0 && (x + y) (x + z) (y + z) ≠ 0, {x, y, z, w}, Integers]

Out[ ]= {{x → 12, y → 1, z → -9, w → -10}}
```

Now this is interesting, the point $\left\{-\frac{12}{10}, -\frac{1}{10}, \frac{-9}{-10}\right\}$ is in our surface:

```
In[ ]:= fermat /. Thread[{x, y, z} → {-12/10, -1/10, -9/-10}]

Out[ ]= 0
```

In principal, `FindInstance` will give a desired number of solutions, but for this problem it will not.

```
In[ ]:= FindInstance[fermatH == 0 && (x + y) (x + z) (y + z) ≠ 0, {x, y, z, w}, Integers , 2]

   FindInstance : The methods available to FindInstance are insufficient to find the requested instances or prove
      they do not exist .

Out[ ]= FindInstance[w³ + x³ + y³ + z³ == 0 && (x + y) (x + z) (y + z) ≠ 0, {x, y, z, w}, ℤ, 2]
```

so we must make do with one solution at a time, even though permutations of the coordinates will give another solution due to the symmetry of the problem.

I pause to give a nice way to get from the `FindInstance` output to the affine rational point. Let

```
In[ ]:= A = {{1, 0, 0, 0, 0}, {0, 1, 0, 0, 0}, {0, 0, 1, 0, 0}, {0, 0, 0, 1, 0}};
        A // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

We take the output of **FindInstance** using only the first instance, changing the conditions may give a new instance

```
In[ ]:= inst = {x, y, z, w} /.
           FindInstance[fermatH == 0 && (x + y) (x + z) (y + z) ≠ 0 && w > 5, {x, y, z, w}, Integers]⟦1⟧
Out[ ]= {6, 1, -9, 8}
```

Now we use

```
In[ ]:= fltMD[inst, A]
```

Out[ ]= $\left\{\frac{3}{4}, \frac{1}{8}, -\frac{9}{8}\right\}$

Further we can replace A by any permutation of the first 3 rows of A to get additional solutions by permuting the components. As the the lower bound for *w* gets larger this will take more time

```
In[ ]:= inst = Timing[{x, y, z, w} /. FindInstance[fermatH == 0 &&
              (x + y) (x + z) (y + z) ≠ 0 && x ^ 2 + y ^ 2 + z ^ 2 + w ^ 2 > 700, {x, y, z, w}, Integers]⟦1⟧]
Out[ ]= {8.26453, {-24, -2, 18, 20}}
```

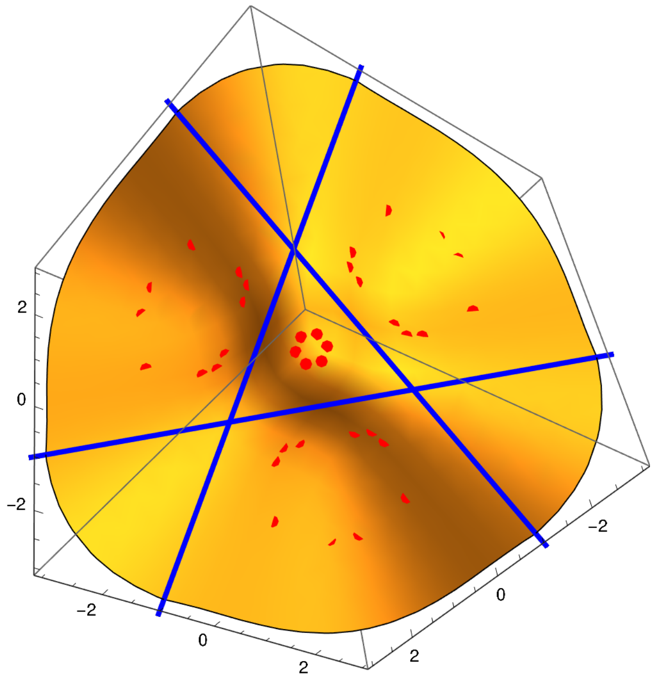
Proceeding this way I found 6 instances which after permuting

```
In[ ]:= fermatH /. Thread[{x, y, z, w} → {-24, -2, 18, 20}]
```

which, after permuting gave 36 different solutions not on the three lines. Plotting I get

In[•]:=
```
Show[ContourPlot3D [x ^ 3 + y ^ 3 + z ^ 3 + 1 == 0, {x, -3, 3}, {y, -3, 3}, {z, -3, 3}, Mesh → None],
 ParametricPlot3D [{{t, -t, -1}, {t, -1, -t}, {-1, t, -t}}, {t, -20, 20}, PlotStyle → Blue],
 Graphics3D [{Red, PointSize [.02], Point[S]}]]
```

Out[•]=



The symmetry is partly due to the symmetry of the surface and our permutations but there are 10 points in 3 of the non-central sectors in somewhat of an oval pattern. The symmetry in the central triangle is completely explained by the 6 symmetries of one instance but not the other symmetries. Perhaps there are 3 other rational curves on this surface? There certainly are lots of other rational points to find here so this is, to me, an open problem.

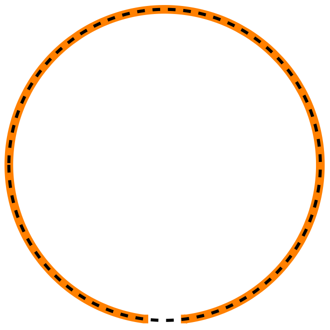
1.7Trigonometric Parameterization

In this section I give some other parameterized surfaces using rational parametric functions as proxies for trigonometric `Cos`, `Sin` parameterizations. It is based on the parameterization of the circle I have been using

In[ ]:=
```
circ2D = {2 t / (1 + t ^ 2), (1 - t ^ 2) / (1 + t ^ 2)}
```

Out[ ]=
$$\left\{\frac{2\,t}{1+t^2},\ \frac{1-t^2}{1+t^2}\right\}$$

In[ ]:=
```
Show[ParametricPlot [{2 t / (1 + t ^ 2), (1 - t ^ 2) / (1 + t ^ 2)}, {t, -15, 15},
   PlotStyle → {Directive [Thickness [.025], Orange]}, PlotRange → Full, Axes → None],
  ParametricPlot [{Cos[u], Sin[u]}, {u, -Pi, Pi}, PlotStyle → Directive [Black, Dashed]],
  ImageSize → Small]
```

Out[ ]=


Often, in this book on surfaces I will use the following curve

```
circ3D = {2 t / (1 + t ^ 2), (1 - t ^ 2) / (1 + t ^ 2), 0}
```

Theoretically the parameter *t* should actually run from $-\infty \le t \le \infty$ where at the endpoints we mean of course the limit. In practice we can use a large bounded range. We will use *s*, *t* exclusively for the rational parameterizations with *u*, *v* used in the trigonometric ones so there will be no notational confusion. Here *u*, *v* will normally run as above $-\pi \le u, v \le \pi$.

I mention here that some of these parameterizations in from the book *CRC Standard Curves and Surfaces with Mathematica* by David H. von Seggern. Others may be found at *Wolfram MathWorld* and the *Wolfram Demonstrations Project.*

1.7.2 Parametric surfaces via trigonometry

The Sphere and hyperboloid

In[ ]:=
```
trigSphere = {Sin[u] Cos[v], Sin[u] Sin[v], Cos[u]};
```
rationalSphere = $\left\{\frac{(1-t^2)\times 2\,s}{(1+t^2)\times(1+s^2)},\ \frac{(1-t^2)\times(1-s^2)}{(1+t^2)\times(1+s^2)},\ \frac{2\,t\,(1+s^2)}{(1+t^2)\times(1+s^2)}\right\}$;

I earlier mentioned the rational parameterization of the hyperboloid, I repeat so we have these all together .

In[ ]:=
```
hyperboloid3D = {(t - s^2 t)/(1 - s^2), (1 + s^2 - 2 s t)/(1 - s^2), (2 s - t - s^2 t)/(1 - s^2)};
```

$$\text{hyperboloid3D} = \left\{\frac{t - s\text{^}2\,t}{1 - s\text{^}2}, \frac{1 + s^2 - 2\,s\,t}{1 - s^2}, \frac{2\,s - t - s^2\,t}{1 - s^2}\right\};$$

The Torus: the standard parameterization is the following where *a* is the large radius and *b* the small. Our torus in Section 4 parameterization is based on this.

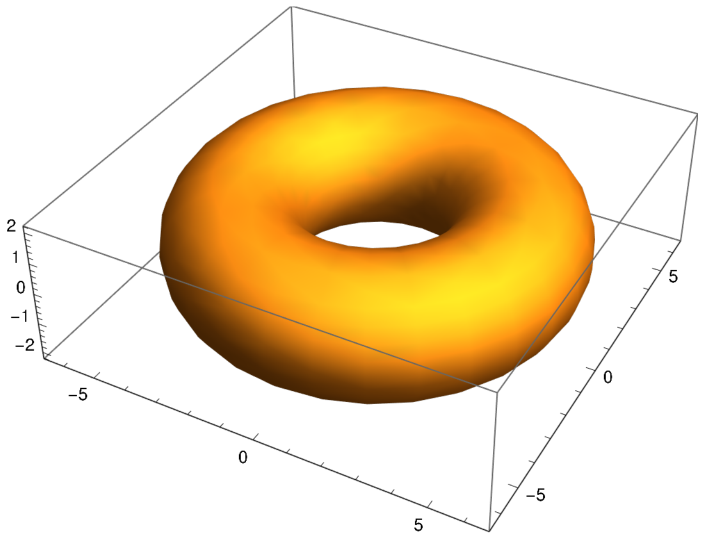
```
trigTorus = {(a + b Cos[v]) Cos[u], (a + b Cos[v]) Sin[u], b Sin[v]};
```

For large radius 4 and small radius 2

In[ ]:=
```
TrigTorus = trigTorus /. {a → 4, b → 2}
```

Out[ ]=
```
{Cos[u] (4 + 2 Cos[v]), (4 + 2 Cos[v]) Sin[u], 2 Sin[v]}
```

In[ ]:=
```
ParametricPlot3D [TrigTorus , {u, -Pi , Pi}, {v, -Pi , Pi}, Mesh → None]
```

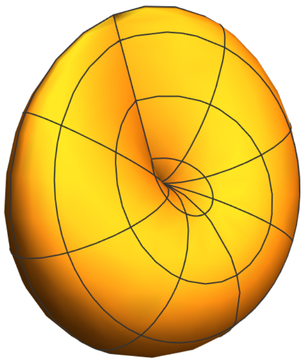
Out[ ]=


The Crosscap

In[ ]:=
```
crocap = {Sin[u] Sin[2 v] / 2, Sin[2 u] Cos[v]^2, Cos[2 u] Cos[v]^2};
```

In[ ]:=
```
ParametricPlot3D [crocap , {u, -Pi , Pi}, {v, -Pi , Pi}, Boxed → False , Axes → False]
```

Out[ ]=



This is algebraic since elementary trig identities, eg. Sin[2 u] = 2 Sin[u] Cos[u], allow one to write these parameters in terms of the proxies for sin3 and cosine. Also the square of the proxies are again rational functions. These equations can get quite involved and the implicit equations may be of very high degree.

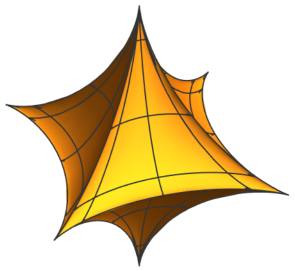
Astroidal Surface

In[ ]:=
```
astroid = {(Cos[u] Cos[v])^3, ( Sin[u] Cos[v])^3, Sin[v]^3}
```

Out[ ]=
```
{Cos[u]^3 Cos[v]^3, Cos[v]^3 Sin[u]^3, Sin[v]^3}
```

In[ ]:=
```
ParametricPlot3D [astroid , {u, -Pi , Pi}, {v, -Pi , Pi},
  MaxRecursion → 3, PlotRange → 1, Axes → False , Boxed → False]
```

Out[ ]=



will be algebraic. von Seggern tells us the equation is

$$x^{2/3} + y^{2/3} + z^{2/3} = 1$$

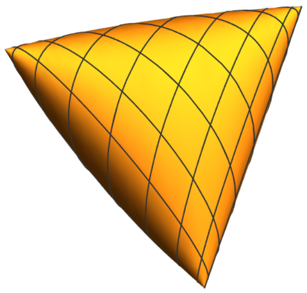
which is not algebraic. But our theorems of Section 1.3 tell us there be algebraic equations as well.

The *Cosine Surface* likewise will be algebraic because of the elementary formula for Cos[u+v] =Cos[u] Cos[v]-Sin[u] Sin[v]

```
In[ ]:= cosSurf = {Cos[u], Cos[v], Cos[u + v]};
```

```
In[ ]:= ParametricPlot3D [cosSurf , {u, -Pi , Pi}, {v, -Pi , Pi},
         MaxRecursion → 3, PlotRange → 1, Axes → False , Boxed → False]
```

Out[ ]=



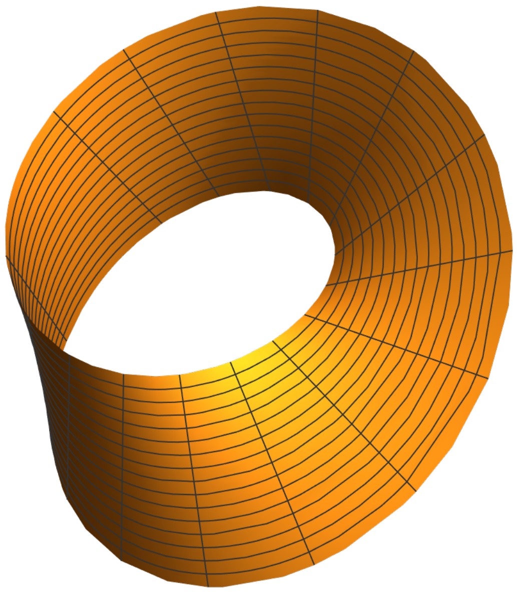
An example of a surface which is not algebraic is the Möbius Strip.

```
In[ ]:= moeband = {Cos[u] (1 + t Cos[u / 2]), Sin[u] (1 + t Cos[u / 2]), t Sin[u / 2]}
```

Out[ ]= $\left\{\left(1+t\,\mathrm{Cos}\left[\frac{u}{2}\right]\right)\mathrm{Cos}[u],\ \left(1+t\,\mathrm{Cos}\left[\frac{u}{2}\right]\right)\mathrm{Sin}[u],\ t\,\mathrm{Sin}\left[\frac{u}{2}\right]\right\}$

```
In[ ]:= ParametricPlot3D [moeband , {u, -Pi , Pi}, {t, -.5, .5}, MaxRecursion → 3,
         PlotRange → All, PlotRange → 1, Axes → False , Boxed → False]
```

Out[ ]=



As pointed out in my *Plane Curve Book* this is a one-sided surface and cannot be a naive algebraic surface. The problem is not combining parameters u, t, rather the Cos$\left[\frac{u}{2}\right]$ can not be expressed polyno-mially in terms of Sin[u] and Cos[u].

The spirals likewise cannot be algebraic because we cannot use the same variable as algebraic and trigonometric parameter. For example van Seggern gives

vSspiral = {a Cos[n v] (1 + Cos[u]) + c Cos[n v], a Sin[n v] (1 + Cos[u]) + c Sin[n v], b v/2/Pi + a Sin[u]};

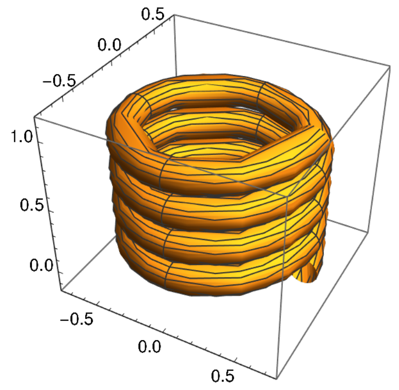
where a, b, c are positive numbers and n is a positive integer.

The example given has

```
In[ ]:= vSspiral1 = vSspiral /. {a → .1, b → 1, c → .5, n → 4}
```

$$\text{Out[ ]=}\ \left\{0.5\ \text{Cos}[4\ v] + 0.1 \times (1 + \text{Cos}[u])\ \text{Cos}[4\ v],\ 0.5\ \text{Sin}[4\ v] + 0.1 \times (1 + \text{Cos}[u])\ \text{Sin}[4\ v],\ \frac{v}{2\pi} + 0.1\ \text{Sin}[u]\right\}$$

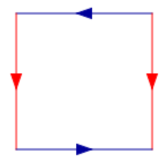
```
In[ ]:= ParametricPlot3D [vSspiral1 , {u, 0, 2 Pi}, {v, 0, 2 Pi}]
```

Out[ ]=


But parameter v is being used in both a trigonometric and analytic parameter in the last coordinate so this will not define a naive implicit surface.

1.7.3 The Klein Bottle

The Klein Bottle is a simple topological surface in 4-space obtained by gluing the sides of the square blue to blue and red to red in the indicated directions without self intersections, the last instruction cannot be done in 3 space.

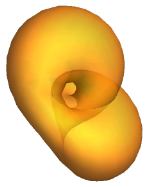


We take our exposition from *Wolfram Mathworld.* An implicit equation of a projection into 3-space is

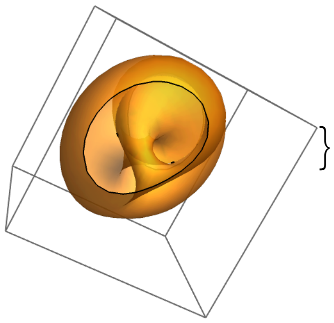
```
In[ ]:= KbottEq = (x^2 + y^2 + z^2 + 2 y - 1) ((x^2 + y^2 + z^2 - 2 y - 1)^2 - 8 z^2) +
          16 x z (x^2 + y^2 + z^2 - 2 y - 1);
```

In[ ]:=
```
{ContourPlot3D[KbottEq == 0, {x, -4, 4}, {y, -4, 4}, {z, -4, 4}, Mesh → None,
   ContourStyle → Opacity[.7], MaxRecursion → 4, Axes → False, Boxed → False],
 ContourPlot3D[KbottEq == 0, {x, -3, 3}, {y, -3, 3}, {z, -3, 3}, Mesh → None,
   ContourStyle → Opacity[.7], MaxRecursion → 4, Axes → False]}
```

Out[ ]= {



,



}

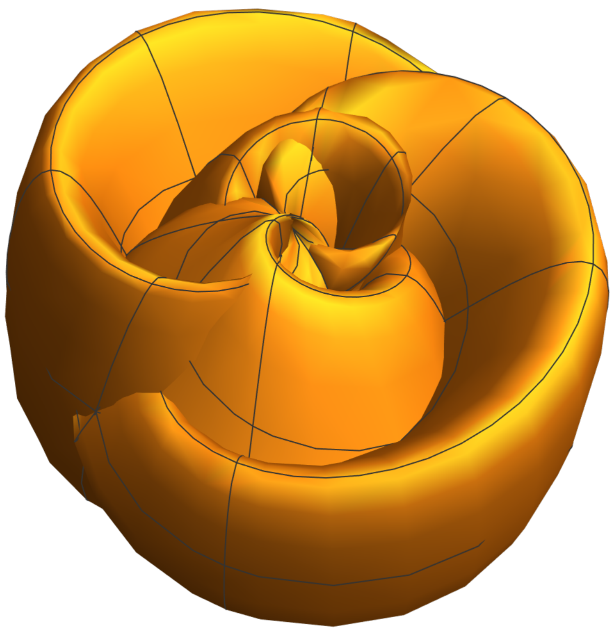
In the right hand plot we slice the surface by sides of the box to better see the interior. Of course projecting causes self intersections. Here is an interesting trigonometric parameterization of an interpretation of this 4 dimensional surface.

In[ ]:=
```
sq2 = N[Sqrt[2]];
kbx = Cos[u] (Cos[.5 u] (sq2 + Cos[v]) + Sin[.5 u] Sin[v] Cos[v]);
kby = Sin[u] (Cos[.5 u] (sq2 + Cos[v]) + Sin[.5 u] Sin[v] Cos[v]);
kbz = Sin[.5 u] Sin[v] + Cos[.5 u] Sin[2 v];
kbPar = {kbx, kby, kbz}
```

Out[ ]=
```
{Cos[u] (Cos[0.5 u] (1.41421 + Cos[v]) + Cos[v] Sin[0.5 u] Sin[v]),
 Sin[u] (Cos[0.5 u] (1.41421 + Cos[v]) + Cos[v] Sin[0.5 u] Sin[v]),
 Sin[0.5 u] Sin[v] + Cos[0.5 u] Sin[2 v]}
```

In[ ]:=
```
ParametricPlot3D [kbPar , {u, 0, 4 Pi},
  {v, 0, 4 Pi}, PlotRange → All, Axes → False, Boxed → False]
```

Out[ ]=



In[ ]:=
```
Simplify[KbottEq /. Thread[{x, y, z} → (kbPar /. {u → 3, v → 2})]]
```

Out[ ]=
```
1.15968
```

This does not satisfy the implicit equation given and is not guaranteed to give such an equation because of the use of half angles , .5 u, .5 v. But it does show another self intersecting parametric surface.

1.8 Fractional Linear Transformations

We have seen Fractional Linear Transformations before in my curve books and even in Chapter 1 of this book . But as a review, in Mathematica they are given by the built - in TransformationFunction . These are also a special formulation of Projective Linear Transformations, the name *Fractional Linear Transfor - mations* comes from the book [Abhyankar].

1.8.1 Basic concepts

As a first example, from Wolfram documentation,

```
In[ ]:= t = RotationTransform [θ, {0, 0, 1}]
```

$$\text{Out[ ]= TransformationFunction}\left[\left(\begin{array}{ccc|c} \text{Cos}[\theta] & -\text{Sin}[\theta] & 0 & 0 \\ \text{Sin}[\theta] & \text{Cos}[\theta] & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\right]$$

As in this example in this Chapter we are only working in 3 - Space so the TransformationMatrix which the argument of this function is a 4 ×4 matrix. In this illustration it is broken up into parts, the upper left 3×3 is the matrix of a linear transformation. The 3×1 matrix to the right is a translation, in this case the zero transformation. With the given bottom row, the transform is an affine transform, that is the affine three space remains fixed. However if the three left hand zeros are replaced by three numbers not all zero and the corner number 1 is replaced by some other non-zero number then we have a projective transformation. Essentially the last row will give a specialization. In this book the transforma tion is always assumed to be invertible. The best check is with our function **matrixrank** with a loose tolerance so it will be numerically well behaved. We want the result to be 4.

In our special case the TransformationFunction will take a list of length 3 as an argument, that is an affine point. It will return another such point. But unless the last row is {0, 0, 0, *a*}, *a* ≠ 0, this point returned in in a different specialization of projective space, so this will be a projective transformation. As in my other books I can define a *push-forward functor* which works on surfaces rather than points, that is, it gives the equation of the surface obtained by applying the transformation function

Examples:

$$\text{In[ ]:= A1} = \left(\begin{array}{ccc|c} \text{Cos[Pi / 4]} & -\text{Sin[Pi / 4]} & 0 & 0 \\ \text{Sin[Pi / 4]} & \text{Cos[Pi / 4]} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right);$$

```
N[A1] // MatrixForm
```

```
Out[ ]//MatrixForm=
⎛0.707107  -0.707107  0.  0.⎞
⎜0.707107   0.707107  0.  0.⎟
⎜   0.         0.     1.  0.⎟
⎝   0.         0.     0.  1.⎠
```

```
In[ ]:= N[TransformationFunction [A1][{1, 2, 3}]]
Out[ ]= {-0.707107 , 2.12132 , 3.}
```

This is a rotation around the z-axis.

```
In[ ]:= TransformationFunction [A1][{0, 0, 2}]
Out[ ]= {0, 0, 2}
```

```
In[ ]:= B = Append[Join[Orthogonalize [RandomReal [{-1, 1}, {3, 3}]], {{0}, {0}, {0}}, 2], {0, 0, 0, 1}];
        B // MatrixForm
Out[ ]//MatrixForm=
        ⎛-0.638947  -0.0671356   0.766315  0⎞
        ⎜ 0.395362   0.825883    0.402003  0⎟
        ⎜-0.659876   0.55983    -0.501153  0⎟
        ⎝    0           0           0     1⎠
```

we get a rotation of affine 3-space with axis some line through the origin if the matrix has determinant 1 , otherwise the determinant is −1 and it is a reflection of affine 3-space with mirror a plane through the origin.

```
In[ ]:= Det[B]
Out[ ]= 1.

In[ ]:= TransformationFunction [B][{1, 2, 3}]
Out[ ]= {1.52573 , 3.25314 , -1.04367}

In[ ]:= If
Out[ ]= If

In[ ]:= B1 = {{1, 0, 0, 2}, {0, 1, 0, -4}, {0, 0, 1, 3}, {0, 0, 0, 1}};
        B1 // MatrixForm
Out[ ]//MatrixForm=
        ⎛1  0  0   2⎞
        ⎜0  1  0  -4⎟
        ⎜0  0  1   3⎟
        ⎝0  0  0   1⎠
```

Then we get a translation of affine space by vector {2,-4,3}

```
In[ ]:= TransformationFunction [B1][{1, 1, 1}]
Out[ ]= {3, -3, 4}

In[ ]:= If
Out[ ]= If

In[ ]:= B2 = Append[Join[Orthogonalize [RandomReal [{-1, 1}, {3, 3}]], {{2}, {-3}, {5}}, 2], {0, 0, 0, 1}];

In[ ]:= B2 // MatrixForm
Out[ ]//MatrixForm=
        ⎛0.642387  -0.221533   0.733663   2⎞
        ⎜0.642974   0.676728  -0.35864   -3⎟
        ⎜ 0.41704  -0.702113  -0.577162   5⎟
        ⎝    0         0          0       1⎠
```

we will get a rotation about a line or a refection about a plane not necessarily through the origin.

```
In[ ]:= TransformationFunction [B2][{1, 1, 1}]
Out[ ]= {3.15452 , -2.03894 , 4.13777}
```

Finally if we just take a random invertible matrix

```
In[ ]:= B4 = RandomReal [{-1, 1}, {4, 4}];
        B4 // MatrixForm
Out[ ]//MatrixForm=
```

$$\begin{pmatrix} 0.0653255 & 0.557682 & -0.528581 & -0.449588 \\ 0.1035 & -0.630132 & 0.570384 & -0.720002 \\ -0.87486 & 0.0275579 & -0.989399 & -0.45492 \\ 0.915743 & 0.779111 & 0.790421 & 0.946909 \end{pmatrix}$$

we get an affine transformation followed by a specialization

```
In[ ]:= TransformationFunction [B4][{1, 1, 1}]
Out[ ]= {-0.389523 , -1.44758 , -1.11267}
```

where this last point is in a different specialization of projective space .

You may have seen in my books and/or articles that I can use transformation functions to define ratio-nal parametric curves.

```
In[ ]:= Clear[t]

In[ ]:= C1 = RandomInteger [{-9, 9}, {4, 4}]
Out[ ]= {{-2, 6, 3, -4}, {0, -6, -8, -9}, {7, 2, -5, -5}, {-3, 6, 6, -6}}

In[ ]:= matrixrank[C1, .00005]
Out[ ]= 4

In[ ]:= TransformationFunction [C1][{t, t^2, t^3}]
Out[ ]=
```

$$\left\{\frac{-4-2t+6t^2+3t^3}{-6-3t+6t^2+6t^3}, \frac{-9-6t^2-8t^3}{-6-3t+6t^2+6t^3}, \frac{-5+7t+2t^2-5t^3}{-6-3t+6t^2+6t^3}\right\}$$

Thus TransformationFunction[A] takes triples of variables as arguments as well as triples of points .

I will often abbreviate **TransformationFunction[A][p**] by my function **fltMD[p,A]**. To see the action on all points of the projective plane I have a function fltiMD, this will take triples or quadruples, affine or projective points and test the result to see if it is affine or projective. This function does not accept variables as arguments.

```
In[ ]:= fltiMD[{1, 2, 3}, B4]
        qi = fltiMD[{1, 2, 3, 0}, B4]
Out[ ]= {-0.0502707 , -1.74485 , -1.28394}
Out[ ]= {0.267793 , -1.47664 , -0.85952}
```

```
In[ ]:= pi = fltiMD[qi, Inverse[B4]]
Out[ ]= {2.40765, -1.13159, -2.0686}
```

Note this last answer can be normalized :

```
In[ ]:= pi / pi⟦1⟧
Out[ ]= {1., -0.47, -0.859179}
```

One important fact we will use later is that these `TransformationMatrices` are themselves homoge - neous things in that we can multiply them by a non-zero constant without changing the result. For example recall above

```
In[ ]:= fltMD[{1, 1, 1}, B4]
Out[ ]= {-0.389523, -1.44758, -1.11267}
```

But replacing B4 by 3*B4

```
In[ ]:= 3 * B4 // MatrixForm
Out[ ]//MatrixForm=
```

$$\begin{pmatrix} 0.0530815 & -0.601544 & -0.410181 & 2.77583 \\ 1.69767 & 1.95791 & 2.50438 & 0.593205 \\ -0.633522 & 1.83377 & 1.53556 & 2.45497 \\ -2.62067 & -2.94403 & -0.126926 & 1.02647 \end{pmatrix}$$

```
In[ ]:= fltMD[{1, 1, 1}, 3 B4]
Out[ ]= {-0.389523, -1.44758, -1.11267}
```

Another important property of these transforms is that multiplication of transformation matrices corresponds to composition of transformation functions. Recall the transformation matrix **A1** above rotates space around the z-axis. B1 translates all points by the vector {2,-4,3}

```
In[ ]:= tr1 = fltMD[{1, 1, 1}, A1]
        fltMD[{0, Sqrt[2], 1}, B1]
        fltMD[{1, 1, 1}, B1.A1]
        fltMD[{1, 1, 1}, A1.B1]
```

Out[ ]= $\{0, \sqrt{2}, 1\}$

Out[ ]= $\{2, -4 + \sqrt{2}, 4\}$

Out[ ]= $\{2, -4 + \sqrt{2}, 4\}$

Out[ ]= $\{3\sqrt{2}, 0, 4\}$

So transforming by **A1** first and then transforming the result by **B1** gives the same result as transforming by the product **B1.A1** Notice that transforming by **A1.B1** gives a different answer because neither composition of transformations or matrix multiplication is commutative.

One consequence of this, since all our transformation matrices are assumed invertible, is that transform ing using the inverse matrix gives the inverse transform. So all these transformation functions are invertible, hence 1-1 and onto.

```
In[ ]:= fltMD[{0, Sqrt[2], 1}, Inverse[A1]]
Out[ ]= {1, 1, 1}
```

1.8.2 The group PGL(4,ℝ)

These 2 properties say the set of transformation functions with invertible transformation matrices is a *group.* (See any abstract algebra book or section 6.1 of my *plane curve book*. This particular group has been extensively studied and is known in the literature as PGL(4,ℝ), the group of invertible 4×4 matrices modulo scalar multiplication. This group of transformations is recognized as the correct group for projective geometry so if there is an invertible transformation function taking surface S1 to surface S2 we say *S1 is projectively equivalent to S2.*

I will try to make a few comments on the relationship between projective linear transformations and our Transformation Functions, it somewhat complicated but not really needed for the rest of this book.

If one is willing to stay entirely within the projective notation, 4 components, then to apply the transfor - mation given by an invertible 4×4 matrix A one just uses matrix multiplication. But remember that the input, matrix and output are all homogeneous so the answer may differ from one's expectations by a scalar multiple.

As an example consider the transformation A which just re-arranges the 4 projective vertex points V, most of which are invisible.

```
In[ ]:= V = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}};
        A = {{0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}, {1, 0, 0, 0}};
        A // MatrixForm
Out[ ]//MatrixForm=
```

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Applying A to each projective point in V we get

```
In[ ]:= A.# & /@ V
Out[ ]= {{0, 0, 0, 1}, {1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}}
```

Notice first if we denote the points in V as v_1, v_2, $v_{3,}$, v_4 then the shift is counterclockwise in that Av_1= v_4, A$v_{4=}$ $v_{3,}$ and so on.

Notice second that this shifts them as columns, not rows. This is somewhat not expected as we thing of transformations matrices from a row point of view, as the first 3 rows contain linear action in the first 3 coordinates and translations in the 4th. The last row is different as it gives denominators. When working with actual numbers (fractions or machine numbers) in our coordinates we can somewhat mimic a projective transformation which involves invisible points using our special version of a transfor-mation function fltiMD.

```
In[ ]:= fltiMD[#, A] & /@ V
Out[ ]= {{0, 0, 0}, {1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}}
```

This is almost the same except the first is reduced to an affine number since the 4th component is non-zero. But if we start with a general invisible point it may look different

```
In[ ]:= w1 = {3, 2, 1, 0}
        w2 = fltiMD[w1, A]
        w3 = fltiMD[w2, A]
        w4 = fltiMD[w3, A]
        fltiMD[w4, A]
Out[ ]= {3, 2, 1, 0}
```

Out[ ]= $\left\{\frac{2}{3}, \frac{1}{3}, 0\right\}$

Out[ ]= $\left\{\frac{1}{2}, 0, \frac{3}{2}\right\}$

```
Out[ ]= {0, 3, 2}
Out[ ]= {3, 2, 1, 0}
```

we cycle around 4 points, 3 of which are affine. In this case if we write each affine point in projective notation with the denominator being the last coordinate then we are simply permuting the coordinates counterclockwise. But if we start with a visible number

```
In[ ]:= u1 = {5, 6, 7};
        u2 = fltMD[u1, A]
        u3 = fltMD[u2, A]
        u4 = fltMD[u3, A]
        fltMD[u4, A]
```

Out[ ]= $\left\{\frac{6}{5}, \frac{7}{5}, \frac{1}{5}\right\}$

Out[ ]= $\left\{\frac{7}{6}, \frac{1}{6}, \frac{5}{6}\right\}$

Out[ ]= $\left\{\frac{1}{7}, \frac{5}{7}, \frac{6}{7}\right\}$

```
Out[ ]= {5, 6, 7}
```

we still cycle though 4 visible points but not in the expected order. The thing to remember is that fltMD and fltiMD are giving results in a different specialization of projective space in each case.

1.8.3 The push-forward operator

The natural action on a surface by an arbitrary transformation is to transform it to its inverse image under the transform. However since our transforms will be invertible if we do a fractional linear transformation from surface S1 to S2 then taking the inverse image of the inverse transformation gives us the image of a surface under the original transformation. This gives us what is called a *push forward functor* instead of just mapping points to points it maps naive surfaces to naive surfaces. The code we will use is called FLTNS which is a special case of FLT3D in my *Space Curve Book.* The full code in GlobalFunctionsS.nb, here is shortened code without checks to show how simple it is. Note that this routine involves homogenizing and specializing even though we have affine equations as input and output. In the *Space Curve Book* I have a general push-forward function for non-invertible transformation matrices, it takes a large part of Chapter 2 of that book and many subroutines to describe.

```
In[ ]:= FLTNS[f_, A_, X_] := Module[{B, d, g, h, t, n},
          B = Inverse[A].Append[X, t];
          d = tDegMD[f, X];
          g = Expand[t^d (f /. Thread[X → X / t])];
          h = Expand[g /. Thread[Append[X, t] → B]];
          Chop[h /. {t → 1}, dTol]]
```

Example : Consider the parabolic ellipsoid, the orange is original, blue is the transform.

```
In[ ]:= f = x^2 + y^2 - z;
```

```
In[ ]:= g1 = FLTNS[f, B1, {x, y, z}]
        g2 = FLTNS[f, B2, {x, y, z}]
        B3 = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, -1, -.4}, {0, 0, 1, .2}};
        B3 // MatrixForm
        g3 = FLTNS[f, B3, {x, y, z}]
```

Out[ ]= $23 - 4x + x^2 + 8y + y^2 - z$

Out[ ]= $37.5401 - 5.23631\,x + 0.461738\,x^2 + 6.60435\,y + 0.526242\,x\,y + 0.871377\,y^2 - 9.02741\,z + 0.846884\,x\,z - 0.413987\,y\,z + 0.666885\,z^2$

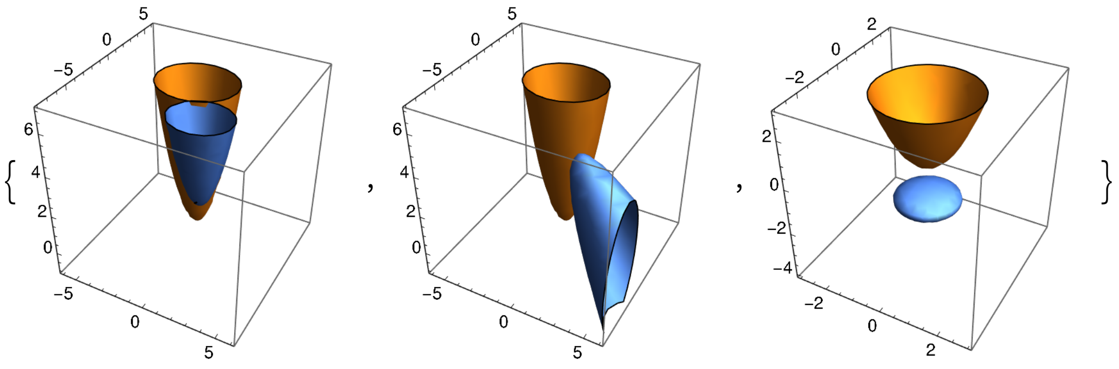
Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -0.4 \\ 0 & 0 & 1 & 0.2 \end{pmatrix}$$

Out[ ]= $10. + 1.\,x^2 + 1.\,y^2 + 15.\,z + 5.\,z^2$

In[ ]:=
```
{ContourPlot3D [{f == 0, g1 == 0}, {x, -6, 6}, {y, -8, 6}, {z, -1, 7}, Mesh → None ,
  ImageSize → Small], ContourPlot3D [{f == 0, g2 == 0}, {x, -6, 6}, {y, -8, 6},
  {z, -1, 7}, Mesh → None , ImageSize → Small], ContourPlot3D [{f == 0, g3 == 0},
  {x, -3, 3}, {y, -3, 3}, {z, -4, 3}, Mesh → None , ImageSize → Small]}
```

Out[ ]= {



}

Note that in the last plot the projective transformation takes the parabolic ellipsoid to an ellipsoid. These surfaces are projectively equivalent.

A nice feature of *Mathematica* is that the built-in **`Inverse`** function for matrices can recognize symbolic matrices in a form that is generically invertible and give the generic inverse. Here is one way we may use it : Consider the plane curve given by

In[ ]:=
```
f = -3 + 3 x - 4 y + 2 x ^ 2 + 2 y ^ 2;
```

We want to transform this to the unit circle . This is actually a 2 dimensional problem but we can consider z a free variable.

In[ ]:=
```
T = {{1, 0, 0, a}, {0, 1, 0, b}, {0, 0, 1, 0}, {0, 0, 0, d}};
g = FLTNS[f, T, {x, y, z}]
```

Out[ ]= $-\frac{3}{d^2}-\frac{3\,a}{d^2}+\frac{2\,a^2}{d^2}+\frac{4\,b}{d^2}+\frac{2\,b^2}{d^2}+\frac{3\,x}{d}-\frac{4\,a\,x}{d}+2\,x^2-\frac{4\,y}{d}-\frac{4\,b\,y}{d}+2\,y^2$

In[ ]:=
```
c0 = g /. Thread[{x, y, z} → {0, 0, 0}]
```

Out[ ]= $-\frac{3}{d^2}-\frac{3\,a}{d^2}+\frac{2\,a^2}{d^2}+\frac{4\,b}{d^2}+\frac{2\,b^2}{d^2}$

In[ ]:=
```
cx = Coefficient [g, x] /. Thread[{x, y, z} → {0, 0, 0}]
```

Out[ ]= $\frac{3}{d}-\frac{4\,a}{d}$

In[ ]:=
```
cy = Coefficient [g, y] /. Thread[{x, y, z} → {0, 0, 0}]
```

Out[ ]= $-\frac{4}{d}-\frac{4\,b}{d}$

```
In[ ]:= sol = Solve[c0 == -2 && cx == 0 && cy == 0, {a, b, c, d}, Reals]
```

```
Out[ ]= {{a → 3/4, b → -1, d → -7/4}, {a → 3/4, b → -1, d → 7/4}}
```

Picking either solution

```
In[ ]:= g1 = g /. sol⟦2⟧
```

```
Out[ ]= -2 + 2 x^2 + 2 y^2
```

which is equivalent to $-1 + x^2 + y^2$. So

```
In[ ]:= T1 = T /. sol⟦2⟧
        FLTNS[f, T1, {x, y, z}]
```

```
Out[ ]= {{1, 0, 0, 3/4}, {0, 1, 0, -1}, {0, 0, 1, 0}, {0, 0, 0, 7/4}}

Out[ ]= -2 + 2 x^2 + 2 y^2
```

1.8.4 Some important transformations .

Here are some transformation functions that will prove useful in the sequel .

1.8.4.1 Miscellaneous transformations

A set of points in ℙ^3 possibly including infinite points, is in general position if no k + 1 points lie in a k - 1 dimensional linear set . (Recall that generically k + 1 points determine a k dimensional set .) A suffi - cient condition that a set of 4 points is in general position is that the matrix with these 4 points as columns (or rows) is invertible. There is also a general position tester in **GlobalFunctionsS.nb** called gpTestMD .

It is a well known fact that there is a projective linear transformation, that is transformation in PGL[4,ℝ], which takes any 4 projective points forming set P_1 in general position to any 4 projective points forming set P_2 in general position. Working strictly projectively with 4×4 matrices as in 2.2.2 one creates a matrix B using the 4 points in P_2 as columns and a matrix A using the 4 points in P_1 as columns. then the desired transformation matrix is **`B.Inverse[A].`**

If all 8 points above are affine, and this can be made to happen by specializing at a plane containing none of them, then there is a routine to find an affine transformation matrix with a Transformation Function that sends the general position points in P_1 to the 4 points in P_2. As you will see this is not so obvious.

In[ ]:=
```
getTransformMatrix [At_, Bt_] := Module[{A, B, d, F, G, a, b, AA, BB, L},
  A = Transpose [At];
  B = Transpose [Bt];
  d = Dimensions [A]⟦1⟧;
  If[! gpTestMD[Transpose [A], d, .0003], Echo["Not  general  Position "];
   Abort[]];
  a = Take[A, All, -1];
 G = Append[Join[IdentityMatrix [d], -a, 2], Append[Table[0, {d}], 1]];
  AA = Transpose [Table[Transpose [A]⟦i⟧ - Flatten[a], {i, d}]];
  BB = Transpose [Table[Transpose [B]⟦i⟧ - Transpose [B]⟦d + 1⟧, {i, d}]];
  L = BB.Inverse[AA];
  F = Append[Join[L, Take[B, All, -1], 2], Append[Table[0, {d}], 1]];
  F.G]
```

Example :

In[ ]:=
```
At = {{1, 0, 5}, {2, -1, 4}, {2, -1, -4}, {-2, -1, -2}};
Bt = {{-1, 2, 4}, {2, -5, 4}, {5, 1, -5}, {5, 5, 4}};
```

In[ ]:=
```
M = getTransformMatrix [At, Bt];
M // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -\frac{3}{16} & -\frac{45}{16} & -\frac{3}{8} & \frac{17}{16} \\ -\frac{11}{8} & \frac{51}{8} & -\frac{3}{4} & \frac{57}{8} \\ -\frac{27}{16} & -\frac{45}{16} & \frac{9}{8} & \frac{1}{16} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

In[ ]:=
```
fltMD[#, M] & /@ At
```

Out[ ]=
```
{{-1, 2, 4}, {2, -5, 4}, {5, 1, -5}, {5, 5, 4}}
```

A transformation matrix produces a rotation if it is an affine transformation with upper left 3×3 submatrix an orthogonal matrix with determinant 1. It is a geometric fact that any rotation of $\mathbb{R}^3$ will have a fixed line as an axis. Mathematica differentiate RotationTransform from RotationMatrix in that in the first case one gets the 4×4 transformation matrix we have been discussing, while RotationMatrix is just the upper left 3×3 matrix. However in either case the built-in rotations are just linear transformations so the origin {0,0,0} is always on the axis. Thus for our use we need a more general construction. Here we will pick two affine intersecting planes and do an orthogonal rotation from one plane to the other. In particular we translate a point on the intersection line two the origin, use a built-in rotation matrix and then translate back. The planes are given by their affine equations in variables x,y,z.

```
In[ ]:= planeRotate3D[plane1_, plane2_] :=
         Module[{p, A1, nplane1, nplane2, w, v, M, A2, nullarr},
          pi = FindInstance[plane1 == 0 && plane2 == 0 &&
              Abs[x] < 8 && Abs[y] < 8 && Abs[z] < 8, {x, y, z}, Reals][[1]];
          If[Length[pi] < 3, Echo["no affine intersection"]; Abort[]];
          p = N[{x, y, z} /. pi];
          A1 = {{1, 0, 0, -p[[1]]}, {0, 1, 0, -p[[2]]}, {0, 0, 1, -p[[3]]}, {0, 0, 0, 1}};
          nplane1 = FLTNS[plane1, A1, {x, y, z}];
          nplane2 = FLTNS[plane2, A1, {x, y, z}];
          v = Grad[nplane1, {x, y, z}];
          w = Grad[nplane2, {x, y, z}];
          M = N[RotationMatrix[{v, w}]];
          A2 = Append[Join[M, {{0}, {0}, {0}}, 2], {0, 0, 0, 1}];
          Inverse[A1].A2.A1]
```

Simple Example, note that even exact input will almost always give numerical output. Also note that this generally gives an affine matrix as output, but never projective.

```
In[ ]:= R1 = planeRotate3D[x - 3, y - 2 z + 3];
        R1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0. & -0.447214 & 0.894427 & 1.65836 \\ 0.447214 & 0.8 & 0.4 & -1.94164 \\ -0.894427 & 0.4 & 0.2 & 3.88328 \\ 0. & 0. & 0. & 1. \end{pmatrix}$$

```
In[ ]:= planeOut = FLTNS[x - 3, R1, {x, y, z}]
Out[ ]= 1.34164 + 0.447214 y - 0.894427 z
```

```
In[ ]:= Expand[planeOut * 3 / planeOut[[1]]]
Out[ ]= 3. + 1. y - 2. z
```

To see that the intersecting line given by the two plane equations is fixed

```
In[ ]:= fixedPt = SolveValues[x == 3 && y - 2 z == -3 && y == RandomInteger[{-5, 5}], {x, y, z}][[1]]
```

Out[ ]= $\left\{3, 2, \frac{5}{2}\right\}$

```
In[ ]:= TransformationFunction[R1][fixedPt]
Out[ ]= {3., 2., 2.5}
```

1.8.5 iTransform

Our most important transform is a projective transform which specializes at a plane, that is makes that plane invisible, but makes the original plane visible as the plane $x + y + z = 1$. One can choose the location of the new visible plane by an additional transformation but I find that this choice works best in most cases. The out put is the transformation matrix, but combining this with FLTNS and Countour - Plot3D make the previously invisible curve visible. The user must specify a plane to specialize but that could be random but must, for technical reasons, intersect the plane $x + y + z = 2$. In particular it cannot be $x + y + z = 1$. Intuitively the choice should be far away from any interesting part of a surface you are working with, but numerically it works best if the coordinates are smallish numbers.

```
In[ ]:= iTransform3D[plane_] := Module[{A1, A2, G},
          G = Grad[plane, {x, y, z}];
          If[G⟦1⟧ == G⟦2⟧ && G⟦2⟧ == G⟦3⟧, Echo["Illegal Plane"]; Abort[];];
          A1 = planeRotate3D[plane, x + y + z - 2];
          A2 = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {1, 1, 1, -2}};
          Chop[A2.A1]]
```

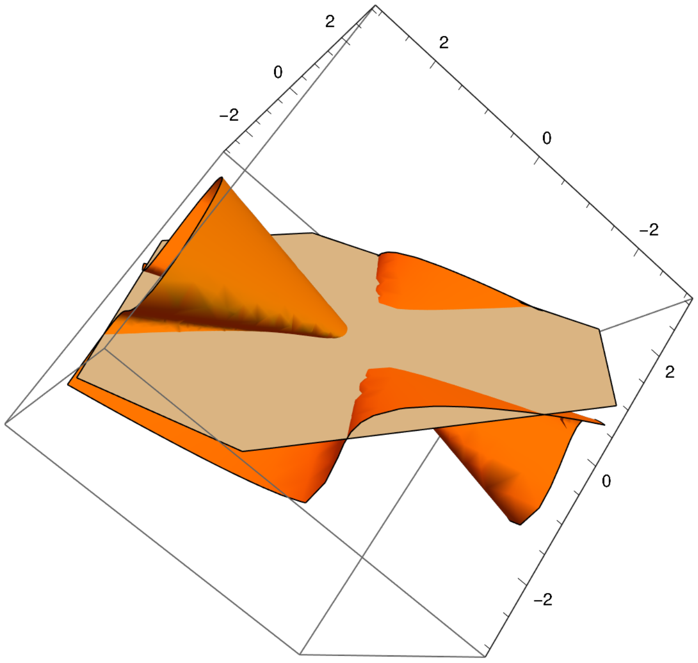
As an example a nice such transform is

```
In[ ]:= Ai = iTransform3D[z - 4]
Out[ ]= {{0.788675, -0.211325, 0.57735, -2.73205}, {-0.211325, 0.788675, 0.57735, -2.73205},
         {-0.57735, -0.57735, 0.57735, 0.535898}, {0, 0, 1.73205, -6.9282}}

In[ ]:= clebsch = 81 (x^3 + y^3 + z^3) - 189 (x^2 y + x^2 z + y^2 x + y^2 z + z^2 x + z^2 y) +
           54 x y z + 126 (x y + x z + y z) - 9 (x^2 + y^2 + z^2) - 9 (x + y + z) - 1;

In[ ]:= fi = FLTNS[clebsch, Ai, {x, y, z}]
```

Out[ ]= $879.578 - 3605.64\,x + 4890.55\,x^2 - 2206.05\,x^3 - 3605.64\,y + 9909.58\,x\,y - 6746.65\,x^2\,y + 4890.55\,y^2 - 6746.65\,x\,y^2 - 2206.05\,y^3 - 1930.07\,z + 5204.67\,x\,z - 3654.95\,x^2\,z + 5204.67\,y\,z - 6690.14\,x\,y\,z - 3654.95\,y^2\,z + 1045.68\,z^2 - 1329.77\,x\,z^2 - 1329.77\,y\,z^2 + 72.3627\,z^3$

```
In[ ]:= ContourPlot3D [{fi == 0, x + y + z == 1}, {x, -3, 3}, {y, -3, 3}, {z, -3, 3},
          Mesh → None, ContourStyle → {Orange, LightGray}, MaxRecursion → 5]
```

Out[ ]=



Here are three variants which allow us to specialize in each affine coordinate direction. They are given as constants.

```
In[ ]:= rrtxyz = 4.325098211016204`

Out[ ]= 4.3251

In[ ]:= ixTransform3D = Chop[planeRotate3D [x + y + z - 1, x].iTransform3D [x + rrtxyz]]
        iyTransform3D = Chop[planeRotate3D [x + y + z - 1, y].iTransform3D [y + rrtxyz]]
        izTransform3D = Chop[planeRotate3D [x + y + z - 1, z].iTransform3D [z + rrtxyz]]

Out[ ]= {{0, 0, 0, 1.1547}, {0.366025 , 1., 0, 2.74354},
         {0.366025 , 0, 1., 2.74354}, {1.73205 , 0, 0, 7.49129}}

Out[ ]= {{1., 0.366025 , 0, 2.74354}, {0, 0, 0, 1.1547},
         {0, 0.366025 , 1., 2.74354}, {0, 1.73205 , 0, 7.49129}}

Out[ ]= {{1., 0, 0.366025 , 2.74354}, {0, 1., 0.366025 , 2.74354},
         {0, 0, 0, 1.1547}, {0, 0, 1.73205 , 7.49129}}
```

```
In[ ]:= clbx = FLTNS[clebsch, ixTransform3D, {x, y, z}]
        clby = FLTNS[clebsch, iyTransform3D, {x, y, z}]
        clbz = FLTNS[clebsch, izTransform3D, {x, y, z}]
```

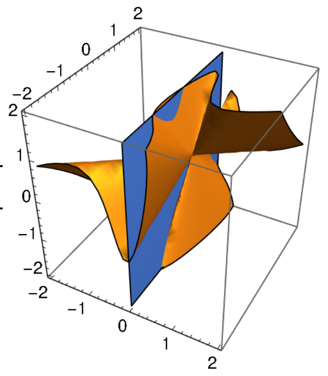
Out[ ]= $35.9 - 543.486\ x + 2029.39\ x^2 + 3362.81\ x^3 - 37.9385\ y + 654.589\ x\ y - 4707.97\ x^2\ y - 120.531\ y^2 + 645.864\ x\ y^2 + 81.\ y^3 - 37.9385\ z + 654.589\ x\ z - 4707.97\ x^2\ z + 190.939\ y\ z + 666.616\ x\ y\ z - 189.\ y^2\ z - 120.531\ z^2 + 645.864\ x\ z^2 - 189.\ y\ z^2 + 81.\ z^3$

Out[ ]= $35.9 - 37.9385\ x - 120.531\ x^2 + 81.\ x^3 - 543.486\ y + 654.589\ x\ y + 645.864\ x^2\ y + 2029.39\ y^2 - 4707.97\ x\ y^2 + 3362.81\ y^3 - 37.9385\ z + 190.939\ x\ z - 189.\ x^2\ z + 654.589\ y\ z + 666.616\ x\ y\ z - 4707.97\ y^2\ z - 120.531\ z^2 - 189.\ x\ z^2 + 645.864\ y\ z^2 + 81.\ z^3$

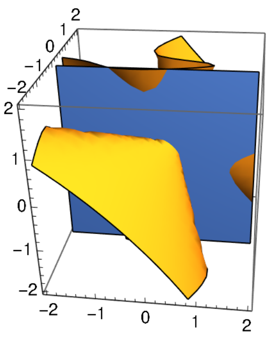
Out[ ]= $35.9 - 37.9385\ x - 120.531\ x^2 + 81.\ x^3 - 37.9385\ y + 190.939\ x\ y - 189.\ x^2\ y - 120.531\ y^2 - 189.\ x\ y^2 + 81.\ y^3 - 543.486\ z + 654.589\ x\ z + 645.864\ x^2\ z + 654.589\ y\ z + 666.616\ x\ y\ z + 645.864\ y^2\ z + 2029.39\ z^2 - 4707.97\ x\ z^2 - 4707.97\ y\ z^2 + 3362.81\ z^3$

```
In[ ]:= {ContourPlot3D[{clbx == 0, x == 0}, {x, -2, 2}, {y, -2, 2}, {z, -2, 2}, Mesh → None,
          MaxRecursion → 4], ContourPlot3D[{clby == 0, y == 0}, {x, -2, 2}, {y, -2, 2},
          {z, -2, 2}, Mesh → None, MaxRecursion → 4], ContourPlot3D[{clbz == 0, z == 0},
          {x, -2, 2}, {y, -2, 2}, {z, -2, 2}, Mesh → None, MaxRecursion → 4]}
```

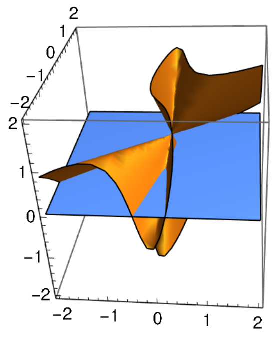
Out[ ]= {



,



,



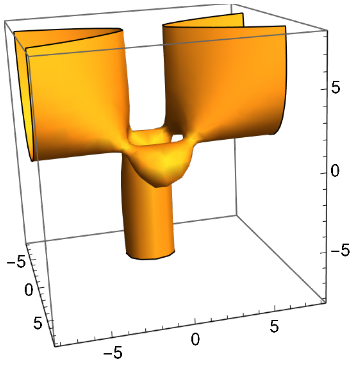
}

This is 3 different views of the same surface but different specializations, in each case the blue plane is the same infinite plane. However the invisible points on each specialization are visible in the others. In this example these look much the same because the Clebsch surface is symmetric in the variables {x,y,z}. In general they can be quite different. Here is an example from Chapter 1.

```
In[ ]:= ts3 = 10.75200000000001` - 6.39999999999992` x - 11.463999999999935` x^2 +
          0.640000000000003` x^3 + 0.9999999999999938` x^4 + 1.5360000000000196` y^2 +
          0.6400000000000121` x y^2 + 0.9999999999999906` x^2 y^2 + 2.879999999999961` x^2 z -
          5.1200000000000205` y^2 z + 3.5839999999999983` z^2 + 3.8400000000000007` x z^2 + 1.` x^2 z^2
```

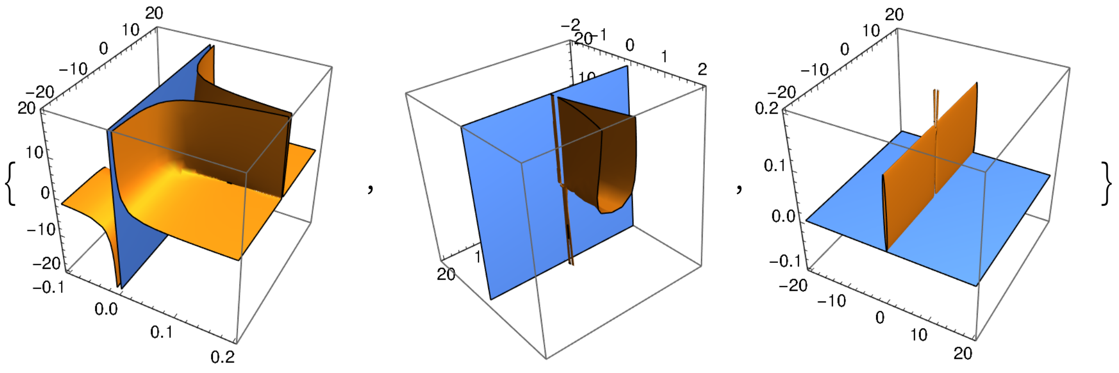
Out[ ]= $10.752 - 6.4\ x - 11.464\ x^2 + 0.64\ x^3 + 1.\ x^4 + 1.536\ y^2 + 0.64\ x\ y^2 + 1.\ x^2\ y^2 + 2.88\ x^2\ z - 5.12\ y^2\ z + 3.584\ z^2 + 3.84\ x\ z^2 + 1.\ x^2\ z^2$

In[ ]:=
```
ContourPlot3D[ts3 == 0, {x, -8, 8}, {y, -8, 8}, {z, -8, 8}, Mesh → None, ImageSize → Small]
```

Out[ ]=


In[ ]:=
```
ts3x = FLTNS[ts3, ixTransform3D, {x, y, z}];
ts3y = FLTNS[ts3, iyTransform3D, {x, y, z}];
ts3z = FLTNS[ts3, izTransform3D, {x, y, z}];
```

In[ ]:=
```
{ContourPlot3D[{ts3x == 0, x == 0}, {x, -.1, .2},
   {y, -20, 20}, {z, -20, 20}, Mesh → None, MaxRecursion → 4],
  ContourPlot3D[{ts3y == 0, y == 0}, {x, -20, 20}, {y, -2, 2}, {z, -20, 20},
   Mesh → None, MaxRecursion → 4], ContourPlot3D[{ts3z == 0, z == 0},
   {x, -20, 20}, {y, -20, 20}, {z, -.1, .2}, Mesh → None, MaxRecursion → 4]}
```

Out[ ]=


1.9Projective Surfaces

I will continue with naive and parametric surfaces but now these surfaces will be projective surfaces in projective real 3 space $\mathbb{RP}^3$. Unless otherwise noted $\mathbb{RP}^3$ will just be denoted $\mathbb{P}^3$ in this chapter. For reasons outlined below one should use the GlobalFunctionsNS.nb dated June 2022 or later rather than earlier versions of Global Functions.

1.9.1 Projective 3 - space

As I have done in my *Plane Curve Book* and *Space Curve Book* I will write affine points as triples, $p = \{a, b, c\}$ and infinite points as quadruples $\{a, b, c, 0\}$. Alternately I may write the affine point as a projective point $p = \{a, b, c, 1\}$. Two affine points are equal if they have the same components but the projective points as quadruples are homogeneous in the sense that $\{a, b, c, d\} = r\,\{a, b, c, d\} = \{r\,a,\ r\,b, r\,c, r\,d\}$ for any non-zero real number. Note that $\{0, 0, 0, 0\}$ is not a projective point, at least one component must be non-zero.

In my previous books I considered infinite points as directions in a general sense, an arrow with no base point or length which could have arrow head on either end or both. Parallel arrows were equivalent. In this book, however, I want to consider infinite points as no different from other points, just points we can't see on ordinary 2D or 3D graphics. So I will call a point of the form $\{a, b, c, 0\}$ an *invisible point.*

The set of invisible points form a *projective plane* in the sense of my plane curve book. It should be noticed that unlike the surfaces in Chapter 1 this is a one sided, non-orientable, surface. If we use coordinates $\{x, y, z, w\}$ for general points then this plane is given by $w = 0$. Actually any plane in $\mathbb{P}^3$ given by an equation $a\,x + b\,y + c\,z + d\,w = 0$ is a copy of the projective plane and its complement is a copy of the ordinary affine 3-space $\mathbb{R}^3$ and will be called a *specialization* of $\mathbb{P}^{3.}$ Unfortunately there is no standard way to give affine coordinates unless it is the complement of the plane $w = 0$. For this reason we will not use specializations in general directly, rather we will specify them by transformations of ordinary $\mathbb{R}^3$ via my FLT's in the next section. Among other reasons this will allow us to directly use Mathematica algorithms designed for 3D, either the built-in ones or ones in my Global Functions notebooks marked 3D or MD. Thus rather than work directly in $\mathbb{P}^3$as is often done in the literature, for example the book by Joe Harris. As we have seen in the space curve book Harris can do nice theory but his book is short of computations and examples.

Because of the homogeneity of projective points in $\mathbb{P}^3$ in order to define projective algebraic subsets we need to use *homogeneous equations.* These are sums of monomials in X, Y, Z and W or other conve - nient letters, all monomials of the same degree. Oftentimes we will get lazy as in the previous para - graph and just use lower case letters but the upper case letters will emphasize that these variables, or more precisely the homogeneous polynomial they denote, are homogeneous. This means that an equation such as

$$W^2 + 2\,W\,X - 5\,X^2 + 3\,W\,Y - 6\,X\,Y + 7\,Y^2 - 4\,W\,Z + 8\,X\,Z + 9\,Y\,Z + 10\,Z^2 = 0$$

is well defined as multiplying each variable by $r \neq 0$ not change the validity of this equation. But note that we may not evaluate an expression such as the left hand side of the above equation at an arbitrary projective point because if the result is not zero then it will depend on r. In particular algebraic sets defined by homogeneous equations do not have a positive or negative side unlike the affine case. Thus surfaces defined by a homogeneous equation may be one-sided, such as the invisible plane $W = 0$.

As in my *Plane Curve Book* given an affine polynomial equation we can homogenize it to get a homoge - neous equation. While the function homogMD still exists in my Global Functions notebook the follow - ing version is preferable as the capitalized variables remind one that we are working homogeneously. *In this book we will change our syntax of functions identified by suffix* `NS` *so the set of variables will be* *denoted v or V to distinguish from the homogeneous variable X. This change is actuated in* ***GlobalFunction*** ***sNS.nb****.*

```
In[ ]:= HomogNS[f_, v_, V_] := Module[{h, fass, hrl, deg},
          If[Length[V] ≠ Length[v] + 1, Echo["List X must have length one more than list x"];
           Abort[]];
          fass = Association[CoefficientRules[f, v]];
          deg = Max[Total[#] & /@ Keys[fass]];
          hrl = Table[Append[k, deg - Total[k]] → fass[k], {k, Keys[fass]}];
          FromCoefficientRules[hrl, V]]

In[ ]:= f = 1 + 2 x + 3 y - 4 z - 5 x ^ 2 - 6 x y + 7 y ^ 2 + 8 x z + 9 y z + 10 z ^ 2;

In[ ]:= hf = HomogNS[f, {x, y, z}, {X, Y, Z, W}]
Out[ ]= W^2 + 2 W X - 5 X^2 + 3 W Y - 6 X Y + 7 Y^2 - 4 W Z + 8 X Z + 9 Y Z + 10 Z^2
```

We still have problems with affine equations that are already homogeneous.

```
In[ ]:= HomogNS[x ^ 2 + y ^ 2 - z ^ 2, {x, y, z}, {X, Y, Z, W}]
Out[ ]= X^2 + Y^2 - Z^2
```

as this looks like a curve in $\mathbb{R}^2$ rather than a surface in $\mathbb{P}^3$ although the capitalized variables indicate they should be regarded as homogeneous and defined only up to a constant multiple. Again using FLT so as to continue working affinely is the best solution.

An affine surface is imbedded in its homogenization by the map $\{x, y, z\} \mapsto \{X, Y, Z, 1\}$

Consider the somewhat random point

```
In[ ]:= p = {2.`, 3.764709339686058`, -2.773802258655994`}
Out[ ]= {2., 3.76471, -2.7738}

In[ ]:= f /. Thread[{x, y, z} → p]
Out[ ]= 0.

In[ ]:= hf /. Thread[{X, Y, Z, W} → Append[p, 1]]
Out[ ]= 0.
```

1.9.2 Specialization and Invisible Points

Homogenization introduces new points with last coordinate zero. But projective space is homoge - neous in the sense that all points are the same, further, as a topological space projective space is compact. Therefore it is in some ways not appropriate to call the new points *infinite.* So I adopt the notation *invisible* points as these new points do not show up on a contour plot.

The opposite of homogenization is specialization. We can declare any plane $aX + bY + cZ + dW = 0$ in projective space to be the set of invisible points by removing them. A copy of affine $\mathbb{R}^3$ remains. The default specialization is simply $W=0$ leaving the original affine space.

If, however, we have a projective surface given by a homogeneous equation in {X,Y,Z,W}, that plane we remove generally has intersected the projective surface in a plane curve, possibly in a finite or even empty set. This I call the *invisible curve.* One wants to identify this curve, or point set. It is important to note that this is still a curve or point set in projective space. One way to think of it as being in affine space is to normalize, and one way to do this is to declare aX+bY+cZ+dW=1

I take, as my example, the hyperboloid

```
In[ ]:= h = x^2 + y^2 - z^2 - 1;
```

Homogenizing we get

```
In[ ]:= H = HomogNS[h, {x, y, z}, {X, Y, Z, W}]
```

Out[ ]= $-W^2+X^2+Y^2-Z^2$

So the infinite curve is

```
In[ ]:= K = H /. {W → 0}
```

Out[ ]= $X^2+Y^2-Z^2$

This may look like the equation of a cone but the point {1,0,1,0} is the same as {2,0,2,0} etc. One way to convert this equation to something that looks like an affine curve is to normalize. We can normalize by setting any fixed but arbitrary homogeneous equation, other than the one we are considering, to 1. In this example the easiest is to set Z=1. So we have the affine equation $x^2+y^2=1$. Actually what we really have is a correspondence

```
{x, y}⟺{X, Y, 1}
```

from points on the affine curve $x^2+y^2=1$ to homogeneous points {X,Y,1} on $\left\{X^2+Y^2-Z^2=0,\ Z=1\right\}$ In this way we may think of the infinite curve of the hyperboloid as a circle.

1.9.3 Infinite Points and curves

We may notice that the equation we get for the infinite curve of a naive surface is, with the possible exception of using different variable names for affine and projective equations, simply the maximal form of the affine equation. For example

```
In[ ]:= maxFormMD[x^2 + y^2 - z^2 - 1, {x, y, z}]
```

Out[ ]= $x^2+y^2-z^2$

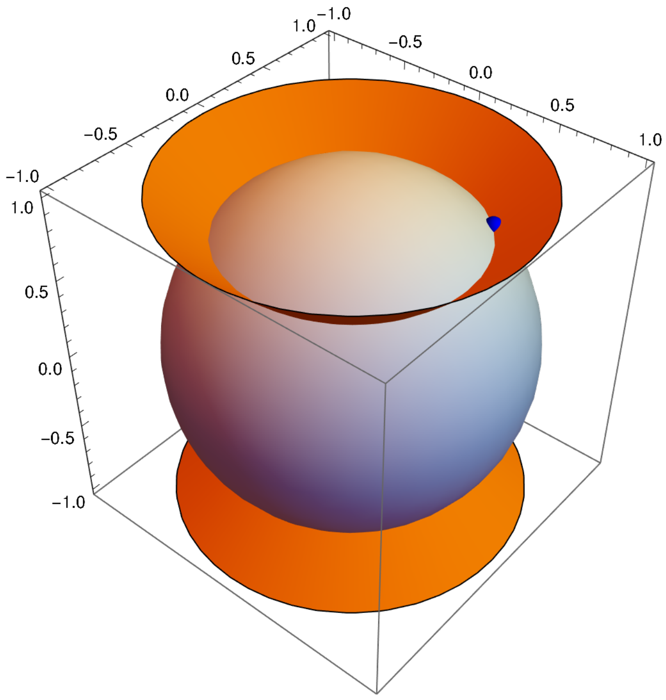
This is homogeneous so we can take one representative for a homogeneous point and normalize it someway, append a zero to get a 4-tuple and call this an *infinite point* as I did for my previous books. An easy way to normalize is use the Mathematica function Normalize. This is equivalent to homoge-neous point on the maximal form as a line and intersecting the line with the unit sphere in $\mathbb{R}^3$, but we get two antipodal points this way. This leads to a nice way to illustrate the infinite curve of an naive affine surface: Plot the maximal form and the unit sphere in the same plot. The infinite curve is then the intersection if one mentally associates each point with its antipodal point. For example the infinite curve of the standard hyperboloid can be shown by

```
In[ ]:= p = Normalize[{1, 1, Sqrt[2]}]
```

Out[ ]= $\left\{\frac{1}{2}, \frac{1}{2}, \frac{1}{\sqrt{2}}\right\}$

```
In[ ]:= Show[ContourPlot3D[{x^2 + y^2 - z^2 == 0, x^2 + y^2 + z^2 == 1},
          {x, -1, 1}, {y, -1, 1}, {z, -1, 1}, Mesh → None,
          ContourStyle → {Orange, LightGray}], Graphics3D[{Blue, Ball[p, .05]}]]
```

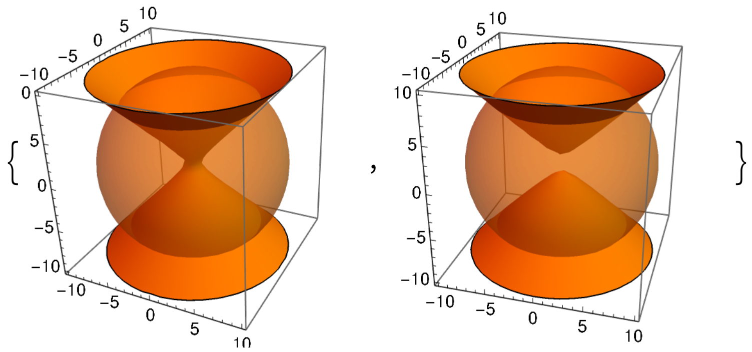
Out[ ]=



Thus the infinite curve is shown twice, once as the circle of intersection of the two surfaces at the top of the plot and with a copy at the bottom. Notice the infinite point indicated at the blue dot is properly written as $\left\{\frac{1}{2}, \frac{1}{2}, \frac{1}{\sqrt{2}}, 0\right\}$ using this formulation.

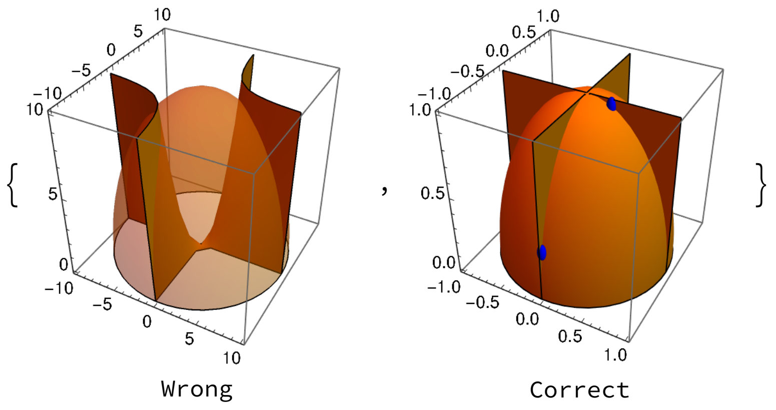
The hyperbolic ellipsoid $x^2 + y^2 - z^2 + 1 = 0$ has the same maximal form so we would get the exact same picture. We can do, by homogeneity, the following to distinguish these plots since the maximal forms differ from the equations by only a small constant.

```
In[ ]:= {ContourPlot3D [{x ^ 2 + y ^ 2 - z ^ 2 == 1, x ^ 2 + y ^ 2 + z ^ 2 == 100},
          {x, -10, 10}, {y, -10, 10}, {z, -10, 10}, Mesh → None,
          ContourStyle → {Orange, Directive [Orange, Opacity[.5]], Gray}],
        ContourPlot3D [{x ^ 2 + y ^ 2 - z ^ 2 == -1, x ^ 2 + y ^ 2 + z ^ 2 == 100},
          {x, -10, 10}, {y, -10, 10}, {z, -10, 10}, Mesh → None,
          ContourStyle → {Orange, Directive [Orange, Opacity[.5]], Gray}]}
```

Out[ ]=


Some may prefer the hemisphere plots of the plane curve book, then the only points which appear twice are those on the xy-plane. Here we illustrate with the saddle surface *z* = xy (see Chapter 2) which has maximal form x y. Here because the maximal form differs by a linear function the plot (left) includ - ing the actual function does not make sense

```
In[ ]:= {Labeled[ContourPlot3D [{z - x y == 0, x ^ 2 + y ^ 2 + z ^ 2 == 100},
           {x, -10, 10}, {y, -10, 10}, {z, 0, 10}, Mesh → None,
           ContourStyle → {Orange, Directive [Orange, Opacity[.5]], Gray}], "Wrong"],
         Labeled[Show[ContourPlot3D [{x y == 0, x ^ 2 + y ^ 2 + z ^ 2 == 1}, {x, -1, 1}, {y, -1, 1},
            {z, 0, 1}, Mesh → None, ContourStyle → {Orange, Directive [Orange, Opacity[1]], Gray}],
           Graphics3D [{Blue, Ball[{0.274721, 0., 0.961524}, .05],
              Ball[{0.`, -0.9486832980505138`, 0.31622776601683794`}, .05]}]], "Correct"]}
```

Out[ ]=


When normalizing using the sphere a great circle represents a line. So in this case the infinite curve is equivalent to the homogeneous curve *X Y*. Note a typical point on the surface *x y* = 0 might be {2,0,7} or {0,-3,1) so they would normalize q1= ={0.274721,0.,0.961524} or q2 ={0.,-0.948683,0.316228} ans so as infinite points would be {0.274721,0.,0.961524,0} or {0.,0.948683,0.316228,0}.

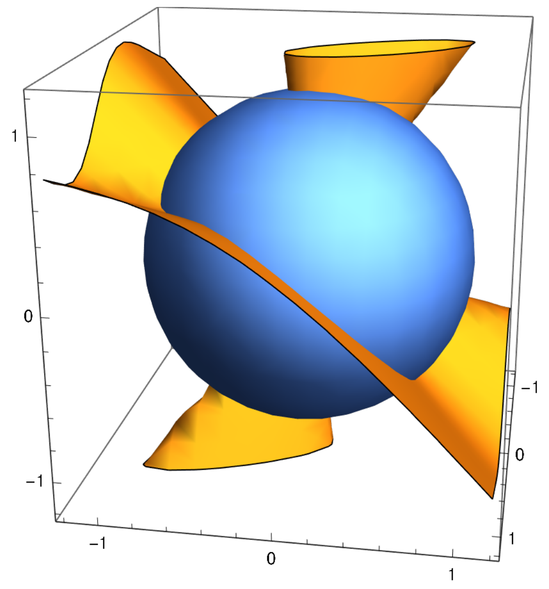
The infinite curve can contain more than one connected component. Here is an example

```
In[ ]:= f1 = -0.9332667921277373` - 0.54659406197207` x + 1.6228493544271605` x^2 -
          0.6266424651018005` x^3 - 0.15904113742115739` y - 0.15141560817564875` x y +
          1.7516424651018003` x^2 y - 0.5139019915209658` y^2 + 2.3341742198323487` x y^2 +
          0.9558892896287468` y^3 + 0.08847551097802434` z - 0.7608711515528794` x z +
          2.763901991520963` x^2 z - 0.9825589880399126` y z + 2.728765877365271` x y z +
          1.0139019915209662` y^2 z - 0.16582578016764984` z^2 + 0.8521234122634711` x z^2 +
          0.019591657532924106` y z^2 + 0.010617061317362703` z^3 - 1

Out[ ]= -1.93327 - 0.546594 x + 1.62285 x^2 - 0.626642 x^3 - 0.159041 y -
         0.151416 x y + 1.75164 x^2 y - 0.513902 y^2 + 2.33417 x y^2 + 0.955889 y^3 +
         0.0884755 z - 0.760871 x z + 2.7639 x^2 z - 0.982559 y z + 2.72877 x y z +
         1.0139 y^2 z - 0.165826 z^2 + 0.852123 x z^2 + 0.0195917 y z^2 + 0.0106171 z^3
```

The infinite curve is

```
In[ ]:= f1i = HomogNS[f1, {x, y, z}, {X, Y, Z, W}] /. {W → 0}

Out[ ]= 0. - 0.626642 X^3 + 1.75164 X^2 Y + 2.33417 X Y^2 + 0.955889 Y^3 + 2.7639 X^2 Z +
         2.72877 X Y Z + 1.0139 Y^2 Z + 0.852123 X Z^2 + 0.0195917 Y Z^2 + 0.0106171 Z^3
```

```
In[ ]:= ContourPlot3D[{f1i == 0, X^2 + Y^2 + Z^2 == 1},
          {X, -1.2, 1.2}, {Y, -1.2, 1.2}, {Z, -1.2, 1.2}, Mesh → None]

Out[ ]=
```



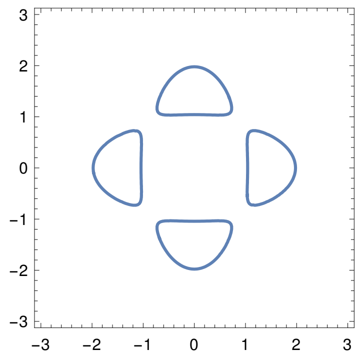
Since this is a real projective plane curve then Harnack's theorem [See Plane Curve Book, Chapter 9] applies. If the degree of this curve is d then the number of topological components in the projective plane must be less than or equal to **compd** $= (d - 1)(d - 2)/2 + 1$. Harnack himself noted that plane curves of degree d with **compd** components do exist, these are called M-curves. Here is how to make a naive surface of degree d such that the infinite curve has compd components if one knows an M-curve for this degree [See discussion in Section 9.2 of Curve Book].

We start with a plane curve *f* of degree *d* with **compd** components. We get a naive surface *g* by applying HomogNS to *f*, note the plane $z-1$ intersects this surface in *f*. Apply the iTransform[z-1] to get a curve with this infinite curve. Here is an example.

Start with the plane curve

```
In[ ]:= f2 = 17 - 20 x^2 + 4 x^4 - 20 y^2 + 17 x^2 y^2 + 4 y^4;
```

```
In[ ]:= ContourPlot[f2 == 0, {x, -3, 3}, {y, -3, 3}, ImageSize → Small]
```

Out[ ]=


This curve has 4 components . Set

```
In[ ]:= F2 = HomogNS[f2, {x, y}, {x, y, z}]
```

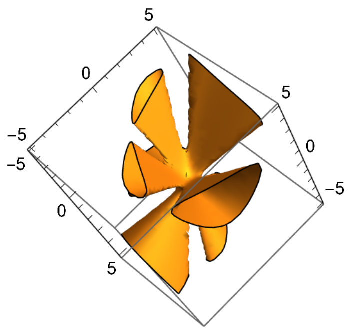
Out[ ]= $4 x^4 + 17 x^2 y^2 + 4 y^4 - 20 x^2 z^2 - 20 y^2 z^2 + 17 z^4$

Now we transform it, and to eliminate the singular point subtract 1. This is our example

```
In[ ]:= F2i = FLTNS[F2, iTransform3D[z - 1], {x, y, z}] - 1
```

Out[ ]= $-0.432084 - 4.52473 x + 11.5184 x^2 - 10.6228 x^3 + 2.53341 x^4 - 4.52473 y + 21.0991 x y - 21.2329 x^2 y + 1.10256 x^3 y + 11.5184 y^2 - 21.2329 x y^2 + 22.1383 x^2 y^2 - 10.6228 y^3 + 1.10256 x y^3 + 2.53341 y^4 + 0.622306 z - 7.35632 x z + 20.4645 x^2 z - 14.2865 x^3 z - 7.35632 y z + 42.0023 x y z - 19.3594 x^2 y z + 20.4645 y^2 z - 19.3594 x y^2 z - 14.2865 y^3 z - 1.0758 z^2 + 2.43598 x z^2 + 6.33598 x^2 z^2 + 2.43598 y z^2 + 22.203 x y z^2 + 6.33598 y^2 z^2 - 0.647809 z^3 + 5.66146 x z^3 + 5.66146 y z^3 + 0.755609 z^4$

```
In[ ]:= ContourPlot3D[F2i == 0, {x, -5, 5}, {y, -5, 5}, {z, -5, 5}, Mesh → None, ImageSize → Small]
```

Out[ ]=


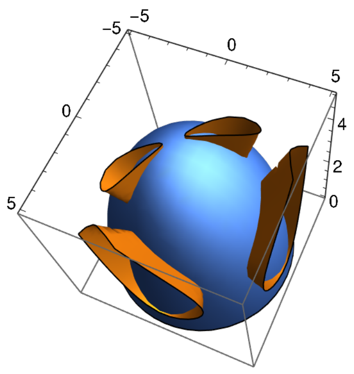
The infinite curve is

```
In[ ]:= F2ic = HomogNS[F2i, {x, y, z}, {X, Y, Z, W}] /. {W → 0}
```

```
Out[ ]= 0. + 2.53341 X^4 + 1.10256 X^3 Y + 22.1383 X^2 Y^2 + 1.10256 X Y^3 + 2.53341 Y^4 -
         14.2865 X^3 Z - 19.3594 X^2 Y Z - 19.3594 X Y^2 Z - 14.2865 Y^3 Z + 6.33598 X^2 Z^2 +
         22.203 X Y Z^2 + 6.33598 Y^2 Z^2 + 5.66146 X Z^3 + 5.66146 Y Z^3 + 0.755609 Z^4
```

We more clearly see the components in the hemisphere plot

```
In[ ]:= ContourPlot3D[{F2ic == 0, X^2 + Y^2 + Z^2 == 25},
         {X, -5, 5}, {Y, -5, 5}, {Z, 0, 5}, Mesh → None, ImageSize → Small]
```

Out[ ]=


1.9.4 Orientable and non orientable surfaces

As we pointed out in Section 1.2 above to even talk about being orientation , one or two sided, we need a smooth surface, otherwise there could be many “sides”. A necessary condition for being orientable is that each projective line, not lying in the surface, must meet the surface in an even number of projec - tive points counted by multiplicity. In particular, a naive surface of affine degree *d* will be orientable if *d* is even, for example quadric surfaces, and one-sided if *d* is odd, for example cubic surfaces. Some care is needed for parametrically defined surfaces.

1.9.5 Parametric curves and Surfaces in Projective Space

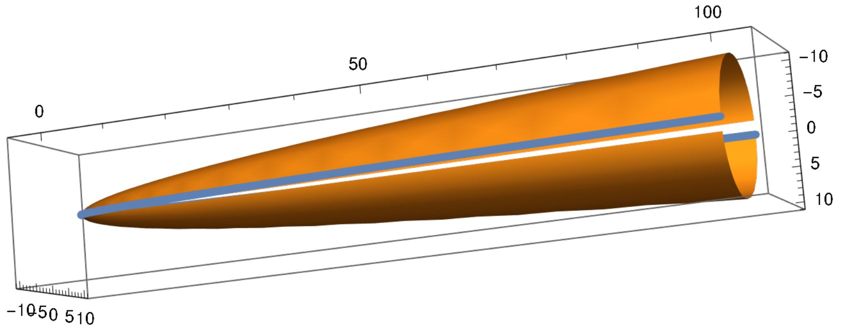
Given a rational parametric surface there are two issues going to projective space. The first is a missing region where the parameters, say *s*, *t*, go to infinity. One solution is to just plot with a large range for the variables. We will look at several examples

The paraboloid is given by (see Chapter 2)

In[ ]:= parParab = $\left\{\frac{2\,t}{1+t\char`^2}\,s,\ \frac{1-t\char`^2}{1+t\char`^2}\,s,\ s\char`^2\right\}$;

In[ ]:=
```
Show[ParametricPlot3D [parParab , {t, 0, 10}, {s, -10, 10}, Mesh → None],
  ParametricPlot3D [{0, t, t^2}, {t, -10, 10}, PlotStyle → Thickness [.01]]]
```

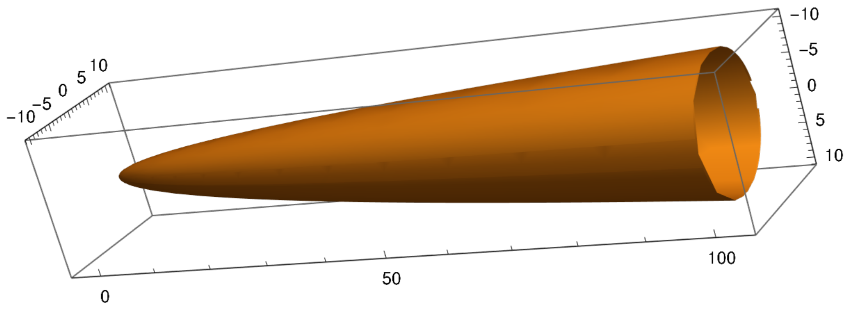
Out[ ]=



The plot shows the missing curve appears to be the parametric parabola {0, *t*, *t*^2}.

In[ ]:=
```
ParametricPlot3D [parParab , {t, -10, 10}, {s, -10, 10}, Mesh → None]
```

Out[ ]=



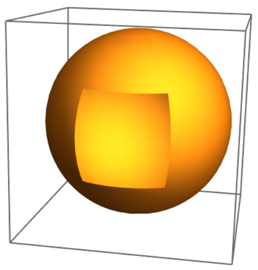
However using negative values of t this parameterization is not 1-1 so the missing region disappears.

Another example is the sphere with parameterization

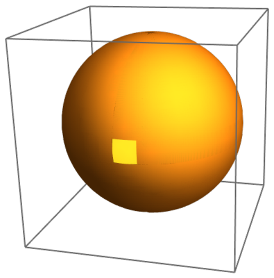
In[ ]:= parSphere = $\left\{\frac{(1-t^2)\times 2\,s}{(1+t^2)\times(1+s^2)}, \frac{(1-t^2)\times(1-s^2)}{(1+t^2)\times(1+s^2)}, \frac{2\,t\,(1+s^2)}{(1+t^2)\times(1+s^2)}\right\}$;

In[ ]:=
```
{ParametricPlot3D [parSphere , {t, -4, 4}, {s, -4, 4}, Mesh → None,
  MaxRecursion  → 5, Axes → False], ParametricPlot3D [parSphere ,
  {t, -16, 16}, {s, -16, 16}, Mesh → None,  MaxRecursion  → 5, Axes → False]}
```

Out[ ]= {



,



}

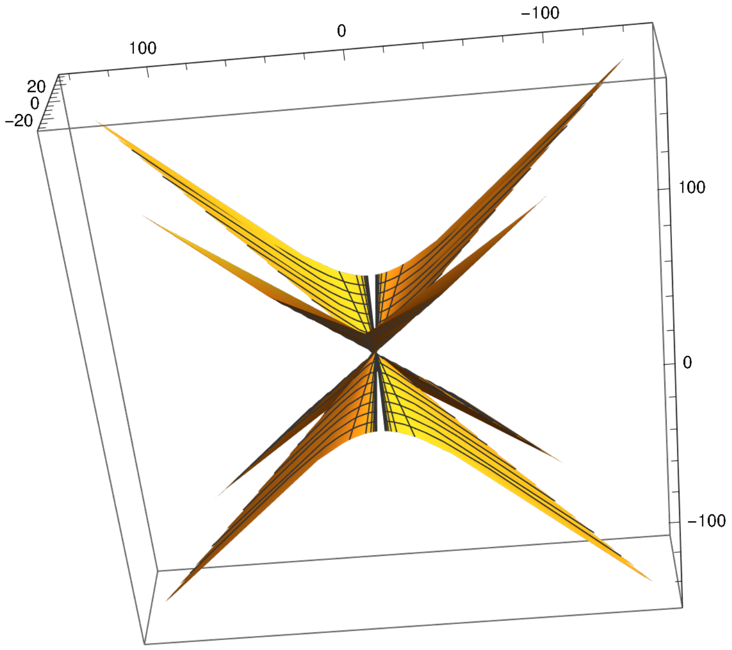
Here is the missing region is a rectangle.

For the hyperboloid

In[ ]:=
```
parHyp = {(t - s^2 t)/(1 - s^2), (1 + s^2 - 2 s t)/(1 - s^2), (2 s - t - s^2 t)/(1 - s^2)};
```

$$\text{parHyp} = \left\{\frac{t - s\text{^}2\,t}{1 - s\text{^}2}, \frac{1 + s^2 - 2\,s\,t}{1 - s^2}, \frac{2\,s - t - s^2\,t}{1 - s^2}\right\};$$

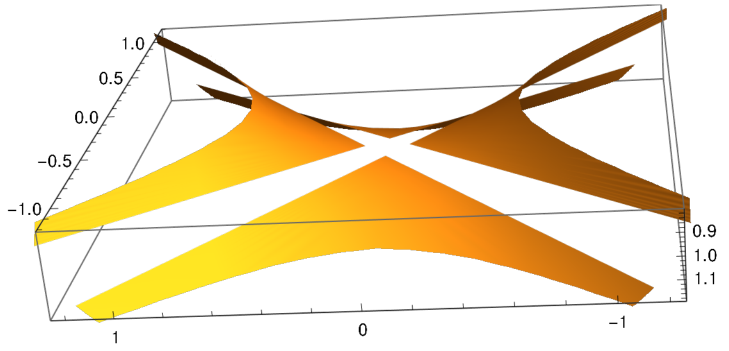
In[ ]:=
```
ParametricPlot3D [parHyp , {t, -30, 30}, {s, -30, 30}, PlotRange → Full]
```

Out[ ]=


The missing curve is the the union of two lines, but this parametric representation does not do a good job of showing the surface. We do have a better parameterization of the saddle surface, however.

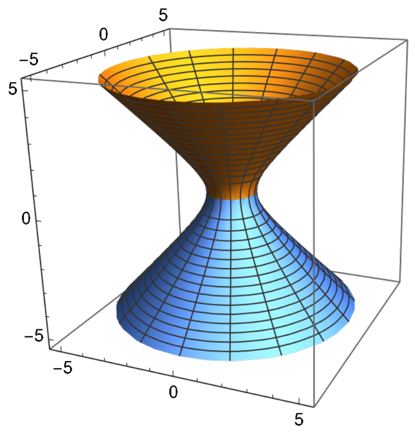
$$\text{parSS} = \left\{\frac{1 + s\,t}{-1 + s\,t}, \frac{s - t}{-1 + s\,t}, \frac{s + t}{-1 + s\,t}\right\};$$

In[ ]:=
```
ParametricPlot3D [parHyp2 , {t, -20, 20}, {s, -20, 20}, Mesh → None]
```

Out[ ]=


If you really want to plot the hyperboloid parametrically [Van Seggern] suggests the parameterization as a surface of revolution

In[ ]:=
```
ParametricPlot3D [
 {{u Cos[v], u Sin[v], Sqrt[u ^ 2 - 1]}, {u Cos[v], u Sin[v], -Sqrt[u ^ 2 - 1]}}, {u, 0, 5}, {v, -Pi, Pi}]
```

Out[ ]=


The second issue is the infinite curve. Some infinite points may arise when one or both parameters go to infinity. Otherwise infinite points may arise when the denominators become zero. One can find some examples of infinite points by normalizing points with large parameters or parameters making the denominator small, but it is hard to get an equation out of this. It is better to implicitize the surface and calculate the infinite points from that.

1.10 Lines on a Projective Surface through a Given Point.

In chapter 2 many quadric surfaces will be ruled surfaces which implies each point is contained in one or several lines of the surface. I show how to find them and their infinite points.

1.10 .1 The method

We start with a smooth real affine point p on a naive surface and find the parametric equation of the real line if it exists. We first find the tangent plane with the global function **tangentPlaneNS** and then intersect this plane with our surface using **NSolve**. If this intersection is empty or imaginary there is no real line otherwise there may be one or more lines. For each solution $q \neq p$ a possible line will have parametric equation line = $p + t\,(q - p)$ which gives the point p for t = 0 and q for t = 1. In Chapter 2 if $q \neq p$ exists the quartic surface contains this line. Otherwise we may wish to check $d - 2$ additional values of t to be sure we have a line.

An important comment is that when we intersect the the tangent plane and surface we get an underde - termined system. There should be at least one solution, the point p, but to get a line we need a solu - tion different from p. **Mathematica** should give a warning message and will provide a pseudo-random rational linear equation to obtain a finite solution set. I discuss this in section 2.3 of my *Plane Curve Book*. This should be sufficient but, as I warn, not always. Unfortunately these pseudo random linear equations are set at the initialization stage of each Mathematica session, the good thing is that if you re-run the example during a given session you will get the same answer, but the down side is that if this pseudo-random equation does not give a desired result then it is not enough to just re-run, you need to provide your own, preferably machine number, linear equation as a third equation. Incidentally, this is one reason I don't provide a global function to find these lines. Also note that the chances are that the original point is not on this pseudo-random or random plane so the points returned will be generally different from *p*, so if there is no line may be returned.

Here are some examples.

Let S be the surface Si = 0 where

```
In[ ]:= S1 = x^2 + y^2 + z^2 - 1;
        p1 = {9/11, 2/11, -6/11};

In[ ]:= S1 /. Thread[{x, y, z} → p1]

Out[ ]= 0

In[ ]:= tp1 = tangentPlaneNS[S1, p1, {x, y, z}]

Out[ ]= -2 + 18 x/11 + 4 y/11 - 12 z/11

In[ ]:= NSolveValues[{S1, tp1}, {x, y, z}, Reals]

        NSolveValues : Infinite solution set has dimension at least 1. Returning intersection of solutions with
          -69046 x/57903 - 142003 y/115806 + 40299 z/38602 == 1.

Out[ ]= {}
```

So there is no real line as expected.

Second example

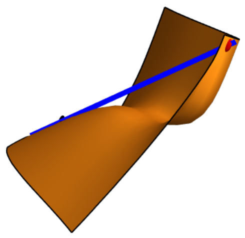
```
In[ ]:= S2 = x^3 - y^3 + z^4 - 1;
        p2 = {-0.8825870838315157`, 1.5`, -1.5`};
        tp2 = tangentPlaneNS[S2, p2, {x, y, z}]

Out[ ]= -8.0625 + 2.33688 x - 6.75 y - 13.5 z

In[ ]:= sol2 = NSolveValues[{S2, tp2}, {x, y, z}]

        NSolveValues : Infinite solution set has dimension at least 1. Returning intersection of solutions with
          -69046 x/57903 - 142003 y/115806 + 40299 z/38602 == 1.

Out[ ]= {{-24.1402, 13.042, -11.2969}, {0.122201, -0.99938, -0.076379},
         {-0.465712 - 1.1209 ⅈ, -0.659138 + 0.648697 ⅈ, -0.348269 - 0.518379 ⅈ},
         {-0.465712 + 1.1209 ⅈ, -0.659138 - 0.648697 ⅈ, -0.348269 + 0.518379 ⅈ}}
```

Pick the second real point to get line

```
In[ ]:= lS2 = p2 + t (sol2⟦2⟧ - p2)
Out[ ]= {-0.882587 + 1.00479 t, 1.5 - 2.49938 t, -1.5 + 1.42362 t}

In[ ]:= S2 /. Thread[{x, y, z} → (lS2 /. t → .8)]
Out[ ]= -0.858857
```

So this line in not in the surface . The picture is

```
In[ ]:= Show[ContourPlot3D [{S2 == 0}, {x, -4, 2}, {y, -1, 2}, {z, -2, 0}, Mesh → None],
         ParametricPlot3D [lS2, {t, -1, 2}, PlotStyle → Blue],
         Graphics3D [{{Black, Ball[p2, .07]}, {Red, Ball[sol2⟦2⟧, .06]}}],
         Axes → None, Boxed → False, ImageSize → Small]
Out[ ]=
```



This line is locally almost in the surface S2 but then diverges from the surface but intersects it transver - sally at the point sol2[[2]].

Here are some examples where we do get lines in the surface. The third example is the standard hyperboloid

```
In[ ]:= S3 = x ^ 2 + y ^ 2 - z ^ 2 - 1;
        p3 = {2.952497092684046` , -1.1325903574074156` , 3.`}
Out[ ]= {2.9525, -1.13259, 3.}

In[ ]:= tp3 = tangentPlaneNS [S3, p3, {x, y, z}]
Out[ ]= -2. + 5.90499 x - 2.26518 y - 6. z

In[ ]:= sol3 = NSolveValues [{S3, tp3}, {x, y, z}]
```

NSolveValues : Infinite solution set has dimension at least 1. Returning intersection of solutions with

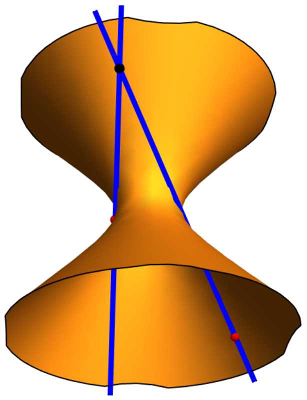
$$-\frac{69046\ x}{57903}-\frac{142003\ y}{115806}+\frac{40299\ z}{38602}==1.$$

```
Out[ ]= {{2.9525, -1.13259, 3.}, {2.9525, -1.13259, 3.}}
```

We get the original point back with multiplicity 2 as warned earlier.

```
In[ ]:= sol3 = NSolveValues [S3 == 0 && tp3 == 0 && RandomReal [{-1, 1}, 3].{x, y, z} == 1, {x, y, z}]
Out[ ]= {{-1.79925, 2.77359, -3.15121}, {-0.225929, -0.990923, -0.181582}}
```

```
In[ ]:= ls3a = p3 + t (sol3⟦1⟧ - p3)
        ls3b = p3 + t (sol3⟦2⟧ - p3)

Out[ ]= {2.9525 - 4.75175 t, -1.13259 + 3.90618 t, 3. - 6.15121 t}

Out[ ]= {2.9525 - 3.17843 t, -1.13259 + 0.141668 t, 3. - 3.18158 t}

In[ ]:= Show[ContourPlot3D [x ^ 2 + y ^ 2 - z ^ 2 == 1, {x, -4, 4}, {y, -4, 4}, {z, -4, 4}, Mesh → None],
         Graphics3D [{{Black , Ball[p3 , .15]}, {Red , Ball[sol3⟦1⟧, .15], Ball[sol3⟦2⟧, .15]}}],
         ParametricPlot3D [{ls3a , ls3b}, {t, -1, 4}, PlotStyle → Blue],
         Axes → None , Boxed → False , ImageSize → Small]

Out[ ]=
```



We get two lines on the surface through this point . We will discuss further in Chapter 2.

For our last example we jump ahead to section 3.8 where we discuss the famous Clebsch Diagonal Cubic which has 7 *Eckart points* which are each contained in three surface lines through the point. Here is one, p4.

```
In[ ]:= S4 = 81 (x ^ 3 + y ^ 3 + z ^ 3) - 189 (x ^ 2 y + x ^ 2 z + y ^ 2 x + y ^ 2 z + z ^ 2 x + z ^ 2 y) +
            54 x y z + 126 (x y + x z + y z) - 9 (x ^ 2 + y ^ 2 + z ^ 2) - 9 (x + y + z) + 1;
        p4 = {1 / 3, 1 / 3, 1 / 3};
```

Note this is a point on the surface .

```
In[ ]:= S4 /. Thread[{x, y, z} → p4]

Out[ ]= 0

In[ ]:= tp4 = tangentPlaneNS [S4, p4, {x, y, z}]

Out[ ]= 24 - 24 x - 24 y - 24 z

In[ ]:= sol4 = NSolveValues [{S4, tp4}, {x, y, z}, Reals]
```

NSolveValues : Infinite solution set has dimension at least 1. Returning intersection of solutions with

$$-\frac{69046\ x}{57903}-\frac{142003\ y}{115806}+\frac{40299\ z}{38602}==1.$$

```
Out[ ]= {{43.5121 , -42.8454 , 0.333333},
         {0.333333 , -0.30901 , 0.975676}, {-0.31871 , 0.333333 , 0.985376}}
```

Here we get 3 solutions different from our chosen point so 3 possible lines.

```
In[ ]:= ls4a = p4 + t (sol4⟦1⟧ - p4)
        ls4b = p4 + t (sol4⟦2⟧ - p4)
        ls4c = p4 + t (sol4⟦3⟧ - p4)
```

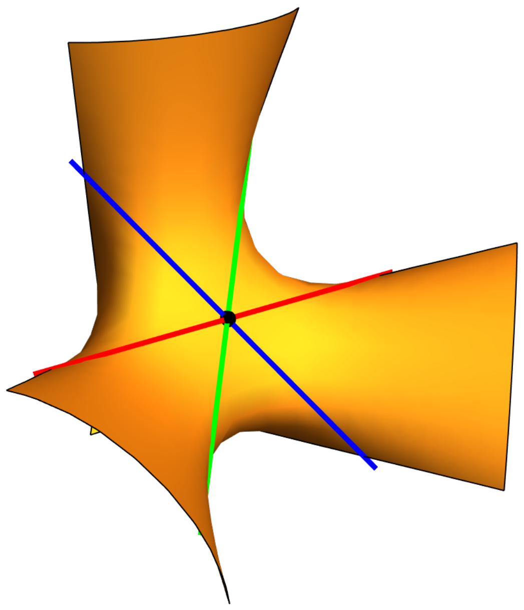
Out[ ]= $\left\{\frac{1}{3} + 43.1787\ t,\ \frac{1}{3} - 43.1787\ t,\ 0.333333\right\}$

Out[ ]= $\left\{0.333333,\ \frac{1}{3} - 0.642343\ t,\ \frac{1}{3} + 0.642343\ t\right\}$

Out[ ]= $\left\{\frac{1}{3} - 0.652043\ t,\ 0.333333,\ \frac{1}{3} + 0.652043\ t\right\}$

We should check an additional random point on each line to see if they actually lie in the surface, we leave this to the reader, but give a plot.

```
In[ ]:= Show[ContourPlot3D [S4 == 0, {x, 0, 1}, {y, 0, 1}, {z, 0, 1}, Mesh → None],
         Graphics3D [{Black , Ball[{1 / 3, 1 / 3, 1 / 3}, .02]}], ParametricPlot3D [{ls4a , ls4b , ls4c},
          {t, -2, 2}, PlotStyle → {Red , Blue , Green}], Axes → None , Boxed → False]
```

Out[ ]=


Note that in section 3.8 we give a different method of finding lines on the surface which gives all such lines at once without needing to know in advance one point on each.

1.10.2 Infinite Points

In constructing our lines we used the general parametric equation $\text{lp} = p + t\{q - p\}$ where p was our given point and q was a second point on the surface and tangent plane at p. From considerations of Section 1.9 we see the infinite point of such a line is obtained simply by appending a zero to the point

$q - p$.

For our example we consider the line ls3a of example S3 . Here

```
In[ ]:= ls3a
Out[ ]= {2.9525 - 4.75175 t, -1.13259 + 3.90618 t, 3. - 6.15121 t}
```

where

```
In[ ]:= p3 - sol3⟦1⟧
Out[ ]= {4.75175 , -3.90618 , 6.15121}
```

that is, the three coefficients of t. So the infinite point is

```
In[ ]:= ip3a = Append[p3 - sol3⟦1⟧, 0]
Out[ ]= {4.75175 , -3.90618 , 6.15121 , 0}
```

Note that the homogenization of S3 with homogenizing variable w is

```
In[ ]:= HS3 = x^2 + y^2 - z^2 - w^2;
In[ ]:= HS3 /. Thread[{x, y, z, w} → ip3a]
Out[ ]= 1.42109 × 10^-14
```

Note also if

```
In[ ]:= q3 = ls3a /. t → 100
Out[ ]= {-472.222 , 389.485 , -612.121}
```

Then

```
In[ ]:= Append[q3 / q3⟦1⟧ * ip3a⟦1⟧, 0]
Out[ ]= {4.75175 , -3.91921 , 6.15948 , 0}
```

is a good approximation to the infinite point ip3a.

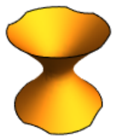
1.11 Overview of this book

In this section we give an overview of the later chapters in the book.

1.11.2 Chapter 2, Quadric Surfaces.

Here we discuss surfaces of degree 2. In this case we can completely classify the possibilities up to projective linear transformations. A nice summary chart is

Projective Real Quadric Surfaces

Type	Not Surface	Degenerate	Cone	Ellipsoid	Hyperboloid
Possible Picture					
example	$(y-2x)^2+(z+3x)^2=0$	$xz=0$	$z^2=x^2+y^2$	$x^2+y^2+z^2=1$	$x^2+y^2-z^2=1$
singularity?	All	line	point	none	none
ruled?	no	two parts	single	none	double
essential ovals?	no	no	no	no	yes
Affine Variants	empty set point, line plane squared	parallel- planes	cylinder Cone	parabolic hyperbolic	elliptic saddle- Surface

In particular note that for non degenerate surfaces the classification hinges on the number of real lines through a point on this surface as discussed in the previous section 1.10. Surfaces with no real lines are ellipsoids, those with one real line through each regular point are cones (cylinders), and those with 2 distinct real lines through each point are hyperboloids. Since projective linear transformations pre-serve lines these are clearly distinct classes under these transformations. Less obvious is the fact that each pair of surfaces in one of these classes equivalent under projective linear transformations. We show this in this chapter. In particular the saddle surface $z = x\,y$ is a hyperboloid and equivalent to any hyperboloid by a projective linear transformation. This does not seem to be clearly understood in the literature.

Since the saddle surface and cylinder $x^2 + y^2 = 1$ clearly have a rational parameterization and there is a well known one for the sphere then all non-degenerate quadric surfaces are rational.

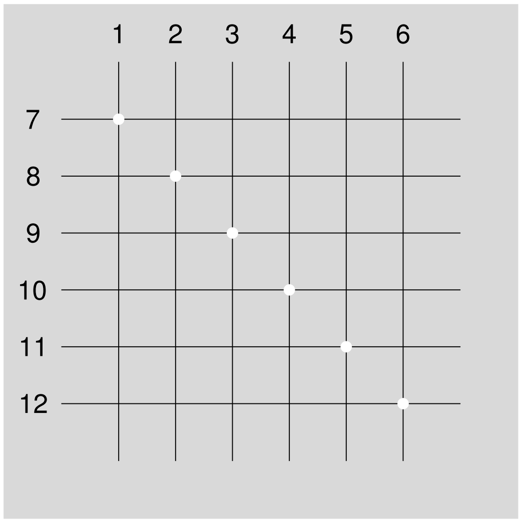
Later in the chapter we explore symmetries of the ellipses and hyperbolas and find that besides the obvious ones there are many non-obvious ones.

1.11.3 Cubic surfaces.

The surprise here is that while all smooth cubic surfaces do not have lines through every point but do have exactly 27 lines in the surface counting complex lines as well as real lines. Further they all have similar configurations. However this does not make them all equivalent under projective linear transformations.

Important in the study of these 27 lines is the Schläfli double 6 configuration.

Out[ ]=



This configuration of lines appears as a sub-configuration of the set of all lines in any smooth cubic surface. Given such a configuration it is then easy to find the additional 15 lines. Conversely given any such configuration of lines we can derive the implicit equation giving the unique cubic surface containing these lines. In fact the discussion of this shows that we only need 6 of these lines to find the unique equation. So if we find one line L1 and 5 skew lines that intersect it, call them L8, L9, L10, L11 and L12, then already we have enough information to find the unique cubic containing them and the remaining 21 lines.

Alternatively, given the cubic surface we will show how to find all the lines at once. It turns out that we can find a non-linear system of 4 equations in 4 unknowns, which Mathematica can solve in less than a second, that allows us to find parametric equations for all the lines. If we are interested we can work backwards to find a double 6 configuration.

We give a famous example, known as the *Clebsch diagonal cubic*

```
cdc = 81 (x ^ 3 + y ^ 3 + z ^ 3) - 189 (x ^ 2 y + x ^ 2 z + y ^ 2 x + y ^ 2 z + z ^ 2 x + z ^ 2 y) +
    54 x y z + 126 (x y + x z + y z) - 9 (x ^ 2 + y ^ 2 + z ^ 2) - 9 (x + y + z) + 1;
```

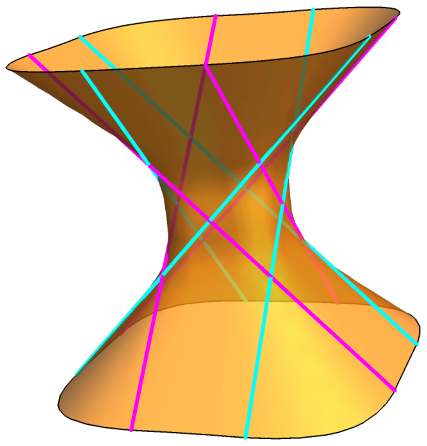
One can find this surface mentioned also by going to WolframAlpha and typing in “clebsch diagonal cubic” or MathWorld at https://mathworld.wolfram.com/ClebschDiagonalCubic.html

1.11.4 Fourth Degree and Related Surfaces

In Chapter 4 we look at some higher degree surfaces, such as the torus. Unfortunately, unlike quadric or cubic surfaces there is no overlying structure. Some of these surfaces, such as the torus, will still have infinite symmetry groups, but others will not. One of the main ideas is to discuss the topic of *geometric point groups they include the the well known crystallographic point groups.*

I, the author, will mention that I was exposed to this idea as an undergraduate math student in an abstract algebra class. The professor, Paul B. Yale, was writing a book about this subject during the time he was teaching the class. His enthusiasm for the subject affected my future as a mathematician. If not for him I probably would not be writing this book 50 years later. I will also mention John C. Baez whose wonderful article *The Octotonians* in the AMS Bulletin, April 2002, rekindled my interest in this subject.

One surface we will study is the quartic hyperboloid $x^4 + y^4 - z^4 = 1$. which does have a finite algebraic symmetry group. This comes partly from the fact that there are only finitely many lines in the quartic hyperboloid.



Another will be the torus and surfaces algebraically equivalent to this. These will have infinitely many algebraic symmetries. We will construct also an eighth degree double torus.

1.11.5 Topology and the topology of complex conic and cubic curves.

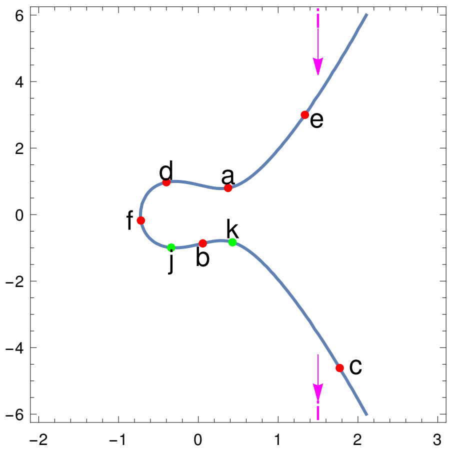
The material here is somewhat separate from the rest of the book, the reader only needs to be familiar with Chapter 1. So it can be read by itself.

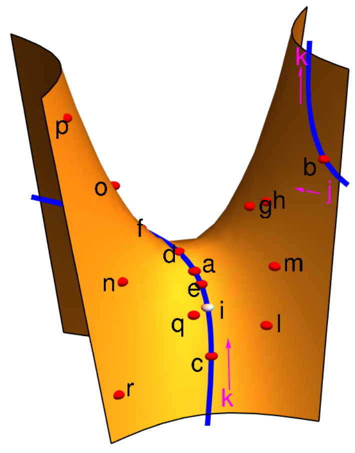
The major point in this chapter is that the sphere and projective hyperboloid are topologically distinct surfaces. We will show this visually using the chromatic number. But the the projective hyperboloid, including variants such as the projective saddle surface, are topologically equivalent to the torus. We prove the later fact by explicit invertible continuous transforms.

As an interlude away from topology I redo some of the material of Chapter 7 of my *Plane Curve Book.* In particular I have correct normal form algorithms for smooth conics and cubics. The normal form for conics is simply $y = x^2$, but the algorithm gives an actual projective linear transform taking the conic to this form. For the cubics I give the correct Weierstrass normal form $y^2 = 4 x^3 - g_2 x - g_3$ Here the constants g_2, g_3 characterize the cubics under projective linear equivalence. So here the normal form algorithm calculates these numbers as well as the projective linear transform connecting the original cubic to it normal form.

I then use this material to first calculate the topology of smooth plane conics and cubics. This material is known since the topology depends only on the genus which is 0 for conics and 1 for cubics. However my explicit approach depends on the earlier results in this Chapter. I show that the complex projective solution set of a conic equation is the sphere, something I have not seemed mentioned explicitly elsewhere while it is well known since the early 1800's that the complex projective solution set of the cubic is the torus. But this is usually known only theoretically, in this chapter I use the Mathematica implementation of the Weierstrass P function to give explicit maps from the solution set to and from the torus. In fact, since the torus is topologically equivalent to the saddle surface $z = x y$ any pair of real numbers gives a solution, usually complex, of the cubic equation. Unlike the torus, however, some of the real points of the saddle surface are infinite so one does not get all solutions this way from the affine saddle surface.

$$0.805458 - 0.996016\, x + 4\, x^3 - y^2 = 0$$





"point"	"solution f8"	"saddle surface"
"a"	{0.373058 , 0.800976 }	{1.17821 , −0.910332 , −1.07256 }
"b"	{0.0557982 , −0.866358 }	{7.05402 , 1.13234 , 7.98754 }
"c"	{1.77165 , −4.61346 }	{2.84656 , −2.19231 , −6.24053 }
"d"	{−0.4 , 0.973583 }	{0.145976 , −0.651173 , −0.095056 }
"e"	{1.33535 , 3.}	{1.65126 , −1.10044 , −1.81711 }
"f"	{−0.719703 , −0.176495 }	{−2.15759 , −0.378341 , 0.816306 }
"g"	{0.454513 − 0.21764 i, −0.713384 + 0.197311 i}	{3, 1.1 , 3.3}
"h"	{0.249724 − 0.0718865 i, −0.776975 − 0.0124136 i}	{4, 1, 4}
"i"	"infinite Point"	{2.20657 , −1.44126 , −3.18024 }
"j"	{−0.339802 , −0.993461 }	"infinite Point"
"k"	{0.427876 , −0.832241 }	"infinite Point"
"l"	{−1.21094 − 0.307908 i, −1.12035 + 2.22907 i}	{5, −0.8 , −4}
"m"	{−0.0439841 − 0.17124 i, −0.935075 − 0.0998141 i}	{5, 0, 0}
"n"	{−0.293159 + 0.652481 i, −1.61489 + 0.336897 i}	{0, −5, 0}
"o"	{0.219746 + 0.391215 i, −0.585984 + 0.343408 i}	$\{-\sqrt{5}, -\sqrt{5}, 5\}$
"p"	{−0.0428533 + 0.392137 i, −1.01089 + 0.30821 i}	{−2, −6, 12}
"q"	{−9.98835 − 0.196065 i, 1.85994 − 63.0407 i}	{1.8 , −2, −3.6}
"r"	{−1.37753 + 0.802384 i, −2.99584 − 2.57118 i}	{1, −7, −7}

References

[Plane Curve Book] Barry H. Dayton, *A numerical approach to Real Algebraic Curves with the Wolfram Language,* Wolfram Media, 2018. The print version is available only from Amazon.com but for Mathematica users notebook versions are available free from Wolfram media (Zip file) or the author's website https://barryhdayton.space/curvebook/ChapterNotebooks.html on a chapter by chapter basis. The latter notebooks contain some of the corrections noted on the author's website.

[Space Curve Book] Available on author's website https://barryhdayton.space/SpaceCurves/spindex.html

[Abhyankar] Shreeram S. Abhyankar, *Algebraic Geometry for Scientists and Engineers* , AMS, 1990.

[Montiel, Ros] Sebastián Montiel, Antonio Ros, Curves and Surfaces, AMS, 2009.